المهارات الرقمية

الصف الحادي عشر - كتاب الطالب

الفصل الدراسي الأول

11

لجنة الإشراف على التأليف

أ.د. وليد خالد سلامة

أ.د.باسل علي محافظة

أ.د. خالد إبراهيم العجلوني

ليلى محمد العطوي

هذا الكتاب جزء من مشروع الشباب والتكنولوجيا والوظائف لدى وزارة الإقتصاد الرقمي والريادة.

الناشر: المركز الوطني لتطوير المناهج

يسر المركز الوطني لتطوير المناهج استقبال آرائكم وملحوظاتكم على هذا الكتاب عن طريق العناوين الآتية:

06-5376262 / 237 | 06-5376266 | P.O.Box: 2088 Amman 11941

@nccdjor | feedback@nccd.gov.jo | www.nccd.gov.jo

قرَّرت وزارة التربية والتعليم تدريس هذا الكتاب في مدارس المملكة الأردنية الهاشـــمية جميعها، بناءً على قرار المجلس الأعلى للمركز الوطني لتطوير المناهج في جلسته رقم (/) تاريخ (/) وقرار مجلس التربية والتعليم رقم (/) تاريخ (/) بدءًا من العام الدراسي (/)

ISBN 978-9923-41-659-4

المملكة الأردنية الهاشمية

رقم الإيداع لدى دائرة المكتبة الوطنية

(3869/7/2024)

الاردن، المركز الوطني لتطوير للمناهج

المهارات الرقمية، الصف الحادي عشر، الفصل الدراسي الاول

عمان، المركز الوطني لتطوير للمناهج، 2024

373.19

/ المهارات الحاسوبية/ / علم الحاسوب/ / المناهج/ / التعليم الثانوي/

يتحمل المؤلف كامل المسؤولية القانونية عن محتوى مصنفه ولا يعبر هذا المصنف عن دائرة المكتبة الوطنية.

فريق التأليف المكلف من شركة عالم الاستثمار للتنمية والتكنولوجيا

د. اسماء حسن حمدان

د. إبراهيم سلامة البلوي

د. محمد جمال عبد الرحمن

حنان حسني أبو راشد

الطبعة الأولى (التجريبية)

1445هـ/ 2024م

المقدمة

انطلاقًا من إيمان المملكة الأردنية الهاشـــمية بأهمية تنمية قدرات الإنسان الأردني، وتسليحه بالعلم والمعرفة؛ ســـعى المركز الوطني لتطوير المناهج، بالتعاون مـــع وزارة التربية والتعليم، إلى تحديث المناهج الدراســـية وتطويرها، لتكون مُعينًا للطلبة على الارتقاء بمستواهم المعرفي والمهاري، ومجاراة أقرانهم في الدول المُتقدِّمة. ونظرًا إلى أهمية مبحث المهارات الرقميـــة ودوره في تنمية مهارات التفكير لدى الطلبة، وفتح آفاق جديدة لهم تُواكِب مُتطلَّبات ســـوق العمل؛ فقد أَوْلى المركز مناهجه عناية فائقة، وأَعدَّها وَفق أفضل الأســـاليب والطرائق المُتَّبَعة عالميًّا بإشراف خبراء أردنيين؛ لضمان توافقها مع القِيَم الوطنية الأصيلة، ووفائها بحاجات الطلبة.

يُعَدُّ مبحث المهارات الرقمية واحدًا من أهمِّ المباحث الدراسية؛ إذ يُمثِّل الخطوة الأولى لتعريف الطلبة بمناحي التكنولوجيا والتطوُّر الرقمي الحديث بصورة موثوقة وآمنة. وقد اشتمل كتاب المهارات الرقمية على موضوعات تراعي التدرُّج في تقديم المعلومة، وعرضها بأســـلوب مُنظَّم وجاذب، وتعزيزها بالصور والأشـــكال؛ ما يُثْري المعرفة لدى الطلبة، ويُعزِّز رغبتهم في التعلُّم، ويُحفِّز هم على أداء أنشطة الكتاب المُتنوِّعة بيسر وسهولة، فضلًا عن تذكيرهم بالخبرات والمعارف التعليمية التي اكتسبوها سابقًا.
روعي في إعداد الكتاب الربط بين الموضوعات الجديدة على نحوٍ شامل ومُتكامِل، وتقديم موضوعاته بصورة شائقة تُعْنى بالســـياقات الحياتية التي تَهُمُّ الطلبة، وتزيد من رغبتهم في تعلُّم المهارات الرقمية. وقد أُلحِق بكل وحدة مقاطع تعليمية مُصوَّرة، تساعد الطلبة على الفهم العميق للموضوع، وتُرسِّخ لديهم ما تضمَّنه من معلومات وأفكار.

ونظرًا إلى ما تُمثِّله الأنشـــطة من أهمية كبيرة في فهم الموضوعات وتعزيـــز الطلاقة الإجرائية لدى الطلبة؛ فقد اشتمل الكتاب على أنشطة مُتنوِّعة تحاكي واقع الطلبة وما يحيط بهم، وتدعم تعلُّمهم، وتُثْري خبراتهم، فضلًا عن اشتماله على روابط إلكترونية يُمكِن للطلبة الاستعانة بها عند البحث في الأوعية المعرفية. ومن ثَمَّ، فإنَّ المهارات الرقمية والتقنية ترتبط ارتباطًا وثيقًا بمسيرة الطلبة التعليمية والمهنية.

ونحن إذ نُقدِّم هذا الكتاب، فإنَّنا نأمل أنْ يُســـهِم في بناء جيل واعٍ ومُبتكِر وقـــادر على التعامل مع التكنولوجيا بمسؤولية وإبداع، وأنْ يكون لَبِنة أساسية في تقدُّم المملكة الأردنية الهاشمية وازدهارها.

المركز الوطني لتطوير المناهج

الفهرس

الخوارزميات والبرمجة (Algorithms and Programing) 8

مُقدِّمة في لغات البرمجة (Introduction to Programming Languages) 10

أوّلًا: لغات البرمجة مُنخفِضة المستوى (Low– Level Languages) 12

ثانيًا: لغات البرمجة عالية المستوى (High– Level Languages) 12

أساسيات لغة البرمجة بايثون (Basics of Python Programming) 20

تثبيت لغة البرمجة بايثون (Python Setup) 21

مُحرّرات النصوص وبيئات التطوير المُتكامِلة (Text Editors and Integrated Development Environment) 24

كتابة برنامج بلغة البرمجة بايثون (Python) وحفظه 25

جملة الإدخال input() 26

طريقة إدخال عدد في لغة البرمجة بايثون (Python) 28

عناصر لغة البرمجة بايثون (Python) 29

أنواع المُتغيّرات في لغة البرمجة بايثون (Python) 32

أولوية العوامل وترابطها (Operators precedence and associativity) 42

الجمل الشرطية (Conditional Statements) 48

أنواع الجمل الشرطية في بايثون 49

الجملة الشرطية (if elif else statement) 53

المُعامِلات المنطقية (Logical Operators) 54

الحلقات (Loops) 62

أنواع الحلقات في برمجية بايثون (Python) 63

حلقات while (while loops) 64

جملة التحكُّم (break) في حلقات (while) 66

جملة التحكُّم (continue) في حلقات (while) 66

جملة (else) مع حلقات (while) 67

حلقات for (for loops) 69

الدالّة range() مع حلقات (for) 70

جملة التحكُّم (break) مع حلقات (for) 72
جملة التحكُّم (continue) مع حلقات (for) 73
جملة (else) مع حلقات (for) 73
حلقات (for) المُتداخِلة (Nested for Loops) 74

القوائم (Lists) 78
القوائم (Lists) 79
الوصول للعناصر في القائمة 82
المرور على القوائم 84
العمليات في القوائم 86
الدوالُّ الجاهزة لمعالجة القوائم 89
سلاسل الحروف (Strings) 92
الأحرف الخاصة 98
القوائم المُركَّبة 99

الدوال البرمجية (Functions) 110
الدوالُّ البرمجية 111
إرجاع النتائج 116
مدى المُتغيّرات (Scope) 118

مُلخَّصُ الوحدةِ 131
أسئلةُ الوحدةِ 134
تقويمٌ ذاتيٌّ (Self Evaluation) 139

الحوسبة الخضراء (Green Computing) 144
الحوسبة الخضراء: تعريفها، وأهميتها 145
تعريف الحوسبة الخضراء 146
أهمية الحوسبة الخضراء 146
طرائق تطبيق الحوسبة الخضراء 147
مُعوّقات تطبيق الحوسبة الخضراء 151
تطبيق الحوسبة الخضراء في الأردن 152

النفايات الإلكترونية (Electronic Waste) 158
تعريف النفايات الالكترونية (E– Waste Definition) 159
إدارة النفايات الإلكترونية (E–waste Management) 161
البصمة الكربونية الرقمية (Carbon Digital Footprint) 166

تطبيقات الحاسوب في الحياة (Computer Applications in our Daily Life) 172
تطبيقات حاسوبية في مجال التعلُّم الإلكتروني (E– Learning) ومجال التعلُّم عن بُعْد (Online Learning) 174
تطبيقات حاسوبية في مجال الصحة 178
تطبيقات حاسوبية في مجال التسوُّق والتسويق الإلكتروني 180
تطبيقات الحكومة الإلكترونية 183
تطبيقات حاسوبية للوسائط المُتعدّدة 184

مُلخَّصُ الوحدةِ 191
أسئلةُ الوحدةِ 192
تقويمٌ ذاتيٌّ (Self Evaluation) 195

دلالات أيقونات الكتاب

نشاط تمهيدي	نشاط استهلالي يربط التعلم السابق بالتعلم الحالي
نشاط عملي	نشاط تطبيقي مرتبط بمهارات الدرس
نشاط	نشاط مرتبط بمحتوى الدرس المعرفي أو المهاري
نشاط فردي	نشاط يطبق بشكل فردي
نشاط جماعي	نشاط يطبق في مجموعات
أبحث	أستخدم شبكة الإنترنت للبحث عن المعلومات

اثراء	توسع في المعلومات مرتبط بمحتوى الدرس
أناقش	عرض الأفكار وتبادلها مع الزملاء والمعلم
إضاءة	معلومة إضافية
أشاهد	عرض محتوى فيديو مرتبط بالمحتوى
مشروع	نشاط تكاملي توظف فيه معارف ومهارات الوحدة
مواطنة رقمية	الإجراءات الواجب اتباعها لتحقيق مبادئ المواطنة الرقمية
المهارات الرقمية	المهارات التكنولوجية التي سأطبقها في الوحدة

الوحدة 1

الخوارزميات والبرمجة
(Algorithms and Programing)

نظرة عامة على الوحدة

سأتعرَّف في هذه الوحدة لغات البرمجة عالية المستوى، وأُقارِنها بلغات البرمجة مُنخفِضة المستوى. كذلك سأتعرَّف كلًّا من المُترجِمات، والمُفسِّرات، وبيئات التطوير المُتكامِلة، بما في ذلك لغة بايثون (Python) بوصفها مثالًا على اللغات المُفسَّرة عالية المستوى، التي تُبيِّن كيف يُمكِن بها استخدام بيئة عمل برمجية بسيطة في إنشاء البرامج وتشغيلها.

سأتعرَّف في هذه الوحدة أيضًا أساسيات اللغة التي لها تعلُّق بالمُتغيِّرات، وأنواع البيانات، والعمليات الحسابية، وبعض جمل الإدخال، والطباعة؛ ما يُمكِّنني – في نهاية الوحدة– من إنشاء برامج قصيرة، تحوي أكثر من مُتغيِّر وعمليات حسابية بسيطة، وذلك باستخدام لغة بايثون (Python).

يُتوقَّع منّي في نهاية الوحدة أنْ أكون قادرًا على:

- شرح ماهيَّة لغات البرمجة، وبيان أهميتها في تطوير البرمجيات.
- تطوير برنامج بسيط باستخدام لغة بايثون (Python)؛ لحَلِّ مشكلة مُعيَّنة، بناءً على الخوارزميات، أو الأفكار التي تخدم المجتمع.
- توضيح قواعد الكتابة الصحيحة للشيفرة البرمجية في لغة بايثون (Python).
- تعريف المُتغيِّرات في لغة بايثون (Python)، واستعمالها لتخزين البيانات وإجراء العمليات عليها.
- كتابة التعابير والعلاقات الحسابية والمنطقية واستخدامها في لغة بايثون (Python).
- كتابة الجمل الشرطية والحلقات، واستعمالها لتنفيذ عمليات مُتكرِّرة واتِّخاذ قرارات منطقية.
- إنشاء القوائم واستخدامها في لغة بايثون (Python)؛ لإدارة مجموعات البيانات.
- تحليل المشكلة وتقسيمها إلى أجزاء صغيرة؛ ما يتيح التعامل معها بفاعلية.
- توثيق الشيفرة البرمجية باستخدام المُخطَّطات وأدوات العرض.

Python

IDLE

Command Line

مُنتَجات التعلُّم (Learning Products):

تصميم برنامج وإعداده لإنشـــاء لعبة تخمين الأرقام باستخدام لغة بايثون (Python).

مشروع

أختار مع أفراد مجموعتي أحد المشـــروعات الآتيـــة لتنفيذه في نهاية الوحدة:

المشروع الأوَّل: تصميم برنامج وإعداده لإنشاء لعبة الألغاز باستخدام لغة بايثون (Python).

المشروع الثاني: تصميم نموذج أوَّلي وإعداده لإنشاء برنامج، بناءً على أفكار تخدم المجتمع والبيئة المحيطة.

المشـــروع الثالث: تصميم نموذج أوَّلي وإعداده لإنشاء برنامج يساعد الطالب على إدارة المهام المنوطة به.

الأدوات والبرامج (Digital Tools and Programs):

Python, IDLE, Command Line, (draw.io), google slides

المهـــارات الرقميـــة (Digital Skills): تصميـــم الخوارزميات، التفكير الحاسوبي، البرمجة، حَلُّ المشكلات البرمجية.

فهرس الوحدة

- الدرس الأوَّل: مُقدِّمة في لغات البرمجة (Introduction to Programming Languages).
- الدرس الثاني: أساسيات لغة البرمجة بايثون (Basics of Python Programming).
- الدرس الثالث: الجمل الشرطية (Conditional Statements).
- الدرس الرابع: الحلقات (Loops).
- الدرس الخامس: القوائم (Lists).
- الدرس السادس: الدوال البرمجية (Functions).

الدرسُ الأوَّلُ

مُقدِّمة في لغات البرمجة
(Introduction to Programming Languages)

الفكرة الرئيسة:

تعرُّف لغات البرمجة، وبيان تصنيفاتها (لغات البرمجة عالية المستوى، لغات البرمجة مُنخفِضة المستوى، لغات البرمجة الكتلية، لغات البرمجة النصية)، والمقارنة بينها، فضلًا عن تعرُّف العلاقة بين الخوارزميات والبرمجة، وتمثيل البرامج بالخوارزميات ومُخطَّطات سَيْر العمليات.

المفاهيم والمصطلحات:

الخوارزميات (Algorithms)، الخوارزمية شبه الرمزية (Pseudocode)، مُخطَّطات سَيْر العمل (Flowcharts)، لغة الآلة (Machine Language)، لغة التجميع (Assembly Language)، الكتل الرسومية (Graphical Blocks)، المُترجِم (Compiler)، المُفسِر (Interpreter).

نتاجات التعلُّم (Learning Objectives):

- أُعرِّف المقصود بلغة البرمجة.
- أُعدِّد بعض لغات البرمجة التي تختلف في مزاياها ووظائفها.
- أُقارِن بين لغة البرمجة الكتلية ولغة البرمجة النصية.
- أُقارِن بين لغات البرمجة عالية المستوى ولغات البرمجة مُنخفِضة المستوى.
- أُوضِّح العلاقة بين الخوارزميات والبرمجة.
- أُمثِّل البرامج بالخوارزميات ومُخطَّطات سَيْر العمليات.

مُنتَجاتُ التعلُّم
(Learning Products)

إعداد عرض تقديمي باستخدام google slides يتضمن شرحا للمشكلة وأسباب اختيارها والحل المقترح/ اللعبة والهدف منها، ووصف لسيناريو اللعبة وقواعد اللعبة منذ بدايتها حتى انتهائها، واعداد مخطط سير عمل (flowcharts) لتصميم لعبة التخمين باستخدام احدى الأدوات الرقمية مثل draw.io، ضمن سياق تصميم لعبة تخمين الأرقام باستخدام برمجية بايثون

نشاط تمهيدي

تعلَّمْتُ في صفوف سابقة كيف أُطوِّر برامج باستخدام برمجية سكراتش (Scratch) كما تعلمت تطوير مواقع الويب باستخدام لغة توصيف النص (HTML). – بالتعاون مع أفراد مجموعتي– أراجع ما تعلَّمْتُه عن هاتين البرمجيتين، ثمَّ أُقارِن بينهما من حيث الهدف، والاستخدام، والواجهة، وبيئة التطوير، وسهولة التعلُّم، ثمَّ أُدوِّن ما أتوصَّل إليه في ملف مُعالِج النصوص (Word).

لغة البرمجة (Programming Language)

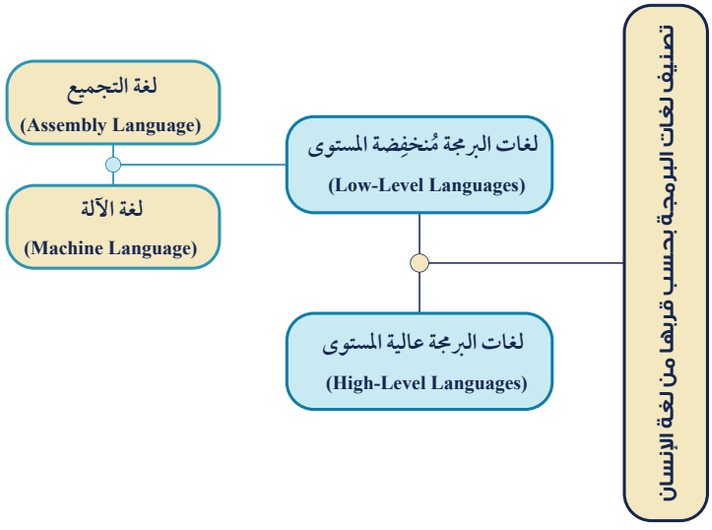
تُعرَّف لغة البرمجة بأنَّها مجموعة من الأوامر والتعليمات التي تُستخدَم في كتابة البرامج والتطبيقات وَفق قواعد مُحدَّدة. وهي تُعَدُّ الأداة الرئيسة التي يستخدمها المُبرمِجون في التفاعل مع جهاز الحاسوب، وتوجيهه لتنفيذ مهام مُعيَّنة.

تُصنَّف لغات البرمجة إلى أنواع عِدَّة، بناءً على وظائف كلٍّ منها، وتطبيقاتها، وطرائق معالجتها، وغير ذلك من المعايير والضوابط.

توجد جملة من المعايير والضوابط الرئيسة التي تَحْكُم تصنيف لغات البرمجة، ويأتي في مُقدِّمة هذه المعايير والضوابط درجة قرب لغات البرمجة من اللغات الإنسانية، أنظر الشكل (1-1). وتأسيسًا على ذلك، يُمكِن تصنيف لغات البرمجة إلى نوعين، هما: لغات البرمجة عالية المستوى، ولغات البرمجة مُنخفِضة المستوى.

إضاءةٌ

برنامج الحاسوب هو مجموعة من الأوامر التي تُكتَب بإحدى لغات البرمجة؛ بهدف حَلِّ مشكلة ما، أو أداء مهمة مُحدَّدة باستخدام جهاز الحاسوب.



الشكل (1-1): تصنيف لغات البرمجة تبعًا لقربها من لغات الإنسان.

أوَّلًا: لغات البرمجة مُنخفِضة المستوى (Low-Level Languages):

تمتاز لغات البرمجة مُنخفِضة المستوى بقربها من لغة الآلة، خلافًا للغات البرمجة عالية المستوى. وهي تنقسم إلى قسمين، هما:

1. **لغة الآلـــة (Machine Language)**: لغة برمجة تحتوي على أوامـــر وتعليمات يُمكِن لجهاز الحاسوب فهمها مباشرة ومعالجتها، خلافًا للإنسان الذي يصعب عليه فهمها. تمتاز هذه اللغة بأنَّها سريعة مقارنةً بلغات البرمجة عالية المستوى.
2. **لغة التجميع (Assembly Language)**: لغة تقوم على اســـتخدام برنامج خاص يُسمّى المجمع (Assembler)، ويعمل على تحويل الأوامر المكتوبة إلى لغة الآلة التي يفهمها جهاز الحاسوب. تمتاز هذه اللغة بأنَّها أسهل من لغة الآلة؛ نظرًا إلى احتوائها على بعض مفردات اللغة الإنجليزية؛ ما يُســـهِّل قراءة برامجها وفهمها. غير أنَّ تنفيذ البرامج المكتوبة بلغة التجميع يكون أبطأ مقارنةً بلغة الآلة.

ثانيًا: لغات البرمجة عالية المستوى (High-Level Languages):

تمتاز لغات البرمجة عالية المســـتوى بمواءمتها للغة التي يفهمها الإنســـان؛ إذ تُستخدم في كتابة البرامج رموز ومفردات قريبة من تلك المُتداوَلة في اللغة الإنجليزية. وقد سُمِّيت هذه اللغات بهذا الاســـم لبُعْدها عن اللغة التي يفهمها جهاز الحاسوب؛ أيْ لغة الآلة. ومن ثمَّ، فهي لا تعتمد على أنواع أجهزة الحاســـوب في أداء وظائفها، وإنَّما صُمِّمت على نحوٍ يجعلها موائمةً لجميع أجهزة الحاسوب، بِغَضِّ النظر عن نوع هذه الأجهزة وأنظمة تشغيلها.

من الأمثلة على لغات البرمجة عالية المســـتوى: لغة بايثـــون (Python)، ولغة جافا (Java)، ولغة سي ++ (C++)، ولغة سي شارب (C#).

كما يُمكِن تصنيف لغات البرمجة إلى نوعين آخرين، هما:

1. **لغات البرمجة الكتلية (Block-Based Programming Languages)**: لغات تُستخدَم فيها الكتل الرســـومية (Graphical Blocks) لتمثيل أجزاء البرامج بدلًا من النصـــوص، مثل لغة البرمجة سكراتش (Scratch).
2. **لغات البرمجة النصية (Text-Based Programming Languages)**: لغات تُستخدَم فيها النصوص لتمثيل أجزاء البرامج بدلًا من الكتل الرسومية، مثل لغة جافا سكريبت (Java Script).

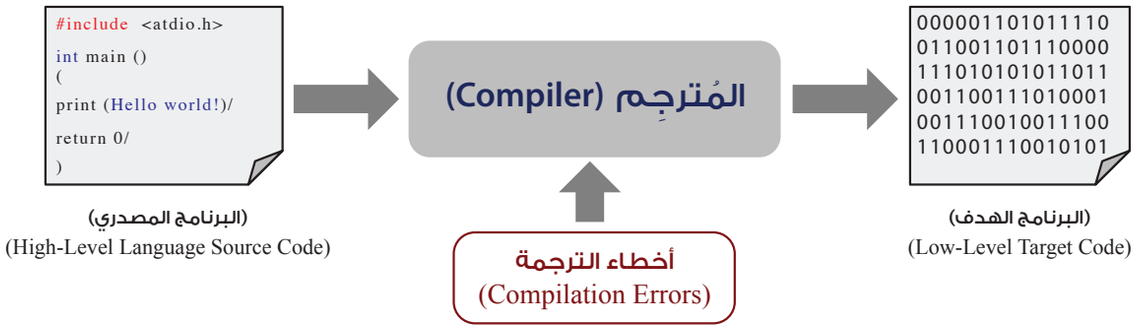
نشاط

أُقارِن وأُناقِش: أُقارِن بيـــن لغات البرمجة المختلفة، ثمَّ أُناقِش زملائي/ زميلاتي في الكيفية التي تتغيَّر فيها طبيعة البرمجة تبعًا لتغيُّر المزايا في كل لغة برمجة.

المُترجِم والمُفسِّر (Compiler and Interpreter)

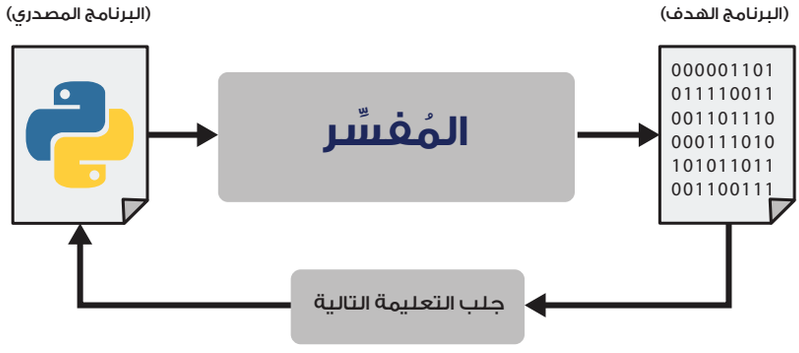
المُترجِم والمُفسِّر هما برنامجان يعملان على تحويل البرنامج المكتوب بلغة برمجة عالية المستوى إلى أوامر مباشرة يفهمها جهاز الحاسوب، ويُسارِع إلى تنفيذها.

- **المُترجِم (Compiler)**: تتمثَّل وظيفة المُترجِم في الفحص الكامل لأيِّ برنامج كُتِب بلغة البرمجة عالية المستوى (البرنامج المصدري)، ثمَّ ترجمته إلى لغة الآلة (البرنامج الهدف)؛ لكيْ تتمكَّن وحدة معالجة البيانات من تنفيذه. ويُمكِن للمُترجِم اكتشاف بعض أنواع من الأخطاء في البرنامج أثناء مرحلة الترجمة، وقبل البَدْء بتنفيذه، أنظر الشكل (1-2).



الشكل (1-2): مبدأ عمل المُترجِم.

- **المُفسِّر (Interpreter)**: يعمل المُفسِّر على تحويل كل جزء من أجزاء البرنامج المكتوب بلغة البرمجة عالية المستوى إلى لغة الآلة، ثمَّ تنفيذ هذه الأجزاء أمرًا بأمر؛ فعند وجود أمر خطأ تتوقف عملية تحويل الأجزاء المُتبقِّية. غير أنَّ المُفسِّر لا يعمل أحيانًا على تحويل البرنامج إلى لغة الآلة بصورة مباشرة، وإنَّما يقوم بتحويل البرنامج إلى لغة وسيطة أولا، ثمَّ يُحوِّل كل جزء من أجزاء البرنامج الناتج إلى لغة الآلة، أنظر الشكل (1-3).



الشكل (1-3): مبدأ عمل المُفسِّر.

بعد ذلك يعمل جهاز الحاسوب على تنفيذ الأوامر التي خضعت للترجمة أو التفسير، ثمَّ يتولّى المُعالِج تنفيذ (Execution) التعليمات تِباعًا وَفق ترتيبها في البرنامج.

نشاط

أبحث وأُقارِن: أبحث في المواقع الإلكترونية الموثوقة في شبكة الإنترنت عن أمثلة على كلٍّ من المُترجِم والمُفسِّـــر، ثمَّ أُقارِن بينهما من حيث آليَّة التنفيذ، والسرعة في التنفيذ، وسهولة اكتشاف الأخطاء.

الخوارزميات (Algorithms)

الخوارزمية هي مجموعة من الخطوات المُرتَّبة والمُتسلسِلة منطقيًّا، تهدف إلى حَلِّ مسألة مُعيَّنة بناءً على معطيات مُحدَّدة. وبعبارة أدقَّ، فإنَّ الخوارزمية تهدف إلى تقديم حَلٍّ منهجي ومُنظَّم للمسائل المختلفة؛ سواء كانت بسيطة أو مُعقَّدة. وهي تُستخدَم على نطاق واسع في علوم الحاسوب لتطوير البرامج والتطبيقات.

يعمل جهاز الحاســـوب على تنفيذ الخوارزميات بعد كتابتها في صورة برنامج باستخدام إحدى لغات البرمجة.

يراعى عند كتابة الخوارزميات مجموعة من المعايير والضوابط، أبرزها:

1. التسلسل المنطقي للخطوات والتعليمات.
2. التحديد الواضح والدقيق للمدخلات التي تدخل الخوارزمية، والمخرجات التي تنتج منها.
3. وضوح الخطوات، وسهولة تتبُّعها.
4. الفاعلية مُمثَّلةً في سرعة تنفيذها.

يُمكِن تمثيل خوارزمية البرنامج بطريقتين، هما:

- **الخوارزمية شـــبه الرمزية (Pseudocode):** تُكتَب الخوارزمية شـــبه الرمزية في مجموعة من الخطوات المُرقَّمة، تُستخدَم فيها لغة الإنسان، والتعابير والرموز الرياضية البسيطة.

مثال: أكتب خوارزمية شبه رمزية لإيجاد ناتج المعادلة: y = a * x + b ، ثمَّ طباعتها.

الحَلُّ: أتَّبع الخطوات الآتية في الحَلِّ:

1. أبدأ.
2. أُدخِل قِيَم المُتغيِّرات: a ، وx، وb.
3. أَجِد ناتج ضرب a * x
4. أَجِد قيمة y بتطبيق المعادلة الآتية: (y = a * x + b).
5. أطبع قيمة y.
6. أتوقَّف.

- **رسم مُخطَّطات سَيْر العمليات (Flowcharts):** تتمثَّل هذه الطريقة في رسم الخوارزمية باستخدام أشكال هندسية مُتعارَف عليها، ومجموعة من الأسهم والخطوط التي تُحدِّد سَيْر الخوارزمية. وكل شكل من هذه الأشكال يدلُّ على خطوة مُعيَّنة من خطوات تمثيل الخوارزمية، أنظر الشكل (1-4).

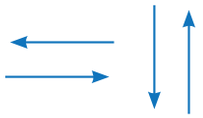
بَدْء الخوارزمية وإنهاؤها:

تنفيذ العملية:

الإدخال أو الإخراج:

اتِّخاذ القرار:

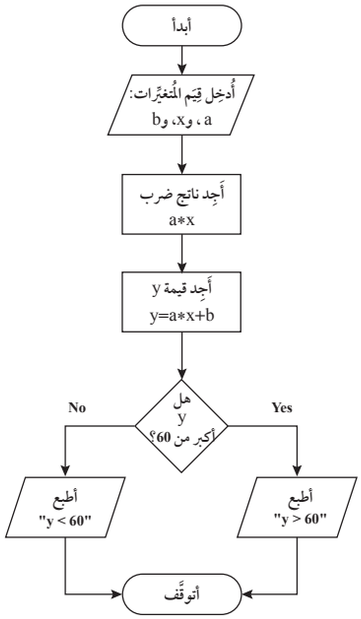
اتجاه سير العمليات:



الشكل (1-4): أكثر الأشكال الهندسية استخدامًا في تمثيل الخوارزميات.

مثال: عُدِّل المثال الســـابق بحيث تطبع الخوارزمية ($y > 60$) إذا كان الناتج أكبر من 60، وتطبع ($y < 60$) إذا كان الناتج أقل من 60، إضافةً إلى طباعة الناتج y.

يُمكِن تمثيل مخطط سير عمل للخوارزمية في هذا المثال كما في الشكل (1-5).



الشكل (1-5): مُخطَّط سَيْر عمل لخوارزمية.

المواطنةُ الرقميةُ

- أحرص دائمًا على اســـتخدام البرمجيات المُرخَّصة قانونيًّـــا، واحترام حقوق المُلْكية الفكرية للمُطوِّرين والشركات.
- أتجنَّب استخدام البرمجيات المُقرصَنة أو البرمجيات غير المُرخَّصة؛ لأنَّها قد تكون غير آمنة، وتُعرِّض جهازي ومعلوماتي الشخصية للخطر.

مشروع

تصميم برنامج وإعداده لإنشـــاء لعبة تخمين الأرقام باستخدام لغة بايثون (Python)/ المهمة (1).

أتعاون مـــع مجموعتي على البدء بالتحضيرات اللازمة لتصميـــم لعبة تخمين للأرقام ذات واجهة نصية عبر برمجية بايثون، باتباع الخطوات الآتية:

في هذا الدرس يكون التركيز على: مرحلة التخطيط والتحليل

- **التعريف للمشكلة/** فكرة اللعبة التي تم اختيارها وتوضيح الهدف منها، وتحديد الجمهور المستهدف.

- **وصف لسيناريو اللعبة:** رسم مخطط لسير العمل في اللعبة في محاولة تشكيل فهم شامل للعمل الذي ســـنقوم به والبدء بتجزئته لنتمكن من إنجازه على مراحل: كيف نبدأ؟ ما عدد اللاعبين؟ ما قواعد اللعبة واجراءاتها؟ متى يعتبر اللاعب فائزاً؟ متى تنتهي اللعبة؟ ...الخ.

- **فريق العمل:** توزيع الأدوار مع الزملاء في المجموعة للعمل على المشروع.

بعد مناقشة الأسئلة الســـابقة والاجابة عليها ضمن المجموعة لنلخص نتائج نقاشنا سنعمل على إعداد عرض تقديمي باستخدام google slides يتضمن شرحا للمشكلة وأسباب اختيارها والحل المقترح/ اللعبة والهدف منها، ووصف لســـيناريو اللعبة وقواعد اللعبة منذ بدايتها حتى انتهائها، وإعداد الخوارزمية ومخطط سير العمليات (flowcharts) لتصميم لعبة التخمين باستخدام إحدى الأدوات الرقمية مثل draw.io. كما تعلمنا في السنوات السابقة.

تقديم فكرة المشـــروع: برمجة لعبة التخمين"نجوم وأقمار". للمصمم تكتب ضمن أيقونة المشروع

بعد الانتهاء من هذه الوحدة ســـتكون قد أنهيت برمجة لعبة بسيطة اسمها "نجوم وأقمار" ، وهي من ألعاب التخمين. فكرة اللعبة قائمة على إضمار عدد مكون 4 أرقام مختلفة يتم اختيار رقم مكون من عدة منازل (عادةً 4 منازل) بشـــكل عشوائي. ومحاولة اللاعب تخمين العدد المضمر بشـــكل صحيح فيقوم بإدخال رقم مكون من نفس عدد المنازل الموجود في الرقم المضمر، يتم تحقيق هذا الهدف مـــن خلال تقديم مجموعة من التخمينات والحصول على تلميحات من عدد النجوم والأقمار بناءً على مدى صحة التخمين. ويشـــارك باللعبة لاعبين، يظهر لهم في البداية قائمة تتضمن تعليمات اللعبـــة وقواعدها، وبدء اللعبة والخروج، تبدأ اللعبة باختيار اللاعب على ابدأ اللعبة، حيث يتم الترحيب باللاعبين والطلب منهم التعريف بأنفســـهم عبر إدخال أسمائهم، يتم ضمر عدد عشوائي من أربع منازل من قبل البرمجية ومن ثم يقوم اللاعب الأول بتوقع رقم ثم الحصول على تلميح ثم توقع رقم آخر بناءً على التلميح وهكذا حتى يستطيع معرفة الرقم أو ينتهي العدد الأقصى من المحاولات المسموحة.

وفي ما يأتي توضيح لطريقة الحصول على النجوم والأقمار:

- **عدد النجوم**: إذا أدخل اللاعب رقماً في إحدى منازل الرقم المضمر وكان التخمين صحيحاً، يحصل اللاعب على نجمة؛ أي أن عدد النجوم يمثل عدد المنازل التي استطاع اللاعب تخمينها بشكل صحيح.
- **عدد الأقمار**: إذا أدخل اللاعب رقماً في إحدى منازل الرقم المضمر وكان هذا الرقم موجوداً في الرقم المضمر، ولكن ليس في مكانه الصحيح، يحصل اللاعب على قمر. بمعنى آخر، عدد الأقمار يمثل الأرقام التي تشكل جزءاً من العدد المضمر لكنها ليست في المكان الصحيح.
- **عدم الحصول على شيء**: إذا كان الرقم المدخل غير موجود في العدد المضمر، فلا يحصل اللاعب على أي شيء.
- يسمح للاعب بعدد محدد من التخمينات (10 محاولات) لتخمين العدد المضمر بشكل صحيح.

أُقيِّمُ تعلُّمي

المعرفةُ: أُوظِّف في هذا الدرس ما تعلَّمْتُه من معارف في الإجابة عن السؤالين الآتيين:

السؤال الأوَّل: أُوضِّح المقصود بكلٍّ ممّا يأتي:

1. لغة البرمجة.

2. لغة الآلة.

3. البرنامج.

السؤال الثاني: أُقارِن بين لغات البرمجة عالية المســـتوى ولغات البرمجة مُنخفِضة المستوى من حيث سهولة القراءة، والتطوير، والأداء، والكفاءة.

المهاراتُ: أُوظِّفُ مهاراتِ التفكيرِ الناقدِ والتواصلِ الرقميِّ والبحثِ الرقميِّ في الإجابةِ عنِ الأسئلةِ الآتيةِ:

السؤال الأوَّل: أُميِّز بين لغات البرمجة الكتلية ولغات البرمجة النصية من حيث آليَّة تمثيل الأوامر، ثمَّ أذكر مثالًا على كلٍّ منهما.

السؤال الثاني: لماذا يُعَدُّ المُترجِم أسرع تنفيذًا من المُفسِّر؟

السؤال الثالث: ما التحدِّيات التي قد يُواجِهها المُبرمِج عند استخدام لغات البرمجة النصية بدلًا من لغات البرمجة الكتلية؟

السؤال الرابع: بناءً على دراستي موضوع (المُفسِّـــر والمُترجِم)، أيُّهما أفضل لتطوير برامج كبيرة ومُعقَّدة؟ أُبرِّر إجابتي.

القِيَم والاتجاهات:

أُخطِّط مع زملائي/ زميلاتي لتشجيع الأطفال على تعلم البرمجة، وأُحدِّد نوع لغة البرمجة (الكتلية أو النصية) التي سأختارها لتعليمهم بحيث تُناسِب أعمار الفئة المستهدفة ، ثمَّ أعد مخططًا للتنفيذ بالاستفادة من العطلة الصيفية ومرافق المدرسة بالتنسيق مع مُعلِّمي/ مُعلِّمتي وأولياء الأمور.

الدرسُ الثاني

أساسيات لغة البرمجة بايثون
(Basics of Python Programming)

مُنتَجاتُ التعلُّم
(Learning Products)

فتح برنامج خاص بلعبة التخمين على برمجية بايثـــون، والبدء بطباعة رسالة الترحيب وخيارات القائمة الرئيسية ، ضمن سياق تصميم لعبة تخمين الأرقام باستخدام برمجية بايثون.

الفكرة الرئيسة:

تعرُّف لغـــة البرمجة بايثـــون (Python)، وتعلُّم كيف يُمكِن تحميل البرنامج، والتعامل مع الشاشة الرئيسة، وإنشاء برنامج بسيط وتنفيذه وحفظه واسترجاعه، إضافةً إلى تعرُّف الأنواع المختلفـــة من البيانات في لغة البرمجـــة بايثون (Python)، واســـتخدام التعابير الحسابية والتعابير المنطقية وتمثيلها في لغـــة البرمجة بايثون (Python)، وفهـــم قواعد كتابة الجمل البرمجية عن طريق التطبيقات العملية.

المفاهيم والمصطلحات:

البرمجـــة بالكائنـــات (Object–Oriented Programming)، تعدُّد المهام (Multitasking)، مُوجِّـــه الأوامر (Command Prompt)، مُحرِّرات النصوص (Text Editors)، تمييز الصيغة (Syntax Highlighting)، التعليقات (Comments)، الكلمات المحجـــوزة (Reserved Words)، المُعرِّفات (Identifiers)، الرموز (Literals)، المســـافات الفارغـــة (Indentations)، تمييز حالة الحـــرف (Case Sensitivity)، أولويات العوامل (Operators Precedence)، الترابط (Associativity).

نتاجات التعلُّم (Learning Objectives):

- أُعرِّف النموذج الأوَّلي للبرنامج.
- أُبيِّن قواعد كتابة الجملة البرمجية بلغة البرمجة بايثون (Python).
- أُوضِّح عناصر لغة البرمجة بايثون (Python)؛ من: ثوابت، ومُتغيِّرات، ورموز، وتعابير، وعلاقات.
- أُجري عمليات حسابية على التعابير الحسابية.
- أكتب كلًّا من العلاقات والعبارات الحسابية والمنطقية باستخدام لغة البرمجة بايثون (Python).

تعرَّفْتُ في الدرس السابق لغات البرمجة النصية (Text-Based Programming Languages) التي تُستخدَم فيها النصوص لتمثيل أجزاء البرنامج بدلًا من الكتل. وتُعَدُّ لغة البرمجة بايثون (Python) أحد أشهر الأمثلة على هذه اللغات؛ فما لغة البرمجة بايثون؟ وما مزاياها؟ وكيف يُمكِن التعامل معها واستخدامها؟

نشاط تمهيدي

أبحث - بالتعاون مع أفراد مجموعتي- في المواقع الإلكترونية الموثوقة في شبكة الإنترنت عن نشأة لغة البرمجة بايثون (Python)، واستخداماتها، وأهم مزاياها، ثمَّ أُناقِش أفراد المجموعات الأُخرى ومُعلِّمي/ مُعلِّمتي فيما أتوصَّل إليه من نتائج.

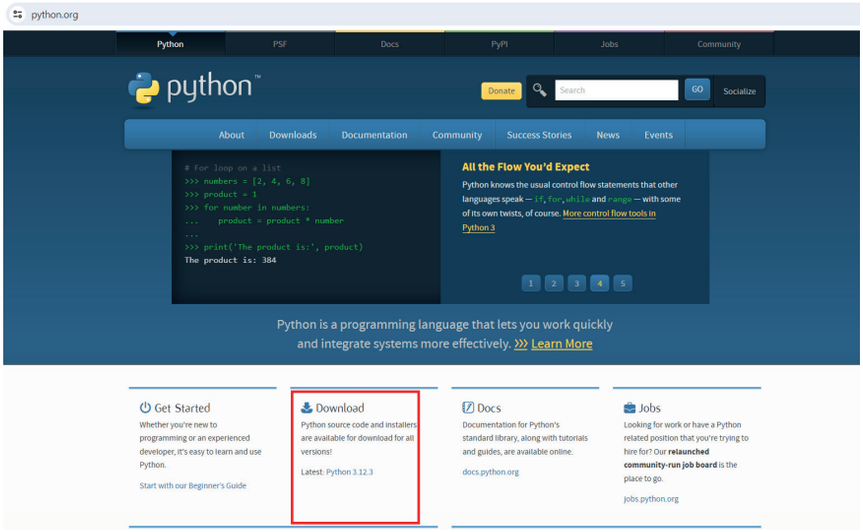
تُعرَّف بايثون (Python) بأنَّها لغة برمجة عالية المستوى، تُستخدَم في أنظمة التشغيل المختلفة، بما في ذلك نظام التشغيل ويندوز (Windows)، ونظام التشغيل (MacOS)، ونظام التشغيل (Linux). وهي تمتاز بأنَّها لغة مفتوحة المصدر؛ ما يعني إمكانية تحميل الرمز (الكود) المصدري الخاص بها، وتعديله، واستخدامه بحرية.

تثبيت لغة البرمجة بايثون (Python Setup)

تأتي معظم أنظمة التشغيل في بيئة بايثون مثبتة مسبقًا، باستثناء نظام التشغيل ويندوز (Windows)، الذي يتطلب تثبيت بايثون يدويًا. وإذا كنا نرغب في استخدام محرر نصوص، فسنحتاج إلى تثبيته أيضًا.

يُمكِن تثبيت لغة البرمجة بايثون (Python) في نظام التشغيل ويندوز (Windows) باتِّباع الخطوات الآتية:

1. **تحميل مُفسِّر لغة البرمجة بايثون (Python):**
   أ. زيارة الموقع الإلكتروني للغة البرمجة بايثون (Python): https://www.python.org/
   ب. الضغط على زِرِّ تحميل آخر إصدار متوافر للغة البرمجة بايثون (Python) كما في الشكل (2-1).



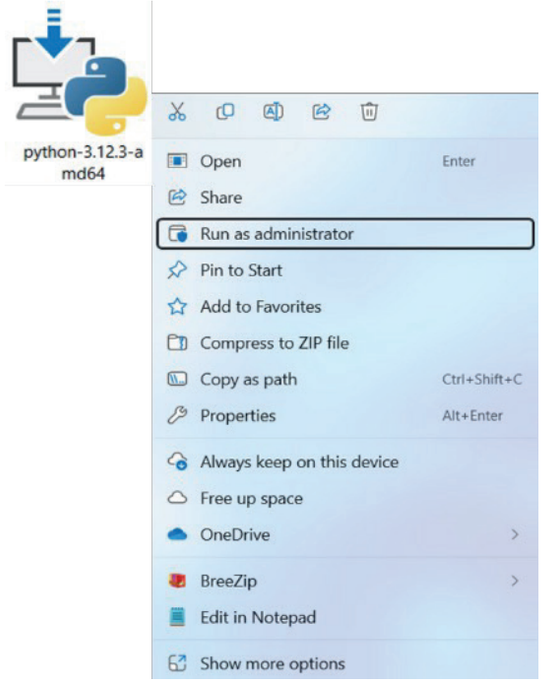
الشكل (2-1): تحميل مُفسِّر لغة البرمجة بايثون (Python).

ملحوظة: يجب التأكُّد أنَّ رقم الإصدار الذي يراد تحميله مُوائِم لنظام التشغيل المُستخدَم.

ج. الضغط على زِرِّ حفظ الملف (Save File) لكي تبدأ عملية التحميل.

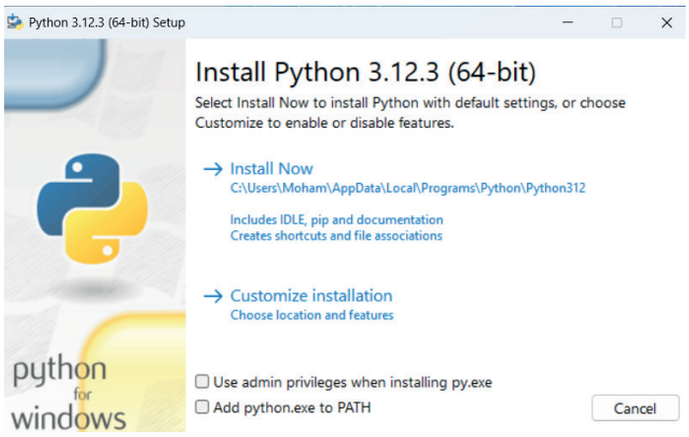
2. **تثبيت مُفسِّر لغة البرمجة بايثون (Python):**

أ. الضغط بزِرِّ الفأرة الأيمن على الملف بعد اكتمال عملية التحميل، ثمَّ الضغط على خيار (Run as Administrator) كما في الشكل (2-2).



الشكل (2-2): تثبيت مُفسِّر لغة البرمجة بايثون (Python).

ب. تفعيل خيار (Use admin privileges when installing py.exe) وخيار (Add path to python.exe) كما في الشكل (2-3).



الشكل (2-3): تفعيل الخيارات، وبَدْء تثبيت مُفسِّر لغة البرمجة بايثون (Python).

ج. الضغط على زِرِّ التثبيت الآن (Install Now) لكي تبدأ عملية التثبيت.

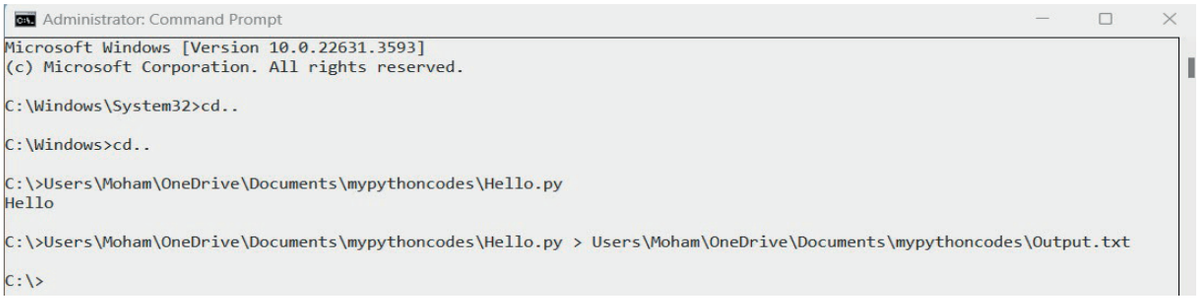
د. الضغط على زِرِّ الإغـــلاق (Close) بعد اكتمال عملية التثبيت بنجاح (Setup Was Successful).

إضاءةٌ

إذا لم يظهر رقم إصدار لغة البرمجة بايثون (Python) عنـــد التحقُّق مـــن تثبيت مُفسِّـــر لغة البرمجة بايثون (Python)، فـــإنَّ أحد أكثر الأسباب شيوعًا لذلك هو عدم تفعيل خيـــار (Add python.exe to PATH) أثناء عملية التثبيت كما في الشكل (2-3).

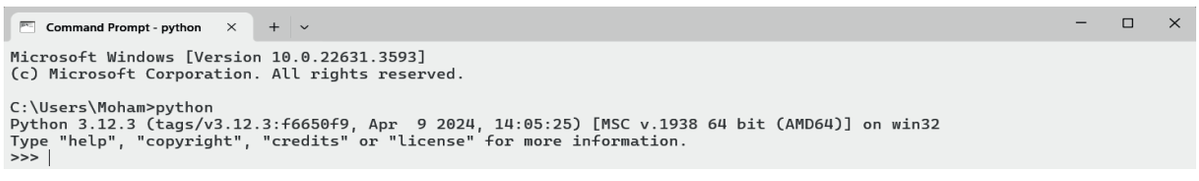
3. **التحقُّق من تثبيت مُفسِّر لغة البرمجة بايثون (Python):**

أ. فتح مُوجِّه الأوامر (Command Prompt)، بالذهاب إلى القائمة الرئيسة، وكتابة كلمتي مُوجِّـــه الأوامر (Command Prompt) في مُربَّع البحـــث، ثم الضغط على Command Prompt. عند إتباع الخطوات السابقة ستظهر الشاشة الموضحة في الشكل (2-4).



الشكل (2-4): شاشة تشغيل مُوجِّه الأوامر (Command Prompt).

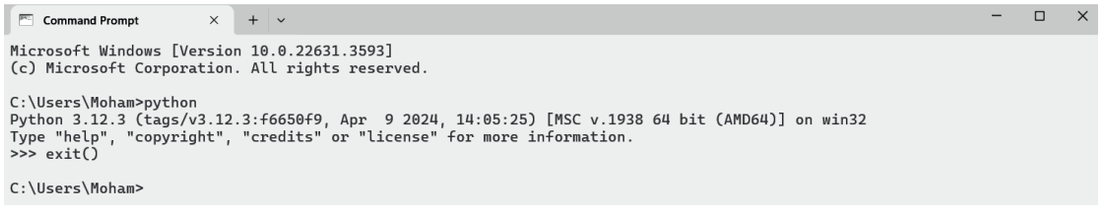
ب. كتابـــة كلمة بايثون (python)، ثمَّ الضغط على زِرِّ الإدخال (Enter)؛ للتحقُّق من تثبيت مُفسِّـــر لغة البرمجة بايثون (Python)، وتعرُّف رقم الإصدار الخاص به كما في الشكل (2-5).



الشكل (2-5): التحقُّق من تثبيت مُفسِّر لغة البرمجة بايثون (Python)، وتعرُّف رقم إصداره.

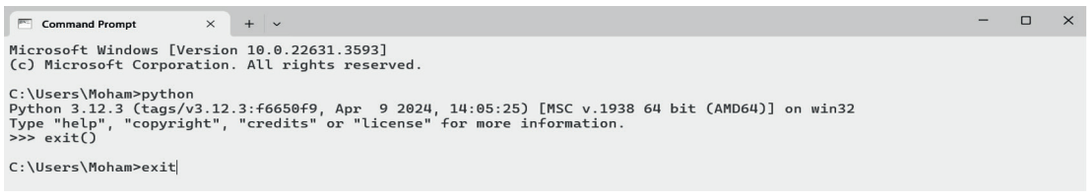
ج. ظهور رقم الإصدار المُثبَّت، وهو 3.12.3، ثـــمَّ بَدْء كتابة الأوامر بلغة البرمجة بايثون (Python)، وتثبيتها مباشرة بعد الرمز <<< .

د. الخروج من لغة البرمجة بايثون (Python) بكتابة كلمة الخروج ()exit، ثمَّ الضغط على زِرِّ الإدخال (Enter) كما في الشكل (2-6).



الشكل (2-6): الخروج من لغة البرمجة بايثون (Python).

هـ. الخروج من مُوجِّه الأوامر (Command Prompt) بكتابة كلمة الخروج (exit)، ثمَّ الضغط على زِرِّ الإدخال (Enter) كما في الشكل (2-7).



الشكل (2-7): الخروج من مُوجِّه الأوامر (Command Prompt).

نشاط عملي

أطبق خطوات تثبيت لغة البرمجة بايثون وأتأكد من تنفيذ الخطوات بشـــكل سليم. ثم أشغل البرمجية وأستكشف الشاشة الرئيسة

مُحرِّرات النصوص وبيئات التطوير المُتكامِلة (Text Editors and Integrated Development Environment)

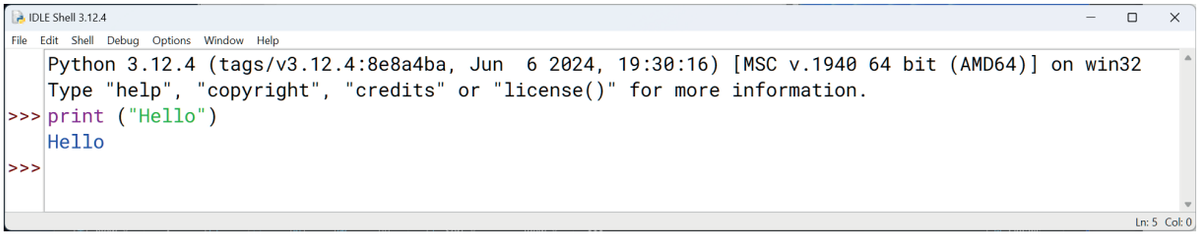
إنَّ استخدام مُحرِّرات النصوص وبيئات التطوير المُتكامِلة (IDEs) في لغة البرمجة بايثون (Python) يعتمد أساسًا على حجم المشـــروع؛ إذ تُخصَّص المُحرِّرات لكتابة برامج بسيطة، في حين تُختار بيئات التطوير المُتكامِلة للمشروعات الكبيرة.

تُعَدُّ بيئة التطوير والتعلُّم المُتكامِلة (IDLE) – التي تُدمَج افتراضيًّا في لغة البرمجة بايثون (Python)– واحدة من أكثر بيئات التطوير شيوعًا، وتمتاز بتوافقها مع نظام التشغيل ويندوز (Windows)، ونظام التشغيل (MacOS)، ونظام التشـــغيل (Unix). توفر (IDLE) نافذة (Shell) لتنفيذ الأوامر وعرض المخرجات، كما توفر مُحرّر نصوص يتيح ميزة تمييز الصيغ (Syntax highlighting) التي تحسن من مقروئية البرنامج، و ميزة إكمـــال الرموز تلقائيًا (Code completion)، بالإضافة إلى مُصحّح أخطاء مُدمَج.

كتابة برنامج بلغة البرمجة بايثون (Python) وحفظه:

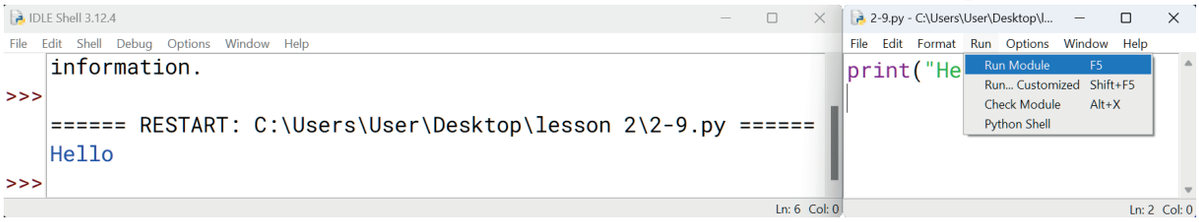
يُمكِن كتابة برنامج ما بلغة البرمجة بايثون (Python) على النحو الآتي:

1. فتح بيئة التطوير والتعلُّم المُتكامِلة للغة البرمجة بايثون (Python)، ثمَّ تشغيلها، فتظهر الشاشة الرئيسة.
2. كتابة أوامر البرنامج بلغة البرمجة بايثون (Python) وتنفيذها. أنظر الشكل (2-8) الذي يُبيِّن الأوامر البرمجية لبرنامج يعمل على طباعة كلمة (Hello).



الشكل (2-8): تنفيذ برنامج لطباعة كلمة (Hello).

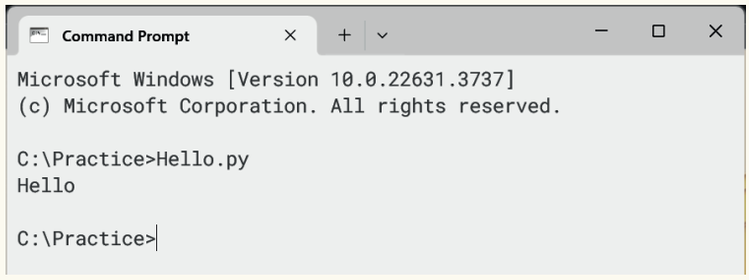
3. كما يمكن انشـــاء صفحة جديدة (New) من قائمة ملف، وبعـــد الانتهاء من كتابة الكود يتم حفظ البرنامج بالضغط على زِرِّ الحفظ باســـم (Save as) من قائمة الملف (File)، ثمَّ تنفيذ البرنامج باختيار خيار تشـــغيل النمط (Run Module) من قائمة التشغيل (Run)، فيظهر ناتج تنفيذ البرنامج في نافذة بيئة التطوير والتعلُّم المُتكامِلة (IDLE Shell) كما في الشكل (2-9).



الشكل (2-9): كتابة برنامج بلغة البرمجة بايثون (Python) وتنفيذه.

إثراءٌ

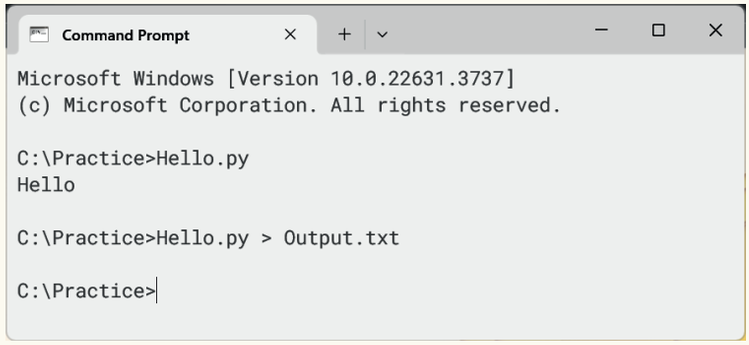
يُمكِن تنفيذ البرنامـــج المكتوب بلغة البرمجة بايثون (Python) باســـتخدام مُوجِّه الأوامر (Command Prompt)، ثمَّ فتح الملف الذي يحوي هذا البرنامج، ثمَّ الضغط على زِرِّ الإدخال (Enter) كما في الشكل (2-10).



الشكل (2-10): تنفيذ برنامج لغة البرمجة بايثون (Python) باستخدام مُوجِّه الأوامر.

يُمكِن حفظ مخرجات تنفيذ البرنامج المطلوب في ملف آخر باتِّباع الخطوات الآتية:

1. فتح الملف الذي يحتوي على برنامج لغة البرمجة بايثون (Python).
2. طباعة الرمز > بعد اسم الملف.
3. تحديد مكان حفظ الملف الذي يحتوي على مخرجات تنفيذ البرنامج.
4. الضغط على زِرِّ الإدخال (Enter)، أنظر الشكل (2-11).



الشكل (2-11): حفظ مخرجات برنامج لغة البرمجة بايثون (Python) باستخدام مُوجِّه الأوامر.

نشاط عملي

أكتب برنامجًا بلغة البرمجة بايثون (Python) لطباعة اسمي على شاشة جهاز الحاسوب، ثمَّ أحفظه. بعد ذلك أُنفِّذ البرنامج للتحقُّق من المخرجات.

جملة الإدخال input()

يُمكِن استعمال لغة البرمجة بايثون (Python) لإنشاء برنامج يتفاعل مع المُستخدِم، وذلك بالطلب إلى المُستخدِم إدخال البيانات المطلوبة بعد تشغيل البرنامج، فيعمل البرنامج على معالجتها. ولكي يتمكَّن المُستخدِم من إدخال هذه البيانات في البرنامج أثناء عمله؛ لا بُدَّ له من استعمال الدالَّة input().

وما إنْ يتمُّ اســـتدعاء هذه الدالَّة، حتّى يظلَّ مُفسِّر بايثون (Python) في وضع الاستعداد، وينتظر من المُستخدِم أنْ يُدخِل البيانات عن طريق لوحة المفاتيح، ويضغط على زِرِّ الإدخال (Enter)، فيعمل مُفسِّـــر بايثون (Python) حينئذٍ على إرجاع ما أُدخِل في صورة نص إلى المكان الذي اســـتُدعِيت منه الدالَّة ()input. وهذا يعني أنَّ الدالَّة ()input تقرأ مدخلات المُستخدِم بوصفها نصًّا، ثمَّ تعيدها بوصفها نصًّا أيضًا، حتّى لو بادر المُستخدِم إلى إدخال عدد ما.

ومن ثَمَّ إذا كان هدف المُســـتخدِم إدخال عدد ما في البرنامج، فإنَّ البرنامج يعمل على تحويل ما تُرجِعه الدالَّة ()input إلى عدد.

نشاط

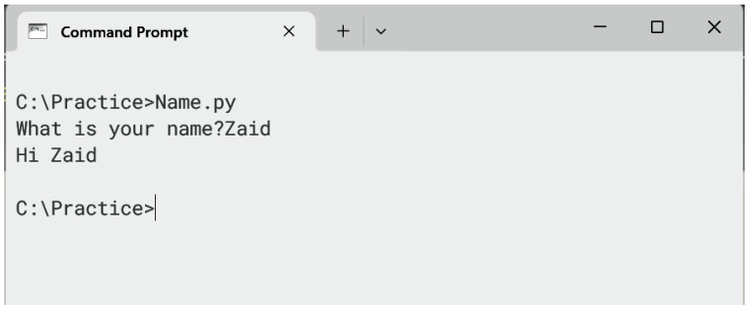
أُجرِّب وأستكشف: أستكشف الشاشة الرئيسة للغة البرمجة بايثون (Python)، وأُجرِّب استدعاء الدالَّة ()input، وإدخال قِيَم نصية وعددية، ثمَّ أُدوِّن النتائج والتحدِّيات التي واجهْتُها، وأُقارِنها بالنتائج والتحدِّيات التي واجهها زملائي/ واجهتها زميلاتي.

مثال:

البرنامج الآتي يقوم بإظهار جملة تطلب من المستخدم أن يدخل اسمه ("What is your name?") ثم يخزنه في متغير اسمه name ، ثم يعرض جملة ترحيب باسم المستخدم المخزن في المتغير name. ستظهر أوامر البرنامج كالآتي:

```
name = input ("What is your name?")
print("Hi", name)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية (بافتراض أنَّ الاسم المُدخَل هو Zaid) على شاشة جهاز الحاسوب:



نشاط عملي

أُجرِّب إنشـــاء برنامج يطلب إلى المُستخدِم إدخال اســـم مدينته المُفضَّلة، ثمَّ يعرض له رسالة تحوي اســـم هذه المدينة. أُشارِك زملائي/ زميلاتي في نتائج برنامجي، وأتعاون معهم/ معهن على إيجاد حلول للمشكلات التي تظهر أثناء تنفيذ المطلوب.

طريقة إدخال عدد في لغة البرمجة بايثون (Python)

يُمكِن للمُســـتخدِم إدخال عدد ما بأنْ يضع الدالَّة input() داخل int()، ثمَّ ينتظر حتّى يتحوَّل العدد المُدخَل إلى عدد صحيح.

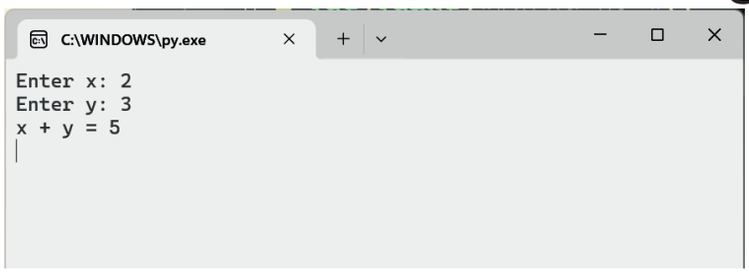
مثال:

لكتابة برنامج يطلب إلى المُستخدِم أنْ يُدخِل قيمة عددية أوَّلية، ثمَّ يُخزِّنها في المُتغيِّر x بعد تحويلها إلى عدد صحيح باستخدام الدالّة int(). بعد ذلك طلب إلى المُستخدِم أنْ يُدخِل قيمة عددية أُخرى، ثمَّ يُخزِّنها في المُتغيِّر y بعد تحويلها إلى عدد صحيح باستخدام الدالَّة int()، ثمَّ يعرض نتيجة جمع القيمتين باستخدام الدالَّة print().

يجب كتابة الأوامر البرمجية الآتية:

```
x = int(input("Enter x: "))
y = int(input("Enter y: "))
print("x + y =", x + y)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

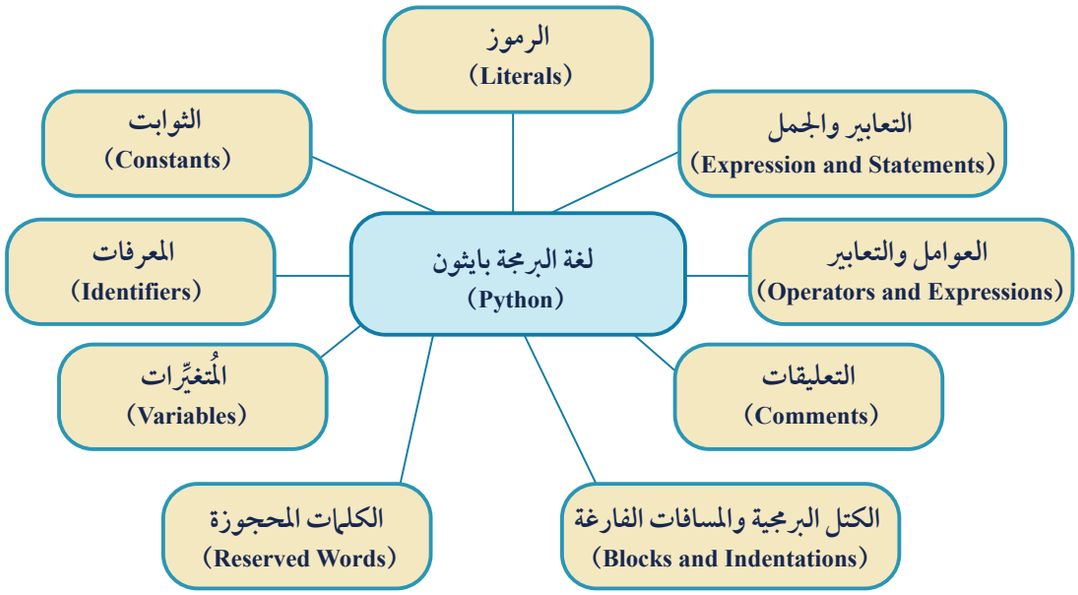


نشاط عملي

أُجرِّب إنشاء برنامج يطلب إلى المُستخدِم أنْ يُدخِل اسمه ومكان ولادته وعمره، ثمَّ يعرض له ذلك كله على شاشة جهاز الحاسوب.

عناصر لغة البرمجة بايثون (Python)

يحتوي البرنامج المكتوب بلغة البرمجة بايثون (Python) على العناصر الأساسية الآتية التي يُبيِّنها الشكل (2-12):



الشكل (2-12): عناصر لغة البرمجة بايثون (Python).

1. **التعليقات (Comments)**: لا تُؤثِّر التعليقات في تنفيذ البرنامج، ولا يُشترَط لذلك وجود عدد مُحدَّد منها؛ فكتابتها في البرنامج أمر اختياري، ولكنْ يُنصَح بإضافتها لأسباب عِدَّة، أبرزها:
   أ. **توثيق البرنامج**: تُسهِم إضافة التعليقات في توثيق آليَّة تطوير البرنامج، وتحديد الهدف منه، ما يساعد على مراجعته أو التعديل عليه بعد مُضِيِّ وقت طويل.
   ب. **تحسين مقروئية البرنامج**: تؤدّي إضافة التعليقات إلى قراءة البرنامج بصورة أفضل؛ ما يُسهِّل على الآخرين عملية فهمه وتعديله وتطويره.

   أمّا الطريقة التي يُمكِن بها كتابة تعليق في البرنامج فتتمثَّل في كتابة الرمز #، ثمَّ كتابة التعليق بعده.
2. **المُعرِّفات (Identifiers)**: أسماء تُستعمَل للدلالة على المُتغيِّرات والدوالِّ والكائنات وغير ذلك من العناصر. تحتوي لغة البرمجة بايثون (Python) على قواعد إلزامية يجب الأخذ بها عند اختيار الأسماء، وهي:
   أ. احتواء المُعرِّف فقط على أحد الحروف (a – z)، أو أحد الحروف (A – Z)، أو الأعداد (0–9)، أو الشرطة السفلية (underscore) (_).
   ب. وجوب أنْ يبدأ المُعرِّف بأحد الحروف الكبيرة (A – Z)، أو أحد الحروف الصغيرة (a – z)، أو الشرطة السفلية (underscore) (_).
   ج. عدم بَدْء المُعرِّف بعدد.
   د. منع استخدام أي كلمة من الكلمات المحجوزة.

مثال:

من أسماء المُعرِّفات المقبولة في لغة البرمجة بايثون: Grade_، number، name. ومن الأسماء غير المقبولة في لغة البرمجة بايثون: @user2، name-، nd. ،

3. **الكلمات المحجـــوزة (Reserved words)**: توجد كلمات محجـــوزة للغة البرمجة بايثون (Python)، لا يُمكِن استخدامها مُعرِّفاتٍ. وهذه الكلمات هي:

and	elif	from	None	return
assert	else	global	nonlocal	True
break	except	if	not	Try
class	exec	import	or	while
continue	False	in	pass	with
def	finally	is	print	yield
del	for	lambda	raise	

4. **الثوابت (Constants)**: قِيَم تظـــلُّ ثابتة، ولا تتعرَّض للتغيير أثناء تنفيذ البرنامـــج. وهي تُصنَّف إلى نوعين اثنين، هما:

أ. **الثوابت الرمزية النصية (Character Constants)**: سلســـلة من الحروف التي تُســـتخدَم في لغة البرمجة، وتُكتَب بين علامتـــي اقتباس، مثل: "Hello"، و "Jordan".

ب. **الثوابـــت العدديـــة (Numerical Constants)**: سلســـلة من الأعداد، تبدأ بالعدد (0)، وتنتهي بالعدد (9)، ويُمكِن أنْ تحتوي على إشـــارة (+) فـــي مُقدِّمتها للدلالة على أنَّ العدد موجب، وقد تحتوي على إشارة (-) للدلالة على أنَّ العدد سالب. وسيقتصر الحديث في هذه الوحدة على الثوابت العددية الحقيقية (real numbers)، مثل: الأعداد الصحيحة، والأعداد العشرية.

إضاءةٌ

يختلف الثابـــت الرمزي عن الثابت العددي في أنَّه لا يُســـتخدَم في العمليات الحســـابية، وإنَّما يُستخدَم في تمثيـــل المعطيات غير العددية.

5. **المُتغيِّرات (Variables)**: رموز تدلُّ على القِيَم المُستخدَمة في البرنامج؛ إذ يُخصَّص للمُتغيِّر مســـاحة تخزينية في ذاكرة البرنامج، وتوضَع القيمة المُرتبِطة بالمُتغيِّر في هذه المســـاحة التخزينية، ويُستخدَم اسم المُتغيِّر في الإشارة إلى تلك القيمة.

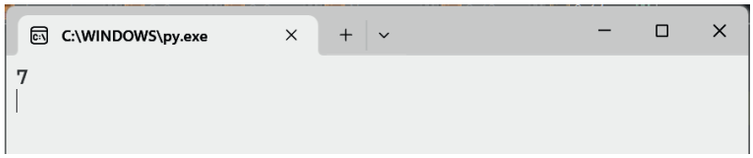
مثال:

يراد تعريف المُتغيِّر المُسمّى (days) في لغة البرمجة بايثون (Python)، وإسناد القيمة (7) إليه، وطباعة قيمته.

يجب كتابة الأوامر البرمجية الآتية:

```
days = 7
print(days)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



مثال:

يُعرِّف البرنامج الآتي مُتغيِّرين، قيمة كلٍّ منهما (99)، ثمَّ يطبع هذه القيمة لكليهما:

```
x = y = 99
print('x =', x)
print('y =', y)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

إضاءةٌ

يُلاحَظ في لغة البرمجة بايثون (Python) أنَّ المُبرمِج ليس مسؤولًا عن تحديد أنواع المُتغيِّرات التي يُعرِّفها في برنامجه؛ فما إنْ يعمل المُبرمِج على تعريف مُتغيِّر ما، ويضع فيه أيَّ قيمة، حتّى يُسارِع مُفسِّر لغة البرمجة بايثون (Python) إلى تحديد نوع هذا المُتغيِّر بناءً على القيمة التي أسندها إليه المُبرمِج بصورة تلقائية وقت التشغيل. ولهذا يجب في لغة البرمجة بايثون (Python) إسناد قيمة إلى المُتغيِّر أثناء تعريفه.

إضاءةٌ

في لغة البرمجة بايثون (Python)، يُمكِن تعريف عدد من المُتغيِّرات ذات القِيَم المتساوية في وقت واحد.

نشاط عملي

أكتب برنامجًا بلغة البرمجة بايثون (Python)؛ لتعريف مُتغيِّرات يُماثِل عددها عدد أيام الأسبوع، واسناد اسم يوم من أيام الأسبوع إلى كلٍّ منها، ثمَّ طباعتها.

أنواع المُتغيِّرات في لغة البرمجة بايثون (Python):

توجد أنواع كثيرة من المُتغيِّرات في لغة البرمجة بايثون (Python)، ويُمكِن إجمال الأنواع الأساسية لهذه المُتغيِّرات في ما يأتي:
المُتغيِّرات العددية (Numbers)، والمُتغيِّرات النصية (Strings)، والمُتغيِّرات المنطقية (Booleans)، والمصفوفات ذات الحجم غير الثابت التي تُسمّى القوائم (Lists)، والمصفوفات ذات الحجم الثابت والقِيَم الثابتة التي لا تقبل التغيير، والتي تُسمّى الصفوف (Tuples)، والمصفوفات ذات الحجم غير الثابت التي لا تحتوي على قِيَم مُكرَّرة (Sets)، والجداول التي تُخزَّن فيها البيانات بصورة مفاتيح (Keys) وقِيَم (Values)، وتُسمّى القواميس (Dictionaries).

إضاءةٌ

المُتغيِّر في لغة البرمجة بايثون (Python) هو من النوع غير الثابت؛ لأنَّه يتغيَّر تلقائيًّا بحسب نوع القيمة التي تُخزَّن فيه.

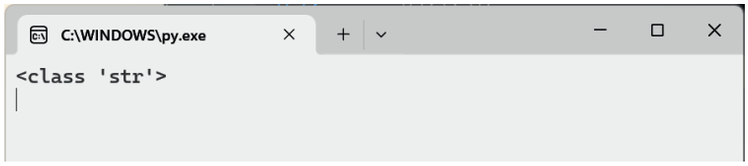
يُمكِنني معرفة نوع أيِّ مُتغيِّر عرَّفْتُه، وأسنَدْتُ إليه قيمة ما، باستخدام الدالَّة type():

type( )

مثال:

```
var = 'Jordan'
print(type(var))
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط عملي

أستخدِم برمجية بايثون (Python) في تحديد نوع كل مُتغيِّر ممّا يأتي:

```
Age = 16
name = "Basem"
is_student = True
height = 1.75
```

في ما يأتي بيان لأنواع المُتغيِّرات المختلفة في لغة البرمجة بايثون (Python):

أ. **المُتغيِّرات العددية (Numbers):** عند تعريف مُتغيِّر عددي وتخزين عدد فيه، فإنَّ مُفسِّر لغة البرمجة بايثون (Python) يعمل تلقائيًّا على تحديد نوع هذا المُتغيِّر بناءً على نوع القيمة الرقمية التي أُسنِدت إليه. وهذه بعض أنواع المُتغيِّرات التي سنستخدمها في هذه الوحدة:

- **Int:** يُستخدَم هذا النوع من المُتغيِّرات في تخزين أعداد صحيحة.
- **Float:** يُستخدَم هذا النوع من المُتغيِّرات في تخزين أعداد تحوي فواصل عشرية.

مثال:

البرنامج الآتي مسؤول عن تعريف المُتغيِّر المُسمّى (x)، وقيمته العدد الصحيح (10)، وتعريف المُتغيِّر المُسمّى (y)، وقيمته العدد العشري (2.5)، ثمَّ طباعة نوع قيم المتغيرات:

```
x = 10
y = 2.5
```

طباعة نوع قيمة المُتغيِّر:

```
print(type(x))
print(type(y))
```

إضاءة

المُتغيِّر الذي يُسنَد إليه ثابت عددي كسري هو إمّا من نوع (int)، وإمّا من نوع (float). والذي يُحدِّد هذين النوعين، ويُميِّز بينهما، هو استخدام الفاصلة العشرية أو عدم استخدامها.

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

<class 'int'>: يشير ذلك إلى أنَّ المُتغيِّر (x) هو من نوع العدد الصحيح (integer).

<class 'float'>: يشير ذلك إلى أنَّ المُتغيِّر (y) هو من نوع العدد العشري (floating point) .

ب. **المُتغيِّرات النصية (Strings):** يتطلَّب تعريف نص ما في لغة البرمجة بايثون (Python) استخدام علامات الاقتباس الفردية (')، أو علامات الاقتباس المزدوجة ("")، أو علامات الاقتباس الثلاثية (""")، علمًا بأنَّه لا فرق بين الرمز (') والرمز ("")؛ إذ يُمكِن استخدام أيٍّ منهما في تعريف نص يتألَّف من سطر واحد. كذلك يُمكِن استخدام الرمز ('') والرمز (""") في تعريف نص كبير يتألَّف من عِدَّة أسطر.

مثال:

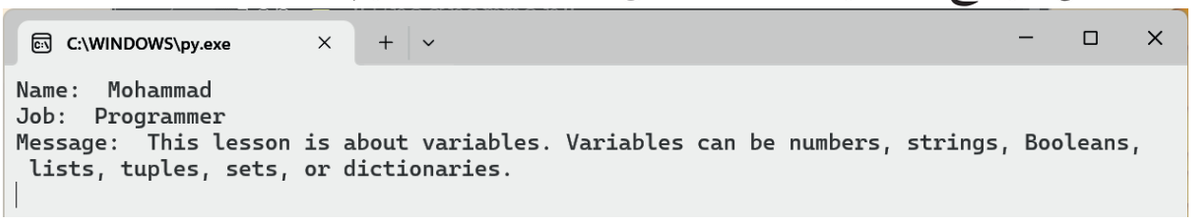
يُمكِن تعريف ثلاثة مُتغيِّرات تحوي قِيَمًا نصيةً بكتابة الأوامر البرمجية الآتية:

```
name = 'Mohammad'
job = "Programmer"
message = '''This lesson is about variables. Variables can be numbers,
strings, Booleans, lists, tuples, sets, or dictionaries.'''
```

لطباعة قِيَم المُتغيِّرات النصية، يجب كتابة الأوامر البرمجية الآتية:

```
print('Name: ', name)
print('Job: ', job)
print('Message: ', message)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط عملي

أُجرِّب بنفسي: أدرس البرنامج التالي، ثمَّ أتوقَّع نتيجة تنفيذه، ثمَّ أُدوِّنها في دفتري. هل أتوقَّع اختلاف الناتج في هذا البرنامج عن ناتج البرنامج السابق؟ أُنفِّذ البرنامج، ثمَّ أُقارِن بين البرنامجين من حيث الناتج.

```
message = '''This lesson is about variables. Variables can be 'numbers',
"strings", 'Booleans', lists, tuples, sets, or dictionaries.'''
print('Message:', message)
```

ج. **المُتغيِّرات المنطقية (Booleans):** متغيرات تستخدم لتخزين قيم منطقية قيمها إما صواب وإما خطأ. فعند تعريف أحد المُتغيِّرات، وإسناد قيمة صحيح (True) أو قيمة خطأ (False) إليه، فإنَّ مُفسِّر لغة البرمجة بايثون (Python) سيَعُدُّه مُتغيِّرًا منطقيًّا.

مثال:

يُمكِن تعريف مُتغيِّر اسمه (passed) وقيمته (True) بكتابة الأمر الآتي:

```
passed = True
```

يُمكِن تنفيذ أمر الطباعة إذا كانت قيمة المُتغيِّر (passed) تساوي (True) بكتابة الأمر الآتي:

```
if passed == True:
    print("passed=True")
```

يُمكِن تنفيذ أمر الطباعة إذا كانت قيمة المُتغيِّر (passed) تساوي (False) بكتابة الأمر الآتي:

```
else:
    print("passed=False")
```

نشاط

أُجرِّب بنفسي: أُنفِّذ البرنامج التالي باســـتخدام برمجية بايثون (Python)، ثمَّ أُقارِن الناتج بناتج البرنامج السابق. هل يوجد اختلاف في الناتج؟ أفسر إجابتي؟

```
passed = True
if passed == 1:
    print("passed=True")
else:
    print("passed=False")
```

6. **الرموز (Literals):** يُستخدَم في لغة البرمجة بايثون (Python) مجموعة من الرموز، أبرزها:

أ. **النصوص (String Literals):**

مثال:

```
a = '''Python for 11th grade'''
print(a)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

ب. **الحروف (Character Literals):**

مثال:

```
a = 'P'
print(a)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

ج. **الأرقام (Numeric Literals):**

مثال:

```
y = 30
print(y)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

30

د. الرموز المنطقية (Boolean Literals):

مثال:

```
a = (1 == True)
b = (1 == False)
c = True + 1
d = False + 4
print(a, b, c, d)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

7. **التعابير والجمل (Expression and Statements)**: يُعرَّف التعبير بأنَّه سلسلة تتألَّف من واحد أو أكثر من القِيَم والمُتغيِّرات والعوامل واســـتدعاءات الدوالِّ، وينتج منها قيمة مُعيَّنة. أمّا الجملة فهي أصغر جزء من البرنامج يقبل التنفيذ، ويؤدّي إلى حدوث تأثيرات عديدة، ولا يُفْضي إلى نتيجة مُحدَّدة أو قيمة مُعيَّنة بعد الانتهاء من تنفيذه. ومن ثَمَّ، فلا يُمكِن – مثلًا– إسناد جملة إلى مُتغيِّر، أو وضعها مباشرة في دالَّة الطباعة ()print؛ لعدم وجود قيمة عائدة منها بعد تنفيذها.

مثال:

في الأمر البرمجي: z = 2 + 3:

(2+3) هو تعبير، أمّا (z = 2 + 3) فهو جملة.

8. **الكتل البرمجية والمسافات الفارغة (Blocks and Indentations)**:

الكتل البرمجية هي مجموعة من الجمل ذات الصلة. وقد تحتوي الكتلة البرمجية على جملة واحدة فقط. أمّا المسافات الفارغة (Indentations) فتضاف إلى البرنامج في لغة البرمجة بايثون (Python)؛ لتحديد الكتل البرمجية وتوضيحها.

مثال:

```
if 1 > 0:
    print("one is greater than zero.")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

مثال:

```
if 1 > 0:
    print("one is greater than zero.")
if 1 > 0:
     print("one is greater than zero.")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

مثال:

```
if 1 > 0:
print("one is greater than zero.")
```

عند تشغيل البرنامج، ستظهر على شاشة جهاز الحاسوب رسالة تفيد بوجود خطأ في البرنامج.

قواعد إضافية لكتابة الجملة البرمجية:

1. **تمييز حالة الأحرف (Case Sensitivity):** تُميِّز لغة البرمجة بايثون (Python) بين الأحرف الكبيرة والأحرف الصغيرة. فمثلًا، كلمة (days) وكلمة (Days) مختلفتان، وهما لا تعنيان شيئًا واحدًا؛ سواء كان استخدامهما للمُتغيِّرات، أو الدوالِّ، أو غير ذلك.
2. **عند اختيار الأسماء (Names)**، يُنصَح باتِّباع القواعد الآتية:
   أ. **اسم المُتغيِّر (Variable Name):** تُستخدَم الأحرف الصغيرة عند وضع أسماء للمُتغيِّرات. وفي حال اشتمل اسم المُتغيِّر على أكثر من كلمة، فإنَّ الشرطة السفلية (underscore) توضَع بين كل كلمتين كما في المثال الآتي:

average_score = 25

إضاءةٌ

المسافة الفارغة المضافة قبل دالَّة الطباعة ()print ليست أمرًا اختياريًّا؛ إذ لو حُذِفت هذه المسافة من البرنامج، لظهرت رسالة تفيد بوجود خطأ في البرنامج. وهذه المسافة الفارغة غير مُقيَّدة بعدد مُعيَّن من المسافات؛ فقد تكون مسافة (space) واحدة أو أكثر. وبالمثل، لا يُشترَط استخدام نفس عدد المسافات في البرنامج كاملًا، وإنَّما يُشترَط الالتزام بنفس عدد المسافات في الكتلة البرمجية الواحدة. وبالرغم من عدم اشتراط استخدام العدد نفسه من المسافات في جميع مراحل البرنامج، فإنَّ ذلك يُعَدُّ من الممارسات البرمجية الجيِّدة.

ب. **اسم الدالَّة (Function Name)**: تُستخدَم الأحرف الصغيرة عند وضع أسماء للدوالِّ. وفي حال اشتمل اســـم الدالَّة على أكثر من كلمة، فإنَّ الشرطة السفلية (underscore) توضَع بين كل كلمتين.

ج. **كتابة أكثر من جملة على السطر نفسه**: تُكتَب كل جملة على سطر واحد في لغة البرمجة بايثون (Python). ولكنْ، يُمكِن كتابة أكثر من جملة على الســـطر نفسه بوضع فاصلة منقوطة بين كل جملتين؛ إذ سيفهم في هذه الحالة مفسر لغة البرمجة بايثون (Python) أنَّ السطر الواحد يحوي أكثر من جملة كما في المثال الآتي:

```
x=10; y=20
```

د. **كتابة أمر واحد على أكثر من ســـطر**: يُمكِن كتابة أمر واحد على أكثر من سطر بوضع الرمز \ في نهاية كل سطر، فيفهم مفســـر لغة البرمجة بايثون (Python) أنَّ الأمر يشمل أكثر من سطر كما في المثال الآتي:

عُرِّفت ثلاثة مُتغيِّرات على النحو الآتي:

```
sales_1 = 120
sales_2 = 200
sales_3 = 187
```

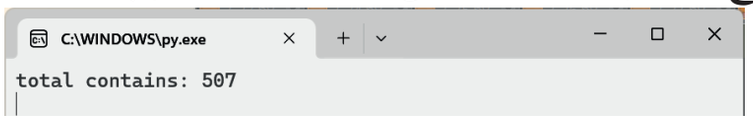
لجمع قِيَم المُتغيِّرات: sales_1، وsales_2، وsales_3، ووضع الناتج في المُتغيِّر (total)، فإنَّ الأمر يكون على النحو الآتي:

```
total = sales_1 + \
sales_2 + \
sales_3
```

لطباعة قيمة المُتغيِّر (total)، يُكتَب الأمر الآتي:

```
print("total contains:", total)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط عملي

المُتغيِّرات الآتية تُمثِّل علامات ثلاثة طُلّاب في اختبار مُعيَّن:

grade_1 = 85

grade_2 = 90

grade_3 = 78

1. أستعمِل كتابة الأمر الواحد على نفس الســـطر لجمع علامات هؤلاء الطُّلّاب، ووضع الناتج في المُتغيِّر (total_grades)، ثمَّ أطبع قيمة (total_grades).
2. أســـتعمِل كتابة الأمر الواحد على أسطر مُتعدِّدة لجمع علامات هؤلاء الطُّلّاب، ووضع الناتج في المُتغيِّر (total_grades)، ثمَّ أطبع قيمة (total_grades).

9. العوامل والتعابير (Operators and Expressions):

تُصنَّف العوامل بحسب استخداماتها إلى سبع مجموعات. وفي ما يأتي بيان لأربع منها:

أ. **العوامل المُستخدَمة في العمليات الحسابية (Arithmetic Operators)**، وهي مُمثَّلة في الجدول (2-1).

الجدول (2-1): العوامل المُستخدَمة في العمليات الحسابية.

اسم العامل	الرمز	مثال توضيحي	الشرح
إضافة (Addition)	+	x+y	إضافة قيمة y إلى قيمة x.
الطرح (Subtraction)	-	x-y	طرح قيمة y من قيمة x.
الضرب (Multiplication)	*	x*y	ضرب قيمة x في قيمة y.
القسمة (Division)	/	x /y	قسمة قيمة x على قيمة y.
باقي القسمة (Modulus)	%	x%y	إرجاع باقي قسمة قيمة x على قيمة y.
القوَّة (Exponentiation)	**	x** y	رفع قيمة x إلى أُسٍّ بقيمة y.
القسمة التحتية (Floor Division)	//	x //y	قسمة قيمة x على قيمة y، وإرجاع أقرب عدد صحيح إلى الناتج (أقل من الناتج، أو يساوي الناتج).

ب. **العوامل المُستخدَمة في المقارنات (Comparison Operators)**، وهي مُمثَّلة في الجدول (2-2).

الجدول (2-2): العوامل المُستخدَمة في المقارنات.

اسم العامل	الرمز	مثال توضيحي	الشرح
يساوي (Equal to)	==	x==y	هل قيمة x تساوي قيمة y؟
لا يساوي (Not equal to)	!=	x!=y	هل قيمة x لا تساوي قيمة y؟
أكبر من (Greater than)	>	x>y	هل قيمة x أكبر من قيمة y؟
أصغر من (Less than)	<	x<y	هل قيمة x أصغر من قيمة y؟

أكبر من أو تساوي (Greater than or equal to)	>=	x>=y	هل قيمة x أكبر من قيمة y أو تساويها؟
أصغر من أو تساوي (Less than or equal to)	<=	x<=y	هل قيمة x أصغر من قيمة y أو تساويها؟

ج. العوامل المُستخدَمة في كتابة الشروط المنطقية (Logical Operators)، وهي مُمثَّلة في الجدول (2-3).

الجدول (2-3): العوامل المُستخدَمة في كتابة الشروط المنطقية.

اسم العامل	الرمز	مثال توضيحي	الشرح
(Logical AND)	and	x and y	– إرجاع (True) فقط إذا كانت قيمتي x و y هي True – إرجاع (False) إذا كانت قيمة x أو قيمة y أو قيمة كلٍّ منهما (False).
(Logical OR)	or	x or y	– إرجاع (False) فقط إذا كانت قيمة x و قيمة y هي False – إرجاع (True) إذا كانت قيمة x أو قيمة y أو قيمة كلٍّ منهما (True).
(Logical NOT)	not	not x	– إرجاع (True) إذا كانت قيمة x هي (False) وإرجاع (False) إذا كانت قيمة x هي (True).

د. **العوامل المُستخدَمة في إعطاء قِيَم للمُتغيِّرات (Assignment Operators)**، وهي مُمثَّلة في الجدول (2-4).

الجدول (2-4): العوامل المُستخدَمة في إعطاء قِيَم للمُتغيِّرات.

اسم العامل	الرمز	مثال توضيحي	الشرح
الإسناد الأساسي (Basic Assignment).	=	x=y	إعطاء x قيمة y.
الإضافة والإسناد (Addition Assignment).	+=	x+=y	زيادة قيمة x بمقدار قيمة y.
الطرح والإسناد (Subtraction Assignment).	-=	x-=y	إنقاص قيمة x بمقدار قيمة y.
الضرب والإسناد (Multiplication Assignment).	*=	x*=y	مضاعفة قيمة x قيمة y من المَرّات.
القسمة والإسناد (Division Assignment).	/=	x /=y	تخزين ناتج قسمة قيمة x على قيمة y في x.
باقي القسمة والإسناد (Modulus Assignment).	%=	x%=y	تخزين باقي قسمة قيمة x على قيمة y في x.
القوَّة والإسناد (Exponentiation Assignment).	**=	x**=y	تخزين قيمة x مرفوعة إلى أُسٍّ بقيمة y في x.
القسمة والإسناد (Floor Division Assignment).	/ /=	x / /=y	تخزين ناتج x / / y في x.

أولوية العوامل وترابطها (Operators Precedence and Associativity)

تُقيَّم العوامل في لغة البرمجة بايثون (Python) وَفقًا لنظام الأولوية، كما هو الحال في الرياضيات، أنظر الجدول (2-5).

الجدول (2-5): ترتيب العوامل الحسابية والعوامل المنطقية والعوامل المُستخدَمة في المقارنات من أعلاها أولوية إلى أقلها أولوية.

الأولوية	اسم العامل	الرمز	الترابط
1	القوَّة (exponentiation).	**	من اليمين إلى اليسار.
2	الجمع والاسناد (Unary plus).	+	من اليمين إلى اليسار.
	الطرح والاسناد (Unary minus).	-	
3	الضرب (Multiplication).	*	من اليسار إلى اليمين.
	القسمة (Division).	/	
	باقي القسمة (Modulus).	%	
	القسمة التحتية (Floor division).	//	
4	الجمع (Addition).	+	من اليسار إلى اليمين.
	الطرح (Subtraction).	-	
5	أكبر من (Greater than).	>	من اليسار إلى اليمين.
	أصغر من (Less than).	<	
	أكبر من أو يساوي (Greater than or Equal to).	=>	
	أصغر من أو يساوي (Less than or Equal to).	<=	
6	يساوي (Equal to).	==	من اليسار إلى اليمين.
	لا يساوي (Not equal to).	!=	
7	(Logical NOT).	not	من اليمين إلى اليسار.
8	(Logical AND).	and	من اليسار إلى اليمين.
9	(Logical OR).	or	من اليسار إلى اليمين.

إضاءةٌ

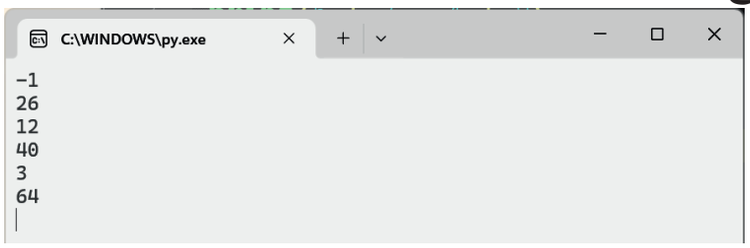
توجد في لغة البرمجة بايثون (Python) عوامـــل أُخرى لم نتطرَّق إليها في هذا الدرس، مثل العوامل المُستخدَمة في البحث في المصفوفات، وسيتمُّ الحديث عنها لاحقًا في هذه الوحدة.

في حال وجود عدد من العوامل لها الأولوية نفسها، فإنَّ ترتيب تنفيذ هذه العوامل يعتمد على قواعد ترابطها (Associativity Rules)، بحيث يكون التنفيذ من اليسار إلى اليمين أو العكس، أمّا العوامل الموجودة بين الأقواس فلها الأولوية العُليا بِغَضِّ النظر عن نوعها.

مثال:

```
print((1 + 3) - (2 + 3))
print(10 + 2 * 8)
print(5 + 2 - 4 + 9)
print((5 + 5) * 4)
print(5 * 2 // 3)
print((2 ** 3) ** 2)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط عملي

أُجرِّب بنفسي: أستكشف الأوامر البرمجية الآتية، ثمَّ أتوقَّع ناتج تنفيذ كلٍّ منها:

```
print(4 + 3 - (2 * 5 / 10))
print(4 * 5 + 5)
print(5 * (2 // 3)
print(2 ** 3 ** 2)
```

أســـتخدِم لغة البرمجة بايثون (Python) في إدخال الأوامر البرمجية وتنفيذها. ما النتيجة التي توصَّلْتُ إليها عند تشغيل البرنامج؟ هل اختلف الناتج عن توقُّعاتي؟ أُبرِّر سبب الاختلاف (إنْ وُجِد).

إثراءٌ

للتعمق في فهم أولوية المعاملات وترابطها أنفذ البرنامج الآتي:

```
meal = 'dates'
money = 0
if meal == "dates" or meal == "sandwich" and money >= 5:
    print("Lunch being delivered.")
else:
    print("Not able to deliver lunch.")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

ألاحظ أنَّ العامل (AND) له أولوية على العامل (OR)؛ لذا سيتمُّ إيجاد ناتج التعبير

meal == "sandwich" and money >= 5

الذي هو (False)، ثمَّ إيجاد ناتج التعبير

meal == "dates" or False

الذي هو (True). وبناءً على ذلك، طُبِعت الجملة الآتية:

Lunch being delivered

نشاط عملي

إذا عُدِّل البرنامج السابق بإضافة الأقواس كما هو مُبيَّن في الأسفل، فما نتيجة البرنامج الجديد؟ أُنفِّذ البرنامج، وأتحقَّق من الناتج.

```
meal = 'dates'
money = 0
if (meal == "dates" or meal == "sandwich") and money >= 5:
    print("Lunch being delivered.")
else:
    print("Not able to deliver lunch.")
```

المواطنةُ الرقميةُ:

- **استخدام المصادر الرسمية:** أُحمِّل برنامج بايثون (Python) من الموقع الإلكتروني الرسمي: python.org، وأتعلَّم من الموارد الموثوقة، وألتزم بشروط الترخيص عند استخدام المكتبات أو الرموز (الأكواد) مفتوحة المصدر.
- **الأمن السيبراني:** أستخدِم برامج مكافحة الفيروسات، وأتجنَّب تحميل البرامج من مصادر غير موثوقة.
- **التعاون المسؤول:** أُشارِك معرفتي بمسؤولية واحترام، وأُساعِد الآخرين على نحوٍ فاعل.

مشروع

المشـــروع: تصميم برنامج وإعداده لإنشاء لعبة تخمين الأرقام باستخدام لغة البرمجة بايثون (Python)/ المهمة (2).

أُكمِل – بالتعاون مع أفراد مجموعتي– تنفيذ مشروع التصميم والتطوير للعبة تخمين الأرقام، وأعمل على اســـتكمال التحضيرات اللازمة للعبة؛ بأنْ أفتـــح برنامجًا خاصًّا بها في برمجية بايثون (Paython)، ثمَّ أبدأ بطباعة رسالة الترحيب وخيارات القائمة الرئيسة (تعليمات اللعبة، ابدأ اللعبة، الخروج من اللعبة)، علمًا بأنَّ المقطع البرمجي المطلوب هو:

```
print("Welcome to Stars and Moons")
print("1. Instructions")
print("2. Start Game")
print("3. Exit")
```

أُجرِّب كتابة برنامج يتولّى تنفيذ ذلك، ثمَّ أُضيف جملة لقراءة الخيار الذي سيختاره اللاعب: (1)، أو (2)، أو (3). غير أنَّه يتعيَّن الانتظار قليلًا إلى حين إضافة (قراءة مدخلات المُستخدِم) إلى البرنامج:

```
option = input("Please select an option (1, 2, 3): ")
```

في هذه المرحلـــة، لا يُمكِنني فعل أيِّ شـــيء بالخيار الذي اختاره اللاعب، وأُراعي الإدخال الصحيح للأوامر البرمجية أثنـــاء كتابة المقطع البرمجي، ثمَّ أحتفـــظ بما كُتِب – ضمن المجموعة– في ملف حتّى يُمكِن لأفراد المجموعة التعديل عليه في الخطوات القادمة.

أُقيِّمُ تعلُّمي

المعرفةُ: أُوظِّفُ في هذا الدرسِ ما تعلَّمْتُهُ منْ معارفَ في الإجابةِ عنِ الأسئلةِ الآتيةِ:

السؤال الأوَّل:

1- ما الدالَّة في لغة البرمجة بايثون (Python) التي تجعل البرنامج تفاعليًّا، وتُمكِّن المُستخدِم من إدخال بيانات في البرنامج أثناء عمله؟

2- فيمَ تختلف كتابة البرنامج في الحالتين الآتيتين:

أ- إدخال المُستخدِم نصًّا في البرنامج.

ب- إدخال المُستخدِم عددًا في البرنامج.

السؤال الثاني: أشرح قواعد كتابة الأسماء في لغة البرمجة بايثون (Python).

السؤال الثالث: ما الفرق بين العامل = والعامل == في لغة البرمجة بايثون (Python)؟ أُدعِّم إجابتي بأمثلة.

المهاراتُ: أُوظِّفُ مهاراتِ التفكيرِ الناقدِ والتواصلِ الرقميِّ والبحثِ الرقميِّ في الإجابةِ عنِ الأسئلةِ الآتيةِ:

السؤال الأوَّل: أتتبَّع البرنامج الآتي من دون تشغيله، ثمَّ أُحدِّد النتيجة المُترتِّبة على عملية التشغيل إذا أدخل المُستخدِم العدد (1)، ثمَّ العدد (2) ثمَّ العدد (3)، ثمَّ العدد (4).

```
x = int(input("Enter x="))
print(5 * (x // 3))
```

السؤال الثاني: أتتبَّع البرنامج الآتي من دون تشغيله، ثمَّ أُحدِّد النتيجة المُترتِّبة على عملية التشغيل.

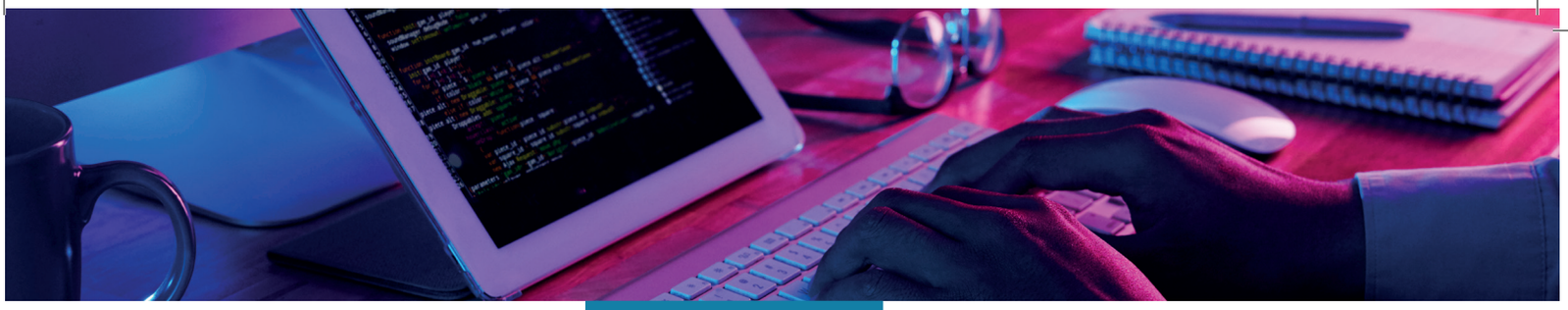
```
meal = 'dates'
money = 10
if meal == "dates" or meal == "sandwich" and money >= 5:
    print("Lunch being delivered.")
else:
    print("Not able to deliver lunch.")
```

السؤال الثالث: أكتشف الأخطاء الواردة في البرنامج الآتي من دون تنفيذه.

```
1st_funding = int(input("Enter 1st funding "))
2nd_funding = int(input("Enter 2nd funding "))
raise = 1st_funding + 2nd_funding
print("Raise =", raise)
```

القِيَمُ والاتجاهاتُ:

أستخدِم إحدى الأدوات التقنية في إعداد قاموس ناطق لمصطلحات لغة البرمجة بايثون (Python)، لمساعدة الطلبة ذوي الإعاقة ثمَّ أعرِضه على زملائي/ زميلاتي في الصف.



الدرسُ الثالث

الجمل الشرطية
(Conditional Statements)

مُنتَجاتُ التعلُّم
(Learning Products)

تعديل البرنامج عبر إضافة الجمل الشرطية لاتخاذ القرارات بناءً على مدخلات المستخدم، ضمن سياق تصميم لعبة تخمين الأرقام باستخدام برمجية بايثون.

الفكرةُ الرئيسةُ:

تعرُّف الجمل الشرطية في لغة البرمجة بايثون (Python)، وبيان كيف يُمكِن كتابتها واستخدامها في تعليق تنفيذ أوامر مُعيَّنة بناءً على شروط يُحدِّدها المُبرمِج، وإجراء تطبيقات عملية لتعزيز الفهم.

المفاهيم والمصطلحات:

الجمل الشرطية (Conditional Statements).

نتاجات التعلُّم (Learning Objectives):

- أكتب جملًا شرطيةً مُركَّبةً ومُترابِطةً باستخدام المُعامِلات المنطقية (مثل: and، وor، وnot) في لغة البرمجة بايثون (Python).
- أستخدِم لغة البرمجة بايثون (Python) في إنشاء برامج تتضمَّن جملًا شرطيةً، وأتتبَّع ناتجها وأُنفِّذها.

لا بد أنك تعرفت الجمل الشرطية وطرق كتابتها في اللغة العربية واللغة الإنجليزية، هل يوجد ارتباط بين مكونات جملة الشرط في هذه اللغات ومكوناتها في لغات البرمجة؟

نشاط تمهيدي

لعبة البطاقات الشرطية

1. أُحضِّر – بالتعاون مع أفراد مجموعتـــي– عددًا من البطاقات المُرقَّمة من (0) إلى (10)، ثمَّ أتنافس مع أفراد المجموعات الأُخرى للفوز في إحدى الألعاب.
2. أبدأ اللعبة بالطلب إلى أحد زملائي/ إحدى زميلاتي في المجموعة ســـحب بطاقة، واتِّباع القواعد الآتية بناءً على العدد المُدوَّن في البطاقة:
   أ. إذا كان العدد المُدوَّن في البطاقة أقل من (5)، فإنَّني أسحب بطاقة أُخرى.
   ب. إذا كان العدد المُدوَّن في البطاقة أكبر من (5)، وكان من الأعداد الزوجية، فإنَّ الدور ينتقل إلى زميل آخر/ زميلة أُخرى في مجموعتي.
   ج. إذا كان العدد المُدوَّن في البطاقة أكبر من (5)، وكان من الأعداد الفردية، فإنَّ الدور ينتقل إلى مجموعة أُخرى.
   د. إذا كان العدد المُدوَّن في البطاقة (0)، فإنَّ المجموعة تخرج من اللعبة.
3. أَلفِتُ انتباه الجميع إلى وجوب تكرار ســـحب البطاقات بالتناوب بين المجموعات، وتنفيذ القواعد السابقة.
4. أجتمِع مع أفراد مجموعتي – بعد انتهاء اللعبة، أو انتهاء الوقت المُحدَّد لها–، ثمَّ أُناقِشهم في ما تعلَّمناه.

كيف تُشبِه هذه اللعبة الجمل الشرطية في البرمجة؟ أُدوِّن توقُّعاتي.

أنواع الجمل الشرطية في بايثون (Python)

تُســـتخدَم الجمل الشـــرطية في تنفيذ مجموعة من الأوامر البرمجية في البرنامج بناءً على شروط يُحدِّدها المُبرمِج. وهي تمتاز بتعدُّد أشكالها، وتفرُّد كل جملة منها بصيغة عامة تُحدِّد طريقة تنفيذها في البرنامج.

أولًا: الجملة الشرطية if (if statement)

تُكتَب الصيغة العامة للجملة الشرطية (if) على النحو الآتي:

```
if condition:
    statements1
```

حيث:

If: كلمة محجوزة في لغة البرمجة بايثون (Python).

condition: الشرط (تعبير منطقي).

statements: أوامر برمجية تُنفَّذ إذا كان الشرط صحيحًا (تحقُّق الشرط).

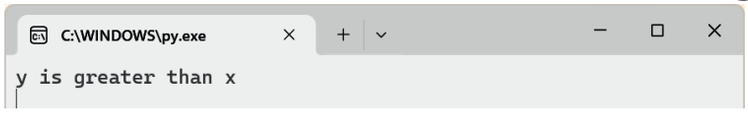
(:): علامة يجب أنْ توضَع بعد الشرط (condition) حتّى يُنفَّذ البرنامج.

مثال:

يطبع البرنامج الآتي عبارة "y is greater than x" إذا كانت قيمة المُتغيِّر (y) أكبر من قيمة المُتغيِّر (x):

```
x = 3
y = 20
if y > x: print("y is greater than x")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



إضاءةٌ

يُمكِن كتابة الجملة الشرطية السابقة بصورة مختصرة كما يأتي:

```
if y > x: print("y is greater than x")
```

يجب أنْ تكون المسافات البادئة صحيحة؛ لكي يعمل المقطع البرمجي (الكود) بصورة صحيحة في لغة البرمجة بايثون (Python).

– أتأكَّد دائمًا أنَّ الأسطر التي تتبع الشرط (مثل السطر الذي يحتوي على كلمة (print) تحوي مسافة بادئة (أربع مسافات أو Tab واحدة).

نشاط عملي

أُجرِّب بنفسي: أُعدِّل المقطع البرمجي (الكود) في المثال السابق لتعيين قيمة (30) للمُتغيِّر (x)، وأتتبَّع النتيجة المُتوقَّعة من دون تشـــغيل البرنامج، ثمَّ أتحقَّق من الناتج عن طريق تنفيذ البرنامج في بيئة بايثون (Python).

ما الذي يجب تعديله في البرنامج لطباعة جملة "x is greater than y" بدلًا من جملة "y is greater than x"؟

ثانيًا: الجملة الشرطية (if else)

تُكتَب الصيغة العامة للجملة الشرطية (if else) على النحو الآتي:

```
if condition:
    statements1
else:
   statements2
```

حيث:

if, else: كلمتان محجوزتان في لغة البرمجة بايثون (Python).

statements1: أوامر برمجية تُنفَّذ إذا كان الشرط صحيحًا (تحقُّق الشرط).

statements2: أوامر برمجية تُنفَّذ إذا لم يكن الشرط صحيحًا (عدم تحقُّق الشرط).

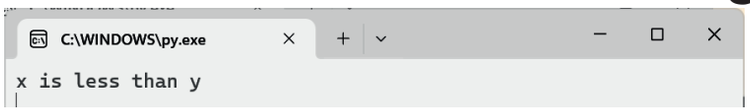
(:): علامة يجب أنْ توضع بعد الشرط (condition) و بعد جملة else.

مثال:

يطبع البرنامج الآتي عبـــارة "y is less than x" إذا كانت قيمة المُتغيِّر (y) أقل من قيمة المُتغيِّر (x)، ويطبع البرنامج عبارة "x is less than y" إذا كانت قيمة المُتغيِّر (x) أقل من قيمة المُتغيِّر (y):

```
x = 80
y = 100
if y < x:
    print("y is less than x")
else:
    print("x is less than y")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية:



إضاءةٌ

يُمكِن كتابة الجملة الشرطية السابقة بصورة مختصرة كما يأتي:

```
x = 80
y= 100
print("y is less than x") if y < x else print("x is less than y")
```

نشاط عملي

أُجرِّب بنفسي: أُعدِّل المقطع البرمجي (الكود) في المثال السابق لتعيين قيمة (100) للمُتغيِّر (x)، وأتتبَّع النتيجة المُتوقَّعة من دون تشـــغيل البرنامج، ثمَّ أتحقَّق من الناتج عن طريق تنفيذ البرنامج في بيئة بايثون (Python).

ثالثًا: الجملة الشرطية (if elif)

تُكتَب الصيغة العامة للجملة الشرطية (if elif) على النحو الآتي:

```
if condition1:
  statements1
elif condition2:
  statements2
```

حيث:

if, elif: كلمتان محجوزتان في لغة البرمجة بايثون (Python).

statements1: أوامر برمجية تُنفَّذ إذا تحقَّق الشرط (condition1).

statements2: أوامر برمجية تُنفَّذ إذا تحقَّق الشرط (condition2).

مثال:

يطبع البرنامج الآتي عبارة "y is greater than x" إذا كانت قيمة المُتغيِّر (y) أكبر من قيمة المُتغيِّر (x)، ويطبع البرنامج عبارة "x equals y" إذا كانت قيمة المُتغيِّر (x) تساوي قيمة المُتغيِّر (y):

```
x = 20
y = 20
if y > x:
    print('y is greater than x')
elif x == y:
    print('x equals y')
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية:

نشاط عملي

أُجرِّب بنفسي: أُعدِّل المقطع البرمجي (الكود) في المثال السابق لتعيين قيمة (30) للمُتغيِّر (x)، وأتتبَّع النتيجة المُتوقَّعة من دون تشـــغيل البرنامج، ثمَّ أتحقَّق من الناتج عن طريق تنفيذ البرنامج في بيئة بايثون (Python).

نشاط فردي

أكتب جملًا برمجية بلغة البرمجة بايثون (Python) لحساب قيمة المُتغيِّر (z) وَفقًا لكل معادلة ممّا يأتي وطباعته:

$z = x^2 * y \quad , x >= 5$

$z = x * (x + y) \ , x == 5$

رابعًا: الجملة الشرطية (if elif else)

تُكتَب الصيغة العامة للجملة الشرطية (if elif else) على النحو الآتي:

```
if  condition1:
 statements1
elif  condition2:
 statements2
else:
 statements3
```

حيث:

if, else, elif: كلمات محجوزة في لغة البرمجة بايثون (Python).

statements1: أوامر برمجية تُنفَّذ إذا تحقَّق الشرط (condition1).

statements2: أوامر برمجية تُنفَّذ إذا تحقَّق الشرط (condition2).

statements3: أوامر برمجية تُنفَّذ إذا لم يتحقَّق أيٌّ من الشرطين (condition2, condition1).

مثال:

يطبع البرنامج الآتي عبارة "y is greater than x" إذا كانت قيمة المُتغيِّر (y) أكبر من قيمة المُتغيِّر (x)، ويطبع البرنامج عبارة "x equals y" إذا كانت قيمة المُتغيِّر (x) تساوي قيمة المُتغيِّر (y)، ويطبع البرنامج عبارة "x is greater than y" إذا كانت قيمة المتغير (y) أصغر من قيمة المتغير (x):

```
x = 50
y = 25
if y > x:
    print("y is greater than x")
elif y == x:
    print("x equals y")
else:
    print("x is greater than y")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية:

نشاط عملي

أُجرِّب بنفسي:

استنادًا إلى المثال السابق، أُجيب عن الأسئلة الآتية:

1. ما النتيجة المُترتِّبة على تشـــغيل البرنامج إذا كانت قيمة (x) تساوي (40)؟ أُعدِّل المقطع البرمجي (الكود) في المثال لتعيين قيمة (40) للمُتغيِّر (x)، وأتتبَّع النتيجة المُتوقَّعة من دون تشغيل البرنامج، ثمَّ أتحقَّق من الناتج عن طريق تنفيذ البرنامج في بيئة بايثون (Python).
2. ما النتيجة المُترتِّبة على تشـــغيل البرنامج إذا كانت قيمة (x) تساوي (10)؟ أُعدِّل المقطع البرمجي (الكود) في المثال لتعيين قيمة (10) للمُتغيِّر (x)، وأتتبَّع النتيجة المُتوقَّعة من دون تشغيل البرنامج، ثمَّ أتحقَّق من الناتج عن طريق تنفيذ البرنامج في بيئة بايثون (Python).
3. ما النتيجة المُترتِّبة على تشـــغيل البرنامج إذا كانت قيمة (x) تساوي (25)؟ أُعدِّل المقطع البرمجي (الكود) في المثال لتعيين قيمة (25) للمُتغيِّر (x)، وأتتبَّع النتيجة المُتوقَّعة من دون تشغيل البرنامج، ثمَّ أتحقَّق من الناتج عن طريق تنفيذ البرنامج في بيئة بايثون (Python).

أبحث

أبحث: أبحث في المواقع الإلكترونية الموثوقة في شبكة الإنترنت عن لغات برمجة أُخرى، ثمَّ أُقارِن طريقة كتابة الجمل الشرطية فيها بطريقة كتابة الجمل الشرطية في لغة البرمجة بايثون (Python).

المُعامِلات المنطقية (Logical Operators)

يُستعمَل المُعامِل المنطقي (and) والمُعامِل المنطقي (or) لربط التعابير المنطقية البسيطة، وتكوين جمل منطقية مُركَّبة، في حين يُستعمَل المُعامِل المنطقي (not) لنفي التعابير المنطقية.

1- المُعامِل المنطقي (and):

قد يتوقَّف تنفيذ أمر برمجي مُعيَّن في البرنامج على تحقُّق مجموعة من الشروط مُجتمِعةً.

مثال:

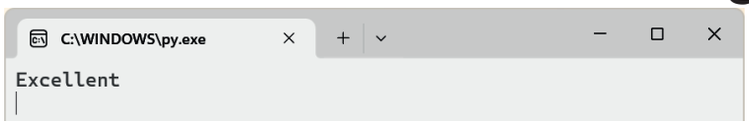
يراد طباعة كلمة "Excellent" إذا كان مُعدَّل الطالب/ الطالبة أكبر من أو يســـاوي (90) وأقل من أو يساوي (100).

الحَلُّ:

يجب استعمال المُعامِل المنطقي (and) للدلالة على تحقُّق الشرطين معًا، وتُكتَب الأوامر البرمجية على النحو الآتي:

```
Avg = 95
if Avg >= 90 and Avg <=100:
    print("Excellent")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية:



نشاط عملي

أُجرِّب بنفسي: أكتب المقطع البرمجي (الكود) السابق، ثمَّ أتحقَّق من النتيجة عن طريق تنفيذ البرنامج.

أُعدِّل قيمة المُتغيِّر (Avg) لتصبح (80)، ثمَّ أُنفِّذ البرنامج. ما الناتج الظاهر على شاشة جهاز الحاسوب؟

أحذِف المسافة البادئة قبل جملة الطباعة، ثمَّ أُنفِّذ البرنامج. ما ناتج تنفيذ البرنامج؟

2- المُعامِل المنطقي (or):

قد يتوقَّف تنفيذ أمر مُعيَّن في البرنامج على تحقُّق شرط من مجموعة شروط.

مثال:

يراد طباعة قيمة المُتغيِّر (x) إذا كانت قيمة هذا المُتغيِّر تساوي (1) أو (2).

الحَلُّ:

يجب استعمال المُعامِل المنطقي (or) للدلالة على تنفيذ جملة الطباعة في حال تحقَّق أحد الشرطين، وتُكتَب الأوامر البرمجية على النحو الآتي:

```
x = 1
if x == 1 or x == 2:
    print('you selected a valid number')
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية:

3- المُعامِل المنطقي (not):

قد يتوقَّف تنفيذ أمر في البرنامج على عدم تحقُّق شرط مُعيَّن.

مثال:

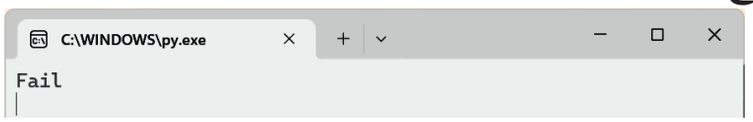
يراد طباعة كلمة "Fail" إذا لم يُحقِّق الطالب/ الطالبة شرط النجاح (Avg >= 50).

الحَلُّ:

يجب استعمال المُعامِل المنطقي (not) للدلالة على عدم تحقُّق الشرط، وتُكتَب الأوامر البرمجية على النحو الآتي:

```
Avg = 49
if not Avg >= 50:
    print("Fail")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية:



نشاط فردي

- أكتب المقطع البرمجي (الكود) السابق بطريقة أُخرى، ثمَّ أتحقَّق من النتيجة عن طريق تنفيذ البرنامج.
- أُعدِّل قيمة المُتغيِّر (Avg) لتصبح (70)، ثمَّ أُنفِّذ البرنامج. ما الناتج الظاهر على شاشة جهاز الحاسوب؟

نشاط جماعي

المُعامِلات المنطقية والجمل الشرطية

أكتب – بالتعاون مع أفراد مجموعتي– مقطعًا برمجيًّا بلغة البرمجة بايثون (Python) لإدخال قيمة إحدى فواتير الشراء وعدد المشتريات، وطباعة عبارة "No discount" إذا كانت قيمة الفاتورة أقل من (100)، أو كان عدد المشتريات أقل من (3)، وطباعة قيمة الخصم الذي يساوي 5% من قيمة الفاتورة إذا كانت قيمتها أكبر من (100)، وكان عدد المشتريات أكبر من أو يساوي (3).

الجملة الشرطية المُركَّبة (Nested Conditional Statements)

تُوفِّر لغة البرمجة بايثون (Python) إمكانية كتابة جمل شرطية مُركَّبة (مُتداخِلة)؛ أيْ وضع جملة شرطية (if) داخل جملة شرطية (if) أُخرى.

مثال:

يطبع البرنامج الآتي عبارة "Above ten" إذا كانت قيمة (x) أكبر من (10)، ثمَّ يتحقَّق إذا كانت قيمة (x) أكبر من (60). فيطبع البرنامج، إضافة إلى العبارة السابقة، عبارة "and also above 60" إذا كانت قيمة (x) أكبر من (60)، و يطبع عبارة "but not above 60" إذا كانت قيمة (x) أقل من (60).

```
x = 68
if x > 10:
    print("Above ten,")
    if x > 60:
        print("and also above 60.")
    else:
        print("but not above 60.")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية:

نشاط فردي

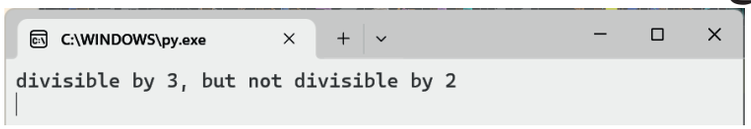
- أُعدِّل قيمة المُتغيِّر (x) لتصبح (68)، ثمَّ أُنفِّذ البرنامج. ما الناتج الظاهر على شاشة جهاز الحاسوب؟
- أُعدِّل قيمة المُتغيِّر (x) لتصبح (15)، ثمَّ أُنفِّذ البرنامج. ما الناتج الظاهر على شاشة جهاز الحاسوب؟
- أحذِف المسافة البادئة قبل جملة الطباعة الأخيرة. ما ناتج تنفيذ البرنامج؟ أُناقِش إجاباتي مع زملائي/ زميلاتي في الصف.

مثال:

يُستخدَم البرنامج الآتي في التحقُّق إذا كانت قيمة المُتغيِّر (y) تقبل القسمة على (3)، وتقبل القسمة على (2)، أو تقبل القسمة على أحدهما ، أو لا تقبل القسمة على أي منهما. ثمَّ يطبع البرنامج العبارة الدالَّة على ذلك.

```
y = 9
if y %2==0:
    if y %3==0:
        print("divisible by 3 and 2")
    else:
        print("divisible by 2, but not divisible by 3")
else:
    if y %3==0:
        print("divisible by 3, but not divisible by 2")
    else:
        print("not divisible by 2 and not divisible by 3")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية:



نشاط عملي

أُجرِّب بنفسي:

استنادًا إلى المثال السابق، أُجيب عن الأسئلة الآتية:

1. ما نتيجة البرنامج إذا كانت قيمة (y) تساوي (8)؟

2. أكتب المقطع البرمجي (الكود) الوارد في المثال السابق في لغة البرمجة بايثون (Python)، وأتتبَّع النتيجة المُتوقَّعة من دون تشغيل البرنامج، ثمَّ أتحقَّق من الناتج عن طريق تنفيذ البرنامج.

3. ما نتيجة البرنامج إذا كانت قيمة (y) تساوي (5)؟

إضاءةٌ

إذا أردْتُ كتابة جملة شـــرطية (if) فارغة لا تحتوي على أيٍّ من الجمل، فإنَّني أضع الجملة (pass) داخل هذه الجملة الشرطية (if)؛ لتجنُّب ظهور رسالة تفيد بوجود خطأ في البرنامج.

نشاط عملي

أُجرِّب بنفسي:

أتتبَّع تنفيذ البرنامج الآتي في بيئة بايثون (Python)، بوجود جملة (pass) تارة، وعدم وجودها تارة أُخرى.

```
x = 4
y = 2
if x > y:
  pass
```

نشاط جماعي

اكتشاف الأخطاء في المقطع البرمجي بلغة البرمجة بايثون (Python)

أُحلِّل – بالتعاون مع أفراد مجموعتي – المقطع البرمجي الآتي، وأكتشِف الأخطاء الواردة فيه من دون تنفيذه، وأعمل على تصحيح هذه الأخطاء، ثمَّ أُنفِّذ المقطع البرمجي بعد تصحيحه.

```
grade = input("Enter your grade: ")

if grade >= 85
   print("A")
elif grade >= 75
   print("B")
elif grade >= 65:
   print("C")
else
   print("F")
```

المواطنةُ الرقميةُ:

- **التعاون والمشـــاركة**: أتعاون مع الزملاء/ الزميلات، وأُشارِكهم في تنفيذ الأنشطة واكتشاف الأخطاء البرمجية وتحليلها؛ ما يُعزِّز لديَّ مهارة التفكير الناقد والعمل الجماعي، ويُســـهِم في بناء مجتمع تعليمي داعم ومُشارِك.
- **الأخلاقيات الرقمية**: أحترِم حقوق المُلْكية الفكرية عند استخدام المقاطع البرمجية الموجودة في شبكة الإنترنت، أو عند التعديل عليها، وأحرِص على أخذ الموافقة المُسبَّقة على ذلك.

مشروع

المشروع:

أُكمِل – بالتعاون مع أفراد مجموعتي – تنفيذ مشروع التصميم والتطوير للعبة تخمين الأرقام؛ بـــأنْ أطبع تعليمات اللعبة عند اختيار اللاعب الرقم (1) من القائمة. وبناءً على ما تعلَّمناه في هذا الدرس، ســـأعمل – ضمن المجموعة – على تعديل ما كتبناه سابقًا (باستخدام الجمل الشرطية) كما يأتي:

1- طباعة شرح عن تعليمات اللعبة إنْ أدخل اللاعب الرقم (1).

2- طباعة جملة: (لم يتمَّ تنفيذ هذه الخصيصة بعدُ" إذا اختار اللاعب الرقم (2) أو الرقم (3).

3- طباعـــة جملة: (الخيار المُدخَل غير معروف، أَدخِل 1 أو 2 أو 3) إذا أدخل اللاعب رقمًا غير الأرقام المُشار إليها.

بعد ذلك أتأكَّد – مع أفراد مجموعتي – أنَّ الشـــروط قد كُتِبت بصورة صحيحة، وأتحقَّق من استجابة البرنامج للخيارات المُدخَلة بشكل مناسب، ثمَّ أحتفظ بما كُتِب – ضمن المجموعة – في ملف حتّى يُمكِن لأفراد المجموعة التعديل عليه في الخطوات القادمة.

أُقيِّمُ تعلُّمي

المعرفةُ: أُوظِّفُ في هذا الدرسِ ما تعلَّمْتُهُ منْ معارفَ في الإجابةِ عنِ الأسئلةِ الآتيةِ:

السؤال الأوَّل: ما الفرق بين الجملة الشـــرطية (if else) والجملة الشرطية (if elif)في لغة البرمجة بايثون (Python)؟

السؤال الثاني: أذكر أمثلة على استخدام جملة (pass) داخل الجملة الشرطية (if).

المهاراتُ:أُوظِّفُ مهاراتِ التفكيرِ الناقدِ والتواصلِ الرقميِّ والبحثِ الرقميِّ في الإجابةِ عنِ الأسئلةِ الآتيةِ:

السؤال الأوَّل: أتتبَّع البرنامج الآتي من دون تشغيله، ثمَّ أذكر النتيجة المُترتِّبة على تشغيله.

```
x = 20
y = 5
z = 30
if  not x <= y and x < z:
    print("y < x < z")
```

السؤال الثاني: اقرأ البرنامج التالي المكتوب بلغة البرمجة بايثون (Python)، ثمَّ أُجيب عن السؤالين الآتيين:

1. مـــا الهدف الرئيس من البرنامج؟ أَصِف ما يقوم به البرنامج عامةً من دون وصف وظيفة كل أمر برمجي فيه.

```
x = int(input("Enter your grade: "))
if x > 84:
    print("Excellent grade.")
```

```
elif x > 76:
    print("Very good grade.")
elif x > 68:
    print("Good grade.")
elif x > 50:
    print("You passed the course.")
else:
    print("You failed the course.")
```

2. ما النتيجة المُترتِّبة على تشـــغيل البرنامج إذا أدخل المُســـتخدِم العدد (49)، ثمَّ العدد (55)، ثمَّ العدد (68)، ثمَّ العدد (70)، ثمَّ العدد (90)، ثمَّ العدد (78)، ثمَّ العدد (76)؟

السؤال الثالث: أكتشف الأخطاء الواردة في البرنامج الآتي من دون تنفيذه.

```
n = int(input("Enter any natural number: ")

if n <= 0:
    print("Wrong input. Please enter a positive number.")
else: sum = 0
while n > 0:
    sum += n
    n -= 1
print("The sum of the natural numbers is: ", sum)
```

القِيَمُ والاتجاهاتُ:
أبحث في تطبيقات الجمل الشـــرطية الحياتية وأجمعها وأوثقها حفظًا لحقوق الملكية الفكرية في ملف ثم أنشرها عبر المواقع الإلكترونية للمدرسة بهدف الإسهام في نشر المعرفة.

الدرسُ الرابع

الحلقات (Loops)

مُنتَجاتُ التعلُّم
(Learning Products)

تعديل البرنامج بإضافة حلقات تكرارية تُمكّن من عرض القائمة بصورة مُتكرّرة حتّى يتمّ اختيار الخيار الصحيح، ضمن سياق تصميم لعبة تخمين الأرقام باستخدام برمجية بايثون (Paython).

الفكرة الرئيسة:

تعرُّف كيف تُكتَب جمل التحكُّم في برمجية بايثون (Python) باستخدام المُعامِلات المنطقية، واختيار أكثر الهياكل البرمجية مناسبة لحَلِّ مشكلات مُعيَّنة بكفاءة وتفسير سبب استخدامها، وكتابة مقاطع برمجية بسيطة وواضحة بحيث يَسهُل على الآخرين فهمها وصيانتها.

المفاهيم والمصطلحات:

الحلقات (Loops)، جمل التحكُّم (Control Statements)، التهيئة (Initialization)، جمل (Statements)، الزيادة (Increment)، النقصان (Decrement)، العنصر (Element)، المصفوفة (Sequence)، النطاق (Range).

نتاجات التعلُّم (Learning Objectives):

- أكتب جمل التحكُّم باستخدام الحلقات (مثل: For، و While) في برمجية بايثون (Python).
- أستخدم أكثر الهياكل البرمجية (مثل: الحلقات، والجمل الشرطية) مناسبة لحَلِّ مشكلات مُعيَّنة بكفاءة.
- أكتب برامج مُتكامِلة على نحوٍ يُسهِّل على الآخرين قراءتها وفهمها.

تعرَّفْتُ سابقًا الحلقات في برمجية سكراتش (Scratch)، واستخدمْتُها في كتابة البرامج التي تتطلَّب تكرار تنفيذ مجموعة من الأوامر البرمجية عددًا من المَرّات. هل تتشابه بِنية الحلقات في لغات البرمجة جميعها؟

نشاط عملي

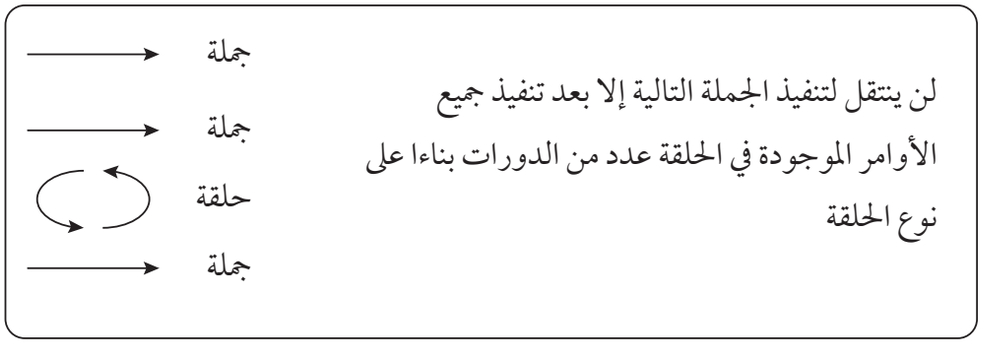
أُفكِّر في روتيني اليومي، وأُحدِّد الأعمال والأنشطة التي أقوم بها كل يوم بانتظام، مثل: الاستيقاظ، وتناول طعام الإفطار، والذهاب إلى المدرسة، والتعلُّم، وتناول طعام الغداء، واللعب، وتناول طعام العشاء، والنوم. بعد ذلك أُدوِّن هذه الأعمال والأنشطة في دفتر ملاحظاتي، ثمَّ أُناقِشها مع زملائي/ زميلاتي ومُعلِّمي/ مُعلِّمتي.

أنواع الحلقات في برمجية بايثون (Python)

يُشبِه الروتين اليومي الحلقة في البرمجة؛ إذ تشهد تكرار نفس المجموعة من الأوامر كل يوم. في ما يأتي مثال بسيط على حلقة في برمجية بايثون (Python) تُمثِّل روتينًا يوميًّا إلى جانب التوقيتات:

```
Python 3.12 (64-bit)
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> daily_routine = [
...     ("6:00 AM", "Wake up"),
...     ("7:00 AM", "Have breakfast"),
...     ("8:00 AM", "Go to school"),
...     ("12:00 PM", "Have lunch"),
...     ("4:00 PM", "Play"),
...     ("6:00 PM", "Have dinner"),
...     ("9:00 PM", "Sleep")
... ]
>>>
>>> for time, activity in daily_routine:
...     print(f"At {time}, it's time to: {activity}")
...
At 6:00 AM, it's time to: Wake up
At 7:00 AM, it's time to: Have breakfast
At 8:00 AM, it's time to: Go to school
At 12:00 PM, it's time to: Have lunch
At 4:00 PM, it's time to: Play
At 6:00 PM, it's time to: Have dinner
At 9:00 PM, it's time to: Sleep
>>>
```

يستفاد من الحلقات في تكرار مجموعة من الأوامر البرمجية مَرّات عديدة؛ فعند البَدْء بتشغيل برنامج يحتوي على حلقة، فإنَّ تنفيذ الجمل يتوقَّف عند هذه الحلقة، حيث يعمل البرنامج على تنفيذ ما في داخل الحلقة من جمل عددًا من المَرّات. وبعد أنْ يخرج البرنامج من تلك الحلقة، فإنَّه يأخذ بتنفيذ بقية الجمل التي تليها، أنظر الشكل (4-1).



الشكل (4-1): مبدأ عمل الحلقات في برمجية بايثون (Python).

تُصنَّف الحلقات في برمجية بايثون (Python) إلى نوعين، هما:

1. **حلقات while (while loops)** : تُستعمَل حلقة (while) لتكرار تنفيذ جملة واحدة أو أكثر طالما تحقَّق شرط مُعيَّن. وفي حال لم يعد هذا الشرط مُتحقِّقًا، فإنَّ البرنامج يتوقَّف عن تنفيذ هذه الجملة أو الجمل.
2. **حلقات for (for loops)**: تُستعمَل حلقة (for) لتكرار مجموعة من الجمل البرمجية عددًا مُحدَّدًا من المَرّات.

تحتوي برمجية بايثون (Python) على جمل تحكُّم (Control Statements) تعمل على ضبط الآليَّة التي تُنفَّذ بها الحلقات. وتشمل هذه الجمل كلًّا من جملة التحكُّم (break)، وجملة التحكُّم (continue):

1. **جملة التحكُّم (break)**: تُستعمَل هذه الجملة لإيقاف الحلقة إذا تحقَّق شرط مُعيَّن، ثمَّ تنفيذ الجمل التي تلي الحلقة في البرنامج.
2. **جملة التحكُّم (continue)**: تُستعمَل هذه الجملة لإيقاف الدورة الحالية في الحلقة، والانتقال إلى الدورة التالية فيها إذا تحقَّق شرط مُعيَّن.

في ما يأتي بيان للحلقات وجمل التحكُّم الموجودة في برمجية بايثون (Python) وطرائق استخدامها في البرامج:

حلقات while (while loops)

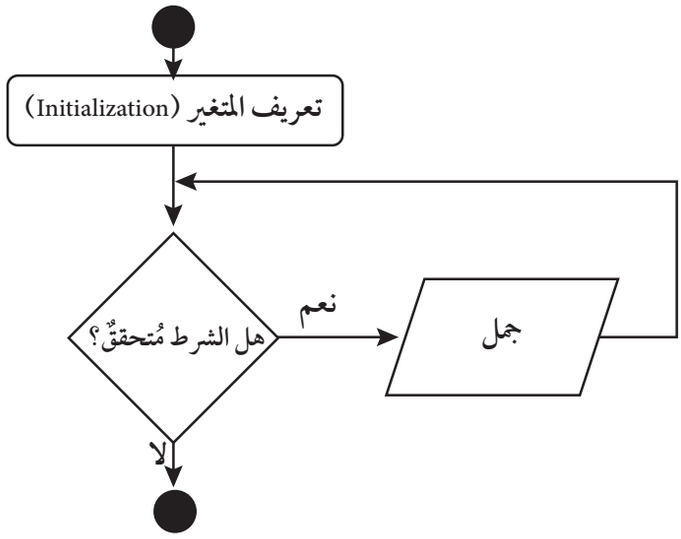
تُعرَّف حلقة (while) باستخدام الكلمة المحجوزة (while)، وتُكتَب صيغتها العامة على النحو الآتي:

```
while  condition:
 statements
 increment or decrement
```

في ما يأتي بيان مُفصَّل لكل عنصر يُمثِّل جزءًا من عملية التعريف بحلقة (while):

- condition: شرط يُحدِّد استمرار تنفيذ الجمل الموجودة في حلقة (while)، ويتمُّ التحقُّق منه في كل دورة؛ إذ يتوقَّف تنفيذ الجمل حين يصبح هذا الشرط غير مُتحقِّق.
- statements: جمل توجد في حلقة (while)، ويُكرِّر البرنامج تنفيذها ما دام الشرط مُتحقِّقًا.
- decrement، أو increment: إجراء يُحدِّد كيف تزداد قيمة العدّاد أو تنقص، وهو يُنفَّذ في كل دورة.

يُبيِّن الشكل (4-2) تمثيلًا لطريقة عمل حلقات (while) باستخدام مُخطَّط سَيْر العمليات.



الشكل (4-2): مُخطَّط سَيْر العمليات لحلقات (while).

مثال

يطبع البرنامج الآتي قيمة العدّاد (count) من (1) إلى (5):

```
count = 1
while count < 6:
    print("Count is", count)
    count+= 1
else:
    print("Loop has ended")
```

إذا تتبَّعْتُ المقطع البرمجي في المثال السابق، أَجِد أنَّ البرنامج يُنفِّذه كما يأتي:

1. Initialization: بَدْء البرنامج بتعريف المُتغيِّر count، وإعطائه القيمة الأوَّلية (1).
2. Condition: تحقُّق البرنامج من أنَّ قيمة المُتغيِّر count أقل من (6). وفي حال كان الشرط صحيحًا، فإنَّ البرنامج يستمر في تنفيذ الجملة داخل الحلقة.
3. Statement: جملة الطباعة لقيمة المُتغيِّر count.
4. Decrement، أو Increment: عمل البرنامج على زيادة قيمة (count) بمقدار (1) في كل دورة؛ ما يُؤثِّر في تحقُّق الشرط في الدورة التالية.

بعد ذلك سيعمل البرنامج على التحقُّق من الشرط مَرَّة أُخرى؛ فإذا تبيَّن أنَّ الشرط لا يزال صحيحًا، فإنَّ البرنامج سيُكرِّر الخطوة الثالثة والخطوة الرابعة. أمّا إذا لم يَعُدِ الشرط صحيحًا، فإنَّ البرنامج يخرج من الحلقة، ويطبع عبارة "Loop has ended".

نشاط عملي

- أُنفِّذ المثال السابق في بيئة بايثون (Python)، وأُلاحِظ الناتج.
- أُعدِّل المقطع البرمجي بتغيير جملة (count += 1) إلى جملة (count += 2)، ثمَّ أُنفِّذ البرنامج. ما الناتج المُترتِّب على تنفيذ البرنامج؟
- أُعدِّل المقطع البرمجي بتغيير الشرط (count = 6) إلى الشرط (count < 6)، ثمَّ أُنفِّذ البرنامج. ما الناتج المُترتِّب على تنفيذ البرنامج؟

أُقارِن إجاباتي بإجابات زملائي/ زميلاتي في الصف.

جملة التحكُّم (break) في حلقات (while)

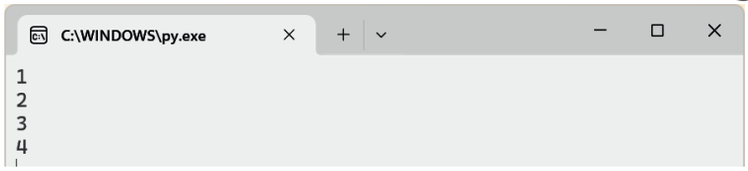
تعرَّفْتُ في بداية الدرس أنَّ جملة التحكُّم (break) تُستعمَل لإيقاف الحلقة إذا تحقَّق شرط مُعيَّن، وأنَّ البرنامج يعمل بعد ذلك على تنفيذ الجمل التي تلي الحلقة فيه.

مثال:

يُنفِّذ البرنامج – الذي ورد ذكره في المثال السابق– جملة (break) إذا أصبحت قيمة المُتغيِّر (count) تساوي (4)، وذلك على النحو الآتي:

```
count = 1
while count < 6:
    print(count)
    if count == 4:
        break
    count += 1
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



جملة التحكُّم (continue) في حلقات (while)

تعرَّفْتُ سابقًا أنَّ جملة التحكُّم (continue) تُستعمَل لإيقاف الدورة الحالية في الحلقة، والانتقال إلى الدورة التالية فيها إذا تحقَّق شرط مُعيَّن.

مثال:

تعمل جملة التحكُّم (continue) على طباعة الأرقام من (1) إلى (6) باستثناء الرقم (2) كما يأتي:

```
count = 0
while count < 6:
    count += 1
    if count == 2:
        continue
    print(count)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

جملة (else) مع حلقات (while)

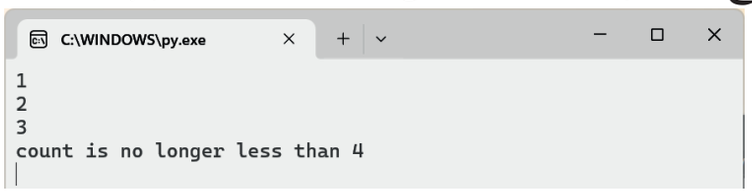
تُستعمَل جملة (else) مع حلقة (while) لتنفيذ مجموعة من الأوامر البرمجية إذا أصبحت قيمة الشرط خطأً (False)؛ أيْ خارج الحلقة.

مثال:

يطبع البرنامج الآتي قيمة العدّاد إذا كانت القيمة أقل من (4). وخلافًا لذلك، فإنَّ البرنامج سيطبع عبارة "count is no longer less than 4":

```
count = 1
while count < 4:
    print(count)
    count += 1
else:
    print("count is no longer less than 4")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط عملي

في المثال السابق، إذا حُذِفت جملة (else)، فهل سيختلف الناتج؟ ماذا أتوقَّع أنْ يكون الناتج؟ أُعدِّل المقطع البرمجي بحذف جملة (else)، ثمَّ أُنفِّذ البرنامج المُعدَّل باستخدام برمجية بايثون (Python)، ثمَّ أُقارِن الناتج الحالي بالناتج السابق، وأُلاحِظ الفرق بينهما إن وجد.

أُناقِش زملائي/ زميلاتي في السؤالين الآتيين:

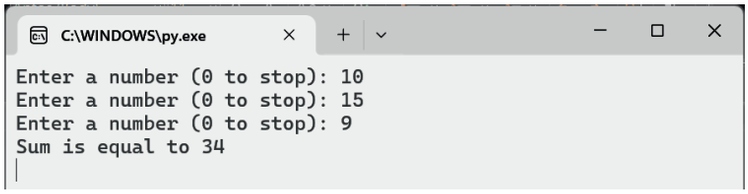
- هل أثَّرت جملة (else) في الناتج؟

مثال:

يعمل البرنامج الآتي على حساب مجموع الأعداد التي أدخلها المُستخدِم حتى يصل المجموع إلى (30) فأكثر، أو حتى يتمَّ إدخال القيمة (0). فإذا وصل المجموع إلى (30) فأكثر، طُبِع المجموع الحالي، وخرج البرنامج من الحلقة. أمّا إذا أُدخِلت القيمة (0) قبل الوصول إلى المجموع (30)، فإنَّ البرنامج يطبع رسالة مفادها أنَّ المجموع أقل من (30)، ثمَّ يعرض قيمة المجموع النهائية.

```
s = 0
a = int(input("Enter a number (0 to stop): "))
while a != 0:
    s += a
    if s >= 30:
        print("Sum is equal to", s)
        break
    a = int(input("Enter a number (0 to stop): "))
else:
    print("Sum is equal to", s, ".")
```

عند تشغيل البرنامج، وإدخال العدد (10)، ثمَّ العدد (15)، ثمَّ العدد (9)، ثمَّ العدد (0)، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



إذا حُذِفت جملة (else) الواردة في المثال السابق، فإنَّ النتيجة الآتية ستظهر على شاشة جهاز الحاسوب عند تشغيل البرنامج:

```
s = 0
a = int(input("Enter a number (0 to stop): "))
while a != 0:
    s += a
    if s >= 30:
        print("Sum is equal to", s)
        break
    a = int(input("Enter a number (0 to stop): "))
```

أُلاحِظ أنَّ وجود جملة (else) يتيح للمُستخدِم التحكُّم في طباعة الجملة الثانية، بحيث لا تُطبَع إلّا بعد خروج البرنامج من الحلقة دون أنْ يتحقَّق الشرط في جملة (if).

نشاط جماعي

أقرأ – بالتعاون مع أفراد مجموعتي– المقطعين البرمجيين الآتيين، وأُحاوِل توقُّع ناتج التنفيذ، ثمَّ أُنفِّذ هذين المقطعين باستخدام برمجية بايثون (Python)، ثمَّ أُقارِن الناتج بما توقَّعْتُه:

```
i = 1
while i == 1:
    print("I am stuck")

while True:
    print("I am stuck")
```

أُناقِش زملائي/ زميلاتي في الأسئلة الآتية:

– لماذا يؤدّي هذان المقطعان البرمجيان إلى حلقة لانهائية؟

– ما الطرائق التي يُمكِن استعمالها لتجنُّب الحلقات اللانهائية؟

– كيف يُمكِن تعديل المقطع البرمجي على نحوٍ يجعل التنفيذ ينتهي في نقطة مُحدَّدة؟

حلقات for (for loops)

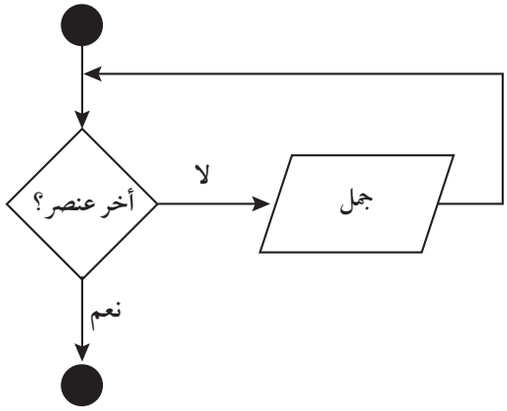
تُعرَّف حلقة (for) باستخدام الكلمتين المحجوزتين (for) و(in) على النحو الآتي:

```
for  element in sequence:
 statements
```

في ما يأتي بيان مُفصَّل لكل عنصر يُمثِّل جزءًا من عملية التعريف بحلقة (for):

- element: مُتغيِّـــر يُعرَف داخل الحلقة، وتوضَع فيه إحدى قِيَم المتتابعة (sequence) التي تُجلَب في كل دورة، وتكون موضوعة بعد هذا المُتغيِّر.
- sequence: سلسلة يريد المُستخدِم الوصول إلى جميع عناصرها.
- statements: جمل موجودة في حلقة (for) التي سيُكرِّر البرنامج تنفيذها في كل دورة.

يُبيِّن الشكل (4-3) تمثيلًا لطريقة عمل حلقات (for) باستخدام مُخطَّط سَيْر العمليات.



الشكل (4-3): مُخطَّط سَيْر العمليات لحلقات (for).

الدالَّة ()range مع حلقات (for)

تُستعمَل الدالَّة ()range لإرجاع سلسلة من الأرقام، تبدأ بالرقم (0) (ما لم يُحدَّد رقم آخر)، وتنتهي برقم مُحدَّد، تزداد ()range بمقدار (1) (ما لم يُحدَّد مقدار آخر للزيادة).

تُستخدَم الدالَّة ()range بثلاث طرائق مختلفة، هي:

- **range(a)**: تُرجِع الدالَّة بهذه الطريقة سلسلة من الأرقام، بَدْءًا بالرقم (0)، وانتهاءً بالرقم (a-1).

مثال:

إذا كانت قيمة (a) هي (5)، فإنَّ الدالَّة ستُرجِع سلسلة الأرقام الآتية: (0)، (1)، (2)، (3)، (4).

- **range(a, b)**: تُرجِع الدالَّة بهذه الطريقة سلسلة من الأرقام، بَدْءًا بالرقم (a)، وانتهاءً بالرقم (b-1).

مثال:

إذا كانت قيمة (a) هي (1)، وقيمة (b) هي (5)، فإنَّ الدالَّة ستُرجِع سلسلة الأرقام الآتية:
(1)، (2)، (3)، (4).

- **range(a, b, c)** تُرجِع الدالَّة بهذه الطريقة سلسلة من الأرقام، بَدْءًا بالرقم (a)، وانتهاءً بالرقم (b-1)، مُتزايِدةً بقفزة مقدارها (c).

مثال:

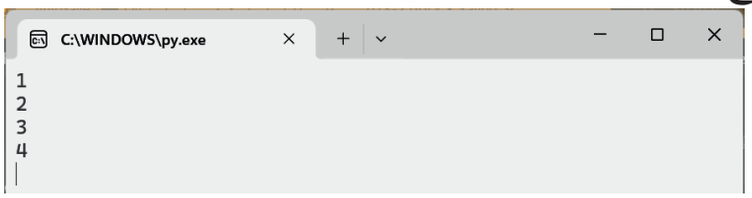
إذا كانت قيمة (a) هي (1)، وقيمة (b) هي (5)، وقيمة (c) هي (2)، فإن الدالَّة ستُرجِع سلسلة الأرقام (1)، (3)؛ ذلك أنَّ السلسلة تبدأ بالعدد (1) والقيمة الابتدائية (a)، ثمَّ تزيد القيمة التالية بمقدار (2)، وتزيد القيمة الابتدائية بمقدار (2)، فتصبح (3). ولا يتمُّ تضمين القيمة (5) في هذه الحالة؛ لأنَّ السلسلة تتوقَّف عند الرقم (1–b)؛ أيْ عند الرقم (4).

مثال:

يطبع البرنامج الآتي القِيَم من 1 إلى 4، حيث القيمة (1) هي قيمة (a) والقيمة (4) هي قيمة (1–b)، وبتزايد (c) مقداره (1):

```
for x in range(1, 5, 1):
    print(x)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط عملي

أُجرِّب وأستنتج:

هل سيختلف الناتج في البرنامج السابق إذا استُخدِم range (1, 5) بدلًا من range (1, 5, 1)؟

إضاءةٌ

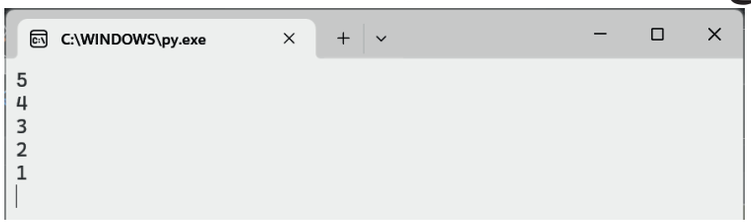
يُمكِن استعمال الدالَّة range(a, b) إذا أراد المُستخدِم أنْ تكون الزيادة بمقدار (1). أمّا إذا أراد المُستخدِم تحديد خطوة مختلفة للزيادة، فيُمكِنه استعمال الدالَّة range(a, b, c) لتحديد قيمة الخطوة (c) . وإذا كانت قيمة الخطوة (c) تساوي (1)، فإنَّ النتيجة لن تختلف؛ لأنَّ range(a, b) تعني ضمنيًّا range (a, b, 1).

مثال:

يُبيِّن البرنامج الآتي استخدام الدالَّة range() في العَدِّ العكسي من (5) إلى (1):

```
for x in range(5, 0, -1):
    print(x)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



إضاءةٌ

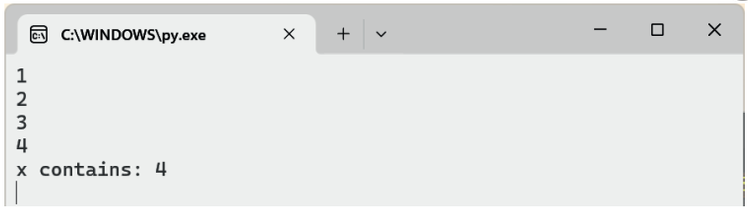
في لغة البرمجة بايثـــون (Python)، يُمكِن الوصول إلى المُتغيِّرات التي عُرِّفت داخل الحلقة من خارج الحلقة.

مثال:

يطبـــع البرنامج الآتي القِيَم من 1 إلى 4، ثمَّ يطبع عبـــارة ("x contains:", x)، حيث (x) هي القيمة النهائية ضمن range():

```
for x in range(1, 5, 1):
    print(x)
print("x contains:",x)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



جملة التحكُّم (break) مع حلقات (for)

تعمل جملة التحكُّم (break) مع حلقات (for) بالطريقة نفســـها التي استُخدِمت فيها مع حلقات (while).

مثال:

يُنفِّذ البرنامج الآتي الحلقة (for)، ويتوقَّف عن ذلك حين تصبح قيمة (x) تساوي (2):

```
for x in range(1, 5, 1):
    print(x)
    if x == 2:
        break
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

نشاط جماعي

أُجرِّب وأستنتج:

في المثال السابق، إذا وُضِعت جملة print() بعد جملة (if)، فماذا سيحدث؟

أتوقَّع ناتج البرنامج، ثمَّ أُنفِّذه باستخدام برمجية بايثون (Python)، وأُلاحِظ الناتج.

جملة التحكُّم (continue) مع حلقات (for)

تُستخدَم جملة التحكُّم (continue) مع حلقات (for) بالطريقة نفسها التي استُخدِمت فيها مع حلقات (while).

مثال:

يطبع البرنامج الآتي الأعداد (1) و(3) و(4) باستخدام جملة التحكُّم (continue) :

```
for x in range(1, 5, 1):
    if x == 2:
        continue
    print(x)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

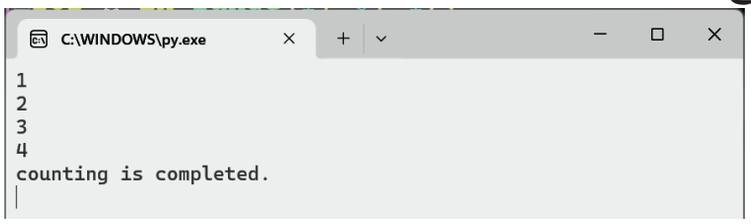
جملة (else) مع حلقات (for)

يُمكِن استعمال جملة (else) مع حلقات (for) لتنفيذ مجموعة من الأوامر عند الخروج من الحلقة.

مثال:

```
for x in range(1, 5, 1):
    print(x)
else:
    print("counting is completed.")
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط جماعي

أُجرِّب وأستنتج:

في المثال السابق، إذا وُضِعت جملة

```
if x == 2: break
```

قبل جملة print(x) في البرنامج، فماذا سيحدث؟ هل سيُنفِّذ البرنامج جملة (else)؟
أتوقَّع ناتج البرنامج، ثمَّ أُنفِّذه باستخدام برمجية بايثون (Python)، وأُلاحِظ الناتج.

حلقات (for) المُتداخِلة (Nested for Loops)

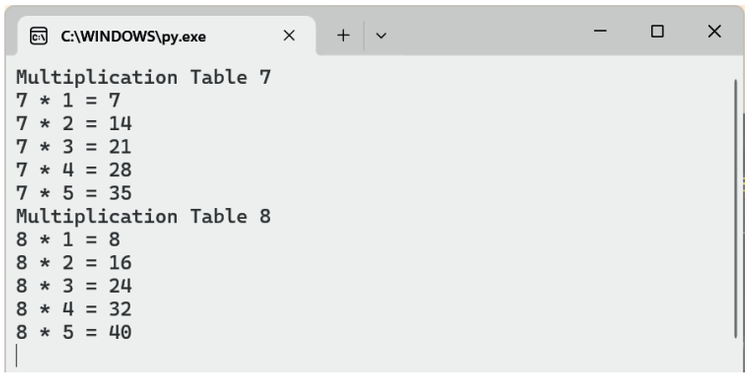
يُمكِن كتابة حلقة (for) في البرنامج داخل حلقة (for) أُخرى، عندئذٍ سيُنفِّذ البرنامج الحلقة الداخلية في كل دورة من دورات الحلقة الخارجية.

مثال:

يطبع البرنامج الآتي العناصر الخمسة الأولى من جدولي الضرب للعددين (8) و(7) باستخدام حلقات (for) المُتداخِلة:

```
for x in range(7, 9):
    print("Multiplication Table", x)
    for y in range(1, 6):
        print(x, "*", y, "=", x * y)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:



نشاط فردي

أتتبَّع المقطع البرمجي السابق، ثمَّ أُدوِّن عدد مَرّات تنفيذ الحلقة الخارجية والحلقة الداخلية.

- أُعدِّل على المقطع البرمجي لطباعة جداول الضرب للأعداد: (7)، (8)، (9)، (10).
- أُنفِّذ المقطع البرمجي في بيئة بايثون (Python)، وأستكشف الأخطاء، ثمَّ أُصحِّحها.

أُقارِن إجاباتي بإجابات زملائي/ زميلاتي في الصف.

إضاءةٌ

إذا أراد المُســـتخدِم كتابة حلقة (for) فارغة (أيْ لا تحتوي على أيِّ جملة)، فإنَّه يضع جملة (pass) داخل هذه الحلقة؛ لكيلا تصله رسالة تفيد بوجود خطأ في البرنامج.

نشاط عملي

أُجرِّب وأستنتج:

أُجرِّب وأستنتج: ما ناتج تنفيذ المقطع البرمجي الآتي مع وجود جملة (pass) ومن دون وجودها؟

```
for x in range(5):
    pass
```

نشاط جماعي

اكتشاف الأخطاء في المقطع البرمجي ضمن لغة البرمجة بايثون (Python)

أُحلِّل – بالتعاون مع أفراد مجموعتي– المقطع البرمجي الآتي، ثمَّ أكتشف الأخطاء الواردة فيه من دون أنْ أُنفِّذه، ثمَّ أكتب المقطع البرمجي الصحيح، وأعمل على تنفيذه:

```
count = 0
for i in range(10)
    if i % 2 = 0:
        while count < 5:
        print("Count is:", count)
        count += 1
else:
print("Done with inner loop")
    if count == 5:
    break
print("Loop ended.")
```

أبحث

أبحثُ في المواقع الإلكترونية الموثوقة في شـــبكة الإنترنت عن اســـتخدامات الحلقات التكرارية في البرمجة، وأهميتها في كتابة البرامج المفيدة في الحياة اليومية.

المواطنةُ الرقميةُ:

- **الأمان الرقمي**: أحرص على عدم تضمين المقطع البرمجي في لغة البرمجة بايثون (Python) أيَّ معلومات شخصية أو مشاركتها في المقطع، وأستخدِم تقنيات البرمجة الآمنة لحماية بياناتي.
- **التعاون والمشـــاركة**: أتعاون مع الزملاء/ الزميلات، وأُشارِكهم في تحليل الأخطاء البرمجية وتصحيحها، وأُشارِك أيضًا في تبادل المعرفة بين أوساط المجتمعات البرمجية الرقمية.

مشروع

المشروع: تصميم برنامج وإعداده لإنشاء لعبة تخمين الأرقام باستخدام لغة بايثون (Python)/ المهمة (4).

أُكمِل – بالتعاون مع أفراد مجموعتي– تنفيذ مشروع التصميم والتطوير للعبة تخمين الأرقام. وكنَّا قد انتهينا في الخطوة الأولى للمشروع من عرض القائمة الرئيسة في اللعبة، ثمَّ استجبنا في الخطوة الثانية لِما يُدخِله اللاعب عن طريق عرض تعليمات اللعبة أو كتابة رسالة بحسب الخيار المُدخَل. ولكنْ، ما الخطوة التي تلي عرض التعليمات وكتابة الرسالة؟

والآن سأعمل – ضمن المجموعة– على تعديل ما كُتِب سابقًا باستخدام الحلقات التكرارية؛ بأنْ أُكرِّر عرض القائمة الرئيسة والخيارات على النحو الآتي:

1- **العودة إلى القائمة الرئيســـة**: سيعود اللاعب إلى القائمة الرئيســـة مَرَّة أُخرى بعد قراءة تعليمات اللعبة بدلًا من انتهاء البرنامج.

2- طلب إدخال خيار صحيح: سيعمل البرنامج على تكرار طلب إدخال خيار صحيح طالما كان الرقم المُدخَل أقل من (1) وأكثر من (3).

أستخدِم حلقة تكرار مُركَّبة لتنفيذ ذلك كما يأتي:

1- **حلقة خارجية**: حلقة لانهائية تعمل على تكرار طباعة القائمة الرئيسة، والطلب إلى اللاعب إدخال أحد الخيارات.

2-**حلقة داخلية**: حلقة تعمل على تكرار طلب الإدخال إذ كان الرقم المُدخَل غير صحيح؛ أيْ أقل من (1) وأكثر من (3).

أتحقَّق – مع أفراد مجموعتي– من كتابة المقطـــع البرمجي بصورة صحيحة من دون وجود أخطاء إملائية أو أخطاء منطقية . كذلك أتحقَّق من أنَّ الحلقات تعمل بصورة صحيحة، ومن تكرار القائمة الرئيســـة عند الحاجة، ثمَّ أحتفظ بما كُتِب – ضمن المجموعة– في ملف حتّى يُمكِن لأفراد المجموعة التعديل عليه في الخطوات القادمة.

أُقيِّمُ تعلُّمي

المعرفة: أُوظِّف في هذا الدرس ما تعلَّمْتُه من معارف في الإجابة عن السؤالين الآتيين:

السؤال الأوَّل: أُقارِن بين حلقات (for) وحلقات (while) باستخدام مُخطَّط سَيْر العمليات لكلٍّ منهما.

السؤال الثاني: ما الطرائق الثلاث التي يُمكِن بها استخدام الدالَّة ()range ؟

المهـــارات: أُوظِّف مهارة التفكير الناقد والمهارات البرمجية والتحليل في الإجابة عن الســـؤالين الآتيين:

السؤال الأوَّل: أكتب برنامجًا لإيجاد مضروب الأعداد من (1) إلى (10) باستخدام حلقات (for).

السؤال الثاني: اقرأ البرنامج التالي المكتوب بلغة البرمجة بايثون (Python)، ثمَّ أُجيب عمّا يأتي:

1. ما الهدف الرئيس من البرنامج؟

---

2. أَصِف ما يقوم به البرنامج عامةً من دون وصف وظيفة أيِّ أمر برمجي فيه.

```
x = 1
while x <= 6:
    y = 1
    while y <= 6:
        print(x, "*", y, "=", x*y)
        y += 1
    x += 1
```

القِيَمُ والاتجاهاتُ

أبحث في شـــبكة الإنترنت عن تقنيات البرمجة الآمنة وطرق حماية البيانات عند مشاركة البرامج عبر مجتمعات البرمجة وألخص ما أتوصل إليه في Google Docs وانشـــره عبر الموقع الإلكتروني للمدرسة.

الدرسُ الخامس

القوائم
(Lists)

مُنتَجاتُ التعلُّم
(Learning Products)

تعديل البرنامج باستخدام قوائم اللعبة في توليد الأرقام العشوائية، ضمن سياق تصميم لعبة تخمين الأرقام باستخدام برمجية بايثون (Paython).

الفكرةُ الرئيسةُ:

تعرُّف كيف يُمكِن التعامل مع القوائم وسلاسل الحروف في برمجية بايثون (Python)، واستخدامها في تخزين مجموعات البيانات المُترابِطة وإدارتها وتنفيذ عمليات فيها.

المفاهيم والمصطلحات:

القائمة (List)، سلسلة الحروف (String)، موقع العنصر في القائمة (أو الحرف في السلسلة) (Index)، إلصاق القوائم أو السلاسل (Concatenation)، القابلية للتغيير (Mutability)، القائمة المُركَّبة (Nested List)، المصفوفة ثنائية الأبعاد (2D Matrix).

نتاجات التعلُّم (Learning Objectives):

- أُعرِّف مفهوم المُتغيِّر (قائمة)، وأُبيِّن استخداماته في البرمجة.
- أُنشِئ قوائم على اختلاف أنواعها (مُتسلسِلة، ومُتغيِّرة، ومُركَّبة)، وأستخدِمها في تخزين مجموعة مُتنوِّعة من القِيَم.
- أُحدِّد كيف يُمكِن تجميع مجموعة القِيَم في قائمة واحدة.
- أُوضِّح أنواع القوائم (المُتسلسِلة، والمُتغيِّرة، والمُركَّبة)، وأُنفِّذ عمليات مختلفة فيها، مثل: الإضافة، والحذف.
- أستخدِم الدوالَّ البرمجية الجاهزة في لغة البرمجة بايثون (Python) لمعالجة القوائم وإجراء بعض العمليات الأساسية فيها.

تُعَـــدُّ القوائم إحدى المزايا المُهِمَّة للغات البرمجة؛ إذ تُســـهِم في إدارة تنظيـــم البيانات، وتتيح للمُستخدِم التعامل معها واسترجاعها بسهولة. ولكنْ، كيف يُمكِن توظيف القوائم في لغة البرمجة بايثون (Python)؟ وهل يختلف ذلك عن توظيفها في لغات البرمجة الأُخرى؟

نشاط تمهيدي

أفترض أنَّني أُريد إنشـــاء برنامج يتولّى قراءة علامـــات (100) طالب في أحد الامتحانات، ثمَّ يعرض هذه العلامات مُرتَّبة بصورة تصاعدية. أُناقِش أفراد مجموعتي في الأسئلة الآتية بناءً على ما تعلَّمْتُه عن لغة البرمجة بايثون (Python):

1. هل يجب علـــيَّ تخزين علامات هؤلاء الطلبة أم يمكنني كتابـــة هذا البرنامج دون الحاجة لتخزين العلامات؟
2. إذا تعيَّن علـــيَّ تخزين علامات هؤلاء الطلبة، فإلى كم متغيرٍ أحتـــاج؟ وما التحدِّيات التي سأُواجِهها عند محاولتي تخزين العلامات؟
3. كيف يُمكِنني طباعة العلامات وترتيبها تصاعديًّا؟ وهل مـــا تعلَّمْتُه عن لغة البرمجة بايثون (Python) سيساعدني على ذلك؟

القوائم (Lists)

تُمثِّل القائمة في لغة البرمجة بايثون (Python) عددًا من القِيَم التي يُخزَّن بعضها مع بعض، وترتبط معًا بمعنى وظيفي مشترك. فمثلًا، يُمكِن للمُســـتخدِم تخزين أسماء الشوارع داخل إحدى المدن في قائمة، وكذلك تخزين أسعار البضائع التي اشتراها أحد العملاء، أو تخزين علامات الطلبة في الصف الحادي عشر.

يتطلَّب استخدام القوائم في لغة البرمجة بايثون (Python) تعريف القائمة أوَّلًا، ثمَّ تخزين العناصر داخلها.

يُمكِن تعريف إحدى القوائم باســـتخدام الأقواس المُربَّعة [ ]، والفصل بين عناصر القائمة بفواصل على النحو الآتي:

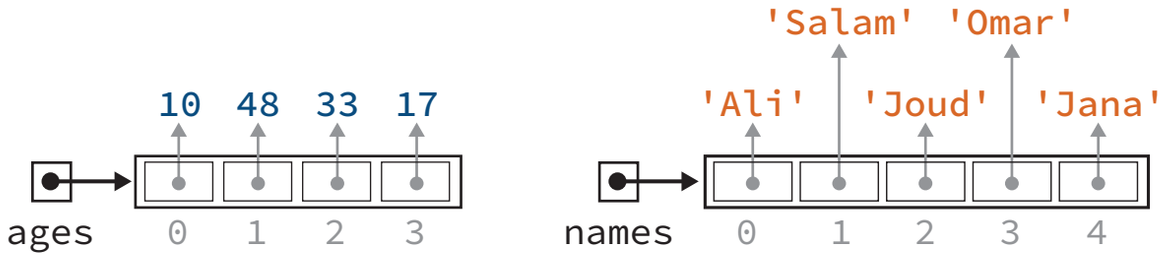
المقطع البرمجي التالي يعرّف قائمة باســـم List name تحتوي على أربعة عناصر، وكل عنصر في القائمة يحتوي على قيمة ثابتة واحدة تتكرر، وهي 'value 1'

```
mylist = ['value1','value1','value1','value1']
```

مثال:

يُبيِّن المقطع البرمجي الآتي تعريفًا لقائمة تُسمّى (names)، وتتضمَّن (5) أسماء، وتعريفًا لقائمة أُخرى تُسمّى (ages)، وتتضمَّن (4) أعمار لأشخاص وتعريفا لقائمة فارغة، أنظر الشكل (5-1).

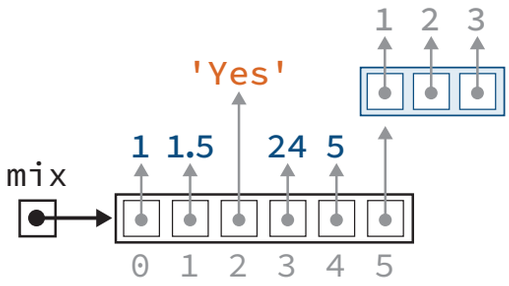
```
names = ['Ali', 'Salam', 'Joud', 'Omar', 'Jana']
ages = [10, 48, 33, 17]
a = []
```



الشكل (5-1): ترتيب عناصر قائمتين في ذاكرة برنامج.

لا يُشترَط في القائمة التي يراد إنشاؤها أنْ تحوي جميعها عناصر من النوع نفسه؛ إذ يُمكِن للمُستخدِم تخزين أرقام صحيحة، وأرقام عشرية، وسلاسل نصية، وقوائم أُخرى في القائمة نفسها كما في المثال الآتي:

```
mix = [1, 1.5, 'Yes', 24, 5, [1, 2, 3]]
```



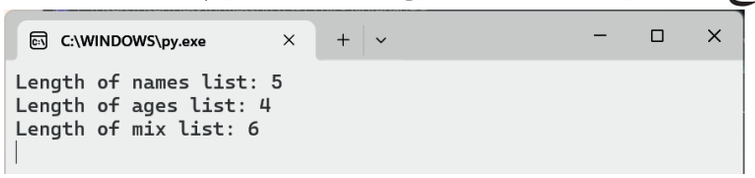
أُلاحِظ من المثال السابق أنَّ طول القائمة هو (6) عناصر، بالرغم من أنَّ بعض هذه العناصر مُركَّبة من عناصر أخرى.

يُمكِن للمُستخدِم معرفة طول أيِّ قائمة باستخدام الدالة len ():

```
names = ['Ali', 'Salam', 'Joud', 'Omar', 'Jana']
ages = [10, 48, 33, 17]
mix = [1, 1.5, 'Yes', 24, 5, [1, 2, 3]]

print("Length of names list:", length_of_names)
print("Length of ages list:", length_of_ages)
print("Length of mix list:", length_of_mix)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة الحاسوب:



نشاط عملي

أُجرِّب وأستكشف:

أعمل – بالتعاون مع أفراد مجموعتي– على استخدام لغة البرمجة بايثون (Python)، لإنشاء قائمة تضمُّ أسماء طلبة المجموعة، وقائمة أُخرى تُبيِّن هواياتهم المُفضَّلة. بعد ذلك أكتب الأوامر البرمجية، وأتحقَّق من صحتها عبر طباعة طول كل من القوائم وطباعة أسماء الطلاب مع هواياتهم المفضلة.

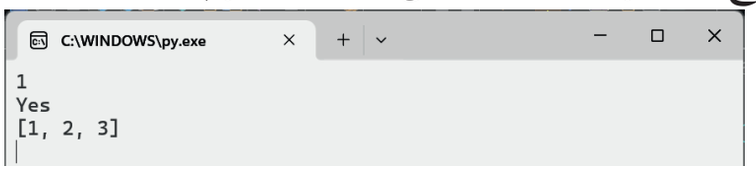
الوصول للعناصر في القائمة

يُمكِن للمُستخدِم الوصول إلى أيِّ عنصر من عناصر القائمة باستخدام الأقواس المُربَّعة ورقم يُمثِّل موقع العنصر (index) في القائمة على النحو الآتي:

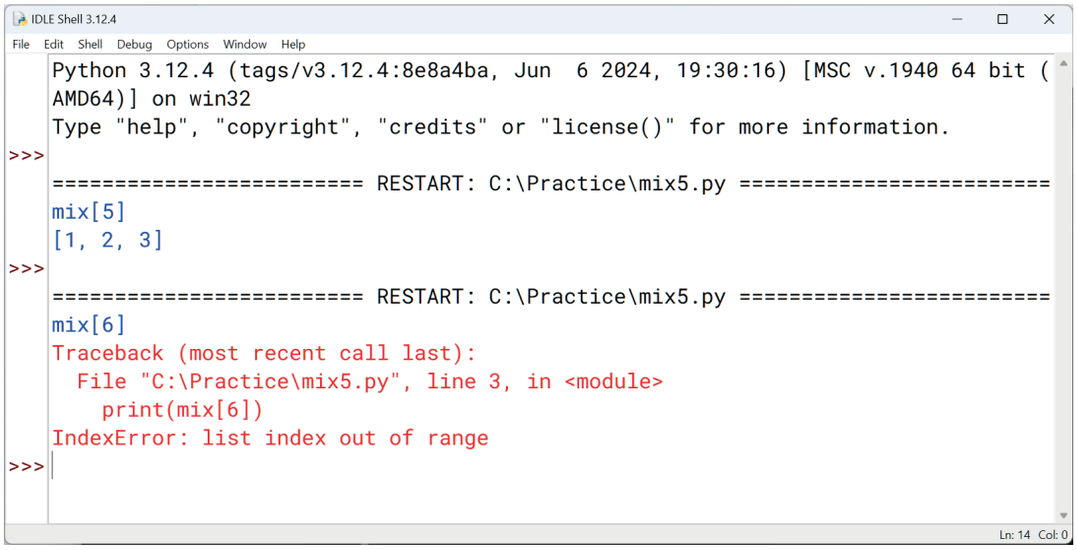
```
mix = [1, 1.5, 'Yes', 24, 5, [1, 2, 3]]

print(mix[0])
print(mix[2])
print(mix[5])
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة الحاسوب:

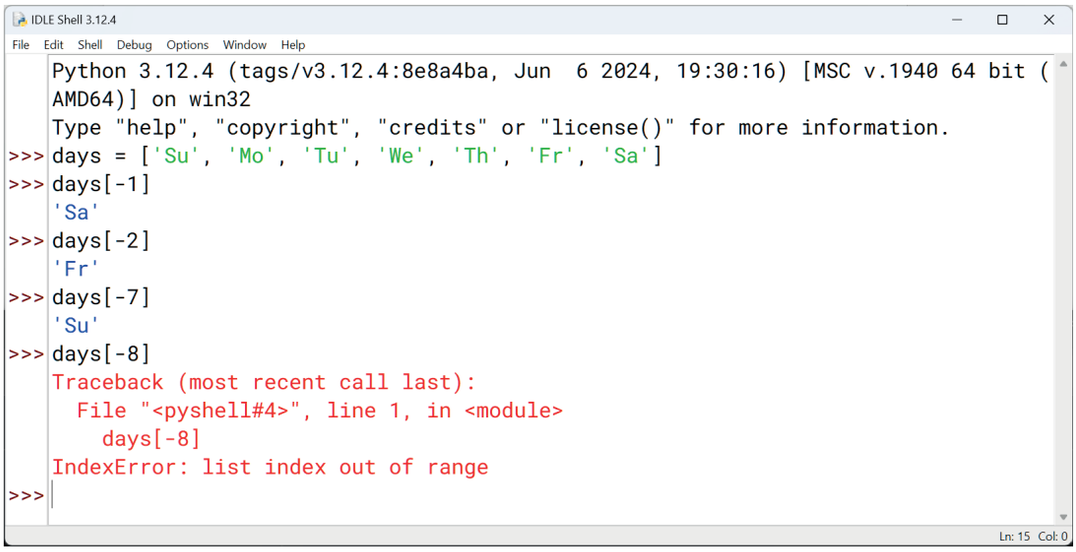


أُلاحِـــظ أنَّ ترقيم مواقع العناصر قد بدأ بالرقم (0)، لا بالرقم (1). وهذا يعني أنَّ قائمة من ســـتة عناصر (مثل mix) سيكون أخر عنصر فيها في الموقع (5) لا في الموقع (6). وإذا حاول المُستخدِم استعمال رقم أعلى من (5)، فإنَّ ذلك سيؤدّي إلى توقُّف البرنامج عن العمل، أنظر الشكل (5-2).



الشكل (5-2): ناتج البرنامج بعد تحديد موقع عنصر من خارج النطاق.

كذلك تتيح لغة البرمجة بايثون (Python) للمُســـتخدِم استعمال أرقام سالبة للوصول إلى العناصر، حيث يرمز الرقم (1-) إلى العنصر الأخير، ويرمز الرقم (2-) إلى العنصر قبل الأخير وهكذا، أنظر الشكل (5-3).



الشكل (5-3): استخدام الأرقام السالبة في الوصول إلى العناصر.

نشاط عملي

- أكتب المقطع البرمجي اللازم لتعريف قائمة (months)، ثمَّ أُضيف العناصر إليها.
- أستخدِم موقع العنصر للوصول إلى العنصر ('May').
- أستخدِم الأعداد السالبة للوصول إلى العنصر ('Jun').
- أُنفِّذ المقطع البرمجي للتحقُّق من صحته، وأتتبَّع الأخطاء، وأعمل على تصحيحها.

إضاءةٌ

استخدام الرقم (1-) يُقلِّل من نسبة حدوث الخطأ مقارنةً باستخدام موقع العنصر الأخير، مثل الرقم (6) في القائمة (days)؛ فالرقم (1-) يشير دائمًا إلى العنصر الأخير بِغَضِّ النظر عن طول القائمة، ويُسهِّل على الجميع فهم العنصر المقصود خلافًا للرقم (6) مثلًا؛ فهذا الرقم يتطلَّب من قارئ البرنامج معرفة أنَّ طول القائمة هو (7)، واستنتاج أنَّ الرقم (6) يُمثِّل موقع العنصر الأخير.

قد يَلزم في بعض البرامج المرور على جميع العناصر في القائمة لأداء وظيفة ما، كما هو الحال في البرنامج الذي يُبيِّنه الشكل (5-4)، والذي يطبع على سطر مُنفصِل كل عنصر من عناصر القائمة المُسمّاة (items).

```
1 for e in items:
2     print(e)
```

الشكل (5-4): طباعة كل عنصر على سطر مُنفصِل في القائمة (items).

كذلك يُمكِن كتابة البرنامج السابق باستخدام نوع آخر من حلقات التكرار، هو حلقة (while)، أنظر الشكل (5-5).

```
1 length = len(items)
2 i = 0
3 while i < length:
4     print(items[i])
5     i += 1
```

الشكل (5-5): طباعة كل حرف على سطر مُنفصِل في القائمة (items).

يُلاحَظ على هذا البرنامج ما يأتي:

1. استعمال المُتغيِّر (i) للمرور على جميع مواقع العناصر في القائمة؛ إذ بدأت قيمة هذا المُتغيِّر عند الرقم (0)، وهو موقع العنصر الأوَّل، وانتهت قيمته عند آخر موقع عنصر في القائمة.
2. الحاجة لمعرفة عدد المواقع التي سنقوم بالمرور عليها؛ ما ألزم استخدام len(items) التي تُبيِّن طول القائمة المُخزَّنة في (items).
3. استخدام شرط حلقة التكرار يتأكد من عدم وصول قيمة المتغير i لطول القائمة، وذلك لأنه لو كان طول القائمة 10 على سبيل المثال فإن آخر عنصر في القائمة موجود في الموقع 9 وليس في الموقع 10.

مثال:

إذا افترضْتُ أنَّ القائمة (found) في الشكل (5-6) تتضمَّن قراءات لجهاز استشعار، وأنَّ القراءات المتتابعة والمتساوية تدلُّ على وجود خطأ ما في القراءة، فإنَّه يُمكِنني التحقُّق من وجود عناصر متتابعة ومتساوية في القائمة كما يأتي:

```
found = False
for i in range(len(readings) - 1):
    if readings[i] == readings[i+1]:
        found = True
        break

print(found)
```

الشكل (5-6): مثال على قوائم العناصر المُتكرِّرة.

في هـــذا المثال، عُقِدت مقارنة بين كل عنصر والعنصر الذي يليـــه؛ فلكل قيمة من قِيَم (i)، تمَّت مقارنة العنصر الموجود في الموقع (i) بالعنصر الموجود في الموقع الذي يليه (i+1).

نشاط

أُحلِّل وأستنتج: استنادًا إلى المثال الوارد في الشكل (5-6):

- ما ســـبب استخدام جملة التحكُّم (break) في السطر (5)؟ وما تأثير حذفها في ناتج تنفيذ البرنامج؟
- مـــا دلالـــة اســـتخدام الجملـــة range(len(readings)-1)؟ وما تأثيـــر تعديلها إلى range(len(readings)) في ناتج تنفيذ البرنامج؟

أُناقِش زملائي/ زميلاتي ومُعلِّمي/ مُعلِّمتي في إجاباتي.

إضاءةٌ

الخطأ المعروف باســـم "off-by-one error"، أو باســـم الخطأ بخطوة واحدة، هو من أكثر الأخطاء البرمجية شـــيوعًا؛ قد يُخطِئ المُبرمِج عند تحديد شـــرط نهاية الدوران في حلقة التكرار (while)، أو عند تحديد المدى فـــي حلقة التكرار (for)؛ ما يدفع البرنامج إلى تنفيذ دورة واحدة أقل من المطلوب أو أكثر منه.

إثراء

يوجد نمط برمجي يستخدم متغير يلعب دور الراية (flag)، ويعتمد في كتابة البرامج على ما يأتي:

1- البَدْء بافتراض نتيجة بحث.

2- المرور على القائمة للتحقُّق من صحة الفرضية.

3- طباعة النتيجة بعد الانتهاء من المرور على القائمة.

في المثال السابق، أُطلِق على المُتغيِّر (found) اسم الراية (flag) التي تُرفَع أو تُنزَل عند اكتشاف وجود صفة ما أثناء عملية المرور. وهذا النمط في كتابة البرامج مفيد في عملية البحث، وله استخدامات كثيرة.

نشاط

أُناقِش وأُجرِّب:

كيف يُمكِن المرور على عناصر قائمة بالعكس؟ أُناقِش زملائي/ زميلاتي ومُعلِّمي/ مُعلِّمتي في هذا السؤال، ثمَّ أُجرِّب تنفيذ ذلك عمليًّا في بيئة بايثون (Python).

العمليات في القوائم

توجد عمليات عِدَّة يُمكِن تنفيذها في القوائم، مثل: الوصول إلى العناصر في قائمة ما، وإضافة عناصر جديدة إليها، وحذف عناصر منها، وترتيب العناصر فيها. وقد نُفِّذت بعض هذه العمليات بصورة فعلية في الأمثلة السابقة لهذا الدرس.

نشاط

أُجرِّب وأُناقِش: أُجرِّب تنفيذ كل جملة من الجمل الآتية:

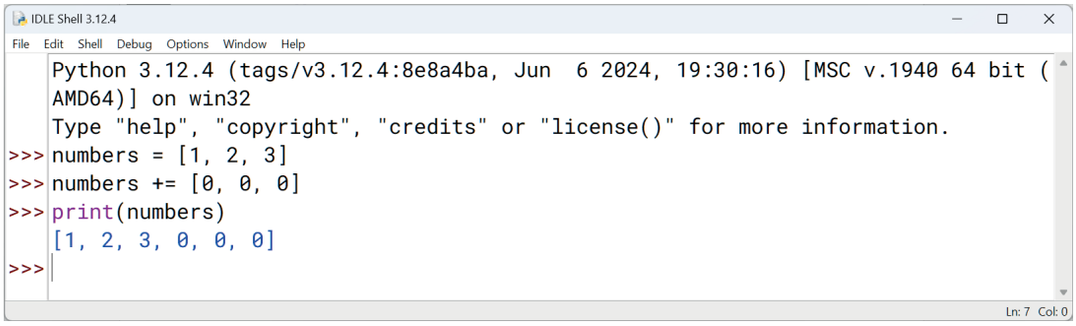
```
[1, 2, 3] + [98, 99, 100]
[1, 2, 3] + 5
[1, 2, 3] + [5]
[1, 2, 3] * 4
[1, 2, 3] * [1, 2, 3]
```

أُناقِش أفراد مجموعتي في السؤالين الآتيين:

- أيُّ الجمل يُمكِن تنفيذها؟
- أيُّ الجمل لا يُمكِن تنفيذها؟

بناءً على ذلك، أُناقِشهم في معنى الجمع والضرب بالنسبة إلى القوائم.

عند استخدام عامل الجمع (+) بين قائمتين، تنتج قائمة جديدة تحوي عناصر القائمتين معًا. ومن ثَمَّ يُمكِن استعمال العامل (=+) لتعديل قائمة، بإضافة عناصر قائمة أُخرى إلى نهايتها كما يأتي:



كذلك يُمكِن استخدام هذا العامل في إنشاء قائمة خطوة بخطوة كما في الشكل (5-7)؛ حيث يقوم البرنامج الموضح في المثال بقراءة أرقام وإضافتها إلى قائمة ويتوقف عند إدخال أي رقم سالب.

```
numbers = []
n = int(input("Enter a positive number: "))
while n >= 0:
    numbers += [n]
    n = int(input("Enter a positive number: "))
print(numbers)
```

الشكل (5-7): مثال على قراءة الأرقام وإضافتها إلى إحدى القوائم.

أُلاحِظ على هذا البرنامج أنَّه قد بدأ بقائمة فارغة، ثمَّ أُضيفت العناصر تِباعًا باستخدام العامل (=+). أُلاحِظ أيضًا أنَّ الرقم (n) أُحيطت به الأقواس المُربَّعة [n]؛ لأنَّ العامل (=+) يؤدّي وظيفته بين قائمتين؛ ما يجعل الجمل، مثل جملة numbers += n (من دون أقواس)، غير صحيحة بحسب قوانين لغة بايثون.

إضافةً إلى العامل + والعامل * (يعمل على تكرار قائمة ما عددًا من المَرّات)، يُمكِن أيضًا المقارنة بين القوائم باستخدام عوامل المقارنة المنطقية (== و =! و < و => و > و =<)، التي تعمل على

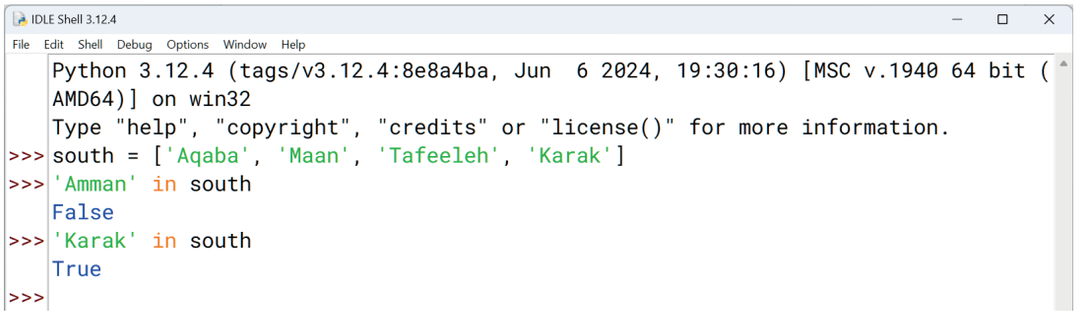
مقارنة كل عنصر في القائمة الأولى بالعنصر الذي يُقابِله في القائمة الثانية.

مثال:

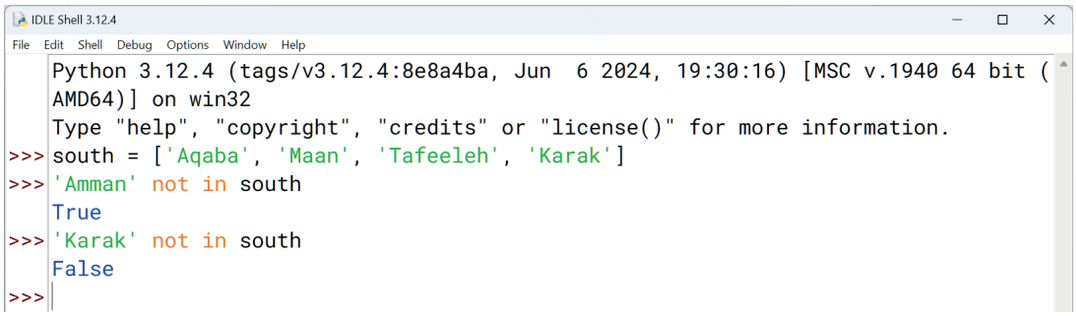
القائمة [1, 2, 3]> مـــن القائمة [3, 1]؛ لأنَّ العنصر الأوَّل(1) في القائمة الأولى أقل من العنصر الأوَّل في القائمة الثانية (3).

في حين أنَّ القائمة [1, 2, 3]> من القائمة [1, 0, 0, 0]؛ لأنَّ العنصر الثاني (2) في القائمة الأولى أكبر من العنصر الثاني (0) في القائمة الثانية.

كذلك يُمثِّل التحقُّق من وجود عنصر ما في القائمة واحدةً من العمليات المُهِمَّة في القوائم. وأسهل طريقة لعمل ذلك هي استخدام العامل (in) على النحو الآتي:



يُمكِن أيضًا استخدام العامل (not in) في التحقُّق من عدم وجود عنصر ما في القائمة كما يأتي:



أبحث

أبحث في المواقع الإلكترونية الموثوقة في شبكة الإنترنت عن استخدامات العوامل الحسابية والعوامل المنطقية في التطبيقات العملية الحياتية للغـــة البرمجة بايثون (Python)، ثمَّ أُدوِّن ما أتوصَّل إليه من نتائج، وأُشارِكها مع زملائي/ زميلاتي في الصف.

نشاط عملي

أُجرِّب وأستكشف:

- أعرّف قائمتين فارغتين ثم أضيف خمسة أعداد إلى كل منهما.

- أستخدم عوامل المقارنة لطباعة عناصر القائمة الأصغر.
- أستخدِم معامل الجمع في ضَمِّ عناصر القائمتين وطباعتهما.

أُنفِّذ البرنامج، ثمَّ أتحقَّق من صحته، وأعمل على استكشاف الأخطاء وتصحيحها.

الدوالُّ الجاهزة لمعالجة القوائم

تعرَّفْتُ سابقًا أنَّه يُمكِن استخدام len(...) في تحديد طول قائمة ما، والآن سأتعرَّف أنَّ العديد من الدوالِّ المُشابِهة تُوفِّرها اللغة للتعامل مع القوائم، أنظر الجدول (5-1) الذي يعرض أمثلة على ذلك، علمًا بأنَّ جميع هذه الأمثلة مُطبَّقة على القائمة: numbers = [1, 3, 5, 3, 2, 4].

الجدول (5-1): أمثلة على بعض الدوالِّ الجاهزة لمعالجة القوائم.

الشرح	المثال
تعمل max(...) على إيجاد أكبر عنصر في القائمة بافتراض أنَّ جميع العناصر هي من النوع نفسه.	`>>> max(numbers)` `5`
تعمل min(...) على إيجاد أصغر عنصر في القائمة بافتراض أنَّ جميع العناصر هي من النوع نفسه.	`>>> min(numbers)` `1`

تعمل sorted(...) على إعادة نسخة مُرتَّبة من القائمة (القائمة الأصلية تظلُّ من دون تعديل).	`>>> result = sorted(numbers)`
	`>>> print(result)`
	`[1, 2, 3, 3, 4, 5]`
	`>>> print(numbers)`
	`[1, 3, 5, 3, 2, 4]`

تعمل count(...) على عَدِّ عدد مَرّات تكرار عنصر ما في القائمة.	`>>> numbers.count(3)`
	`2`
	`>>> numbers.count(7)`
	`0`

تعمل index(...) على إيجاد موقع أوَّل ظهور لعنصر ما في القائمة.	`>>> print(numbers)`
	`[1, 3, 5, 3, 2, 4]`
	`>>> numbers.index(5)`
	`2`
	`>>> numbers.index(3)`
	`1`

يتبيَّن من الجدول السابق وجود طريقتين مختلفتين لاستخدام الدوالِّ الجاهزة:

- الطريقة الأولى تُستخدَم في (min)، و(max)، و(sorted)، و(len)، و(print)، وتتمثَّل في إرسال القائمة إلى الدالة.
- الطريقة الثانية تُستخدَم في (count)، و(index)، وتتمثَّل في استدعاء الوظيفة عن طريق ذكر اسم القائمة متبوعًا بنقطة، ثمَّ ذكر اسم الدالة.

يُذكَر أنَّ الدوال في كلتا الطريقتين لا تقوم بإجراء إجراء أيِّ تعديل على القائمة، وإنَّما تكتفي بالمرور على القائمة لحساب نتيجة ما، ثمَّ إعادة هذه النتيجة. غير أنَّ ذلك ليس مُطَّرِدًا في جميع الدوال، كما هو الحال في تلك الواردة في الجدول (5-2)؛ إذ تعمل جميعها على تعديل القائمة.

الجدول (5-2): أمثلة على وظائف جاهزة لمعالجة القوائم وتعديلها.

الشرح	المثال

تعمل sort() على ترتيب عناصر القائمة (تُعدَّل القائمة نفسها بدلًا من إرجاع نسخة مُرتَّبة كما في sorted(...)).	`>>> numbers = [1, 3, 5, 3, 2, 4]`
	`>>> numbers.sort()`
	`>>> print(numbers)`
	`[1, 2, 3, 3, 4, 5]`

تعمل reverse() على عكس ترتيب العناصر في القائمة.	`>>> numbers = [1, 3, 5, 3, 2, 4]`
	`>>> numbers.reverse()`
	`>>> print(numbers)`
	`[4, 2, 3, 5, 3, 1]`

تعمل append(...) على إضافة عنصر في آخر القائمة.	`>>> numbers = [1, 3, 5, 3, 2, 4]`
	`>>> numbers.append(99)`
	`>>> print(numbers)`
	`[1, 3, 5, 3, 2, 4, 99]`

تعمل remove(...) على حذف أوَّل ظهور لعنصر في القائمة.	`>>> numbers = [1, 3, 5, 3, 2, 4]`
	`>>> numbers.remove(3)`
	`>>> print(numbers)`
	`[1, 5, 3, 2, 4]`
	`>>>`

إضاءةٌ

من المفيد جدًّا تعرُّف أهمِّ الدوال الجاهزة التي تُوفِّرها اللغة، لكنَّ معظم المُبرمِجين المحترفين لا يحفظون جميع هذه الدوال، وإنَّما يعودون مِرارًا إلـــى الموقع الإلكتروني للغة البرمجة بايثون (Python)؛ للبحث عن الدوال المناسبة لبرامجهم، أو للتحقُّق من كيفية استخدام بعض هذه الدوال.

أتعرَّف الوظائف الجاهزة التي لها تعلُّق بالقوائم عن طريق الرابط الإلكتروني الآتي:

https://docs.python.org/3/tutorial/datastructures.html

أو عن طريق مسح الرمز سريع الاستجابة (QR Code) المجاور.

نشاط عملي

أُجرِّب وأستكشف:

- أستخدِم الدوالَّ الجاهزة في طباعة أكبر عنصر وأصغر عنصر في القائمتين اللتين أنشأْتُهما في النشاط السابق.
- أستخدِم الدوالَّ الجاهزة في طباعة عناصر القائمة الأولى بشكل عكسي.

يوجد تشابه بين سلاســـل الحروف والقوائم، يتمثَّل في أنَّ كلتيهما تتألَّف من عناصر متتابعة. غير أنَّ جميع العناصر في السلســـلة هي حروف (أيْ رموز تُمثِّل حروفًا أبجديةً، أو أرقامًا، أو علاماتِ ترقيم، أو غيرَ ذلك)؛ لذا يُمكِنني تطبيق معظم ما تعلَّمْتُه عن القوائم على سلاسل الحروف.

نشاط فردي

أُجرِّب وأستكشف:

أُعرِّف سلسلة الحروف `name = 'Amina-Abdu'`، ثمَّ أُجرِّب تنفيذ الجمل الآتية:

```
name[0]
sorted(name)
max(name)
name += 'L'
name * 2
name < 'Zaid'
'Abd' in name
name[5] = 'h'
name.sort()
```

- أيُّ هذه الجمل تمكَّنْتُ من تنفيذها؟
- أيُّ هذه الجمل لم أتمكَّن من تنفيذها؟
- ما العامل المشترك بين الجمل التي لم أتمكَّن من تنفيذها؟

يُمكِن الوصول إلى أيِّ حرف في السلسلة باستخدام الموقع (index)، ويُمكِن أيضًا استخدام الدوالِّ، مثل: sorted() و max() و len() و count(). وكذلك استخدام العوامل، مثل: + و* كما تعلَّمْتُ سابقًا، فضلًا عن إمكانية اختبار وجود عنصر في السلسلة باستخدام العامل (in) والعامل (not in)، بالرغم من وجود فروق بســـيطة بينها؛ إذ تتحقَّق العوامل في القوائم من وجود عنصر ما في إحدى القوائم، في حين تتحقَّق العوامل في السلاسل من وجود سلسلة أُخرى داخل السلسلة نفسها.

مثال:

يتحقَّق البرنامج الآتي من وجود السلسلة ('.gov.jo') ضمن السلسة (url):

```
url = input("ما موقعك الإلكتروني؟")
if '.gov.jo' in url:
    print('يبدو أنك أدخلت موقعاً حكومياً أردنياً')
else:
    print('شكراً جزيلاً')
```

الدوالُّ الجاهزة الخاصة بسلاسل الحروف

توجد دوالُّ جاهزة تختصُّ بسلاسل الحروف، أنظر الجدول (5-3) الذي يعرض أمثلة على ذلك، علمًا بأنَّ جميع هذه الأمثلة مُطبَّقة على السلسلة: text = 'Hello there'.

الجدول (5-3): أمثلة على بعض الدوالِّ الجاهزة لسلاسل الحروف.

الشرح	المثال

تعمل upper() على إنشاء نسخة من السلسلة، تحوي الحروف نفسها، ولكنْ بعد تحويل الأحرف الصغيرة إلى أحرف كبيرة.	`>>> text = 'Hello there'`
	`>>> text.upper()`
	`'HELLO THERE'`

تعمل lower() على إنشاء نسخة من السلسلة، تحوي الحروف نفسها، ولكنْ بعد تحويل الأحرف الكبيرة إلى أحرف صغيرة.	`>>> text.lower()` `'hello there'`

تعمل replace(a, b) على إنشاء نسخة من السلسلة بعد استبدال كل ظهور لـa بـb.	`>>> text.replace('Hello', 'Hi')`
	`'Hi there'`
	`>>> text.replace('e', 'X')`
	`'HXllo thXrX'`

تعمل isalpha() على التأكُّد أنَّ جميع حروف السلسلة هي حروف أبجدية، ومن ثَمَّ كانت نتيجة الاستدعاء الأوَّل هي (False)؛ نظرًا إلى وجود فراغ بين الكلمتين في السلسة، خلافًا للاستدعاء الثاني؛ إذ كانت نتيجته (True)؛ لأنَّ السلسلة تحوي فقط حروفًا أبجديةً.
تعمل isnumeric() على التأكُّد أنَّ السلسلة تحوي فقط أرقامًا، في حين تعمل islower() وisupper() على التأكُّد أنَّ السلسلة تحوي فقط حروفًا أبجديةً صغيرةً أو حروفًا كبيرةً.
	`>>> text.isalpha()`
	`False`
	`>>> text = "HelloThere"`
	`>>> text.isalpha()`
	`True`
	`>>> text.isnumeric()`
	`False`
	`>>> text = "HELLO"`
	`>>> text.islower()`
	`False`
	`>>> text.isupper()`
	`True`
	`>>>`

إثراء

توجد عمليات أُخرى تُوفِّرها اللغة للتعامل مع السلاسل، ويُمكِنني تعرُّفها عن طريق الموقع الإلكتروني الرسمي للغة البرمجة بايثون (Python):

https://docs.python.org/3/library/stdtypes.html#string-methods

أو عن طريق مسح الرمز سريع الاستجابة (QR Code) المجاور.

تتمثَّل أهمُّ الفروق بين السلاســـل والقوائم في أنَّ السلاسل في لغة البرمجة بايثون (Python) غير قابلة للتغيير (Immutable)، وأنَّ القوائم قابلة للتغيير (Mutable). ولهذا لا يُمكِن -مثلًا- تغيير أحد الحروف في السلسلة، في حين يُمكِن تغيير عنصر ما في القائمة، أنظر الشكل (5-8).

```
numbers = [0, 1, 2]
numbers[0] = 9
print(numbers)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة الحاسوب:

```
name = 'Jana'
name[0] = 'D'
print(name)
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة الحاسوب:

```
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    name[0] = 'D'
TypeError: 'str' object does not support item assignment
```

الشكل (5-8): مثال يُبيِّن الفرق بين السلاسل والقوائم.

أُلاحِظ أنَّ جميع العمليات في الشكل السابق لا تُعْنى بتعديل السلسلة، وإنَّما تُعْنى بإنشاء نسخة جديدة مُعدَّلة منها. فمثلًا، عند تنفيذ الجمل الآتية، فإنَّ السلسلة لن تتغيَّر:

```
>>> text = 'Hello there'
>>> text.replace('Hello', 'Hi')
'Hi there'
>>> text.upper()
'HELLO THERE'
>>> print(text)
Hello there
>>>
```

وهذا يتأكَّد عند استدعاء () replace و () upper على السلسلة؛ إذ تُنشَأ نسخ جديدة مُعدَّلة. ونظرًا إلى عدم حفظ هذه النسخ أو طباعتها؛ فإنَّ السلسلة لم تتأثَّر بهذه العمليات. وبالرغم من ذلك، يُمكِن حفظ النسخ المُعدَّلة للسلسلة كما في الجمل الآتية:

```
>>> text = 'Hello there'
>>> text = text.replace('Hello', 'Hi')
>>> text = text.upper()
>>> print(text)
HI THERE
>>>
```

كذلك يُمكِن المرور على عناصر القوائم المُركَّبة، وتطبيق بعض المهام عليها كما في القوائم البسيطة.

مثال:

عند إنشاء حساب جديد في موقع إلكتروني، فإنَّ هذا الموقع يتحقَّق من (جودة) كلمة السِّرِّ؛ سعيًا لتقليل خطر اكتشافها من طرف المُخترِقين.

يتضمَّن هذا المثال إنشاء برنامج يعمل على استيفاء كلمة السِّرِّ للشرطين الآتيين:

- اشتمال كلمة السِّرِّ على (10) أحرف فأكثر.
- احتواء كلمة السِّرِّ على أحرف إنجليزية صغيرة، وأحرف إنجليزية كبيرة، وأرقام، ورموز غير الأحرف والأرقام.

يبدأ البرنامج المرور على أحرف كلمة السِّرِّ، وعَدَّ مَرّات تكرار كلٍّ من الأحرف الأبجدية (الصغيرة والكبيرة) والأرقام والرموز، ثمَّ يتأكَّد أنَّ عدد مَرّات التكرار لكل ما سبق لا يساوي صفرًا، أنظر الشكل (5-9).

```
psw = input("Enter Password: ")
small = 0
capital = 0
number = 0
special = 0

for c in psw:
    if c.islower():
        small += 1
    elif c.isupper():
        capital += 1
    elif c.isdigit():
        number += 1
    else:
        special += 1

if len(psw) < 10:
    print("Password must be >= 10 characters long.")

if small == 0:
    print("Password must contain small letters.")

if capital == 0:
    print("Password must contain capital letters.")

if number == 0:
    print("Password must contain numbers.")

if special == 0:
    print("Password must contain special characters.")

if small != 0 and capital != 0 and number != 0 and special != 0 and
len(psw) >= 10:
    print("Strong password!")
```

الشكل (5-9): مثال على سلسلة كلمة المرور.

نشاط عملي

أُنفِّذ البرنامج في المثال السابق في بيئة بايثون (Python)، وأُلاحِظ الناتج، ثمَّ أستكشف الأخطاء التي قد تحدث أثناء تنفيذي للبرنامج، وأعمل على تصحيحها.

نشاط جماعي

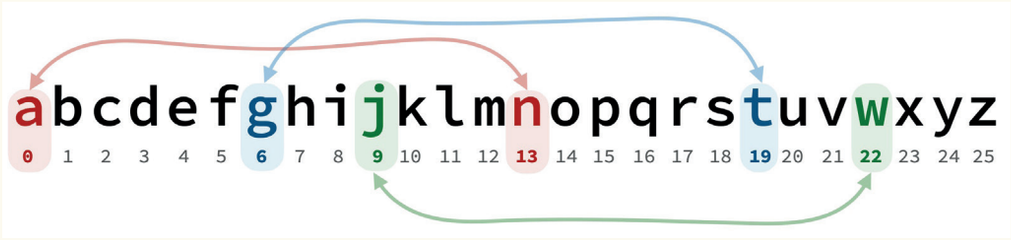
أستكشف وأُناقِش:

أتتبَّع الأوامر البرمجية في المثال السابق، وأستكشف إمكانية كتابتها بطريقة أُخرى، ثمَّ أُناقِش زملائي/ زميلاتي في أهمية كل أمر، وتأثير حذفه في تنفيذ البرنامج بصورة صحيحة.

إثراء

يُمكِن كتابة برنامج لتشفير الرسائل بناءً على فكرة قديمة جدًّا، تُنسَب إلى القيصر الروماني يوليوس، الذي يقال إنَّه وظَّف هذه الفكرة في مراسلاته.

تقوم الفكرة على تمثيل كل حرف من الحروف الأبجدية بحرف آخر يبعد عنه مسافة مُحدَّدة، مثل استبدال كل حرف من الحروف الأبجدية بالحرف الذي يبعد عنه (13) خطوة؛ إذ يُستبدَل حرف الـ'a' بحرف الـ'n'، وحرف الـ'w' بحرف الـ'j'، وهكذا كما هو مُوضَّح في الشكل الآتي:



بناءً على ذلك، فإنَّ تشفير كلمة 'abu' هو 'noh'، وتشفير كلمة 'bad' هو 'onq'.

أُلاحِظ أنَّه لحساب موقع الحرف البديل يجب إضافة (13) خطوة، لكنَّ ذلك قد يجعل الموقع أكبر من (25) كما هو الحال بالنسبة إلى الحرف (w)؛ إذ سيكون الناتج هو (35) إنْ أُضيف (13) إلى موقع الحرف (22)، في حين يجب استبدال الحرف (w) بالحرف (j) الموجود في الموقع (9). ومن ثَمَّ يُمكِن طرح (13) بدلًا من إضافة (13)؛ ما يفي بالغرض (22 – 13= 9).

وتأسيسًا على ذلك، فإنَّ البرنامج سيظهر على النحو المُبيَّن في الشكل الآتي:

```
chars = 'abcdefghijklmnopqrstuvwxyz'
result = ''

text = input("Enter the text to encrypt/decrypt: ")

for c in text:
    if c.isalpha():                 # هل الحرف أبجدي؟
        c = c.lower()               # حول الحرف الكبير لحرف صغير
        i = chars.index(c)          # احسب موقع الحرف الأصلي

        new_i = i+13                # احسب موقع الحرف الجديد
        if new_i >= len(chars):
            new_i = i - 13

        c = chars[new_i]            # استبدل الحرف

    result += c                     # أضف الحرف للرسالة المشفرة

print(result)
```

الأحرف الخاصة

تحمـــل بعض الأحرف معاني خاصة في لغـــة البرمجة بايثون (Python)؛ لـــذا يجب الانتباه عند استخدامها في السلاسل. فمثلًا، تُســـتعمَل علامة التنصيص ' ' وعلامة التنصيص " " لتحديد بداية سلسلة الحروف ونهايتها؛ ما يجعل استخدام هذه العلامة حرفًا داخل السلسلة مشكلةً. ولهذا تسمح لغة بايثون (Python) بالتفرقة بين علامة التنصيص التي هي جزء من السلسلة وعلامة التنصيص التي تُحدِّد بداية السلسلة ونهايتها، وذلك عن طريق استخدام الرمز \ قبل علامة التنصيص كما يأتي:

```
>>> print('What is the difference between \' \' and " "? ')

What is the difference between ' ' and " "?
>>>
```

أُلاحِظ أنَّ الرمز \ لم يُضَف قبل علامة التنصيص "؛ لأنَّ العلامة التي اســـتُخدِمت في تحديد بداية السلسلة ونهايتها هي علامة '.

كذلك يُستخدَم الرمز \ قبل بعض الحروف لإعطائها معنًى خاصًّا، مثل: '\n' التي تعني سطرًا جديدًا، و'\t' التي تعني مسافة مُطوَّلة:

```
>>> print('Name:\tJamilah\nAge:\t16 years old')

Name:   Jamilah
Age:    16 years old
>>>
```

ولكنْ، كيف يُستخدَم الرمز \ حرفًا داخل السلسلة إنْ كان يحمل معنًى خاصًّا؟ يُمكِن فعل ذلك عن طريق إضافة رمز \ آخر قبله كما يأتي:

```
>>> print('I can print \\')

I can print \
>>>
```

نشاط فردي

أكتب أوامر برمجية، أستخدِم فيها الأحرف الخاصة، وأُنفِّذها، ثمَّ أُلاحِظ ناتج التنفيذ كل مَرَّة.

تعرَّفْتُ سابقًا أنَّ عناصر القائمة قد تحوي جملة من القوائم؛ ما يعني إمكانية إنشاء قائمة من مجموعة قوائم. وهذا النوع من القوائم المُركَّبة مُنتشِر ومُهِمٌّ لكثير من التطبيقات؛ إذ يُمكِن – مثلًا– في الرياضيات تمثيل المصفوفة ثنائية الأبعاد (2D Matrix) في صورة قائمة مُركَّبة تحوي أرقامًا، وكذلك تمثيل رقعة الشطرنج في صورة مصفوفة ثنائية الأبعاد تحوي أحجارًا، وغير ذلك كثير.

إنشاء القوائم المُركَّبة

يُمكِن إنشاء قائمة مُركَّبة كما في الجملة الآتية:

```
a  = [[0, 0, 0],
      [0, 0, 0],
      [0, 0, 0]]
```

أُلاحِظُ أنَّ هذه القائمة تتألَّف من (3) قوائم، وأنَّ كل قائمة منها تتألَّف من (3) عناصر؛ أيْ إنَّ هذه القائمة المُركَّبة تحوي (3) صفوف و(3) أعمدة.

يُمكِن الوصول إلى الصف الأوَّل باستخدام a[0]، والوصول إلى الصف الثاني باستخدام a[1]، وهكذا. كذلك يُمكِن الوصول إلى أيِّ عنصر من عناصر القائمة المُركَّبة عن طريق تحديد موقع الصف، ثمَّ تحديد موقع العنصر داخل هذا الصف (أيْ رقم العمود). فمثلًا، العنصر الأوَّل في القائمة المُركَّبة موجود في المكان a[0][0]، والعنصر الأخير في هذه القائمة موجود في المكان a[2][2]، وهكذا.

مثال:

يعمل البرنامج الآتي على تعيين قيمة (99) للعنصر [2][1]، ثمَّ طباعة عناصر القائمة:

```
>>> a[1][2] = 99
>>> print(a)

    [[0, 0, 0], [0, 0, 99], [0, 0, 0]]
```

مثال :

يطبع البرنامج الآتي عناصر القائمة المُركَّبة، ويضع كل صف على سطر مُنفصِل:

```
>>> for row in a:
...     print(row)

    [0, 0, 0]
    [0, 0, 99]
    [0, 0, 0]
>>>
```

إنَّ هذه الطريقة اليدوية في تعيين قِيَم العناصر تُستخدَم فقط في إنشاء القوائم المُركَّبة الصغيرة. أمَّا القوائم المُركَّبة الكبيرة (مثل مصفوفة تتألَّف من (1000) صف و(2000) عمود) فيتطلَّب إنشاؤها

اســـتخدام حلقة تكرار، بَدْءًا بإعداد قائمة فارغة، وانتهاءً بإضافة كل صف إلى القائمة داخل حلقة التكرار كما يأتي:

```
a = []
for i in range(1000):
    row = [0]*2000
    a.append(row)
```

المرور على القوائم المُركَّبة

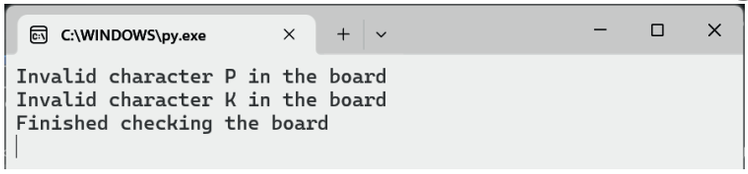
لا يختلـــف المرور على القائمة المُركَّبة عن المرور علـــى أيِّ قائمة أُخرى، لكنَّنا نحتاج كثيرًا إلى استخدام حلقات تكرار مُركَّبة (حلقة للمرور على كل صف داخلها، وحلقة للمرور على كل عنصر في الصف)؛ لأنَّ عناصر القائمة تتألَّف أساسًا من قوائم.

مثال:

يستخدم البرنامج الآتي حلقة التكرار في المرور على كل عنصر في رقعة (اسمها board) مُخصَّصة للعبة (XO)؛ بُغْيَةَ التحقُّق من عدم وجود أحرف في الرقعة، ما عدا 'X' أو 'O' أو '-':

```
board = [['-', 'X', 'P'],
         ['-', 'X', '-'],
         ['K', 'O', '-']]
for row in board:
    for c in row:
        if c != 'X' and c != 'O' and c != '-':
            print('Invalid character ' + c + ' in the board')
print('Finished checking the board')
```

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة الحاسوب:

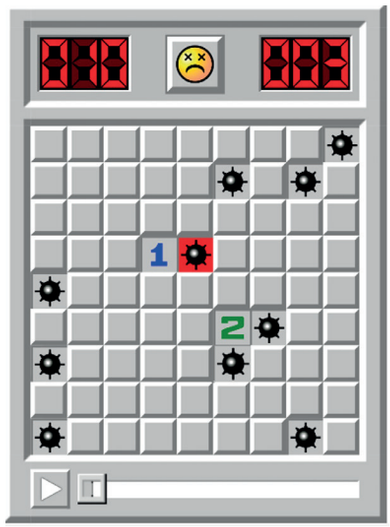


صحيح أنَّ التعامل مع القوائم المُركَّبة يتطلَّب كثيرًا استخدام حلقات تكرار مُركَّبة، لكنَّ ذلك ليس شرطًا.

تطبيقات عملية

سنستعرض الآن مثالين على كيفية المرور على القوائم المُركَّبة، وهما يُمثِّلان جزءًا من بعض الألعاب التي تُمارَس على رقعة يُمكِن تمثيلها في صورة قائمة مُركَّبة.

مثال:



تُعَدُّ لعبة كاسحة الألغام (Minesweeper) واحدة من الألعاب الكلاسيكية القديمة. وتقوم فكرة هذه اللعبة على محاولة تجنُّب الضغط على مُربَّع يحوي قنبلة؛ إذ تُقدِّم اللعبة تلميحات – عند الضغط على بعض المُربَّعات– عن عدد القنابل المَخْفية حول المُربَّع الذي يضغط عليه اللاعب.

في هذا المثال، لن نكتب برنامجًا كاملًا لهذه اللعبة، وإنَّما سنكتفي بمحاكاة عملية العَدِّ لعدد القنابل الموجودة حول كل مُربَّع، وتخزين هذا العدد في المُربَّع نفسه.

لنفترض أنَّ اللعبة مُمثَّلة بقائمة مُركَّبة، تحوي في كل عنصر علامة '-' في حال عدم وجود قنبلة، أو علامة '*' في حال وجود قنبلة. ووظيفتنا في هذا البرنامج هي استبدال رقم يُمثِّل عدد القنابل المُلاصِقة لذلك المُربَّع (مراعين المُربَّعات التي على يمين المُربَّع المطلوب، والمُربَّعات التي على شماله، والمُربَّعات التي فوقه، والمُربَّعات التي تحته مباشرة) بكل علامة '-'، أنظر الشكل (5-10).

	0	1	2	3	4
0	–	–	*	*	–
1	–	*	–	–	–
2	–	*	*	–	*
3	–	–	–	–	–
4	*	–	–	–	–

	0	1	2	3	4
0	0	2	*	*	1
1	1	*	3	1	1
2	1	*	*	2	*
3	1	1	1	0	1
4	*	1	0	0	0

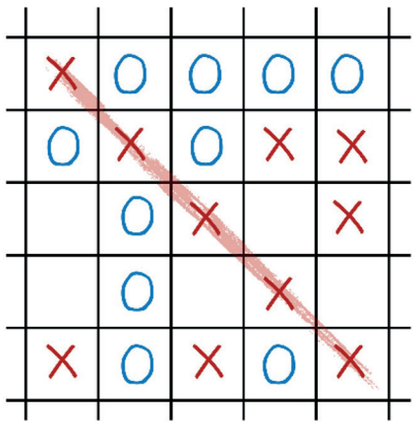
الشكل (5-10): صورة القائمة قبل تنفيذ البرنامج وبعد تنفيذه.

يكتفـــي البرنامج بالمرور على كل موقع في القائمة المُركَّبة، وعَدِّ القنابل التي حوله؛ أيْ إنَّه لن يمرَّ على كل عنصر في هذه الأثناء؛ لأنَّ الهدف ليس فقط معرفة قيمة العنصر، وإنَّما معرفة قيمة العناصر التـــي حوله أيضًا. ومن ثَمَّ يجب معرفة موقع العنصر؛ للتأكُّد مـــن القِيَم المُخزَّنة في المواقع التي حوله، أنظر الشـــكل (5-11) الذي يُبيِّن كيف يُنفَّذ المطلوب بافتراض أنَّ القائمة المُركَّبة تُسمّى (board).

```
board = [
['-', '*', '-', '-'],
['-', '-', '*', '-'],
['*', '-', '-', '-'],
['-', '*', '-', '*']
]
for i in range(len(board)):
    for j in range(len(board[i])):
        if board[i][j] == '-':
            count = 0
        if i > 0 and board[i-1][j] == '*':
            count += 1
        if i < len(board) - 1 and board[i+1][j] == '*':
            count += 1
        if j > 0 and board[i][j-1] == '*':
            count += 1
        if j < len(board[i]) - 1 and board[i][j+1] == '*':
            count += 1
        board[i][j] = str(count)
for row in board:
    line = ' '.join(row)
    print(line)
```

الشكل (5-11): صورة القائمة المُركَّبة للعبة كاسحة الألغام.

مثال:



تُعَدُّ لعبة (XO) واحدة من الألعاب المشهورة التي تُلعَب عادةً باستخدام رقعة حجمها 3x3، ويُمكِن

ممارستها أيضًا باستخدام رقعة مُربَّعة بِغَضِّ النظر عن حجمها.
في هذا المثال، لن نكتب برنامجًا كاملًا لهذه اللعبة، وإنَّما سنكتفي بكتابة الجزئية التي تتحقَّق من فوز أحد اللاعبين، وهو ما يتطلَّب المرور على ثلاثة أشياء، هي:

- كل صف.
- كل عمود.
- القُطْران.

سنبدأ أوَّلًا بالصفوف، ونتحقَّق من عدد (X) وعدد (O) في كل صف؛ فإذا كان العدد لأيٍّ منهما مساويًا لطول الصف، عَلِمْنا أنَّ أحد اللاعبين قد فاز.

```
# عدد الصفوف وهو نفسه عدد الأعمدة لأن الرقعة مربعة
N = len(board)
for row in board:
    if row.count('X') == N:
        print('X wins!')
        break
    if row.count('O') == N:
        print('O wins!')
        break
```

أمّا بالنســـبة إلى الأعمدة، فإنَّ الأمر مختلف بعض الشيء. فكل عمود يحوي عناصر من صفوف مختلفة؛ ما يتطلَّب المرور على كل عمود بصورة يدوية حيث (j) يمثل موقع كل عمود و (i) يمثل كل صف:

```
for j in range(N):
    # 1
    countX = 0
    countY = 0


for j in range(N):
    countX = 0
    countY = 0
    for i in range(N):
        if board[i][j] == 'X':
            countX += 1
        elif board[i][j] == 'O':
            countY += 1
        if countX == N:
            print('X wins')
            break
        if countY == N:
            print('Y wins')
            break
```

في هذا المثال، يؤدّي المقطع البرمجي ثلاث مهام في كل عمود (j)، هي:

1. تصفير عدّادين؛ أحدهما لحرف (X)، والآخر لحرف (O).
2. المرور على جميع العناصر في العمود (j) لعَدِّ مَرّات تكرار حرف (X) وحرف (O) في ذلك العمود، وذلك باستخدام حلقة تكرار تمرُّ على جميع الصفوف في القائمة المُركَّبة، وتتحقَّق من العنصر (j) في تلك القائمة.
3. التأكُّد – بعد الانتهاء من عَدِّ الأحرف في العمود (j)– أنَّ العدد مســـاوٍ لطول العمود، وهو ما يعني أنَّ أحد اللاعبين قد فاز.

وأمّا القُطْران فيُمكِننا التحقُّق منهما باستخدام حلقة تكرار تعمل على تغيير الصف والعمود في كل دورة؛ إذ يبدأ القُطْر الأوَّل عند [0][0]، ثمَّ ينتقل إلى [1][1]، ثمَّ ينتقل إلى [2][2]، وهكذا.

```
countX = 0
countY = 0
for i in range(N):
      if board[i][N-i-1] == 'X':
            countX += 1
      elif board[i][N-i-1] == 'O':
            countY += 1
if countX == N:
      print('X wins')
if countY == N:
      print('Y wins')
```

في حين يبدأ القُطْر الثاني عند [0][N−1]، ثمَّ ينتقل إلى [1][N−2]، ثمَّ ينتقل إلى [2][N−3]، وهكذا.

```
countX = 0
countY = 0
for i in range(N):
      if board[i][N-i-1] == 'X':
            countX += 1
      elif board[i][N-i-1] == 'O':
            countY += 1
if countX == N:
      print('X wins')
if countY == N:
      print('Y wins')
```

المواطنةُ الرقميةُ:

- **الأخلاقيات الرقمية**: أحترم آراء الآخرين وأفكارهم عند مناقشة التعليمات البرمجية أو مناقشة المشروعات، وأُقدِّم النقد البَنّاء والمساعدة للآخرين عند مراجعة تعليمات البرمجية.
- **الوعي بالأمن السيبراني**: أُدرِك أهمية استخدام برامج مكافحة الفيروسات وتحديث أنظمة التشغيل بانتظام أثناء العمل في بيئة بايثون (Python).

مشروع

تصميم برنامج وإعداده لإنشاء لعبة تخمين الأرقام باستخدام لغة بايثون (Python)/ المهمة (5).

أُكمِل - بالتعاون مع أفراد مجموعتي- تنفيذ مشروع التصميم والتطوير للعبة تخمين الأرقام. وكنَّا قد انتهينا في الخطوات السابقة من عرض القائمة الرئيسة في اللعبة، بالاستجابة لِما يُدخِله اللاعب عن طريق عرض تعليمات اللعبة، أو كتابة رسالة بحسب الخيار المُدخَل، وتعديل البرنامج مُمثَّلًا بتكرار عرض القائمة الرئيسة والخيارات. والآن سأعمل - ضمن المجموعة- على إضافة العنصر الرئيس في اللعبة، وهو إدخال البرنامج للرقم العشوائي.

سأستعمل - مع أفراد مجموعتي- دالة (random) لتوليد رقم مخفي في بداية اللعبة، يتألَّف من سلسلة تحوي (4) أعداد مختلفة، ثمَّ أُخزِّن الأرقام المُمكِنة (المُحتمَلة) في قائمة، وأعمل على تغيير ترتيبها باستخدام الدالَّة random.shuffle.

أتحقَّق - مع أفراد مجموعتي- من استخدام القوائم بصورة صحيحة، ومن تفعيل عملية توليد الأرقام، ثمَّ أحتفظ بما كُتِب - ضمن المجموعة- في ملف حتّى يُمكِن لأفراد المجموعة التعديل عليه في الخطوات القادمة.

أُقيِّمُ تعلُّمي

المعرفة: أُوظِّف في هذا الدرس ما تعلَّمْتُه من معارف في الإجابة عن الأسئلة الآتية:

السؤال الأوَّل: ما أبرز أوجه التشابه وأوجه الاختلاف بين القوائم والسلاسل؟

السؤال الثاني: أكتب جملة يُمكِن استخدامها في حذف الحرف الأخير من سلسلة تُسمّى (str1).

السؤال الثالث: أذكر طريقتين مختلفتين (أو أكثر) لتنفيذ كل مهمة من المهام الآتية، ثمَّ أُحدِّد أيُّ هذه الطرائق أسهل أو أفضل:

1. طباعة العنصر الأوَّل في قائمة تتألَّف من (10) عناصر.
2. طباعة العنصر الأخير في قائمة تتألَّف من (10) عناصر.
3. التأكد من وجود حرف 'a' في سلسلة ما.
4. ترتيب عناصر قائمة ما (باستخدام الدوالِّ الجاهزة).
5. إيجاد العنصر الأكبر في قائمة ما.
6. حذف جميع عناصر قائمة ما.

المهارات: أُوظِّف مهارة التفكير الناقد ومهارة حَلِّ المشكلات والمهارات البرمجية في الإجابة عن الأسئلة الآتية:

السـؤال الأوَّل: أتتبَّع البرنامج الآتي من دون تشغيله (أتتبَّع كل جزء من البرنامج بصورة مُنفصِلة)، ثمَّ أذكر ناتج تشغيله.

```
a = [1, 3, 5, 2, 0, 4]
# 1
for i in range(len(a)):
    print(max(a[i:]))
# 2
for i in range(1, len(a)):
    print(max(a[:i]))
# 3
for e in a[::-1]:
    a += [e]
print(a)
```

السؤال الثاني: أُحدِّد الهدف الرئيس لكل برنامج من البرامج المُبيَّنة في الجدول الآتي، ولا أَصِف ما يقوم به البرنامج في كل سطر بصورة مُنفصِلة، وإنَّما أستعمل بضع كلمات لتلخيص المهام التي يؤدّيها البرنامج بوجه عام.

```
result = []
for e in a:
    if e < 0:
        result = [e] + result
    else:
        result += [e]

print(result)
```

```
found = False
for x in a:
    for y in a:
        if x + y == 0:
            found = True
            break
```

```
for row in a:
    temp = row[0]
    row[0] = row[-1]
    row[-1] = temp
```

```
j = len(a) - 1
i = 0
for row in a:
    temp = row[i]
    row[i] = row[j]
    row[j] = temp
    i += 1
    j -= 1
```

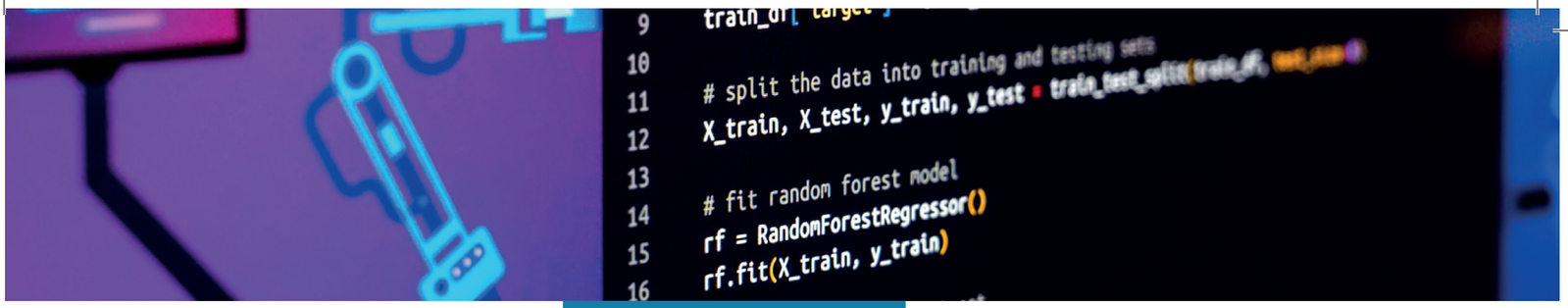
السؤال الثالث: أكتب برنامجًا يُحقِّق كلًّا من المهام الآتية:

- **المهمة الأولى**: بناء قائمة من أرقام يدخلها المستخدم ثم طباعة عدد الأرقام الزوجية الموجودة في القائمة.
- **المهمة الثانية**: بناء قائمتين من أرقام يدخلها المستخدم ثم التحقق من وجود أي عنصر من عناصر القائمة الأولى في القائمة الثانية.
- **المهمة الثالثة**: قراءة سلسلة أحرف من المستخدم ثم التأكد من وجود (3) أحرف متتابعة ومتساوية داخل السلسلة.

السؤال الرابع: أكتب حزءاً من برنامج يتحقق من أن قائمة ثنائية الأبعاد اسمها board هي قائمة مربعة (أي أن عدد الصفوف مساو لعدد الأعمدة).

القِيَم والاتجاهات:

أُعِدُّ – بالتعاون مع أفراد مجموعتي – كُتيِّبًا إرشاديًّا يُبيِّن أكثر الأخطاء البرمجية التي تعرضنا لها أثناء تطبيق المقاطع البرمجية وتوثيق الحلول بهدف مساعدة الطلبة الآخرين على مواجهة المشكلات البرمجية. وأستعين بأحد برامج التصميم لتنفيذه، ثمَّ أنشره بين طلبة المدرسة بعد مراجعته وتدقيقه مع معلمي/ معلمتي وفي الموقع الإلكتروني الخاص بها.



الدرسُ السادس

الدوال البرمجية
(Functions)

مُنتَجاتُ التعلُّم
(Learning Products)

أكتب برامج بلغة البايثون (Python) تستدعي دوال برمجية جاهزة.

الفكرة الرئيسة:

تعرُّف آليَّة تجزئة المشكلة إلى أجزاء صغيرة، وكتابتها على أساس أنَّها جمل برمجية أو وحدات، وتعلُّم كيفية توثيق البرامج عن طريق التصميم والتطوير. كذلك تعرُّف الدوالِّ البرمجية، وتعلُّم كيف يُمكِن استخدام الدوالِّ الجاهزة واستيرادها، وكيف يُمكِن تعريف دوالَّ جديدة واستخدامها في البرامج.

المفاهيم والمصطلحات:

الدالَّة البرمجية (Function)، الوحدة البرمجية (Module)، استيراد الوحدات (Importing Modules)، مدى المُتغيِّر (Variable's Scope)، المدخلات أو مُعامِلات الدالَّة (Function Parameters)، التوثيق (Documentation)، سلاسل التوثيق (Docstrings).

نتاجات التعلُّم (Learning Objectives):

- أُعرِّف المقصود بالوحدات البرمجية (Modules).
- أُجزِّئ المشكلة إلى أجزاء صغيرة، وأعمل على تصميم كل جزء منها وبرمجته.
- أُحدِّد الطريقة الفضلى لتمثيل أجزاء المشكلة في صورة جمل برمجية، أو روتين فرعي، أو وحدات، أو كائنات.
- أستعمل لغة البرمجة بايثون (Python) لاستدعاء روتين فرعي جاهز بناءً على وقوع حدث مُحدَّد.

تعرَّفْتُ سابقًا أنَّ الدالَّة البرمجية (Function) مقطع برمجي له اسم يؤدّي وظيفة ما، ويُمكِن استدعاؤه باستخدام اسمه؛ فكيف تُستخدَم الدوالُّ البرمجية في لغة البرمجة بايثون (Python)؟ وما أهميتها في تصميم البرامج وتسهيل قراءتها والتعامل معها؟

نشاط تمهيدي

تعرَّفْتُ في الدروس السابقة الدوالَّ الجاهزة في لغة البرمجة بايثون (Python)، التي يُمكِن استخدامها في معالجة القوائم والسلاسل أو أداء مهام أُخرى. أكتب قائمة تحوي هذه الدوالَّ، ثمَّ أُصنِّفها بناءً على التشابه في كيفية استخدامها. بعد ذلك أبحث في أهمِّ الجوانب التي تختلف فيها الدوالُّ بعضها عن بعض.

الدوالُّ البرمجية

إذا افترضْتُ أنَّ الأمر ()print لم يكن موجودًا في لغة البرمجة بايثون (Python)، وأنَّه يتعيَّن عليَّ دائمًا طباعة أيِّ شيء على الشاشة بكتابة كامل الكود البرمجي الذي يقوم بالتعامل مع نظام التشغيل وأنواع البيانات المختلفة من أجل إظهارها على الشاشة بالشكل الصحيح، فإنَّ ذلك سيكون مُرهِقًا لي بلا شكٍّ، ويجعل قراءة البرنامج عسيرة. ولهذا، فإنَّ لغة البرمجة بايثون (Python) وفَّرت علينا هذا الجُهْد والعناء بتقديمها مقاطع برمجية جاهزة مُدقَّقة وخالية من الأخطاء، بحيث يُمكِننا استدعاؤها بكل سهولة عن طريق اسمها، واستخدامها في برامجنا من دون حاجة إلى كتابة الأوامر دائمًا.

الدوالُّ البرمجية الجاهزة

تُوفِّر لغة البرمجة بايثون (Python) عددًا كبيرًا من الدوالِّ البرمجية الجاهزة، وقد استخدمنا العديد منها في أمثلة سابقة ورد ذكرها في هذه الوحدة. ونظرًا إلى كثرة هذه الدوالِّ، فإنَّه يصعب على المُبرمِجين تذكُّرها جميعًا، أو حفظ كيفية استخدامها. ولهذا يُعَدُّ الموقع الإلكتروني للغة البرمجة بايثون (Python) الملاذ والصديق لكل مُبرمِج مُحترِف؛ إذ يُواظِب كلٌّ منهم على زيارته باستمرار؛ للبحث عن دالَّة لوظيفة ما، أو تذكُّر كيف تعمل إحدى الدوالِّ.

الوحدات البرمجية (Modules)

تم تنظيم عدد الدوالِّ الكبير في لغة البرمجة بايثون (Python) عن طريق جمعها في وحدات (modules)، تحتوي الوحدة الواحدة منها على دوالَّ برمجية تشترك معًا في الغرض والاستخدام. فمثلًا، تحتوي وحدة (time) على دوالَّ لها علاقة بالوقت والتاريخ، وتحتوي وحدة (math) على دوالَّ لها علاقة بالعمليات الرياضية، وهكذا.

يُبيِّن الجدول (6-1) مجموعة من الوحدات في لغة البرمجة بايثون (Python)، وأمثلة على الدوالِّ البرمجية التي تنتمي إلى هذه الوحدات.

الجدول (6-1): مجموعة من الوحدات البرمجية، وأمثلة على الدوالِّ البرمجية التي تنتمي إليها.

الوحدة module	مثال على الدالَّة البرمجية التي تنتمي إلى الوحدة	
math	مضروب الرقم 10	math.factorial(10)
	لوغاريتم الرقم 2 للأساس 8	math.log(2 ,8)
	الرقم 3 مرفوع إلى الأُسِّ 5	math.pow(5 ,3)
	الجذر التربيعي للرقم 2	math.sqrt(2)
	جيب تمام الزاوية 3 , 14159265	math.cos(3 , 14159265)
	جيب الزاوية 3 , 14159265	math.sin(3 , 14159265)
random	رقم صحيح عشوائي يقع بين 1 و10	random.randint(10 ,1)
	رقم عشري عشوائي يقع بين 0 و1	random.random()
	1, 5, 7] عنصر عشوائي من القائمة]	random.choice([7 ,5 ,1])
	نسخة من القائمة مخلوطة بصورة عشوائية	random.shuffle([4 ,3 ,2 ,1])
statistics	المُتوسِّط الحسابي للقائمة	statistics.mean([2 ,1 ,8 ,8 ,10 ,1 ,1])
	الوسيط الحسابي للقائمة	statistics.median([2 ,1 ,8 ,8 ,10 ,1 ,1])
	العنصر الأكثر تكرارًا في القائمة	statistics.mode([2 ,1 ,8 ,8 ,10 ,1 ,1])

إضاءةٌ

يُمكِن تعرُّف جميع الوحدات الجاهزة في لغة البرمجة بايثون (Python) عن طريق الرابط الإلكتروني الآتي:

https://docs.python.org/3/library/index.html

أو عن طريق مسح الرمز سريع الاستجابة QR Code المجاور.

أُلاحِظ أنَّ بعض الدوالِّ تستقبل مدخلات، وأنَّ بعضها الآخر لا يستقبل أيَّ مدخلات. فمثلًا، الدالَّة sqrt(...) تستقبل رقمًا واحدًا، والدالَّة pow(...) تستقبل رقمين اثنين، والدالَّة median(...) تستقبل قائمة، في حين أنَّ الدالَّة random() لا تستقبل أيَّ مدخل.

أُلاحِظ أيضًا أنَّ جميع هذه الدوالِّ تعمل على إرجاع النتائج، بالرغم من أنَّ ذلك ليس شـــرطًا في الدوالِّ. فمثلًا، تطبع الدالَّة print(...) على الشاشـــة، ولا تُرجِع أيَّ شيء؛ ما يُفسِّر سبب استدعائها من دون تخزين نتيجتها. أمّا الدالَّة input(...) فتعمل على إرجاع ما أدخله المُستخدِم. ومن ثَمَّ، فإنَّ الطريقة المُستخدَمة في استدعاء كلٍّ من هاتين الدالَّتين مختلفة:

```
name = (input)
print(name)
```

نحتاج إلى تخزين النتيجة في مُتغيِّر

لا نحتاج إلى تخزين أيِّ نتيجة

والشـــيء نفســـه ينطبق على الدالَّة sorted(...) والدالَّة sort()؛ إذ تعمل الأولى على إرجاع إحدى القوائم المُرتَّبة، في حين تعمل الثانية على ترتيب القائمة نفسها، ولا تُرجِع أيَّ نتيجة، أنظر الجدول (6-2).

الجدول (6-2): دوالُّ برمجية وأثرها في البرنامج.

نحتاج إلى طباعة القائمة الراجعة من الدالَّة حتّى تظهر النتيجة	`print(sorted(mylist))`
يُمكِننا أيضًا تخزين النتيجة الراجعة من الدالَّة لاستخدامها لاحقًا	`result = sorted(mylist)`
هذا الاستدعاء لا أثر له؛ لأنَّ القائمة المُرتَّبة الراجعة من الدالَّة لم يتمَّ حفظها أو طباعتها	`sorted(mylist)`
لا داعي لتخزين أيِّ شـــيء أو طباعته؛ لأنَّه تمَّ الانتهاء من ترتيب القائمة	`mylist.sort()`
سنجد القائمة مُرتَّبة عند طباعتها	`print(mylist)`

استيراد الوحدات (Import)

تُمثِّل كل وحـــدة ملفًا يحوي دوالَّ برمجية خاصة بها (إضافةً إلـــى تعريفات أُخرى). ولهذا، فإنَّ استخدام هذه الدوالِّ يتطلَّب أوَّلًا استيراد الوحدة باستخدام كلمة (import) كما في المثال الآتي:

```
import math
print(math.sqrt(2))
```

يعمل المُبرمِجون عادةً على وضع جملة الاستيراد في رأس البرنامج حتّى يتضح - ابتداءً- لقارئ البرنامج أيُّ الوحدات يراد اســـتخدامها. وبعد استيراد الوحدة، يُمكِن استدعاء الدوالِّ الخاصة بها بنفس الطريقة التي ورد ذكرها في الجدول (6-1).

أُلاحِظ أنَّه عند اســـتدعاء دالَّة من وحدة ما، فإنَّ اســـم هذه الوحدة يوضَع قبل اســـم الدالَّة، مثل math.sqrt(2). ويُمكِن التخلُّص من ذلك باســـتيراد الدالَّة نفسها، مثل الدالَّة sqrt التي تُستورَد من وحدة (math) كما يأتي:

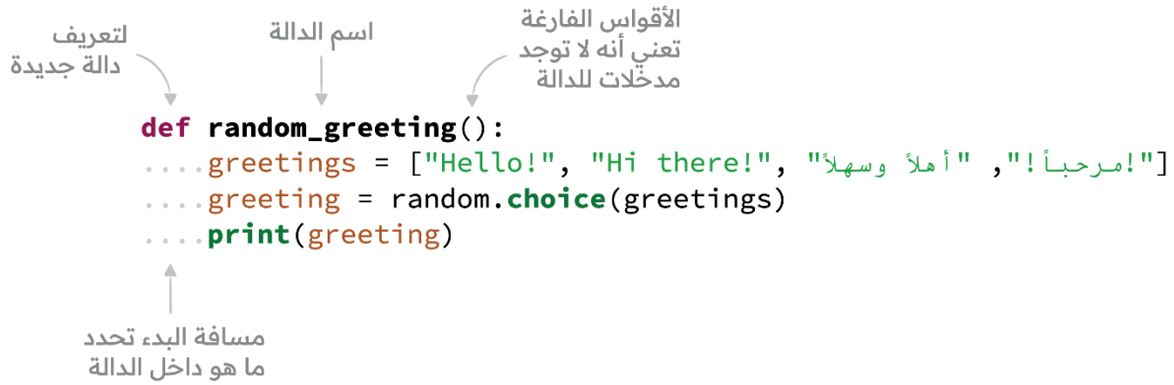
```
from math import sqrt
print(sqrt(2))
```

بعد ذلك تُســـتخدَم الدالَّة sqrt مباشرة من دون حاجة إلى اسم الوحدة، ولكنْ يجب الحذر حينئذٍ؛ لأنَّ هذه الجملة لا تستورد الوحدة، وإنَّما تستورد فقط الدالَّة؛ لذا لا يُمكِن بعد استيرادها استخدام دالَّة أُخرى من الوحدة نفسها من دون استيرادها أو استيراد كامل الوحدة كما أشرنا آنفًا.

تعريف دوالَّ برمجية جديدة

قد يحتاج المُســـتخدِم إلى تعريف دوالَّ برمجية خاصة به؛ إمّـــا لتنظيم البرامج، وإمّا لاحتواء هذه الدوالِّ على وظائف تَلزمه في البرنامج بصورة مُتكرِّرة، وإمّا لاستدعاء آخرين هذه الدوالَّ للإفادة منها.

يُبيِّن المثال الوارد في الشكل (6-1) كيفية تعريف دالَّة بسيطة تُسمّى random_greeting، وتُبادِر إلى إلقاء تحية عشوائية كلَّما تمَّ استدعاؤها:



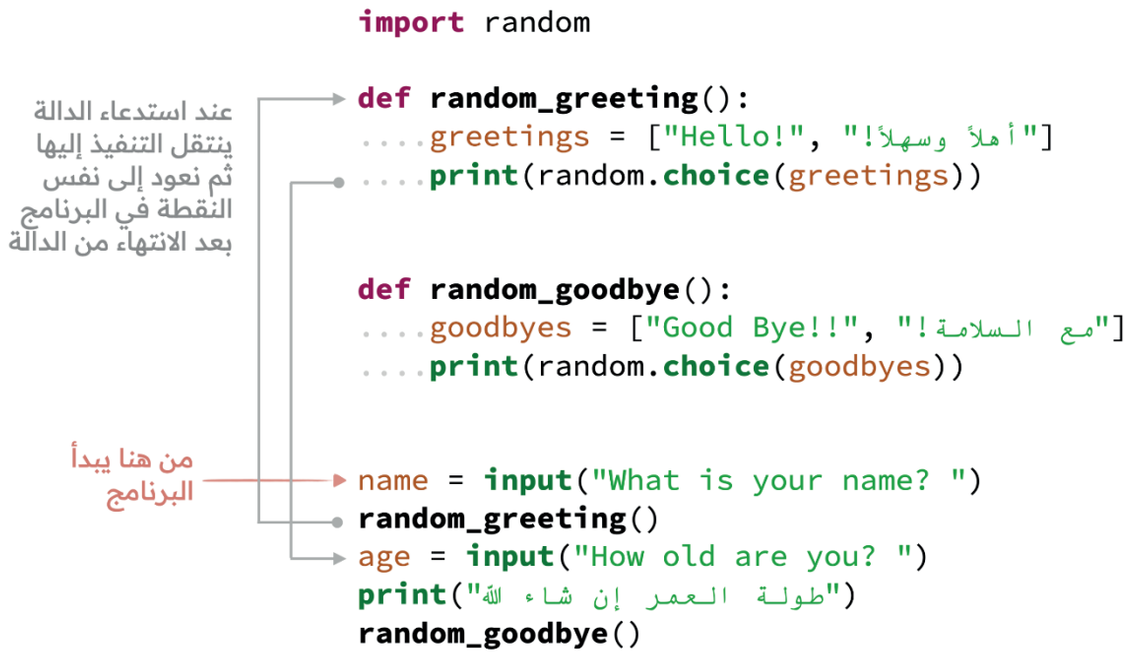
الشكل (6-1): مثال على كيفية تعريف دالَّة بسيطة.

يجب تعريف الدالَّة بكتابة كلمة (def) متبوعةً باســـم الدالَّة، ثمَّ بأقواس ونقطتين رأسيتين. ويُمكِن للمُفسِّر التمييز بين ما بداخل الدالَّة وخارجها عن طريق مسافة البَدْء التي تكون في أوَّل السطر.

بعد تعريف الدالَّة، يُمكِن استدعاؤها في أيِّ مكان داخل الملف نفسه كما يأتي:

```
random_greeting()
```

يُبيِّن الشكل (6-2) برنامجًا مُتكامِلًا يعمل على تعريف دالَّتين واستدعائهما. ولمّا كانت الدوالُّ لا تُنفَّذ إلا بعد اســـتدعائها، بِغَضِّ النظر عن مكانها في الملف، فإنَّ البرنامج سيبدأ العمل من السطر الذي يُسأَل فيه عن الاسم، بالرغم من أنَّ هذا السطر يأتي بعد تعريف الدوالِّ.



الشكل (6-2): مثال على برنامج مُتكامِل يعمل على تعريف دالَّتين جديدتين واستدعائهما.

نشاط عملي

أُجرِّب بنفسي: أقوم بطباعة البرنامج الموضح في الشكل 6-2 ثم أقوم بتشغيله عدة مرات وملاحظة اختلاف المخرجات في كل مرة، ثم أقوم بتجربة التالي ومناقشته مع زملائي:

- نقل كامل تعريف دالة الوداع random_goodbye لتصبح قبل دالة الترحيب random_greeting ، هل أثر ذلك على صحة عمل البرنامج ولماذا؟
- نقل كامل تعريف الدالتين ليصبحا في نهاية البرنامج بدلاً من بدايته (أي بعد الجمل التي تسأل عن الاسم والعمر وتقوم باستدعاء الدوال) ، هل أثر ذلك على صحة عمل البرنامج ولماذا؟

يتطلَّب استخدام بعض الدوالِّ توافر مجموعة من البيانات، تعمل الدوالُّ على تحليلها ومعالجتها. فمثلًا، لا يُمكِن استخدام دالَّة الجذر التربيعي قبل استقبال رقم يُمكِّن من حساب جذره التربيعي. ولهذا يجب تحديد عدد مدخلات (أو مُعامِلات) الدالَّة (parameters) عند تعريفها، وإعطاء كل مدخل (مُعامِل) اسمًا.

مدخلات الدالة

```
def print_square(width, height):
    line = '*' * width
    for i in range(height):
        print(line)


print_square(5, 4)
print()
print_square(6, 3)
```

نتيجة تنفيذ البرنامج السابق.

```
  5
┌─────┐
*****
*****
4
*****
*****
└─────┘

  6
┌──────┐
******
3******
└──────┘
```

الشكل (6-3): تعريف دالَّة برمجية جديدة.

إرجاع النتائج

إنَّ الـــدوالَّ التي عَمِلْنا على تعريفها في الأمثلة الســـابقة لا تُرجِع أيَّ نتيجـــة. والمثال الوارد في الشكل (6-4) يُبيِّن كيف يُمكِن إرجاع نتيجة من دالَّة، وكيف تُستقبَل النتيجة عند استدعاء الدالَّة.

```
import math

def area(radius):
    return math.pi * radius**2


print(area(5))
result = area(5)
print(result)

area(5)
```

طريقتان صحيحتان لاستدعاء الدالة

ستضيع النتيجة العائدة من الدالة إن قمنا باستدعائها بهذه الطريقة

الشكل (6-4): مثال على دالَّة برمجية تُرجِع نتيجة عند استدعائها.

عند تشغيل البرنامج، ستظهر النتيجة الآتية على شاشة جهاز الحاسوب:

78.53981633974483

78.53981633974483

نشاط عملي

أُجرِّب وأســـتنتج: أُدخِل البرنامج الوارد في الشكل (6-4)، ثمَّ أُعدِّل الأوامر البرمجية، بحيث تطلب من المُستخدِم إدخال نصف القُطْر، ثمَّ استدعاء الدالَّة وطباعة النتيجة.

تعمل الدالَّة في المثال السابق على استقبال رقم يُمثِّل نصف قُطْر دائرة، وإرجاع مساحة هذه الدائرة، علمًا بأنَّ عملية إرجاع النتيجة تمَّت باستخدام (return)؛ إذ أُنهِي تنفيذ الدالَّة عند تلك النقطة، وتمَّ إرجاع النتيجة. ولهذا لن يُنفَّذ أيُّ أمر يأتي بعد هذه الجملة في الدالَّة (في حال وجود أيِّ سطر).

يُمكِن اســـتخدام أكثر من جملة (return) في الدالَّة نفسها، بحيث تُنفَّذ كل جملة في حالة مختلفة عن الأُخرى، أنظر الشكل (6-5) الذي يُبيِّن مثالًا على دالَّة تعمل على إرجاع حرف مختلف يُمثِّل مُعدَّل الطالب بحسب علامته:

```
def letter_grade(grade):
    if grade >= 90:
        return 'A'

    if grade >= 80:
        return 'B'

    if grade >= 70:
        return 'C'

    if grade >= 60:
        return 'D'

    return 'F'
```

لن نصل لهذا السطر إن تم تنفيذ جملة return السابقة

سنصل لهذا السطر فقط إن لم يتم تنفيذ أي من جمل return السابقة

الشكل (6-5) مثال على استخدام جملة (return) لإرجاع نتيجة ما.

نشاط جماعي

أعمل - بالتعاون مع أفراد مجموعتي- على تعديل البرنامج السابق وتطويره، بحيث يتمكَّن من جمع العلامات وإيجاد المُعدَّل النهائي، ثمَّ أســـتخدِم دالَّـــة letter_grade() في تصنيف المُعدَّل.

إضاءةٌ

يجب التحقُّق من وجود جملة (return) لكل حالة مُمكِنة في الدالَّة التي تعيد نتيجة ما.

نشاط جماعي

اقرأ البرنامج الآتي جيِّدًا، ثمَّ أُحلِّل – بالتعاون مع أفراد مجموعتي– الأوامر البرمجية الواردة في الشكل (6-6)؛ لتحديد الأخطاء المُحتمَلة، ثمَّ اقتراح الحلول المناسبة، وتجربتها في بيئة بايثون (Python).

```
def  a bsolute_value(x):
if x > 0:
    return x
if x < 0:
    return -x
x = float(input("Enter a number to find its absolute value: "))
print("The absolute value is:", absolute_value(x))
```

الشكل (6-6): مثال على الأخطاء البرمجية.

مدى المُتغيِّرات (Scope)

لكل مُتغيِّر مدى (scope) يُمكِن استخدامه في هذا المُتغيِّر. فمثلًا، مدى المُتغيِّر (AVOGADRO) في البرنامج الوارد في الشكل (6-7) هو جميع الأسطر التي تلي أوَّل سطر استُخدِم فيه المُتغيِّر؛ لذا يُمكِن استخدام هذا المُتغيِّر داخل الدوالِّ التي تأتي بعده، وكذلك استخدامه خارج هذه الدوالِّ. أمّا المُتغيِّر (result) فمداه هو جميع الأسطر التي تليه؛ لذا لا يُمكِن استخدامه داخل الدوالِّ (تأتي الدوالُّ في الملف قبل أوَّل استخدام للمُتغيِّر).

مدى AVOGADRO

```
AVOGADRO = 6.022 * 10**23

def atoms_to_moles(atoms):
    moles = atoms / AVOGADRO
    return moles

def moles_to_atoms(moles):
    atoms = moles * AVOGADRO
    return atoms

num_atoms = 1.204e24
result = atoms_to_moles(num_atoms)
print("Number of atoms: ", num_atoms)
print("Moles: ", result)
```

مدى moles و atoms

مدى atoms و moles

مدى num_atoms

مدى result

الشكل (6-7): مثال على مدى المُتغيِّرات.

أُلاحِـــظ أنَّ مدى كلٍّ من المُتغيِّر (atoms) والمُتغيِّر (moles) ينتهي بانتهاء الدالَّة؛ لذا يُطلَق على هذا النوع من المُتغيِّرات اسم المُتغيِّرات المحلية (local variables)؛ فهي مُعرَّفة فقط محليًّا داخل الدالَّة. أمّا مدى كلٍّ من المُتغيِّـــر (AVOGADRO) والمُتغيِّر (result) فغير محصور داخل الدالَّة؛ لذا يُطلَق على هذا النوع من المُتغيِّرات اسم المُتغيِّرات العامة (global variables).

يُذكَر أنَّ المُتغيِّر (atoms) والمُتغيِّر (moles) المُعرَّفين داخل الدالَّة الأولى هما مُتغيِّران مختلفان عن المُتغيِّر (atoms) والمُتغيِّـــر (moles) المُعرَّفين داخل الدالَّة الثانية؛ فكلٌّ من هذه المُتغيِّرات مُعرَّف محليًّا في مدى ما، ولا علاقة له بالمُتغيِّرات المُعرَّفة في المدى الآخر.

نشاط عملي

أُجرِّب وأستنتج:

- أُعدِّل على البرنامج الوارد في الشـــكل (6-7) بتنفيذ الاســـتدعاء print(atoms) خارج الدالَّتين. ما تأثير ذلك في تنفيذ البرنامج؟
- أُعدِّل على البرنامج نفسه بتنفيذ الاستدعاء print(num_atoms) داخل إحدى الدالَّتين. ما تأثير ذلك في تنفيذ البرنامج؟

عند اســـتدعاء دالَّة ما، فإنَّ مكانًا خاصًّا لها يُحجَز في الذاكرة، ويُســـمّى إطـــار ذاكرة التكديس (Stack Frame)؛ إذ يُستعمَل هذا المكان لحفظ كل ما يتعلَّق بالدالَّة من مُتغيِّرات محلية، ثمَّ يُتخلَّص منه عند الانتهاء من تنفيذ الدالَّة.

لتوضيح ذلك، أتتبَّع البرنامج الوارد في الشكل (6-8).

```python
def fun1(x):
    x = 3

def fun2(x):
    x = 5

x = 7
fun1(x)
fun2(x)
print('x: ', x)
```

الشكل (6-8): مثال على استدعاء الدالة وارسال قيمة المتغير x إلى متغير محلي

أُلاحِظ من البرنامج السابق ما يأتي:

7. تعريف المُتغيِّر (x) في المدى العام بقيمة (7).
8. استدعاء الدالَّة fun1، ثمَّ إرسال القيمة (7) إلى مُتغيِّر محلي داخل الدالَّة، يُسمّى (x).
9. تغيير قيمة المُتغيِّر المحلي (x) إلى (3).
10. انتهاء الدالَّة، والتخلُّص من المُتغيِّر المحلي (x) الذي يحمل القيمة (3).
11. العودة إلى المدى العام، حيث يوجد مُتغيِّر يُسمّى (x)، وتبلغ قيمته (7).
12. استدعاء الدالَّة fun2، ثمَّ إرسال القيمة (7) إلى مُتغيِّر محلي داخل الدالَّة، يُسمّى (x).
13. تغيير قيمة المُتغيِّر المحلي (x) إلى (5).
14. انتهاء الدالَّة، والتخلُّص من المُتغيِّر المحلي (x) الذي يحمل القيمة (5).
15. العودة إلى المدى العام، حيث يوجد مُتغيِّر يُسمّى (x)، وتبلغ قيمته (7).
16. اكتمال طباعة قيمة المُتغيِّر (x).

إذن، ناتج تنفيذ البرنامج هو (7).

نشاط عملي

أُجرِّب بنفسي:

ما نتيجة البرنامج الوارد فيما يلي؟ كيف أُفسِّر هذه النتيجة؟

```python
x = 6
def f():
    x = 1
f()
print(x)
```

سلاسل التوثيق

يحرص المُبرمِجون على توثيق المعلومات الأساسية (Documenting) حول الدوالِّ التي يكتبونها (مثل: الهدف من الدالَّة، ومُعامِلاتها (مدخلاتها)، وما تعمل على إرجاعه)؛ ما يُسهِّل عليهم إعادة استخدام هذه الدوالِّ.

يُمكِن توثيق الدوالِّ باستخدام الدالَّة **help** كما في الشكل (6-9) الآتي:

```
>>> help(len)
Help on built-in function len in module builtins:

len(obj, /)
    Return the number of items in a container.

>>> help(sum)
Help on built-in function sum in module builtins:

sum(iterable, /, start=0)
    Return the sum of a 'start' value (default: 0) plus an iterable of numbers

    When the iterable is empty, return the start value.
    This function is intended specifically for use with numeric values and may
    reject non-numeric types.
```

الشكل (6-12): قراءة التوثيق باستخدام الدالَّة help.

كذلك يُمكِن توثيق الدوالِّ التي يكتبها المُستخدِم بإضافة شرح داخل سلسلة تعريف الدالَّة مباشرة كما في الشكل (6-9) الآتي:

```
>>> def is_even(x):
...     """checks if x is even"""
...     return x % 2 == 0
```

الشكل (6-9): توثيق الدوالِّ بإضافة شرح داخل سلسلة تعريف الدالَّة مباشرة.

يُطلَق على هذه السلسلة اسم سلسلة التوثيق (Docstring)، ويُمكِن للغة البرمجة بايثون (Python) تعرُّفها مباشرة، وإدراك أنَّها توثيق للدالَّة؛ فما إنْ تُستخدَم الدالَّة help، حتّى يظهر الشرح الذي كتبه المُستخدِم.

تصميم البرامج باستخدام الدوالِّ

بعد تعرُّف العديد من التفاصيل المُتعلِّقة بتعريف الدوال البرمجية واستخداماتها، لا بُدَّ من استعمالها لتنظيم عملية تصميم البرامج وكتابتها؛ كي تصبح أسهل للقراءة والتصحيح والتعديل وإعادة الاستخدام.

مثال:

يطلب البرنامج الآتي إلى طالب في مدرســـة إدخال بريده الإلكتروني والبريد الإلكتروني الخاص بأحد والديه، ثمَّ يطلب إليه إعادة إدخال البريد الإلكترونـــي إنْ لم يكن البريد المُدخَل يتبع نمطًا صحيحًا.

إذا كُتِب هذا البرنامج من دون استخدام الدوالِّ، فإنَّه سيبدو على النحو الظاهر في الشكل (6-10):

```
email1 = ''
while True:
    email1 = input("Enter your email: ")

    valid = True

    if '@' not in email1 or '.' not in email1:
        valid = False
    elif email1.startswith('@') or email1.endswith('@'):
        valid = False
    elif email1.startswith('.') or email1.endswith('.'):
        valid = False
    elif email1.index('.') < email1.index('@'):
        valid = False

    if valid:
        break

    print("Invalid email!")


email2 = ''
while True:
    email2 = input("Enter your parent's email: ")

    valid = True

    if '@' not in email2 or '.' not in email2:
        valid = False
    elif email2.startswith('@') or email2.endswith('@'):
        valid = False
    elif email2.startswith('.') or email2.endswith('.'):
        valid = False
    elif email2.index('.') < email2.index('@'):
        valid = False

    if valid:
        break

    print("Invalid email!")
```

الشكل (6-10): برنامج تحقُّق من صحة البريد الإلكتروني بعد إدخاله من دون استخدام الدوالِّ البرمجية.

يُلاحَظ عند النظر مباشرة إلى هذا البرنامج وجود تكرار يُمكِن التخلُّص منه عن طريق تعريف دالَّة؛ للتحقُّق من صحة البريد الإلكتروني، ثمَّ اســـتدعاء هذه الدالَّة داخل حلقة التكرار كما في الشكل (6-11).

```python
def is_valid(email):
    if '@' not in email or '.' not in email:
        return False
    if email.startswith('@') or email.endswith('@'):
        return False
    if email.startswith('.') or email.endswith('.'):
        return False
    if email.index('.') < email.index('@'):
        return False

    return True

email1 = ''
while True:
    email1 = input("Enter your email: ")
    if is_valid(email1):
        break

    print("Invalid email!")

email2 = ''
while True:
    email2 = input("Enter your parent's email: ")
    if is_valid(email2):
        break

    print("Invalid email!")
```

الشكل (6-11): برنامج تحقُّق من صحة البريد الإلكتروني بعد إدخاله باستخدام تعريف لدالَّة برمجية جديدة.

ونظرًا إلى وجود دالَّة تتحقَّق من نمط البريد الإلكتروني؛ فإنَّه يُمكِن الآن استخدامها في أيِّ مكان للتأكُّد من البريد الإلكتروني من دون حاجة إلى إعادة كتابة ذلك الجزء مَرَّة أُخرى.

نشاط فردي

أُجرِّب بنفسي:

أُدخِل البرنامج المُبيَّن في الشكل (6-11) في بيئة بايثون (Python)، ثمَّ أستخدِمه في التحقُّق من صحة البريد الإلكتروني لثلاثة من زملائي.

إضاءةٌ

إذا أردنا تغيير الطريقة التي نستخدمها في التحقُّق من صحة البريد الإلكتروني، فإنَّنا نكتفي بتغيير هذه الطريقة داخل الدالَّة، بِغَضِّ النظر عن عدد المَرّات التي استخدمنا فيها هذه الدالَّة. أمّا في حال عدم وجود الدالَّة فيجب علينا تغيير هذه الطريقة في كل مكان تحقَّقنا فيه من صحة البريد الإلكتروني.

إثراءٌ

يُمكِن اســـتخدام الدوالِّ البرمجية في تسهيل عملية تصميم الخوارزميات، وتجزئة المشكلة إلى أجزاء صغيرة؛ تمهيدًا لحَلِّها.

فمثلًا، إذا أراد المُســـتخدِم كتابة برنامج يُعْنى برسم أشكال كما في الشكل الآتي، تعيَّن عليه إدخال عدد المُثلَّثات، فيعمل البرنامج على رسمها:

يُمكِن تبسيط المسألة بتقسيمها إلى مسألتين؛ الأولى: رسم مُثلَّث مقلوب، والثانية: رسم عدد من المُثلَّثات المقلوبة التي يتناقص حجم كلٍّ منها.

خطوات العمل:

1. التفكير في كيفية رسم مُثلَّث مقلوب، وكتابة دالَّة تتولّى عملية الرسم.

عدد الفراغات		عدد النجمات
0	*********	9
1	.*******	7
2	..*****	5
3	...***	3
4	....*	1

الشكل 6-12: شكل مثلث مقلوب يتألَّف من أسطر

يُبيِّن الشكل (6-12) أنَّ كل مُثلَّث مقلوب يتألَّف من أسطر، وأنَّ كل سطر يحوي عددًا من الفراغات والنجوم، وأنَّ عدد الفراغات يزداد في كل سطر بمقدار (1)، وعدد النجوم يقل بمقدار (2).

2. تعريف دالَّة تستقبل حجم المُثلَّث، مُمثِّلًا بعدد النجوم في السطر العلوي، وتطبع هذه النجوم وَفقًا للعلاقة السابقة.

```
def draw_triangle(n):
  spaces = 0
  while n > 0:
      print(' '*spaces + '*' * n)
      spaces += 1
      n -= 2
```

أَعُدُّ حتّى نصل إلى نجمة واحدة في السطر

3. التفكير في كيفية رسم مجموعة من المُثلَّثات التي تتناقص حجومها، بَدْءًا بالبحث في كيفية استنتاج حجم المُثلَّث الأكبر من عدد المُثلَّثات. وبالنظر إلى الأمثلة السابقة، يتبيَّن أنَّ حجم المُثلَّث الأصغر هو (3)، وأنَّ حجوم المُثلَّثات تزداد بمقدار (2) وصولًا إلى عدد المُثلَّثات المطلوب؛ لذا يُمكِن حساب حجم أكبر مُثلَّث باستخدام المعادلة الآتية:

- size = 1 + 2*n، حيث n عدد المُثلَّثات المطلوب.
- استدعاء دالَّة لرسم مُثلَّث كبير الحجم، ثمَّ تقليل الحجم بمقدار (2)، ثمَّ استدعاء الدالَّة مَرَّة أُخرى لرسم مُثلَّث آخر، وهكذا حتّى يتمَّ الوصول إلى الحجم (1).

```
n = int(input("How many triangles? "))
size = 1 + 2*n
while size > 1:
    draw_triangle(size)
    size -= 2
```

عند تشغيل البرنامج، سيظهر المُثلَّث الأوَّل على النحو الذي خُطِّط له، خلافًا لبقية المُثلَّثات؛ إذ ستظهر مائلة بصورة غير صحيحة كما في الشكل الآتي:

```
*******
 *****
  ***
   *
*****
 ***
  *
***
 *
```

تشغيل البرنامج للحصول على النتيجة وارفاق صورة لها

يُعْزى سبب ذلك إلى خطأ في تصميم دالَّة رسم المُثلَّث؛ إذ لا تبدأ جميع المُثلَّثات من أوَّل السطر، ويجب إزاحة المُثلَّث إلى اليمين كلَّما كان أصغر؛ لذا يجب أوَّلًا تعديل الدالَّة على نحوٍ يُمكِّنها من تسلُّم مقدار الإزاحة (إلى جانب حجم المُثلَّث)، ثمَّ إرسال هذا المقدار إلى الدالَّة، فيبدأ المُثلَّث الأوَّل من دون أيِّ إزاحة، ثمَّ يأخذ كَمُّ الإزاحة يتزايد بمقدار (1) كلَّما قَلَّ حجم المُثلَّث.

```
def draw_triangle(n, shift):
    spaces = 0
    while n > 0:
      print(' '*shift + ' '*spaces + '*' * n)
      spaces += 1
      n -= 2
n = int(input("How many triangles? "))
size = 1 + 2*n
shift = 0
while size > 1:
    draw_triangle(size, shift)
    size -= 2
    shift += 1
```

المواطنةُ الرقميةُ:

- **المسؤولية الرقمية**: أحرص على الإحاطة بالتطوُّرات الجديدة في مجال الأمان الرقمي والتكنولوجيا، وأُطوِّر مهاراتي باستمرار.
- **المشاركة الفاعلة**: أُشارِك في المناقشات والمنتديات بفاعلية وإيجابية، وأُسهِم في بناء مجتمع رقمي صحي.
- **التعاون الإلكتروني**: أُوظِّف الأدوات الرقمية في العمل الجماعي البَنّاء والتعاون الفاعل مع الآخرين.
- **دعم المبادرات الرقمية الإيجابية**: أُشارِك في المبادرات الرقمية التي تُسهِم في تعزيز الوعي الرقمي والمواطنة الرقمية.

المشروع: تصميم برنامج وإعداده لإنشاء لعبة تخمين الأرقام باستخدام لغة بايثون (Python)/ المهمة (6).

وصلنا الآن إلى مرحلة نستطيع فيها تجميع الأجزاء المُتفرِّقة التي كتبناها سابقاً لإنهاء كتابة لعبة (نجوم وأقمار).

أُراجِع مُخطَّط سَيْر العمليات الذي أنشأْتُه في الدرس الأوَّل من هذه الوحدة، ثمَّ أُنعِم النظر في المُخطَّط الآتي الذي يعرض الصورة العامة لأهمِّ أجزاء البرنامج، وكيفية انتقاله من جزء إلى آخر.

مُخطَّط سَيْر العمليات (Flowchart) للعبة تخمين الأرقام:

1. **بداية اللعبة:**
   أ. عرض رسالة الترحيب.
2. **عرض القائمة الرئيسة:**
   أ. الخيار الأوَّل: عرض تعليمات اللعبة.
   ب. الخيار الثاني: بَدْء اللعبة.
   ج. الخيار الثالث: الخروج من اللعبة.
3. **بَدْء اللعبة:**
   أ. توليد الرقم المُضمَر.
   ب. بَدْء المحاولات (الحدُّ الأقصى هو (10) محاولات):
   - إدخال التخمين.
   - التحقُّق من صحة التخمين.
   - حساب عدد النجوم والأقمار.
   - عرض النتائج.

التحقُّق من الفوز:

- في حال الفوز: عرض رسالة التهنئة.
- في حال الخسارة: الاستمرار في المحاولات.

4. **نهاية اللعبة:**
   أ. عرض رسالة النهاية.

والآن سأبدأ تعديل برنامج اللعبة وتحسينه بناءً على ما تعلَّمْتُه في هذه الوحدة من تقنيات برمجية، وأستخدِم الدوالَّ البرمجية في تسهيل قراءة البرنامج على النحو الآتي:

1. طباعة الرسالة الترحيبية للعبة، وكذلك خيارات القائمة الرئيسة def welcome() .

مشروع

2. طباعة تعليمات اللعبة (def about()؛ ذلك أنَّ الأجزاء طويلة نسبيًّا، ووجودها داخل البرنامج قد يجعل قراءته مُهِمَّة عسيرة، ومن ثَمَّ يُمكِن عزلها في دوالَّ مُنفصِلة.

أُلاحِظ وجود أجزاء من اللعبة يجب إعادة اســـتخدامها في أكثـــر من مكان. فمثلًا، لا يُظهِر المُخطَّط الأجـــزاء المُتعلِّقة باللعب للاعبين اثنين؛ لذا يُمكِن افتراض أنَّ كتابة هذا الجزء من اللعبة سيتضمَّن الأجزاء الآتية:

1. التأكُّد أنَّ الرقم الذي أدخله اللاعب هو من الأرقام المسموح بها (يتألَّف من أربعة أعداد).
2. عَدُّ النجوم في الرقم المُدخَل.
3. عَدُّ الأقمار في الرقم المُدخَل.

أتجنَّب كتابة هذه الأجزاء في أكثر من مكان فـــي البرنامج، وذلك بتعريف دوالَّ خاصة بها، واستدعائها عند الحاجة كما يأتي:

def is_valid(guess):

استقبال سلسلة، والتأكُّد أنَّها تتألَّف من (4) أعداد، وتعيد (True) أو (False).

def count_stars(guess, secret):

استقبال الرقم المُضمَر، وتوقُّع اللاعب، وإعادة رقم يُمثِّل عدد النجوم بناءً على ذلك.

def count_moons(guess, secret):

استقبال الرقم المُضمَر، وتوقُّع اللاعب، وإعادة رقم يُمثِّل عدد الأقمار بناءً على ذلك.

والآن ســـأعمل – ضمن المجموعة– على كتابة هذه الدوالِّ (يُمكِن نسخ المقطع البرمجي الذي كتبناه سابقًا) واستخدامها، وأُحاوِل الجمع بينها، بحيث يحاكي منطق البرنامج مُخطَّط سَيْر العمليات الذي اتَّفق عليه أفراد المجموعة.

أُقسِّم البرنامج إلى دوالَّ، مثل:

welcome(), about(), is_valid(), count_moons(), count_stars() .

تلميحات:

1. لا بُدَّ من وجـــود حلقة تكرار (لانهائية) تحيط باللعبة كلها، بحيث يعود اللاعب دائمًا إلى رسالة الترحيب بعد الانتهاء من أيِّ لعبة.
2. لا داعي الآن لكتابة الجزء المُتعلِّق باللعب للاعبين اثنين؛ إذ يُمكِن تأجيل ذلك إلى وقت لاحق.
3. يُمكِن فصل الجزء المُتعلِّـــق باللعب للاعب واحد في دالَّة مُنفصِلة بهدف تنظيم البرنامج، وكذا الحال بالنسبة إلى الجزء الذي سيُكتَب لاحقًا، ويتعلَّق باللعب للاعبين اثنين.

أعمل – مع أفـــراد مجموعتي– على اختبار البرنامج بصورة شـــاملة، وأتحقَّق من تصحيح الأخطاء، وأحرص على إضافة أيِّ تحسينات وإبداعات شخصية تُثْري المشروع (اللعبة).

أُقيِّمُ تعلُّمي

المعرفة: أُوظِّف في هذا الدرس ما تعلَّمْتُه من معارف في الإجابة عن الأسئلة الآتية:

السؤال الأوَّل: أُوضِّح المقصود بكلٍّ ممّا يأتي:

1. مُعامِلات الدالَّة.

2. استدعاء الدالَّة.

3. مدى المُتغيِّر.

4. استيراد الوحدة.

السؤال الثاني: ما الفرق بين المدى العام والمدى المحلي للمُتغيِّرات؟

السؤال الثالث: ما الفرق بين الدالَّتين sort وreverse والدالَّتين sorted وreversed؟

المهارات: أُوظِّف مهارة التفكير الناقد ومهارة حَلِّ المشكلات والمهارات البرمجية في الإجابة عن الأسئلة الآتية:

السؤال الأوَّل: أكتب دالَّة تستقبل (3) أرقام، وتُرجِع الرقم الوسيط، ثمَّ أكتب برنامجًا بسيطًا يستدعي هذه الدالَّة.

السؤال الثاني: أكتب دالَّة تســـتقبل رقمين؛ أحدهما يُمثِّل الطول، والآخر يُمثِّل العرض، ثمَّ أطبع مستطيلًا مرسومًا من علامة '#'، ثمَّ أكتب برنامجًا بسيطًا يستدعي هذه الدالَّة.

السؤال الثالث: أكتب دالَّة تستقبل قائمتين، وتتأكَّد أنَّ كل عنصر في القائمة الأولى يساوي العنصر المُقابِل له في القائمة الثانية (يجب أنْ تُرجِع الدالَّة (True) أو (False))، ثمَّ أكتب برنامجًا بســـيطًا يختبر هذه الدالَّة.

السؤال الرابع: أكتب برنامجًا يحتوي على دالَّة من تصميمي، ويطبع جدول الضرب لجميع الأرقام من (0) إلى (9)، بَدْءًا بطباعة جدول الصفر كاملًا، ثمَّ طباعة جدول الـرقم (1) كاملًا، وهكذا.

السؤال الخامس: أكتب برنامجًا يحتوي على دالَّة من تصميمي، ويطبع شكل المَعين كما في الأمثلة الآتية:

```
  *       *       *
 ***     ***     ***
***     *****   *****
 *      *****  *******
         ***   *******
          *     *****
                 ***
                  *
```

(تلميح: يُمكِن استخدام دالَّتين؛ واحدة لطباعة النصف العلوي، وأُخرى لطباعة النصف السفلي).

القِيَمُ والاتجاهاتُ:
أُطلِق – بالتعاون مع زملائي– مبادرة (نادي البرمجة الصيفـــي)، وأُوظِّف في ذلك الكُتيِّب الذي أنشأْتُه سابقًا، وأُسهِم في تنمية مهارات طلبة مدرستي (في الصفوف 7–10) التي تتعلَّق بأساسيات لغة البرمجة بايثون (Python).

مُلخَّصُ الوحدةِ

تعرَّفْتُ في هذه الوحدة مفهوم كلٍّ من البرمجة، والخوارزميات، والبرامج، إضافةً إلى أساسيات لغة البرمجة بايثون (Python)، وقواعد كتابة الأوامر فيها، وكيفية توظيفها في كتابة البرامج المفيدة وتنفيذها.

في ما يأتي أبرز الجوانب التي تناولتها هذه الوحدة:

- لغة البرمجة هي مجموعة من الأوامر، تُكتَب وَفق قواعد مُحدَّدة، ويراد تحويلها إلى تعليمات يُمكِن لجهاز الحاسوب تنفيذها.
- تُصنَّف لغات البرمجة إلى نوعين رئيسين، هما: لغات عالية المستوى، ولغات مُنخفِضة المستوى.
- استخدام المُبرمِجين كلًّا من المُترجِم (Compiler) والمُفسِّر (Interpreter) في تنفيذ البرامج.
- استخدام الدالَّة ()input في تمكين المُستخدِم من إدخال البيانات في البرنامج.
- اشتمال العناصر الرئيسة للغة البرمجة بايثون (Python) على كلٍّ من التعليقات (Comments) التي تُحسِّن مقروئية البرنامج، وتشرح أجزاء المقاطع البرمجية؛ والمُعرِّفات (Identifiers) التي هي أسماء مُستخدَمة للمُتغيِّرات والدوالِّ التي يجب أنْ تتبع قواعد مُعيَّنة؛ والكلمات المحجوزة (Reserved Words) التي لا يُمكِن استخدامها مُعرِّفاتٍ بسبب حجز اللغة لها؛ والثوابت (Constants) التي تظلُّ ثابتة طوال مُدَّة تنفيذ البرنامج، إضافةً إلى المُتغيِّرات التي تُعرَّف بتخصيص مساحة تخزينية في الذاكرة لقِيَم المُتغيِّرات. تشمل أنواع المُتغيِّرات في لغة البرمجة بايثون (Python) كلًّا من الأعداد الصحيحة (int)، والأعداد العشرية (Float)، والنصوص (Strings)، والقِيَم المنطقية (Booleans)، والقوائم (lists)، والصفوف، والمجموعات (sets)، والقواميس (Dictionaries).
- العوامل الحسابية (Arithmetic Operators) في لغة البرمجة بايثون (Python) والتي تشمل على كلٍّ من الجمع، والطرح، والضرب، والقسمة، وباقي القسمة، والقوَّة، والقسمة التحتية. أمّا العوامل المُستخدَمة في المقارنات (Comparison Operators) فتشمل المساواة، وعدم المساواة، وأكبر من، وأصغر من، وأكبر من أو يساوي، وأصغر من أو يساوي.
- العوامل المنطقية (Logical Operators) والتي تشمل على كلٍّ من (AND)، و(OR)، و (NOT).

أمّا العوامل المُستخدَمة في إعطاء المُتغيِّرات قِيَمًا فتشمل الإسناد الأساسي، والإضافة والإسناد، والطرح والإسناد، والضرب والإسناد، والقسمة والإسناد، وباقي القسمة والإسناد، والقوَّة والإسناد، والقسمة التحتية والإسناد. وبينما تُحدِّد أسبقية العوامل ترتيب تنفيذ العوامل في التعابير، فإنَّ ترابط العوامل يُحدِّد اتجاه تنفيذ العوامل المُتماثِلة من حيث الأسبقية.

- تعريف الكتل البرمجية بمجموعة من الجمل ذات الصلة التي تُحدَّد باستخدام المسافات الفارغة، وهي مسافات ضرورية لتحديد الكتل البرمجية في لغة البرمجة بايثون (Python).
- استعمال الجمل الشرطية في لغة البرمجة بايثون (Python) لتعليق تنفيذ أوامر مُعيَّنة بناءً على شروط يُحدِّدها المُبرمِج. والجمل الشرطية الأساسية هي: (if)، و(if elif else)، (if else). أمّا الجملة الشرطية (if) فتُستعمَل لكتابة شرط مُعيَّن، في حين تُستعمَل الجمل الشرطية (if else) و(if elif) و(if elif else) لكتابة شروط مُتعدِّدة. أمّا العوامل المنطقية مثل: (and) و (or) و(not) فتُستعمَل لربط الشروط بعضها ببعض، في حين يُعَدُّ استخدام المسافات البادئة الصحيحة مُهِمًّا لضمان عمل المقطع البرمجي بصورة صحيحة، وبيان كيفية كتابة الجمل الشرطية المُتداخِلة، وكيف يُمكِن استخدام جملة 'pass' عند الحاجة إلى ترك جملة شرطية فارغة؛ تجنُّبًا لحدوث الأخطاء.
- استعمال الحلقات في لغة البرمجة بايثون (Python) لتكرار مجموعة من الجمل مَرّات عديدة.
- يوجد نوعان رئيسان من الحلقات في لغة البرمجة بايثون (Python)، هما: حلقات 'while'، وحلقات `for` . أمّا حلقات 'while' فتُستعمَل لتكرار تنفيذ جملة أو أكثر طالما تحقَّق شرط مُعيَّن، وإلّا توقَّف البرنامج عن تنفيذ الجمل. وأمّا حلقات 'for' تُستخدَم في تنفيذ المقطع البرمجي مَرّات مُحدَّدة.
- استعمال جمل التحكُّم لضبط سَيْر تنفيذ الحلقات. أمّا جملة التحكُّم 'break' فتُستعمَل لإيقاف الحلقة عند تحقُّق شرط مُعيَّن، ثمَّ تنفيذ الجمل التي تلي الحلقة. وأمّا جملة التحكُّم 'continue' فتُستعمَل لإيقاف الدورة الحالية في الحلقة، والانتقال إلى الدورة التالية عند تحقُّق شرط مُعيَّن. كذلك يُمكِن استعمال جملة `else` مع الحلقات لتنفيذ مقطع برمجي إضافي بعد انتهاء الحلقة.
- استعمال الدالَّة ()range لإنشاء سلسلة من الأرقام وَفقًا للمُعامِلات المُحدَّدة.
- إنشاء قوائم في لغة البرمجة بايثون (Python)، واستخدامها في تخزين مجموعة من القِيَم المُتنوِّعة. ويُمكِن إضافة عناصر إلى القائمة أو حذفها، والوصول إلى العناصر باستخدام الفهارس. كذلك يُمكِن تقطيع القوائم للوصول إلى أجزاء منها خلال مُدَد زمنية مُحدَّدة، فضلًا عن إنشاء قوائم مُركَّبة تحتوي على قوائم أُخرى؛ ما يسمح بتمثيل البيانات ثنائية الأبعاد مثل المصفوفات.

- إنشاء سلاســـل نصية والتعامل معها في لغة البرمجة بايثون (Python)، وإمكانية الوصول إلى أجزاء من هذه السلاسل باستخدام الفهارس والتقطيع، فضلًا عن دمجها، وإجراء عمليات فيها، تشمل العمليات الأساسية في القوائم والسلاسل النصية كلًّا من الجمع والتكرار. كذلك يُمكِن استعمال المُعامِل `+` لجمع القوائم، واستعمال المُعامِل `*` لتكرار العناصر، واستعمال الدوالِّ الجاهزة لمعالجة القوائم، مثل: ()len، و()max، و()min، فضلًا عن استخدام حلقات التكرار والشروط في التحقُّق من القِيَم المُخزَّنة في القوائم المُركَّبة.
- استخدام الدوالِّ الجاهزة في معالجة القوائم والسلاسل النصية في لغة البرمجة بايثون (Python)؛ إذ تُستخدَم الدالَّة ()split والدالَّة ()join في تقسيم السلاسل النصية وجمعها، في حين تُستخدَم الدالَّة ()sort والدالَّة ()reverse في ترتيب القوائم وعكس ترتيبها.
- تجزئة المشـــكلة الكبيرة إلى أجزاء صغيرة يُمكِن تحليلها وكتابتها بوصفها وحدات برمجية أو كائنات. وكذلك اســـتيراد الوحدات البرمجية (Modules) في لغة البرمجة بايثون (Python)، واســـتخدام الدوالِّ الجاهزة التي تُوفِّرها هذه الوحدات؛ فضـــلًا عن تعريف الدوالِّ البرمجية الخاصة لتنفيذ وظائف مُحدَّدة يُمكِن استخدامها في البرامج وإعادة استخدامها فيه. يضاف إلى ذلك تعرُّف المقصود بنطاق المُتغيِّرات، وما يتفرَّع منها من مُتغيِّرات محلية (Local Variables) ومُتغيِّرات عامة (Global Variables).
- توثيق البرامج باستخدام سلاسل التوثيق (Docstrings)، وكذلك استخدام التعابير الشرطية في التحقُّق من الشروط داخل الدوالِّ، وكيفية إرجاع النتائج باستخدام جملة (return).

أسئلةُ الوحدةِ

السؤال الأوَّل: أختار رمز الإجابة الصحيحة في كلٍّ ممّا يأتي:

1. إحدى الآتية تُعَدُّ لغة برمجة عالية المستوى:
   أ. لغة الآلة.
   ب. لغة التجميع.
   ج. لغة البرمجة بايثون (Python).
   د. اللغة الثنائية.

2. يعمل المُترجِم (Compiler)على ترجمة:
   أ. اللغة عالية المستوى إلى لغة الآلة دفعة واحدة.
   ب. اللغة عالية المستوى إلى لغة الآلة سطرًا بسطر.
   ج. لغة الآلة إلى لغة عالية المستوى.
   د. لغة التجميع إلى لغة عالية المستوى.

3. ناتج print(2**3) في برمجية بايثون (Python) هو:
   أ. (5).
   ب. (6).
   ج. (7).
   د. (9).

4. البيانات التي تُستعمَل لتخزين النصوص في برمجية بايثون (Python) هي من نوع:
   أ. int
   ب. (float)
   ج. (string)
   د. (Bool)

5. إحدى الجمل الآتية تتسبَّب في حدوث خطأ في برمجية بايثون (Python):
   أ. print("Hello, World!")
   ب. print("Hello" + "World")
   ج. print("Hello", "World")
   د. print("Hello" + 2)

6. يُمكِن إنشاء قائمة في برمجية بايثون (Python) باستخدام:
   أ. list = []
   ب. list = {}
   ج. list = ()
   د. list = ""

7. تعمل الدالَّة ()len في برمجية بايثون (Python) على:
   أ. إيجاد عدد الأحرف في سلسلة نصية.
   ب. إيجاد عدد العناصر في قائمة ما.
   ج. إيجاد عدد الأحرف في سلسلة نصية، وإيجاد عدد العناصر في قائمة ما.
   د. لا شيء ممّا ذُكِر.

8. الكلمة المفتاحية التي تُستعمَل لبدء الدالَّة في برمجية بايثون (Python) هي:
   أ. fun
   ب. def
   ج. function
   د. define

9. ناتج print(type((10)) في برمجية بايثون (Python) هو:
   أ. <'class 'str> .
   ب. <'class 'int>
   ج. <'class 'float>.
   د. <'class 'bool>

10. إحدى الآتية تُمثِّل الطريقة الصحيحة لبَدْء حلقة (for) في برمجية بايثون (Python):
   أ. for (i = 0; i < 10; i++)
   ب. for i in range(10)
   ج. for i in 0 to 10
   د. loop i in range(10)

11. الوظيفة التي تؤدّيها جملة التحكُّم (break) في حلقة (for) هي:
   أ. إنهاء التكرار الحالي، وبَدْء تكرار جديد.
   ب. إنهاء الحلقة بصورة كاملة.
   ج. تجاوز التكرار الحالي.
   د. إعادة الحلقة إلى وضع البداية.

12. ناتج print("Hello" + "World") في برمجية بايثون (Python) هو:
    أ. HelloWorld.
    ب. Hello World.
    ج. Hello+World.
    د. HelloWorld.

13. ناتج 5 % 2 في برمجية بايثون (Python) هو:
    أ. 2 (2).
    ب. (2.5).
    ج. (1).
    د. (0,5).

السؤال الثاني: أُميِّز الجمل الصحيحة من الجمل غير الصحيحة في ما يأتي:

1. في لغة البرمجة بايثون (Python)، تُستعمَل عبارة (if) لإنشاء حلقة.
2. يُمكِن للقوائم في لغة البرمجة بايثون (Python) تخزين عناصر تحوي أنواعًا مختلفةً من البيانات.
3. العامل= في لغة البرمجة بايثون (Python) يُستخدَم في المقارنة بين قيمتين.
4. يُستخدَم العامل =+ في إضافة قيمة إلى أحد المُتغيِّرات، وإسناد النتيجة إلى هذا المُتغيِّر.
5. تُستخدَم الكلمة المفتاحية (elif) في لغة البرمجة بايثون (Python) للتعامل مع شروط مُتعدِّدة.
6. يجب دائمًا أنْ تعيد الدوالُّ قيمة في لغة البرمجة بايثون (Python).
7. تُستعمَل علامات الاقتباس الفردية والمزدوجة لتعريف سلسلة نصية في لغة البرمجة بايثون (Python).
8. تتطلَّب حلقة (for) في لغة البرمجة بايثون (Python) وجود مُتغيِّر فهرسة صريح.
9. في لغة البرمجة بايثون (Python)، يكون ناتج كلٍّ من 3 * 2 و3 ** 2 مُتماثِلًا.
10. تبدأ التعليقات في لغة البرمجة بايثون (Python) بالرمز / / .

السؤال الثالث: أملأ الفراغ بما هو مناسب في الجمل الآتية:

1. ..... : عامل يُستعمَل لجمع رقمين في لغة البرمجة بايثون (Python).
2. .....: مجموعة من العناصر، مُرتَّبة وقابلة للتغيير في لغة البرمجة بايثون (Python).
3. .....: الكلمة المفتاحية لتعريف دالَّة في لغة البرمجة بايثون (Python).
4. تُستعمَل الدالَّة ..... لطباعة المخرجات في لغة البرمجة بايثون (Python).

5. تُستخدَم ..... في تكرار مجموعة من الجمل.
6. في لغة البرمجة بايثون (Python)، تستمر حلقة (while) في التنفيذ ما دام ..... صحيحًا.
7. في لغة البرمجة بايثون (Python)، تُستعمَل الدالَّة ..... لتحويل قيمة إلى عدد صحيح.
8. يُطلَق على الخطأ الناجم عن صياغة غير صحيحة في لغة البرمجة بايثون (Python) اسم .....
9. .....: الكلمة المفتاحية لاستيراد وحدة في لغة البرمجة بايثون (Python).
10. يُستعمَل المُعامِل ..... لدمج النصوص في لغة البرمجة بايثون (Python).

السـؤال الرابع: أكتب برنامجًا بلغة البرمجة بايثون (Python)، يأخذ رقمًا بوصفه مدخلًا، ويطبع مُربَّعه.

السـؤال الخامس: أكتب برنامجًا بلغة البرمجة بايثون (Python)، يستخدم حلقة (for) في طباعة الأرقام من (1) إلى (10).

السؤال السـادس: أكتب برنامجًا بلغة البرمجة بايثون (Python)، يُستخدَم في حساب مجموع كل الأعداد الزوجية التي تقع بين العدد (1) والعدد (50).

السؤال السابع: أكتب برنامجًا بلغة البرمجة بايثون (Python)، يُدخِل قائمة من الأرقام، ويطبع أكبر رقم فيها.

السؤال الثامن: في ما يأتي مجموعة من المقاطع البرمجية المكتوبة بلغة البرمجة بايثون (Python). أتتبَّع الأوامر في هذه المقاطع، وأكتشف الأخطاء الموجودة في البرنامج من دون تنفيذه، ثمَّ أقترح طرائق لتصحيحها:

```
x = input("Enter a number: ")
if x > 10:
    print("x is greater than 10")
else:
    print("x is less than or equal to 10")
1st_number = int(input("Enter first number: "))
2nd_number = int(input("Enter second number: "))
sum = first_number + second_number
print("The sum is:", sum)
code 8-2
i = 1
for i <= 10:
    print(i)
    i += 1
```

السؤال التاسع: أُعدِّل المقطع البرمجي الآتي؛ لكي يتمكَّن البرنامج من قبول ما يُدخِله المُستخدِم من مدخلات، ثمَّ يطبع عبارة تُبيِّن نوع العدد (فردي أو زوجي):

```
number = int("Enter a number: ")
if number % 2 == 0:
    print("The number is even")
else:
    print("The number is odd")
```

السؤال العاشر: أكتب برنامجًا بلغة البرمجة بايثون (Python)، تُستخدَم فيه الدوالُّ (Functions) لتنفيذ مجموعة من العمليات الحسابية (الجمع، الطرح، الضرب، القسمة) بناءً على مدخلات المُستخدِم.

السؤال الحادي عشر: أكتب برنامجًا بلغة البرمجة بايثون (Python) لاختبار قوَّة كلمات المرور بناءً على مجموعة من المعايير، مثل: الطول، ووجود الحروف الكبيرة والحروف الصغيرة، والأرقام، والرموز الخاصة، علمًا بأنَّ البرنامج سيطلب من المُستخدِم إدخال كلمة المرور، ثمَّ يتحقَّق من درجة قوَّتها وتعقيدها، ثمَّ يعرض نتيجة الاختبار.

تقويمٌ ذاتيٌّ (Self Evaluation)

بعدَ دراستي هذهِ الوحدةَ، اقرأُ الفقراتِ الواردةَ في الجدولِ الآتي، ثمَّ أضعُ إشارةَ (✔) في العمودِ المناسبِ:

مؤشرات الأداء	نعمْ	لا	لستُ مُتأكِّدًا
أُعرِّف المقصود بلغة البرمجة.	○	○	○
أُعدِّد بعض لغات البرمجة التي تختلف في مزاياها ووظائفها.	○	○	○
أُقارِن بين لغة البرمجة الكتلية ولغة البرمجة النصية.	○	○	○
أُقارِن بين لغات البرمجة عالية المستوى ولغات البرمجة مُنخفِضة المستوى.	○	○	○
أُوضِّح العلاقة بين الخوارزميات والبرمجة.	○	○	○
أُمثِّل البرامج بالخوارزميات ومُخطَّطات سَيْر العمليات.	○	○	○
أُعرِّف النموذج الأوَّلي للبرنامج.	○	○	○
أُبيِّن قواعد كتابة الجملة البرمجية بلغة البرمجة بايثون (Python).	○	○	○
أُوضِّح العناصر الآتية للغة البرمجة بايثون (Python): الثوابت، المُتغيِّرات، الرموز، التعابير، العلاقات.	○	○	○
أُجري عمليات حسابية على التعابير الحسابية.	○	○	○
أكتب العلاقات والعبارات الحسابية والعبارات المنطقية باستخدام لغة البرمجة بايثون (Python).	○	○	○
أكتب جملًا شرطيةً مُركَّبةً ومُترابِطةً من خلال المُعامِلات المنطقية (مثل: And، Or، Not) باستخدام لغة البرمجة بايثون (Python).	○	○	○
أستخدِم لغة البرمجة بايثون (Python) في تنفيذ مجموعة من الأوامر.	○	○	○

مؤشرات الأداء	نعمْ	لا	لسْتُ مُتأكِّدًا
أكتب جمل التحكُّم في برنامج ما باستخدام الحلقات (مثل: For، While) في برمجية بايثون (Python).	○	○	○
أستخدِم أكثر الهياكل البرمجية مناسبة (مثل: الحلقات، والجمل الشرطية) في حَلِّ مشكلات مُعيَّنة بكفاءة.	○	○	○
أكتب شيفرة برمجية على نحوٍ يُسهِّل على الآخرين قراءتها وفهمها.	○	○	○
أُوضِّح مفهوم المُتغيِر (قائمة)، وأُبيِّن استخداماته في البرمجة.	○	○	○
أُنشِئ مختلف أنواع القوائم (مثل: المُتسلسِلة، والمُتغيِّرة، والمُركَّبة)، ثمَّ أستخدِمها في تخزين مجموعة مُتنوِّعة من القِيَم.	○	○	○
أُحدِّد كيف يُمكِن وضع مجموعة من القِيَم في قائمة واحدة.	○	○	○
أُوضِّح الأنواع المختلفة للقوائم (المُتسلسِلة، المُتغيِّرة، المُركَّبة)، ثمَّ أُنفِّذ عمليات مختلفة فيها، مثل: الإضافة، والحذف، والفرز، والتكرار.	○	○	○
أستخدِم لغة البرمجة بايثون (Python) في تمثيل مختلف أنواع القوائم، وتخزين البيانات المُتنوِّعة فيها.	○	○	○
أُعرِّف كلًّا من الوحدات البرمجية (Modules)، والكائنات البرمجية (Objects).	○	○	○
أُجزِّئ المشكلة إلى أجزاء صغيرة، ثمَّ أُصمِّم كل جزء منها، وأُبرمِجه.	○	○	○
أُحدِّد الطريقة الفضلى لتمثيل أجزاء المشكلة في صورة جمل برمجية، أو روتين فرعي، أو وحدات، أو كائنات.	○	○	○
أستخدِم لغة البرمجة بايثون (Python) في استدعاء روتين فرعي جاهز بناءً على وقوع حدث مُحدَّد.	○	○	○

تعليماتٌ للمراجعةِ والتحسينِ: إذا اخترْتُ (لا) أوْ (لسْتُ مُتأكِّدًا) لأيٍّ منَ الفقراتِ السابقةِ، فأتَّبِعُ الخطواتِ الآتيةَ لتجنُّبِ ذلكَ:

- أُراجِع المادة الدراسية؛ بأنْ أُعيد قراءة المحتوى المُتعلِّق بالمعيار.
- أطلب المساعدة؛ بأنْ أُناقِش مُعلِّمي/ مُعلِّمتي أو زملائي/ زميلاتي في ما تعذَّر عليَّ فهمه.
- أستخدِم مراجع إضافية؛ بأنْ أبحث عن مراجع أُخرى مثل الكتب، أو أستعين بالمواقع الإلكترونية الموثوقة التي تُقدِّم شرحًا وافيًا للموضوعات التي أَجِد صعوبة في فهمها.

تأمُّلاتٌ ذاتيةٌ

عزيزي الطالبَ/ عزيزتي الطالبةَ:
التأمُّلاتُ الذاتيةُ هيَ فرصةٌ لتقييمِ عمليةِ التعلُّمِ، وفهمِ التحدِّياتِ، وتطويرِ اســـتراتيجياتٍ لتحسينِ عمليةِ التعلُّمِ مستقبلًا. أملأُ الفراغَ في ما يأتي بالأفكارِ والتأمُّلاتِ الشخصيةِ التي يُمكِنُ بها تحقيقُ أفضلِ استفادةٍ منَ التجربةِ التعليميةِ:

تعلَّمْتُ في هذهِ الوحدةِ:

يُمكِنُني أنْ أُطبِّقَ ما تعلَّمْتُهُ في:

الصعوباتُ التي واجهْتُها أثناءَ عمليةِ التعلُّمِ:

ذلَّلْتُ هذهِ الصعوباتِ عنْ طريقِ:

يُمكِنُني مستقبلًا تحسينُ:

الوحدة
2

الحوسبة والحياة
(Computing and Life)

نظرة عامة على الوحدة

سأتعرَّف في هذه الوحدة مجموعة من القضايا البيئية والاجتماعية المُرتبِطة بالحوسبة، بما في ذلك مفهوم الحوسبة الخضراء، وأهميتها، وكيف يُمكِن الإسهام في تطبيقها. وكذلك مفهوم النفايات الإلكترونية، والطرائق الصحيحة للتخلُّص منها، إضافةً إلى الأدوات الحاسوبية الصديقة للبيئة.

سأتعرَّف أيضًا أهمية تطبيقات الحاسوب في الحياة الاجتماعية والاقتصادية، مثل: التعلُّم الإلكتروني، والتعليم عن بُعْد، واستخدام الحاسوب في مجال الصحة، والتسوُّق والتسويق الإلكتروني، والحكومة الإلكترونية. كذلك سأتعرَّف تطبيقات الحاسوب المُتقدِّمة في مجال صناعة الأفلام، والتصميم ثلاثي الأبعاد، والرسوم المُتحرِّكة، والطباعة ثلاثية الأبعاد، وكيفية استعمالها لدعم الابتكار والإبداع في مختلف المجالات.

يُتوقَّع منّي في نهاية الوحدة أنْ أكون قادرًا على:

- تعريف الحوسبة الخضراء، وبيان أهميتها.
- الإسهام في تطبيق الحوسبة الخضراء عمليًّا.
- تعريف النفايات الإلكترونية.
- توضيح طرائق التخلُّص الآمنة من النفايات الإلكترونية.
- ذكر بعض الأدوات الحاسوبية الصديقة للبيئة.
- استخدام تطبيقات الحاسوب في مجال الصحة، والتعليم، والاقتصاد، والحياة.
- توضيح أثر استخدام تطبيقات الحاسوب في مجال التعليم، والصحة، والاقتصاد.
- بيان أهمية تطبيقات الحاسوب (صناعة الأفلام، التصميم ثلاثي الأبعاد، الرسوم المُتحرِّكة، الطباعة ثلاثية الأبعاد، الوسائط المُتعدِّدة) في تنفيذ المشروع.

Python

Canva

Google Forms

مُنتَجات التعلُّم: (Learning Products)

تصميم مشروع ريادي رقمي، يقوم على استخدام أحد تطبيقات الحاسوب، ويتناول القضايا البيئية والاجتماعية المُتعلِّقة بالحوسبة وأثرها في الفرد والمجتمع.

مشروع

أختار مع أفراد مجموعتي أحد المشروعين الآتيين لتنفيذه في نهاية الوحدة:

المشروع الأوَّل: تنظيم حملة تثقيفية عن الحوسبة الخضراء؛ لتعزيز وعي المجتمع المدرسي بأهمية الحوسبة الخضراء.

المشروع الثاني: تنظيم مسابقة تحمل عنوان (البرمجة الخضراء)، وتشترط كتابة تعليمات برمجية - باستخدام لغة البرمجة بايثون (Python) - تهدف إلى الحدِّ من استهلاك الطاقة، والتركيز على تحسين كفاءة المقطع البرمجي، والتقليل من استخدام الموارد الحاسوبية.

MS Words

Gantt Charts

الأدوات والبرامج: (Digital Tools and Programs)

Canva, Paython, Google Forms, MS Words, Gantt Charts,

التفكير الحاسوبي، التعاون الرقمي، الابتكار العالمي، التصميم الرقمي، التعليم المستمر، التواصل الرقمي.

فهرس الوحدة

- الدرس الأوَّل: الحوسبة الخضراء (Green Computing).
- الدرس الثاني: النفايات الإلكترونية (Electronic Waste).
- الدرس الثالث: تطبيقات الحاسوب في الحياة اليومية (Computer Application in our Daily Life).

الدرسُ الأوَّلُ

الحوسبة الخضراء
(Green Computing)

مُنتَجاتُ التعلُّمِ
(Learning Products)

تحديد الفكرة الرئيسة للمشروع الريادي الرقمي الذي يتناول القضايا البيئية والاجتماعية المُتعلّقة بالحوسبة، ثمَّ إعداد خُطَّة مُفصَّلة للمشروع، تشمل تحديد الأهداف، والأدوات اللازمة، وتوزيع الأدوار، ووضع جدول زمني مُفصَّل لتنظيم جميع الخطوات اللازمة لتنفيذ المشروع بفاعلية.

الفكرة الرئيسة:

تعرُّف مفهوم الحوسبة الخضراء، وبيان أهميتها في الحياة، والإسهام في تطبيقها عمليًّا.

المفاهيم والمصطلحات:

الحوسبة الخضراء (Green Computing)، نجمة الطاقة (Energy Star).

نتاجات التعلُّم (Learning Objectives):

- أُعرِّف مفهوم الحوسبة الخضراء.
- أُبيِّن أهمية الحوسبة الخضراء وفوائدها ومزاياها.
- أُسهِم في تطبيق الحوسبة الخضراء في حياتي اليومية.

أصبحت وسائل التكنولوجيا جزءًا لا يتجزَّأ من حياتنا اليومية، وغلب استخدام أدواتها على معظم أنشطتنا وممارساتنا الحياتية؛ فهل يُمكِن للتكنولوجيا بأدواتها ووسائلها أنْ تكون ضارَّة أو تُؤثِّر سلبًا في البيئة؟

نشاط تمهيدي

هل يُمكِن لاستخدام أجهزة الحاسوب والإلكترونيات المختلفة في أنشطتي اليومية أنْ يُسهِم في زيادة التلوُّث البيئي؟ كيف يُمكِن لأنشطتي الحياتية أنْ تزيد من نِسَب التلوُّث البيئي؟ سأفكر في هاتين المسألتين، ثمَّ أُشارِك زملائي/ زميلاتي في أفكاري.

الحوسبة الخضراء: تعريفها، وأهميتها (Green Computing: Definition and Importance).

يسود اعتقاد بين الناس أنَّ أجهزة الحاسوب لا تضرُّ بالبيئة، وأنَّها تستهلك كمِّيات قليلة من الطاقة. وهذا الاعتقاد غير صحيح؛ فهي قد تلحق ضررًا كبيرًا بالبيئة، وتُضاعِف من مشكلة التلوُّث البيئي؛ إذ أشارت بعض الدراسات إلى أنه من بين 250 مليار دولار تُنفَق سنويًّا على تشغيل أجهزة الحاسوب في مختلف أنحاء العالَم ما نسبته 15٪ فقط من الطاقة هو الذي يُستهلَك في العمليات الحاسوبية الفعلية، في حين تُهدَر بقيَّة الطاقة أثناء عدم استخدام أجهزة الحاسوب، وتركها في وضع التشغيل. ولا شكَّ في أنَّ هذه الطاقة المُستهلَكة تُعَدُّ سببًا رئيسًا لانبعاثات غاز ثاني أكسيد الكربون. ومن ثَمَّ، فإنَّ الطاقة المُدَّخَرة في أجهزة الحاسوب وعمليات الحوسبة تُؤدّي - في حال استخدامها- إلى تلويث البيئة بأطنان من انبعاثات الكربون سنويًّا.

تعريف الحوسبة الخضراء

تُعرَّف الحوسبة الخضراء بأنَّها الاستخدام البيئي المسؤول لأجهزة الحاسوب والموارد التكنولوجية ذات الصلة، الذي يَحُدُّ من التأثير السلبي لتكنولوجيا المعلومات والاتصالات في البيئة. وتحقيقًا لهذا الهدف؛ تُستخدَم أفضل الطرائق والوسائل في تصميم أجهزة الحاسوب والخوادم، وتصنيعها، وإعادة تدويرها؛ ما يُقلِّل من آثارها الضارَّة بالبيئة.

يُطلَق على الحوسبة الخضراء أيضًا اسم التكنولوجيا الخضراء (Green IT)، أو التكنولوجيا المستدامة (Sustainable IT).

أهمية الحوسبة الخضراء

تُسهِم الحوسبة الخضراء في تقليل استهلاك الطاقة، وتَحُدُّ من انتشار النفايات الإلكترونية؛ ما يُفْضي إلى خفض التكاليف التشغيلية، وتعزيز مبدأ الاستدامة البيئية. ولهذا تعمل الحوسبة الخضراء على تحسين كفاءة الطاقة، واستخدام مصادر الطاقة المُتجدِّدة، وتدوير النفايات الإلكترونية. وهي تهدف إلى الحدِّ من نِسَب الانبعاثات الكربونية الناجمة عن استخدام تكنولوجيا المعلومات؛ ما يُسهِم في حماية البيئة، وإنتاج تكنولوجيا نظيفة ومستدامة وصديقة للبيئة.

يُمكِن إجمال العوامل الرئيسة التي تَحْكُم عمل الحوسبة الخضراء في ما يأتي:

1. **كفاءة الطاقة (Energy Efficiency)**: يتمثَّل ذلك في تصنيع أنظمةٍ لتكنولوجيا المعلومات مُوفِّرةٍ للطاقة الكهربائية، مثل: الأجهزة الحاصلة على تصنيف نجمة الطاقة (Energy Star)، ومصادر الطاقة المُتجدِّدة، والبرمجيات التي تستهلك قليلًا من الطاقة.

2. **ترشـــيد الموارد (Resource Reduction)**: يتمثَّل ذلك في تقليل اســـتخدام المواد الخطرة والموارد غير المُتجدِّدة، وتعزيز فكرة إعادة التدوير.
3. **افتراضية الخوادم (Virtualization Servers)**: يتمثَّل ذلك في اســـتعمال خادم مادي واحد لإدارة أنظمة التشـــغيل المُتعدِّدة واستخدام التجزئة الافتراضية لتشغيل مجموعة متنوعة من التطبيقات على الخادم نفســـه وتقليل عدد الخوادم في مراكز البيانات، ما يؤدي إلى ترشيد استهلاك الطاقة.
4. **الحوسبة الســـحابية (Cloud Computing)**: يتمثَّل ذلك في استخدام الموارد المشتركة في مراكز البيانات المركزية على السحابة الإلكترونية؛ ما يُوفِّر كثيرًا من الطاقة مقارنةً باستخدام الخوادم الفردية ومراكز البيانات الفعلية.
5. **تصميم مراكز البيانـــات (Data Center Design)**: يتمثَّل ذلك في تصنيع أنظمة تبريد مُوفِّرة للطاقة، وإعداد ترتيبات أفضل للخوادم، وتوزيع الطاقة على نحوٍ يَحُدُّ من استهلاكها.
6. **إدارة النفايات الإلكترونيـــة (E-waste Management)**: يتمثَّل ذلك في التخلُّص الآمن من النفايات الإلكترونية، وإعادة تدويرها؛ ما يَحول دون تلويث البيئة بالمُكوِّنات الخطرة، مثل المعادن الثقيلة.
7. **المشتريات المستدامة لتكنولوجيا المعلومات (Sustainable IT Procurement)**: يتمثَّل ذلك في شراء الأنظمة التكنولوجية الصديقة للبيئة، مثل: الأجهزة المُوفِّرة الطاقة، والأجهزة التي تحوي على القليل من المُكوِّنات الخطرة.
8. **العمل عن بُعْد، والتعاون الافتراضي (Telecommuting and Virtual Collaboration)**: يتمثَّل ذلك في تقليص عمليات السفر والتنقُّل؛ ما يُفْضي إلى تحجيم البصمة الكربونية وخفضها.
9. **كفاءة البرمجيات (Software Efficiency)**: يتمثَّل ذلك في ابتكار حلول برمجية تُستخدَم فيها الموارد بأكثر الطرائق فاعلية.
10. **دعم مصادر الطاقة المُتجـــدِّدة (Promotion of Renewable Energy Sources)**: يتمثَّل ذلك في اســـتخدام موارد الطاقة المُتجدِّدة، والإفادة منها في تشـــغيل البِنية التحتية لتكنولوجيا المعلومات.

طرائق تطبيق الحوسبة الخضراء

تتعدَّد أوجه تطبيق الحوســـبة الخضراء، ويُمكِن إجمالها في ثلاثة مستويات؛ الأوَّل يُمثِّله الأفراد، والثاني يُمثِّله المجتمع، والثالث تُمثِّله المؤسسات والشركات. وفي ما يأتي بيان لذلك:

1. طرق تطبيق الحوسبة الخضراء على مستوى الأفراد:

يُمكِن للأفراد الإسهام في تحسين بيئة التكنولوجيا المستدامة عالميًّا باتِّباع الخطوات الآتية:

أ. **إطفاء الأجهزة غير المُستخدَمة**: يجب فصل أجهزة الحاسوب والأجهزة الإلكترونية عن مصدر التيار الكهربائي بعد الانتهاء من استخدامها؛ للحدِّ من استهلاك الطاقة.

ب. **ضبط الأجهزة على وضع السكون**: يجب برمجة الأجهزة على وضع السكون في حال عدم استخدامها مُدَدًا طويلة؛ ما يُسهِم في ترشيد استهلاك الطاقة.

ج. **ضبط إعدادات الطاقة**: يجب اختيار الوضع المُوفِّر للطاقة في الأجهزة الإلكترونية عند ضبط إعدادات الطاقة فيها؛ ما يَحول دون استهلاك كثير من الطاقة.

د. **استخدام الأجهزة المُوفِّرة للطاقة**: يُفضَّل شراء الأجهزة التي تحمل ملصق نجمة الطاقة (Energy Star)، وتستهلك طاقة أقل، وتُحافِظ – في الوقت نفسه– على الطاقة المُستخدَمة بكفاءة عالية.

هـ. **إعادة التدوير**: يجب التخلُّص من الأجهزة الإلكترونية التالفة بصورة آمنة وصحيحة، تتمثَّل في إعادة التدوير؛ ما يُقلِّل من انتشار النفايات الإلكترونية، ويَحُدُّ من تلوُّث البيئة.

و. **التقليل من عمليات الطباعة وإعادة تعبئة أحبارها**: يُنصَح بالطباعة على وجهي الورقة، وتصغير حجم الخط عند الطباعة؛ ما يُرشِّد استهلاك الورق والحبر، علمًا بأنَّ إعادة تعبئة حبر الطابعة أفضل من شراء القطعة الخاصة بذلك (Cartridges) في تطبيق الحوسبة الخضراء.

ز. **تخطيط عمليات الشراء لأجهزة الحاسوب والأجهزة الإلكترونية**: يجب التفكير مَلِيًّا قبل شراء أجهزة الحاسوب والأجهزة الإلكترونية، والتأكُّد أنَّها تُناسِب طبيعة الاستخدام، وتفي بالغرض المنشود.

نشاط فردي

أجمع بعض المعلومات عن المراكز المحلية لإعادة تدوير الإلكترونيات، ثمَّ أُنظِّم حملة لجمع الأجهزة الإلكترونية القديمة من الطلبة ومَرافق المدرسة، وإرسالها إلى مراكز إعادة التدوير.

إضاءةٌ

نجمة الطاقة (Energy Star): برنامج حكومي أطلقته الولايات المتحدة الأمريكية عام 1992م بوساطة وكالة حماية البيئة (EPA) ووزارة الطاقة (DOE)؛ لتعزيز الكفاءة في استخدام الطاقة، والحدِّ من آثارها الضارَّة بالبيئة.

تضع الشركات الصانعة ملصق نجمة الطاقة (Energy Star) على مُنتَجاتها بعد الوفاء بالمعايير والضوابط الصارمة بهذا الخصوص؛ للدلالة على أنَّ هذه المُنتَجات تستهلك طاقة أقل، وتُسهِم في حماية البيئة من التلوُّث.

تشمل المُنتَجات الحاصلة على هذا الاعتماد كلًّا من أجهزة الحاسوب، والشاشات، والأجهزة المنزلية، وأنظمة الإضاءة، وما شابه.

2. طرائق تطبيق الحوسبة الخضراء على مستوى المجتمع:

يُمكِن تطبيق الحوسبة الخضراء على مستوى المجتمع باتِّباع الخطوات الآتية:

أ التوعية والتثقيف

ب تشجيع السياسات البيئية

ج التعاون مع المُنظَّمات البيئية

أ. **التوعية والتثقيف:** يُقصَد بذلك نشـــر الوعي بأهمية الحوسبة الخضراء وفوائدها؛ ما يُحفِّز المجتمع على اتِّخاذ خطوات فاعلة للحدِّ من الآثار التكنولوجية الضارَّة بالبيئة.

ب. **تشجيع السياسات البيئية:** يتمثَّل ذلك في دعم الأنظمة والتشريعات التي تُعزِّز استخدام التكنولوجيـــا الصديقة للبيئة، مثل القوانين الضريبية المُحفِّزة للشـــركات التي تنتهج مبادئ الحوسبة الخضراء في أعمالها وأنشطتها.

ج. **التعاون مع المُنظَّمات البيئية:** تحرص المؤسســـات والشـــركات والدوائر الحكومية والخاصة على العمل مع المُنظَّمات التي تهتمُّ بالبيئة، وتُســـهِّل سُبُل تطبيق مبادرات الحوسبة الخضراء؛ ما يُعزِّز الجهود المشتركة لتحقيق أهداف التنمية البيئية المستدامة.

نشاط جماعي

تصميم ملصقات إرشادية لأحد تطبيقات الحوسبة الخضراء.

أنشر – بالتعاون مع أفراد مجموعتي – الوعي بأهمية الحوسبة الخضراء في المجتمع، وأعمل على تصميم ملصقات إرشادية لأحد تطبيقات الحوسبة الخضراء (مثل: توفير الطاقة، وإدارة النفايات الإلكترونية) باســـتخدام برنامج (Canva)، أو غيره مـــن برامج التصميم الخاصة بإنشاء الملصقات، ثمَّ أُعلِّق الملصقات في مختلف مَرافق المدرسة؛ سعيًا لزيادة وعي الطلبة ومجتمع المدرسة بأهمية الحوسبة الخضراء.

3. طرائق تطبيق الحوسبة الخضراء على مستوى المؤسسات والشركات:

يُمكِن للمؤسسات والشركات أنْ تُسهِم في تطبيق الحوسبة الخضراء بالتزام جملة من الإجراءات، أبرزها:

أ. **تصميم الأجهزة بكفـــاءة عالية:** يؤدّي اعتماد مواصفات خاصـــة في تصميم أجهزة الحاسوب إلى ترشيد استهلاك الطاقة، وضمان عمل الأجهزة مُدَدًا طويلةً؛ ما يزيد من أَمَدِ التحديث المستمر، ويُخفِّض استهلاك الموارد بصورة كبيرة.

ب. **استخدام الطاقة المُتجدِّدة:** يؤدّي استعمال مراكز البيانات لمصادر الطاقة المُتجدِّدة إلى تقليل الاعتماد على الوَقود الأحفوري، وخفض نِسَب الانبعاثات الكربونية.

ج. **إدارة النفايات الإلكترونية:** يتمثَّل ذلك في سَـــنِّ تشريعات تُعزِّز إعادة تدوير الأجهزة الإلكترونية، وإتلافها بصورة صحيحة تَحُدُّ من آثارها الســـلبية في البيئة، إضافةً إلى استخدام المواد الخام المستدامة والقابلة لإعادة التدوير.

د. **افتراضية الخوادم:** يُقصَد بذلك تخصيص خوادم افتراضية مُتعدِّدة لجهاز واحد؛ سعيًا لتقليل استهلاك الطاقة والمساحة.

هـ. **تحسين كفاءة البرمجيات:** يتمثَّل ذلك في تطوير برمجيات تستهلك طاقة أقل، وتعمل بكفاءة أكثر، فضلًا عن إطالة عمر الأجهزة؛ لتقليل الحاجة إلى استبدالها المُتكرِّر.

و. **تصميم مبانٍ خضراء ومستدامة:** يُسهِم التصميم الجيِّد للمؤسسات والمباني في ترشيد استهلاك الطاقة، ويتمثَّل ذلك في اعتماد أنظمة حديثة للتدفئة والتبريد والتهوية، تتضمَّن استخدام ممرّات باردة أو ممرّات ساخنة بحسب الحاجة.

ز. **شـــراء الأجهزة والمعدّات التي تُرشِّد اســـتهلاك الطاقة:** يتمثَّل ذلك في شراء أجهزة حاسوب مُوفِّرة للطاقة، مثل أجهزة الحاسوب المحمولة (Laptop) التي تستهلك طاقة أقل ممّا تستهلكه الأجهزة المكتبية (Desktop).

أبحث

أبحثُ في المواقع الإلكترونية الموثوقة في شـــبكة الإنترنت عن أنواع أجهزة الحاســـوب المختلفة، وأُقارِن بينها من حيث مُعدَّل اســـتهلاكها للطاقة في الســـاعة الواحدة، ثمَّ أُشارِك زملائي/ زميلاتي ومُعلِّمي/ مُعلِّمتي في النتائج التي أتوصَّل إليها.

مُعوِّقات تطبيق الحوسبة الخضراء

تُواجِه الحوسبة الخضراء تحدِّيات عديدة تُؤثِّر سلبًا في تطبيقها، وتتمثَّل أبرزها في ما يأتي:

1. **التكلفة العالية:** تُحجِم بعض الشـــركات عن انتهاج طريق الحوسبة الخضراء بسبب التكلفة المادية المرتفعة بالنسبة إليها. فقد يتطلَّب اســـتخدام الأجهزة والتكنولوجيا المُوفِّرة للطاقة وجود استثمارات أوَّلية ضخمة؛ ما يُمثِّل عائقًا أمام الشركات والأفراد.

2. **التدريب**: يتطلَّب تطبيق مبادئ الحوسبة الخضراء تدريب الموظفين على كيفية استخدام التكنولوجيا الخضراء؛ ما يُرهِق كاهل بعض المؤسسات والشركات.
3. **التحديث المستمر**: يتعيَّن على المؤسسات والشركات – في ظلِّ التطوُّر السريع للتكنولوجيا– متابعة آخر التحديثات والتقنيات الجديدة في مجال الأجهزة الإلكترونية وأجهزة الحاسوب؛ لضمان ديمومة ترشيد الاستهلاك في الطاقة. وهذا يُحتِّم عليها الاستغناء عن الأجهزة القديمة التي لديها، وشراء أجهزة جديدة؛ ما يُمثِّل تحدِّيًا رئيسًا لها.
4. **البِنية التحتية**: قد تكون البِنية التحتية القائمة غير مُلائِمة لتطبيق مبادئ الحوسبة الخضراء؛ ما يُحتِّم على المؤسسات والشركات إدخال كثير من التعديلات والتحديثات الإضافية.
5. **ثقافة الوعي البيئي**: يُعَدُّ الجهل بأهمية الحوسبة الخضراء أحد أبرز التحدِّيات التي تَحُدُّ من تطبيق مبادئ الحوسبة الخضراء؛ إذ لا تَحْفل كثير من المؤسسات والشركات بالمشكلات البيئية (مثل التغيُّر المناخي) عند تصنيع الأجهزة التكنولوجية أو شرائها.

تطبيق الحوسبة الخضراء في الأردن

يبذل الأردن كثيرًا من الجهود الدؤوبة لتطبيق مبادئ الحوسبة الخضراء في مختلف المؤسسات والوزارات الحكومية، مثل: وزارة الاقتصاد الرقمي والريادة، ووزارة البيئة، ووزارة الطاقة والثروة المعدنية. كذلك تبنَّت العديد من الشركات في القطاع الخاص فكرة الحوسبة الخضراء، وأخذت تَغُذُّ الخُطى نحوها؛ سعيًا لتحسين كفاءة مواردها، وتقليل انبعاثات غاز الكربون، وتوفير الطاقة. وقد تمثَّل ذلك في اتِّخاذ العديد من الإجراءات والمبادرات التي تدعم الاستدامة البيئية، مثل:

1. **مبادرات التوعية**: يتمثَّل ذلك في تنظيم حملات توعية تُعرِّف الناس بأهمية الحوسبة الخضراء وفوائدها.
2. **التشريعات والسياسات**: يتمثَّل ذلك في وضع القوانين والتشريعات التي تُعزِّز استخدام التكنولوجيا الصديقة للبيئة، وتُحفِّز المؤسسات والشركات على تطبيق مبادئ الحوسبة الخضراء، بما تُقدِّمه لها من حوافز وتسهيلات.
3. **الاستثمار في الطاقة المُتجدِّدة**: يتمثَّل ذلك في تنفيذ مشروعات الطاقة الشمسية التي تُزوِّد مراكز البيانات بحاجتها من الطاقة، وتُخفِّف العِبء والضغط على الشبكة التقليدية للطاقة.
4. **التعاون مع الشركات**: يتمثَّل ذلك في تحفيز الشركات على تبنّي مبادئ الحوسبة الخضراء، عن طريق تقديم الحوافز المالية والتقنية لها؛ ما يُسهِم في تعزيز الاستدامة البيئية، ويَحُدُّ من تَبِعات البصمة الكربونية.

إضاءةٌ

ظهر في الآونة الأخيرة مصطلح يُسمّى الترميز الأخضر أو البرمجة الخضراء (Green Coding)، ويُقصَد به اعتماد التعليمات البرمجية (تُعرَف أيضًا باسم البرامج) التي تُسهِم في المحافظة على البيئة، ولا تُلحِق ضررًا كبيرًا بها. ويتمثَّل ذلك في اعتماد تعليمات برمجية فاعلة تستهلك طاقة أقل، وتحسين استخدام البيانات، وتقليل النفايات الإلكترونية.

نشاط

أبحث وأُشارِك:

توجد تقنيات عديدة يُمكِن للمُطوِّرين استخدامها في تنفيذ الأُسس التي يقوم عليها الترميز الأخضر. أبحث في المواقع الإلكترونية الموثوقة في شبكة الإنترنت عن تلك التقنيات، وعن بعض الأمثلة التي تُعزِّز هذا الجانب، ويُمكِن تنفيذها للحدِّ من الآثار التكنولوجية الضارَّة بالبيئة، ثمَّ أُشارِك زملائي/ زميلاتي ومُعلِّمي/ مُعلِّمتي في النتائج التي أتوصَّل إليها.

أُراعي ما يأتي بعد دراسة موضوع (الحوسبة الخضراء):

- **الاستخدام المسؤول للتكنولوجيا:**
  - أُطفِئ الأجهزة الإلكترونية في حال عدم اســـتخدامها، وأُفعِّل أوضاع توفير الطاقة فيها (مثل ضبطها على وضع السكون)، وأُحدِّث أنظمة التشغيل والبرامج بانتظام واستمرار؛ لتحسين كفاءة هذه الأجهزة، وتقليل استهلاك الطاقة.
  - أُعيد تدوير الأجهزة الإلكترونية القديمة بصورة صحيحة.
  - أختار الأجهزة المُوفِّرة للطاقة، وأشـــتري منها فقط ما يفي بحاجاتي؛ لتقليل استهلاك الطاقة، والحدِّ من الفضلات الإلكترونية.
- **التعليم والتوعية:**
  - أُشارِك الأصدقاء والعائلة في ما أعرفه من معلومات عن الحوسبة الخضراء.
  - أُثقِّف نفسي والآخرين بخصوص التأثير الســـلبي للتكنولوجيا في البيئة، وكيف يُمكِن التقليل من هذا التأثير.
- **الاستخدام الذكي للتكنولوجيا:**
  - أُحدِّد وقتًا لاستخدام الأجهزة الإلكترونية؛ سعيًا لتقليل استهلاك الطاقة.
  - أعتمد الحوسبة السحابية؛ لتقليل الحاجة إلى الأجهزة الفردية القوية.
  - أُشارِك في المبادرات والحملات التي تُعزِّز الوعي بأهمية الحوسبة الخضراء والاستدامة البيئية.

مشروع

المشروع: تنفيذ مشروع ريادي رقمي يتناول القضايا البيئية والاجتماعية المُتعلِّقة بالحوسبة، باستخدام إحدى تطبيقات الحاسوب/ المهمة (1).

ضمن إطار التحضيرات لإطلاق مشـــروع ريادي رقمي يتناول القضايا البيئية والاجتماعية المُتعلِّقة بالحوسبة، سأتعاون مع أفراد مجموعتي على تنفيذ المرحلة الأولى من المشروع.

المرحلة الأولى: التصميم والتخطيط.

تحديد فكرة المشـــروع: أختار فكرة مشروع ريادي رقمي، تُركِّز على أحد الموضوعات التي تتناولها الوحدة، مثل: الحوســـبة الخضراء، والنفايات الإلكترونيـــة، والبرمجة الخضراء، واستخدام تطبيقات الحاســـوب في مجال الصحة، والتعليم، والاقتصاد، والحياة. بعد ذلك أجتمع مع أعضاء الفريق لمناقشة الفكرة، وتحديد الهدف الرئيس للمشروع.

وضع خُطَّة المشـــروع: أســـتخدِم في كتابة خُطَّة المشـــروع تطبيقـــات (Office)، مثل: (Microsoft Word)، و(Google Docs)؛ على أنْ تتضمَّن الخُطَّة ما يأتي:

- **الأهداف**؛ أيْ ما نريد تحقيقه من المشروع.
- **الأدوات اللازمة**؛ أيِ الأجهزة والبرامج التي سنستخدمها في تنفيذ المشروع.
- **الجدول الزمني**؛ أيْ تخصيص الوقت اللازم لتنفيذ كل مرحلة من مراحل المشروع.
- **توزيع الأدوار**؛ أيْ تحديد المهام لكل عضو في الفريق.

أتحقَّق من توثيق جميع مراحل المشـــروع، وأحتفظ بالملفات اللازمة لاستكمال المراحل اللاحقة.

أُقيِّمُ تعلُّمي

المعرفة: أُوظِّف في هذا الدرس ما تعلَّمْتُه من معارف في الإجابة عن الأسئلة الآتية:

السؤال الأوَّل: أُوضِّح المقصود بالحوسبة الخضراء، وأُبيِّن أهميتها.

السؤال الثاني: ما الإجراءات التي أستطيع تنفيذها وحدي لتطبيق مبدأ الحوسبة الخضراء؟

السؤال الثالث: ما الأمور التي يجب على المجتمع مراعاتها في تبنّي مبادئ الحوسبة الخضرا ء؟

السؤال الرابع: أُعلِّل ما يأتي:

1. يُعَدُّ استخدام جهاز الحاســـوب المحمول (Laptop) أفضل من استخدام جهاز الحاسوب المحمول (Desktop) في تطبيق مبدأ الحوسبة الخضراء.

2. يســـاعد تصميم ممرّات باردة أو ممرّات ساخنة في المؤسســـات والمباني على تطبيق مبدأ الحوسبة الخضراء.

المهارات: أُوظِّف مهارة التواصل الرقمي ومهارة البحث الرقمي في الإجابة عن السؤال الآتي:

أكتب – بالتعاون مع أفراد مجموعتي– تقريرًا عن دور المؤسسات الحكومية والشركات العامة والخاصة في تطبيق مبدأ الحوسبة الخضراء، وأستعين لذلك بمواقع ذات صلة بالموضوع، بإشراف مُعلِّمي/ مُعلِّمتي.

القِيَم والاتجاهات:

أُنشِئ دفتر يوميات (Dairy Book) باستخدام تطبيق (CANVA)، ثمَّ أُطبِّق إجراءات الحوسبة الخضراء في المنزل والمدرسة، ثمَّ أُدوِّن ما أقوم به يوميًّا على مدار أسبوع كامل، وأُقيِّم الإجراءات التي نفَّذْتُها، وأقيس درجة تأثيرها في توفير الطاقة.

الدرسُ الثاني

النفايات الإلكترونية
(Electronic Waste)

مُنتَجاتُ التعلُّم
(Learning Products)

استكمال المرحلة التخطيطية، والتحضير لتنفيذ المشروع الريادي الرقمي الذي يتناول القضايا البيئية والاجتماعية المُتعلّقة بالحوسبة، وذلك بتجميع الموارد اللازمة.

الفكرة الرئيسة:

تعرّف مفهوم النفايات الإلكترونية (e-waste)، وطرائق التخلُّص الآمنة منها، وكذلك تعرُّف بعض الأدوات الحاسوبية الصديقة للبيئة.

المفاهيم والمصطلحات:

النفايات الإلكترونية (E-waste)، البصمة الكربونية الرقمية (Carbon Digital Footprint)، الأدوات الحاسوبية الصديقة للبيئة (Echo-Friendly Computing Tools).

نتاجات التعلُّم (Learning Objectives):

- أُعرِّف مفهوم النفايات الإلكترونية.
- أُوضِّح طرائق التخلُّص الآمنة من النفايات الإلكترونية.
- أذكر أدوات حاسوبية صديقة للبيئة.

هل سبق أنْ سمعْتُ بمفهوم النفايات الإلكترونية؟ هل تكوَّنت لديَّ فكرة عن هذا المفهوم؟ لماذا تُعَدُّ الأجهزة الإلكترونية التالفة من النفايات؟

نشاط تمهيدي

أُفكِّر في الأسئلة الآتية، ثمَّ أُشـــارِك أفراد مجموعتي في تجربتي الشخصية المُتعلِّقة بموضوع النفايات الإلكترونية:

- ماذا أفعل بالأجهزة الإلكترونية بعد تلفها أو استبدالها؟
- في رأيي، ما الطريقة الصحيحة للتخلُّص من أجهزة الحاسوب والأجهزة الإلكترونية التالفة؟
- كيف تُؤثِّر هذه الطريقة في البيئة والصحة والمجتمع؟

أستمِع لتجارب أفراد مجموعتي بهذا الخصوص، وأتبادل معهم الأفكار والمُقترَحات والحلول، ثمَّ أُدوِّن النتائج التي نتوصَّل إليها في المجموعة. بعد ذلك أُعِدُّ – بالتعاون مع أفراد مجموعتي – عرضًا تقديميًّا قصيرًا يتناول تلك النتائج.

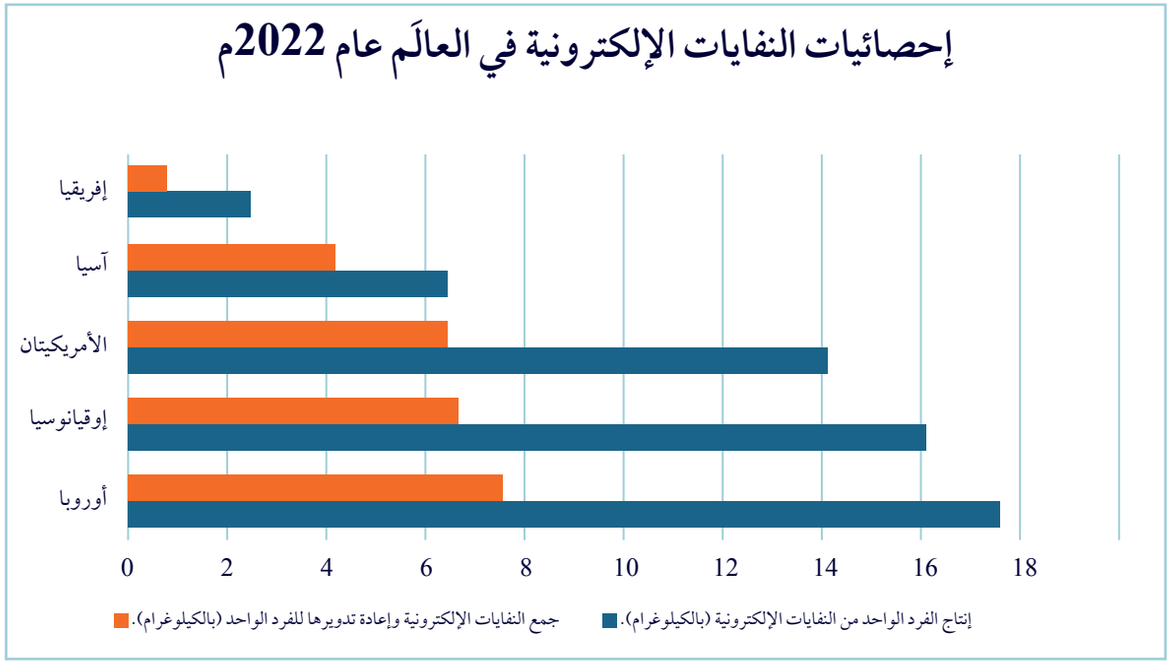
تعريف النفايات الالكترونية (E- Waste Definition)

تُعـــرَف النفايـــات الإلكترونية باســـم المُخلَّفات الإلكترونيـــة أو النفايـــات الرقميـــة؛ وهي أجهزة إلكترونية قديمة أوشـــك عمرهـــا الافتراضي على الانتهاء، واستُبدِل بها أجهزة أُخرى جديدة أو حديثة. ومن الأمثلة عليها: الحواسيب، والهواتف المحمولة، وأجهزة التلفاز، والأجهزة الإلكترونية المنزلية.

تحتوي النفايات الإلكترونية على مواد خطرة وسامَّة (مثل: الزئبق، والرصاص)، يُمكِنها أنْ تُلوِّث البيئة، وتضرَّ بالصحة العامة إذا لم يحسن التخلُّص منها بصورة آمنة. وهي تُعَدُّ مصدرًا رئيسًا ومَعينًا لا ينضب للنفايات الصُّلْبة العالمية، التي تتراكم بكمِّيات ضخمة ســـنويًّا، ولا يعاد تدوير معظمها بطرائق صحيحة؛ ما يُلحِق ضررًا كبيرًا بالبيئة.

تشـــير كثير من الدراســـات المُتخصِّصة إلى أنَّ النفايات الإلكترونية هي أسرع نموًّا بين النفايات الصُّلْبة على مســـتوى العالَم؛ إذ تزداد بمُعدَّل يفوق نموَّ السُّـــكّان بنحو ثلاثة أضعاف. وبحسب بيانات مُنظَّمة الصحة العالمية، فقد شـــهد عام 2019م تدوير أقل مـــن ربع النفايات الإلكترونية على المستوى الرســـمي في مختلف دول العالَم، علمًا بأنَّ هذه النفايات تحوي موارد قيِّمة يُمكِن استعادتها والاستفادة منها إذا أُعيد تدويرها بصورة صحيحة؛ ما يجعلها مصدرًا مُهِمًّا للدخل. غير أنَّ البلدان ذات الدخل المُنخفِض أو الدخل المُتوسِّط لا تُلقي بالًا إلى هذا الجانب، وتعاني نقصًا في القوانين وضعفًا في التدريب وخللًا في البِنية التحتية؛ ما يُعرِّض سُكّانها لمخاطر جَمَّة.

ثـــمَّ جاء تقرير الأُمم المتحدة الرابع عـــن النفايات الإلكترونية (GEM) مُبيِّنًـــا أنَّ توليد النفايات الإلكترونية وتكدُّسها ينمو بسرعة تفوق خمسة أضعاف مُعدَّل إعادة التدوير المُوثَّقة، أنظر الشكل (1-1)؛ ففي عام 2022م، أنتج العالَم قرابة (62) مليون طن من النفايات الإلكترونية، بزيادة نسبتها 82٪ على عام 2010م. ومن المُتوقَّع أنْ يصل الرقم إلى نحو (82) مليون طن بحلول عام 2030م. أمّا ما جُمِع وأُعيد تدويره من هذه النفايات فكان أقل من الربع، بما نســـبته 22.3٪ من المجموع الكلي للنفايات الإلكترونية؛ ما تســـبَّب في هدر كثير من الموارد الطبيعية التي بلغت قيمتها (62) مليار دولار، وزاد من مخاطر التلوُّث بصورة كبيرة. ولا شـــكَّ في أنَّ التحدِّيات التي تُواجِهها كثير من دول العالَم (مثل: التقدُّم التكنولوجي، وزيادة الاســـتهلاك، ودورة الحياة القصيرة للمُنتَجات) قد أسهمت في زيادة الفجوة واتِّساع الهُوَّة بين توليد النفايات والجهود المبذولة لإعادة تدويرها.



Source: The Global E-waste Monitor 2024

الشكل (1-1): إحصائيات النفايات الإلكترونية في مختلف دول العالَم عام 2022م بحسب تقرير الأُمم المتحدة (مُراقِب النفايات الإلكترونية العالمي لعام 2024م).

إثراءٌ

شهدت دورة الألعاب الأولمبية في طوكيو عام 2020م كثيرًا من التحضيرات والتجهيزات، وكان لافتًا فيها اعتماد مُقترَح صنع الميداليات من مواد أُعيد تدويرها، بوصف ذلك جزءًا من مبادرة أوسع تهدف إلى تعزيز الاستدامة البيئية والمحافظة على موارد البيئة. ومن ثَمَّ، فقد أمكن صنع الميداليات الذهبية والفضية والبرونزية من مواد توجد في النفايات الإلكترونية التي يعاد تدويرها، مثل: الهواتف المحمولة القديمة، والأجهزة الإلكترونية الصغيرة.

بدأ القائمون على هذا المُقترَح حملتهم عام 2017م، وتمكَّنوا من جمع (16.5) كلغ من الذهب، وهو ما يُمثِّل 54٪ من الكمِّية المطلوبة، و(1800) كلغ من الفضة، بما نسبته 43.9% من الكمِّية اللازمة لطلاء ميداليات أصحاب المركز الثاني في البطولة الأولمبية.

وتحقيقًا لهذا الهدف؛ فقد بدأ العمل على تفكيك الأجهزة والمعدّات، وتحويلها إلى معادن خام؛ ما زاد من حصيلة ما جُمِع من المعادن النفسية؛ إذ بلغ مجموع الكمِّية المُستخرَجة من البرونز نحو (2700) كلغ بحلول عام 2018م، في حين أسهمت التبرُّعات في زيادة كمِّيات الذهب والفضة المُستخرَجة لتصل إلى (28.4) كلغ من الذهب، و(3500) كلغ من الفضة.

إدارة النفايات الإلكترونية (E-waste Management)

تهدف إدارة النفايات الإلكترونية إلى استعادة النفايات الإلكترونية، ومعالجتها، وإعادة تدويرها، أو تجديدها؛ للاستفادة منها، واستخدامها في مختلف مناحي الحياة مَرَّة أُخرى. غير أنَّ عملية إعادة التدوير الإلكتروني تُواجِه تحدِّيًا كبيرًا؛ نظرًا إلى طبيعة هذه الأجهزة؛ فهي مُعقَّدة، ومصنوعة من الزجاج والمعدن والبلاستيك بنِسَب مُتفاوِتة.

تشمل عملية إدارة النفايات الإلكترونية المراحل الآتية:

1. **جمع النفايات؛** إذ يتمُّ تجميع النفايات الإلكترونية من مصادر مختلفة.
2. **تفكيك النفايات؛** إذ يتمُّ فصل مكونات النفايات الإلكترونية بعضها عن بعض؛ لتحديد ما يُمكِن أنْ يعاد استخدامه.
3. **تنظيف البيانات؛** أي التأكُّد أنَّ البيانات لم تَعُدْ صالحة للاستخدام.
4. **إعادة التدوير؛** أيْ فصل الأجزاء والمواد لاستخدامها في مُنتَجات جديدة.
5. **التجديد؛** أيْ إعادة استخدام الأجزاء القيِّمة لإطالة أَمَدِ عمر المعدّات الأُخرى.

تمرُّ عملية معالجة النفايات الإلكترونية بالمراحل الآتية:

1. **التفكيك؛** أيْ إزالة المُكوِّنات المُهِمَّة من النفايات الإلكترونية لتجنُّب التلوُّث بالمواد السّامَّة خلال العمليات اللاحقة.
2. **المعالجة الميكانيكية،** وهي تشـــمل عملية الســـحق وعملية الفرز للنفايات الإلكترونية؛ ما يُسهِّل استخراج المواد القابلة لإعادة التدوير، وفصل المواد الخطرة.
3. **التكرير؛** إذ يســـاعد التكرير على استعادة المواد الخام من دون إلحاق ضرر كبير بالبيئة. وفي هذه المرحلة، يتمَّ تنقية الكســـور أو تعديلها؛ اســـتعدادًا للبيع بوصفها مواد خام ثانوية، أو للتخلُّص منها بصورة آمنة.

يُذكَر أنَّ عملية التفكيك تُفْضي إلى إزالة المُكوِّنات الأساسية، في حين تؤدّي المعالجة الميكانيكية إلى فصل المـــواد القابلة لإعادة التدوير، ومعالجة المواد الخطـــرة، وتصفية الانبعاثات الغازية، ومعالجة المُخلَّفات؛ ما يَحُدُّ من تأثيرها الضارِّ بالبيئة.

أبحث

أبحثُ في المواقع الإلكترونية الموثوقة في شبكة الإنترنت عن مصطلح إعادة الاستخدام (Reuse)، ومصطلح إعادة التدوير (Recycle)، ومصطلح إعادة التصنيع (Upcycling)، ومصطلح التقليل (Reduction)، ثمَّ أُعِدُّ تقريرًا عن ذلك، وأُشارِكه زملائي/ زميلاتي في الصف.

استراتيجيات إدارة النفايات الإلكترونية

تُعَدُّ إدارة النفايات الإلكترونية عملية مُهِمَّة لضمان تطبيق مبادئ الحوســـبة الخضراء، والحدِّ من التأثير البيئي الضارِّ للأجهزة الإلكترونية المُهمَلة.

تشمل استراتيجيات إدارة النفايات الإلكترونية ما يأتي:

1. **التقليل، وإعادة الاستخدام، وإعادة التدوير**: يُقصَد بذلك تحفيز الأشخاص على تقليل استهلاكهم للأجهزة الإلكترونية، وإعادة استخدام الأجهزة قَدْر الإمكان، وتعزيز عملية إعادة التدوير المسؤولة للأجهزة التالفة أو المُستهلَكة.
2. **مسؤولية المُنتَج طويلة المدى**: يشمل ذلك التزام الشركات المُصنِّعة للأجهزة الإلكترونية بتطبيق البرامج التي تُحمِّلها مسؤولية دورة الحياة لمُنتَجاتها، بما في ذلك التخلُّص الآمن منها بعد انتهاء عمرها الافتراضي.
3. **التشريعات والتنظيمات**: يجب سَنُّ القوانين التي تدعم إدارة النفايات الإلكترونية، بما في ذلك ضوابط التخلُّص الآمن من النفايات الإلكترونية، وأهداف إعادة التدوير.
4. **التوعية العامة والتعليم**: يتمثَّل ذلك في تثقيف أفراد المجتمع، وتوعيتهم بأهمية إدارة النفايات الإلكترونية على نحوٍ مسؤول، وتعريفهم بمزايا إعادة التدوير.
5. **البِنية التحتية اللازمة لجمع النفايات الإلكترونية وإعادة تدويرها**: يكون ذلك بإنشاء نقاط تجميع مُعتمَدة للنفايات الإلكترونية.
6. **أمان البيانات**: يُقصَد بذلك مسح جميع البيانات الشخصية والبيانات المُهِمَّة قبل إعادة تدوير الأجهزة الإلكترونية.
7. **التجديد والتبرُّعات**: يُقصَد بذلك تجديد الأجهزة الإلكترونية، ثمَّ التبرُّع بها للمدارس، أو للمُنظَّمات غير الربحية، أو لفئات المجتمع المحرومة.
8. **التصميم البيئي**: يُقصَد بذلك تبنّي ممارسات التصميم البيئية التي تُسهِّل عملية إعادة التدوير.
9. **التعاون والشراكات**: يجب تعزيز أواصر التعاون بين الشركات المُصنِّعة، والتُّجّار، والمُنظَّمات ذات العلاقة، والحكومات.
10. **البحث والابتكار**: يجب دعم البحوث العلمية التي تهدف إلى تطوير تقنيات إعادة التدوير.

الآثار البيئية الناجمة عن سوء إدارة النفايات الإلكترونية

إنَّ التعامل الخطأ مع المُخلَّفات الإلكترونية، وغياب شروط السلامة العامة والوقاية الضرورية أثناء التعامل مع المواد السّامَّة في هذه المُخلَّفات؛ يُمثِّل خطرًا على الصحة، وتهديدًا للموارد الطبيعية، وبخاصة التربة والمياه.

يُبيِّن الجدول (2-1) أبرز العناصر والمواد السّامَّة الموجودة في النفايات الإلكترونية بحسب ما أوردته وزارة البيئة الأردنية.

الجدول (2-1): أبرز العناصر والمواد السّامّة في النفايات الإلكترونية.

اسم المادة السّامّة	آثارها ومخاطرها	مكان وجودها
الزرنيخ	– اضطراب في النموِّ. – أمراض القلب. – الأمراض السرطانية. – داء السُّكَّري.	– الميكروويف. – لوحات الدارات الإلكترونية. – عاكس التيار. – المُحرِّكات.
الكادميوم	– فقدان الكالسيوم. – هشاشة العظام. – تلف الرئتين. – الوفاة.	– بطّاريات الهواتف المحمولة.
الكروم	– تهيُّج الجلد. – الطفح الجلدي.	– صناعة البلاستيك.
النحاس	– التهاب الحلق والرئتين. – تلف الكبد والكُلى.	– الأسلاك النحاسية. – لوحات الدارات الإلكترونية.
الرصاص	– اضطراب في النشاط المعرفي واللفظي. – الشَّلَل. – الغيبوبة. – الوفاة.	– أجهزة الحاسوب. – الشاشات. – أجهزة التلفاز. – البطّاريات.
النيكل	– الأمراض السرطانية.	– البطّاريات القابلة للشحن.
الفضة	– مـــرضِ argyria (بُقَع زرقاء وبُقَع رمادية تنتشر على الجلد).	– الهواتف المحمولة.
البريليوم	– الأمراض السرطانية.	– الموصِلات.
البلاستيك، والبولفينيل كلوريد	– الإضرار بجهاز المناعة. – الأمراض السرطانية.	– الشاشات. – لوحات المفاتيح. – الفأرة. – جهاز الحاسوب المحمول. – مفتاح (USB).

نشاط

أُصمِّم ملصقًا للتوعية بمخاطر المواد السّـــامَّة في النفايات الإلكترونية باستخدام أحد برامج التصميم، ثمَّ أُشارِك الطلبة وأولياء الأمور في الملصق عبر الوسائل الإلكترونية المتوافرة.

الإدارة الفردية للنفايات الإلكترونية

في ما يأتي بعض النصائح التي تُسهِم في تخلُّصي من النفايات الإلكترونية بطرائق صحيحة وآمنة:

1. **الوعي بمفهوم النفايات الإلكترونية:** يتعيَّن عليَّ إدراك مخاطر النفايات الإلكترونية، مُمثَّلةً في المواد السّامَّة التي تحويها، والتي قد ينتهي المَطاف بمعظمها إلى مَدافن النفايات. ولهذا، فإنَّ تعرُّفي مُكوِّنات النفايات الإلكترونية يُعَدُّ أُولى خطوات التخلُّص منها.
2. **تقليل كَمِّ النفايات الإلكترونية:** يُمكِنني الحدُّ من النفايات الإلكترونية بشراء ما يَلزمني فقط، واختيـــار المُنتَجات طويلة الأجل، والمُنتَجات المُوفِّرة للطاقـــة، وإطالة أَمَدِ عمر الأجهزة بإصلاحها بدلًا من استبدال أجهزة جديدة بها.
3. **التعاون مع المؤسســـات والوزارات،** والمشاركة في المشروعات التي تُعْنى بتدوير النفايات الإلكترونية على المستوى المحلي.

إضاءةٌ

مشروع (**تفكيك**): مشروع استثماري أردني، أُنشِئ للتخلُّص من النفايات الإلكترونية بصورة آمنة وصحيحة.

أتعرَّف مزيدًا من التفاصيل عن هذا المشـــروع، وأزور الموقع الرســـمي الإلكتروني للمشروع؛ بمسح الرمز سريع الاستجابة (QR Code) المجاور:

أبحث

أبحثُ عن مشـــروعات محلية في محافظتي، تُعْنى بإدارة النفايات الإلكترونية، ثمَّ أُشـــارِك النتائج التي أتوصَّل إليها مع زملائي/ زميلاتي في الصف.

البصمة الكربونية الرقمية (Carbon Digital Footprint)

تُعرَّف البصمة الكربونية الرقمية بأنَّها التأثير السلبي في البيئة الناجم عن استخدام التكنولوجيا الرقمية وممارسة الأنشطة الرقمية عبر شبكة الإنترنت، مُمثَّلًا في انبعاثات الكربون، واستهلاك الطاقة؛ فكل عمل نقوم به في شبكة الإنترنت، أو في أجهزتنا الرقمية، ينتهي به الحال إلى التخزين، وهو جزء من بصمتنا الكربونية الرقمية التي تُؤثِّر سلبًا في البيئة.

مثال:

إذا اعتدْتُ مشاهدة جهاز التلفاز مُدَّة ساعة واحدة أو ساعتين يوميًّا كل عام، فهذا يعني أنَّني أستخدِم ما يكفي من الكهرباء لتشغيل ثلّاجتي مُدَّة تصل إلى نصف عام تقريبًا. وفي عام 2020م، بلغت البصمة الكربونية لإحدى القنوات ما يُعادِل تشغيل مدينة تحوي (150000) منزل.

نشاط فردي

قياس بصمتي الكربونية الرقمية (Digital Carbon Footprint).

يُمكِنني قياس بصمتي الكربونية الرقمية باتِّباع الخطوات الآتية:

1. زيارة موقع (Digital Carbon Footprint) عن طريق الرابط الإلكتروني الآتي:

https://www.digitalcarbonfootprint.eu

أو مسح الرمز سريع الاستجابة (QR Code) المجاور:

2. اختيار الجهاز الذي سأستخدِمه.
3. تعديل بيانات الاستخدام.
4. تأمُّل كمِّية غاز ثاني أُكسيد الكربون Co^2 الذي أُسهِم في إطلاقه في البيئة.

نشاط جماعي

أبحثُ – بالتعاون مع أفراد مجموعتي– في المواقع الإلكترونية الموثوقة في شبكة الإنترنت عن أدوات حاسوبية صديقة للبيئة، ثمَّ أُعِدُّ عرضًا تقديميًّا عنها باستخدام إحدى الأدوات الرقمية (مثل تطبيق العروض التقديمية (google slide)، ثمَّ أعرِضه أمام زملائي/ زميلاتي في الصف.

نشاط فردي

أفترض أنَّني أُريد الإسهام في الحدِّ من انتشار النفايات الإلكترونية؛ بأنْ أُفكِّر في إجراء تعديل على أحد مُكوِّنات الحاسوب ليصبح صديقًا للبيئة. أُشارِك أفكاري مع زملائي/ زميلاتي في الصف، ثمَّ أُناقِشهم فيها.

إضاءةٌ

التأثير السلبي في الاستهلاك والإنتاج

12 RESPONSIBLE CONSUMPTION AND PRODUCTION

يَغْلب على أنماط الاستهلاك والإنتاج اليوم الاعتماد على مصادر الطاقة التقليدية (غير المُتجدِّدة). ولا شكَّ في أنَّ الاستخدام المُفرِط للمياه والأراضي، وانبعاثات الغازات الدفيئة، وتوليد النفايات وإدارتها، والتخلُّص غير الآمن من النفايات السّامَّة؛ يُلْقي بظلاله القاتمة على البيئة.

ولهذا يجب اتِّخاذ قرارات مُهِمَّة، وتطبيق إجراءات صارمة؛ لخفض الهدر في الغذاء إلى النصف، وضمان الإدارة السليمة للنفايات الكيميائية، والحدِّ من الاستهلاك غير المسؤول، وتشجيع السياحة الخضراء.

إضاءةٌ

الأردن لإعادة تدوير أجهزة الحاسوب والأجهزة الإلكترونية: مؤسسة حديثة تهدف إلى التعامل الصحيح مع جميع أجهزة الحاســـوب والنفايات الإلكترونية التي تضرُّ بالبيئة، واستخدام طرائق احترافية لإعادة التدوير والتجديد؛ ما يُســـهِم في المحافظة على البيئة، وحمايتها من مخاطر التلوُّث.

وقد أبدت المؤسسة استعدادها لمساعدة مختلف الشركات والمؤسسات على التخلُّص من النفايات الإلكترونية بصورة آمنة، وزيادة الوعي بالآثار السلبية التي تُخلِّفها هذه النفايات على البيئة.

أتعرَّف مزيدًا من التفاصيل عن هذه المؤسسة بمسح الرمز سريع الاستجابة (QR Code) الآتي:

المواطنةُ الرقميةُ:

أُراعي ما يأتي بعد دراسة موضوع (النفايات الإلكترونية):

- **الاستخدام المسؤول للتكنولوجيا:** أحرص على شـــراء الأجهزة الإلكترونية الضرورية فقط، وأختار المُنتَجات التي تُوفِّر قَدْرًا كبيرًا من الطاقة، وتمتاز بعمرها الافتراضي الطويل.
- **إعـــادة التدوير والتبرُّع:** أتبرَّع بالأجهزة الإلكترونية التي لا تزال تعمل، أو أُعيد تدويرها بصورة صحيحة في مراكز إعادة التدوير المُعتمَدة.
- **التوعية بالمخاطر والتثقيف:** أُشارِك أفراد العائلة والأصدقاء في المعلومات المُتعلِّقة بمخاطر النفايات الإلكترونية وأهمية إعادة التدوير الآمن لها.

مشروع

المشروع: تنفيذ مشروع ريادي رقمي يتناول القضايا البيئية والاجتماعية المُتعلِّقة بالحوسبة، باستخدام إحدى تطبيقات الحاسوب/ المهمة (2).

أعمل ـ بالتعاون مع أفراد مجموعتي- على اســـتكمال تنفيذ المرحلة الأولى من المشروع باتِّباع الخطوات الآتية:

1. تجميع الموارد اللازمة لتنفيذ المشروع.
2. جمع المواد اللازمة لتنفيذ المشروع، مثل: النصوص، والصور، ومقاطع الفيديو، والأدوات التقنية.
3. عقد اجتماعات دورية مع أعضاء الفريق؛ لمتابعة سَيْر العمل في المشروع، وتبادل الأفكار.
4. التأكُّد أنَّ كل عضو في الفريق يؤدّي المهام المنوطة به وَفقًا للخُطَّة الموضوعة.

أُقيِّمُ تعلُّمي

المعرفة: أُوظِّف في هذا الدرس ما تعلَّمْتُه من معارف في الإجابة عن الأسئلة الآتية:

السؤال الأوَّل: ما المقصود بالنفايات الإلكترونية؟

السؤال الثاني: ما تأثير النفايات الإلكترونية في البيئة؟

السؤال الثالث: ما الطرائق الصحيحة والآمنة للتخلُّص من النفايات الإلكترونية؟

المهارات: أُوظِّف مهارات التفكير الناقد والتواصل الرقمي والبحث الرقمي في الإجابة عن السؤال الآتي:

بعد الاطِّلاع على مشـــروعات إدارة النفايات الإلكترونية المحلية، والتواصل مع أفراد مجموعتي ومُعلِّمي/ مُعلِّمتي، أُعِدُّ خُطَّة مشروع لإدارة النفايات الإلكترونية في المدرسة، وأُضمِّنه ما يأتي:

1. الأهداف.

2. أسماء المشاركين/ المشاركات.

3. الخُطَّة الزمنية.

4. الإجراءات.

5. توزيع المهام.

6. خطوات التنفيذ.

7. المصادر والمراجع.

8. التقييم.

القِيَمُ والاتجاهاتُ

أُنشِئ وثيقة باستخدام برمجية (Word)، ثمَّ أُدوِّن فيها الخطوات التي أَتَّبِعها يوميًّا للتقليل من بصمتي الكربونية الرقمية.

الدرسُ الثالث

تطبيقات الحاسوب في الحياة
(Computer Applications in our Daily Life)

الفكرة الرئيسة:

تعرُّف تطبيقات حاسوبية في مجال التعلُّم الإلكتروني، والتعلُّم عن بُعْد، والصحة، والتسوُّق والتسويق الإلكتروني، وغير ذلك من مختلف مجالات الحياة، وبيان أهميتها في الحياة اليومية.

المفاهيم والمصطلحات:

التعلُّم الإلكتروني (E- learning)، التعلّم عن بُعْد (Online Learning)، الحكومة الإلكترونية (E- government)، الطباعة ثلاثية الأبعاد (3D-Printing).

نتاجات التعلُّم (Learning Objectives):

- أذكر تطبيقات حاسوبية في مجال التعلُّم الإلكتروني.
- أستخدِم تطبيقات حاسوبية في مجال التعلُّم عن بُعْد.
- أذكر تطبيقات حاسوبية في مجال الصحة، وأُبيِّن أهميتها.
- أستخدِم تطبيقات حاسوبية في مجال التسوُّق والتسويق الإلكتروني.
- أُبيِّن أهمية تطبيقات الحكومة الإلكترونية في تسهيل المعاملات.
- أُوضِّح أهمية بعض تطبيقات الحاسوب في الحياة، مثل: صناعة الأفلام، والتصميم ثلاثي الأبعاد، والطباعة ثلاثية الأبعاد، والرسوم المُتحرِّكة، والوسائط المُتعدِّدة.
- أستخدِم بعض تطبيقات الحاسوب في تنفيذ مشروع ريادي.

مُنتَجاتُ التعلُّم
(Learning Products)

إنتاج المحتوى الرئيس للمشروع الريادي الرقمي الذي يتناول القضايا البيئية والاجتماعية المُتعلّقة بالحوسبة وَفقًا للخُطَّة الموضوعة، باستخدام أحد تطبيقات الحاسوب، ومراجعته، ونشره في العالَم الرقمي.

يستخدم كثير من الأشخاص تطبيقات حاسوبية مُتنوِّعة؛ كلٌّ بحسب حاجاته واهتماماته. وفي ظلِّ تطوُّر العالَم الرقمي واتِّساعه، ظهرت تطبيقات أُخرى تتســـق مع المستجدّات التكنولوجية؛ فما مستقبل تطبيقات الحاسوب؟ وكيف أتخيَّل العالَم الرقمي في المستقبل القريب والمستقبل البعيد؟

نشاط تمهيدي

أُفكِّر في الأسئلة الآتية:

- ما التطبيقات الحاسوبية التي أستخدِمها في حياتي اليومية؟
- ما مجالات الحياة التي أستخدِم فيها التطبيقات الحاسوبية؟
- كيف تُسهِم هذه التطبيقات في تسهيل شؤون حياتي؟

أُشارِك تجربتي مع زملائي/ زميلاتي في الصف، ثمَّ أُناقِشهم في تجاربهم.

تؤدّي التطبيقات الحاسوبية المختلفة اليوم دورًا مُهِمًّا في إنجاز المهام اليومية على نحوٍ أكثر سرعة وفاعلية؛ ســـواء كان ذلك في مجال التعلُّم، أو التســـوُّق، أو الصحة، أو غير ذلك من المجالات. وقد أسهمت هذه التطبيقات إسهامًا كبيرًا في تحسين مختلف مناحي الحياة، وزيادة إنتاجية الأفراد والمؤسسات، وتوفير سُبُل الراحة في العديد من جوانب الحياة اليومية.

نشاط جماعي

استكشاف التطبيقات الحاسوبية في مختلف مجالات الحياة.

أختار – بالتعاون مع أفراد مجموعتي– واحدًا من المجالات الآتية:

التعليم، الصحة، المعاملات الحكومية، التسوُّق والتسويق الإلكتروني.

ثمَّ أبحث – بالتعاون معهم– عن التطبيقات الحاســـوبية المُستخدَمة في المجال المختار، وأجمع أمثلة على تطبيقات حاسوبية شـــائعة في هذا المجال، وأُوضِّح أهمية استخدام هذه التطبيقات في المجال المختار.

بعد ذلك أُلخِّص – بالتعاون معهم– النتائج التي توصَّلنا إليها، ثمَّ أُعِدُّ معهم عرضًا تقديميًّا عن المجال الذي اخترناه، ثمَّ أعرِض نتائج البحث أمام أفراد المجموعات الأُخرى، وأُناقِشـــهم فيها.

أصبح التحوُّل الرقمي في مجال التعليم ضرورة لا مَفَرَّ منها في ظلِّ التطوُّرات التقنية المستمرة. وقد بدأ هذا التحوُّل في الظهور منذ استخدام الحاسوب في مجال التعليم خلال عقد التسعينيات من القرن الماضي، ثمَّ تزايدت أهميته أثناء جائحة كورونا التي أفضت إلى واقع جديد تطلَّب إيجاد حلول تعليمية عن بُعْد؛ لضمان ديمومة العملية التعليمية التعلُّمية.

مزايا التحوُّل الرقمي في التعليم

يُمكِن إجمال مزايا التحوُّل الرقمي في التعليم في ما يأتي:

1. **تعزيز مهارات الطلبة التقنية:**

يساعد التحوُّل الرقمي الطلبة على اكتساب المهارات التقنية اللازمة لمواكبة التطوُّرات الحديثة في سوق العمل، مثل: مهارات استخدام الحواسيب، والبرمجة، والتعامل مع البرمجيات المختلفة.

2. **تسهيل الوصول إلى المعلومة:**

يتيح استخدام التكنولوجيا الوصول إلى المعلومات بسهولة وسرعة؛ إذ يُمكِن للطلبة والمُعلِّمين/ المُعلِّمات الاطِّلاع على الموارد التعليمية عبر شبكة الإنترنت في مختلف الأحوال والأماكن والأوقات.

3. **المرونة في عملية التعلُّم والتعليم:**

يمتاز التعليم الرقمي بمرونة كبيرة، تتيح للطلبة والمُعلِّمين/ المُعلِّمات تحديد أوقات الدراسة والتدريس التي تُناسِبهم، فضلًا عن إتاحة المجال أمام الطلبة للتعلُّم بالوتيرة التي تفي بحاجاتهم، وتراعي أحوالهم؛ ما يُعزِّز جانب الفهم لديهم.

4. **الترشيد في النفقات والتكاليف:**

يمتاز التعليم الرقمي بالاعتماد على الموارد الرقمية المتوافرة في شبكة الإنترنت؛ ما يُقلِّل الحاجة إلى استخدام الكتب المدرسية والأدوات التعليمية التقليدية، ومن ثَمَّ يُقلِّل من التكاليف التي تتطلَّبها عملية التعليم.

5. **التحفيز على التفاعل والابتكار:**

تُحفِّز وسائل التكنولوجيا الحديثة الطلبة على التفاعل والمشـــاركة في العملية التعلُّمية التعليمية بطرائق جديدة ومُبتكَرة، مثل: اســـتخدام الوســـائط المُتعدِّدة، والألعاب التعليمية، والمسابقات التفاعلية.

6. **تحسين الإنتاجية:**

يُعزِّز التحـــوُّل الرقمي الإنتاجية لدى الطلبة والمُعلِّمين/ المُعلِّمـــات؛ ما يزيد من فاعلية العملية التعليمية التعلُّمية وكفاءتها.

7. **التكيُّف مع شخصية الجيل الجديد:**

يتناغم التعليم الرقمي مع الأساليب والوســـائل التربوية التي يُفضِّلها الجيل الجديد، مثل: التعلُّم الذاتي، واستخدام مقاطع الفيديو والرسوم.

أهمية التحوُّل الرقمي في التعليم

أســـهم التحوُّل الرقمي في التعليم إســـهامًا فاعلًا في إحداث تغييرات جوهرية، شملت مختلف جوانب العملية التعلُّمية التعليمية. وهذه أبرزها:

1. **استدامة التعليم**: جعل التحوُّل الرقمي التعليم متاحًا ومتوافرًا للطلبة كافَّةً في مختلف الأوقات والأحوال، لا سيَّما الطارئة منها، مثل جائحة كورونا.
2. **توفير الوقت**: أتاح التحوُّل الرقمي فـــي التعليم للطلبة والمُعلِّمين/ المُعلِّمات توفير الوقت الذي كان يُقضى في الانتقال إلى المدارس والمؤسسات التعليمية.
3. **تحســـين جودة التعليم**: أسهم اســـتخدام الأدوات التكنولوجية المُتقدِّمة في تحسين جودة التعليم، بما وفَّرته من تجارب وخبرات ومهارات تعليمية مُتنوِّعة تُناسِـــب مختلف حاجات الطلبة.

لا يقتصر التحوُّل الرقمي في التعليم على إدخال التكنولوجيا في الغرف الصفية فحسب، بل يتطلَّب انتهاج أســـاليب تعليمية جديدة تُوائِم ضرورات العصر الحديـــث، وحاجات الجيل الجديد من الطلبة؛ إذ يُمكِن باســـتخدام الأدوات الرقمية المناسبة تحسين جودة التعليم، وإعداد الطلبة إعدادًا جيِّدًا للوفاء بمُتطلَّبات سوق العمل مستقبلًا.

يوجد العديد من أدوات التحوُّل الرقمي في التعليم، ويُمكِن إجمال أبرزها في ما يأتي:

1. **أنظمة إدارة التعلُّـــم (Learning Management systems: LMS)**: تُعرَّف أنظمة إدارة التعلُّم بأنَّها برامج حاسوبية مُصمَّمة لإدارة عملية التدريب والتعليم ومتابعتها وتقييمها.
2. **المِنصّات التعليمية**: تُوفِّر هذه المِنصّات دورات تعليمية عبر شبكة الإنترنت.
3. **التطبيقات التعليمية المُحمَّلة في الهواتف والحواسيب الذكية**: تُسهِّل هذه التطبيقات عملية الوصول إلى المواد التعليمية، وتساعد الطلبة على التعلُّم الذاتي.

من الأمثلة الشائعة على التطبيقات الحاسوبية في هذا مجال التعلُّم الإلكتروني:

1. **(Google Classroom)**: مِنصَّة تعليمية تُعزِّز سُبُل التواصل والتعاون بين المُعلِّمين/ المُعلِّمات والطلبة، وتتيح للمُعلِّمين/ للمُعلِّمات إنشاء صفوف افتراضية، ودعوة الطلبة إلى الانضمام إليها. وكذلك مشـــاركة الموارد التعليمية والواجبات، وإدارة النقاشات، وإجراء التقييمات إلكترونيًّا، فضلًا عن متابعة الطلبة وتوجيههم وإرشادهم.
2. **(Moodle)**: نظـــام لإدارة التعلُّم مفتوح المصدر. وفيه يُقـــدَّم العديد من الدروس والموارد التعليمية عبر شبكة الإنترنت.
3. **(Microsoft Teams)**: مِنصَّـــة للتعلُّم الإلكتروني والتواصل بيـــن المجتمعات المختلفة. وفيها يُمكِن للمُســـتخدِم إجراء محادثات نصية ومرئية وصوتية، وعقد اجتماعات عبر شبكة الإنترنت. كذلك تتيح المِنصَّة للمُســـتخدِم مشاركة الموارد، وإدارة العديد من المهام، وهي تُعْنى أساسًا بتقديم خدمات تعليمية وتربوية.
4. **(Google Meet)**: أداة لعقد الاجتماعات والمحاضـــرات الافتراضية، وهي تدعم التفاعل المباشر بين المُعلِّمين/ المُعلِّمات والطلبة.
5. **(Coursera)**: مِنصَّة تُقدِّم دورات تدريبية عبر شبكة الإنترنت بالتعاون مع جامعات عالمية؛ ما يمنح الطلبة تعليمًا فريدًا بِغَضِّ النظر عن المكان والزمان.
6. **(Khan Academy)**: مِنصَّة تُقدِّم دورات تعليمية مجّانية عبر شـــبكة الإنترنت في مجموعة مُتنوِّعة من الموضوعات.
7. **(Kahoot)**: تطبيق يتيح للطلبة إنشاء ألعاب ومســـابقات تعليمية تفاعلية، ثمَّ مشاركتها عن طريق أجهزة الهواتف الذكية والأجهزة اللوحية وأجهزة الحاســـوب. كذلك يعرض التطبيق النتائج والترتيب العام للمتسابقين بعد كل سؤال، ويتيح للطلبة الاندماج في العملية التعلُّمية التعليمية عن طريق اللعب التفاعلي. ويُعَدُّ التطبيق أداة شائعة للتعلُّم النشط.
8. **(Flipgrid)**: تطبيق يتيح للمُعلِّمين/ للمُعلِّمات والطلبة تسجيل مقاطع فيديو قصيرة لمشاركة الأفكار والمناقشات؛ ما يُعزِّز سُبُل التفاعل والنقاش داخل الغرف الصفية الافتراضية.

9. **(Quizlet)**: أداة تعليمية تتيح للمُعلِّمين/ للمُعلِّمات والطلبة إنشاء بطاقات تعليمية، واختبارات، وألعاب تعليمية، تُحسِّن عمليتي الفهم والتذكُّر، وتدعم مجموعة مُتنوِّعة من الموضوعات.

أبحث

أبحثُ في المواقع الإلكترونية الموثوقة في شـــبكة الإنترنت عن تطبيقات حاسوبية أُخرى تُستخدَم في مجال التعليم عن بُعْد، ومجال التعلُّم الإلكتروني، ثمَّ أُشارِك ما أتوصَّل إليه من نتائج مع زملائي/ زميلاتي في الصف عن طريق اللوح التفاعلي (Jamboard).

نشاط فردي

أزور مِنصَّة التعلُّم الأردنية (JoLearn) عن طريق الرابط الإلكتروني الآتي:

https://jolearn.jo

أو مسح الرمز سريع الاستجابة (QR Code) المجاور:

ثمَّ أبحث في كيفية الدخول إلى حســـابي في الصفحة الرئيســـة للمِنصَّة؛ لاستكشاف الموارد التعليمية الإلكترونية التي تُقدِّمها المِنصَّة لي بوصفي طالبًا/ طالبةً، ثمَّ أُشارِك ما أتوصَّل إليه من نتائج مع زملائي/ زميلاتي في الصف.

نشاط فردي

التسجيل في مساق إلكتروني.

1. أختار إحدى المِنصّات التعليمية، مثل: مِنصَّـــة إدراك (Google)، أو مِنصَّة (Coursera)، أو مِنصَّة (Udacity).
2. أبحث عن مســـاق إلكتروني أو دورة تدريبية ذات صلة بموضوعات الوحدة الدراسية، مثل: الحوسبة الخضراء، وإدارة النفايات الإلكترونية.
3. أُسجِّل في المساق، ثمَّ أبدأ رحلة التعلُّم.
4. أُنهي جميع الدروس والمهام المطلوبة في المساق.
5. أُشارِك تجربتي مع زملائي/ زميلاتي في الصف.
6. أُناقِش زملائي/ زميلاتي في إيجابيات المساق والتحدِّيات التي واجهْتُها أثناء عملية التعلُّم.

يشـــهد قطاع الرعاية الصحية في العالَم تحوُّلًا تقنيًّا مُهِمًّا؛ مـــا جعل مفهوم التحوُّل الرقمي ركيزة أساسية لتطوير القطاع الصحي مستقبلًا، وغدت تطبيقات التحوُّل الرقمي في هذا القطاع أكثر تنوُّعًا وشمولًا، وهو ما أسهم بفاعلية في تحسين مستوى الرعاية الصحية، ومكَّن المستشفيات والمراكز الصحية من تقديم خدمات أكثر كفاءة وفاعلية، فضلًا عن توفير الوقت والجُهْد، وتقليل التكاليف والنفقات، وتوخّي الدِّقَّة في التشخيص والعلاج، وتقديم أفضل خدمات الرعاية الصحية، لا سيَّما في المناطق النائية.

في ما يأتي بعض الأمثلة على هذه التطبيقات:

1. **الســـجل الصحي الإلكتروني**: تعمل تطبيقات الســـجل الصحي الإلكترونـــي على تخزيـــن المعلومات الطبية ومشاركتها بين مُقدِّمي الرعاية الصحية بصورة آمنة؛ ما يتيح للأطباء الوصول السريع إلى بيانات المرضى، ويُمكِّنهم من توخّي الدِّقَّة في التشخيص واختيار العلاج المناسب.

يُعَدُّ نظام حكيم (Hakeem) في الأردن واحدًا من الأمثلة على الســـجلّات الصحية الإلكترونية؛ إذ يُوفِّر سجلّات طبية إلكترونية مُتكامِلة، تُسهِّل على الأطباء الوصول إلى معلومات المرضى على نحوٍ سريع وآمن؛ ما يزيد من دِقَّة التشخيص وفاعلية العلاج. يُســـهِم هذا النظام أيضًا في تحسين إدارة الرعاية الصحية، وتقليل الأخطاء الطبية.

2. **تطبيقات الصحة الرقمية**: تشمل تطبيقات الصحة الرقمية تتبُّع اللياقـــة البدنية، والصحة الذكيـــة، والمراقبة الذاتية للصحة؛ إذ تُمكِّن المرضى مـــن متابعة حالتهم الصحية بأنفســـهم، والتفاعل مع مُقدِّمي الرعاية الصحية بصورة أفضل. ومـــن أمثلتها: تطبيـــق (Fitbit)، وتطبيق (Apple Health) التي يتيح تتبُّع النشـــاط البدنـــي والنوم، ومراقبة الصحة العامة.

3. **الذكاء الاصطناعي والتحليل الضخم للبيانات**: يُمكِن للذكاء الاصطناعي والتحليل الضخم للبيانات مساعدة المؤسسات الصحية على تحليل كَمِّ البيانات الهائل، والكشـــف عن الأنماط الصحية المُتعدِّدة، فضلًا عن الإسهام في تحرّي

نوعية العلاج بدِقَّة، والتنبُّؤ بالأوبئة، وتحسين خدمات الرعاية الصحية المستدامة.

من الأمثلة على هذه التطبيقات: برنامج تحليل البيانات الصحية (هدى)، الذي يعمل على قراءة البيانات الضخمة وتحليلها، وتقديم الحلول والتوصيات والتقارير الدقيقة.

4. **الروبوتات والأتمتة:** تُستخدَم الروبوتات في بعض المستشفيات والعيادات لأداء مهام عِدَّة، مثل: إيصال الأدوية، وعمليات التنظيف، ومراقبة المرضى؛ ما يُقلِّل من الأخطاء البشرية، ويزيد من جودة الخدمات المُقدَّمة، لا سيَّما في ظلِّ استخدام الروبوتات الجراحية في العمليات الدقيقة والعمليات المُعقَّدة.
5. **الاستشارات عبر شبكة الإنترنت، والتطبيب عن بُعْد:** تُوفِّر التقنيات الرقمية استشارات طبية وخدمة التشخيص عن بُعْد عبر شبكة الإنترنت؛ ما يُسهِّل وصول الرعاية الصحية إلى المناطق النائية، أو تلقّيها في الحالات الطارئة. ومن أمثلتها: مِنصَّة (Med Jordan) للتطبيب عن بُعْد؛ إذ تُقدِّم هذه المِنصَّة خدمات استشارية طبية عبر شبكة الإنترنت؛ ما يتيح للمرضى تلقّي الرعاية الصحية اللازمة من دون حاجة إلى زيارة المراكز الطبية. وقد أسهم هذا التطبيق إسهامًا فاعلًا في توفير الوقت والجُهْد، وتقليل الازدحام والتجمُّع في العيادات الطبية.
6. **الطباعة ثلاثية الأبعاد والتخصيص:** تُستخدَم تقنية التصنيع ثلاثية الأبعاد في إنتاج أجهزة طبية مُتخصِّصة، وإجراء عمليات جراحية مُحدَّدة تبعًا لكل حالة مرضية؛ ما يُحسِّن من فاعلية العلاج، ويُقلِّل من المخاطر. ومن أمثلتها: الطباعة ثلاثية الأبعاد للأجهزة التعويضية المُتخصِّصة التي تُعتمَد فيها قياسات دقيقة جدًّا.

نشاط فردي

أزور الموقع الرسمي الإلكتروني لبرنامج حكيم (HAKEEM)

عن طريق مسح الرمز سريع الاستجابة (QR Code) المجاور:

ثمَّ أبحث في هدف البرنامج، والخدمات التي يُقدِّمها للمريض، والمنشآت الصحية والتطبيقات الإلكترونية التابعة له، ثمَّ أُشارِك ما أتوصَّل إليه من نتائج مع زملائي/ زميلاتي في الصف.

أبحث

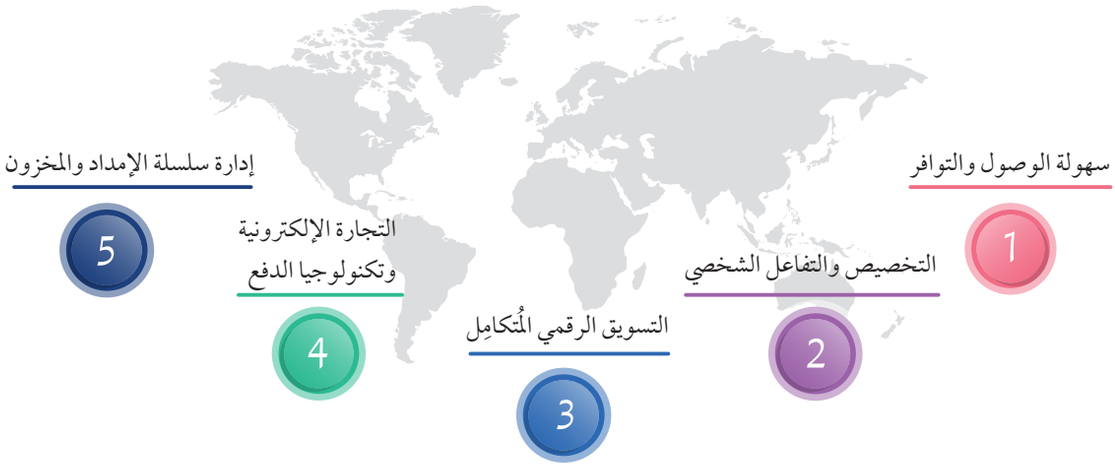
أبحثُ في المواقع الإلكترونية الموثوقة في شبكة الإنترنت عن أمثلة أُخرى على التطبيقات الحاسوبية في مجال الصحة، ثمَّ أُشـــارِك ما أتوصَّل إليه من نتائج مع زملائي/ زميلاتي في الصف عن طريق اللوح التفاعلي (Jamboard).

تطبيقات حاسوبية في مجال التسوُّق والتسويق الإلكتروني

يشـــهد العالَم اليوم تزايدًا ملحوظًا في اســـتخدام التكنولوجيا الرقمية في عمليات البيع والشراء والتســـوُّق والتسويق عبر شـــبكة الإنترنت، والهواتف الذكية، ووســـائل التواصل الاجتماعي، والتطبيقات المُخصَّصة لذلك؛ ما يُســـهِم في دفع عجلة الاقتصاد، والنهوض بحركة التجارة، لا سيَّما في ظلِّ ارتفاع أسعار الوَقود، وصعوبة الوصول إلى المتاجر التقليدية، وتعذُّر زيارة الأسواق المحلية البعيدة والأسواق العالمية.

مزايا التحوُّل الرقمي في مجال التسوُّق والتسويق الإلكتروني

مزايا التحوُّل الرقمي في مجال التسوُّق والتسويق الإلكتروني



أخذ التسوُّق والتسويق الإلكتروني يُنافِس الطرائق التقليدية في عمليات البيع والشراء والمعاملات التجارية؛ نظرًا إلى ما يتصف به من مزايا، أبرزها:

1. **سهولة الوصول والتوافر:**
   أ. إمكانية التســـوُّق الدائم للمُستهلِكين والعملاء؛ ما يُوفِّر عليهم الوقت والجُهْد، ويُهيِّئ لهم سُبُل الراحة والدَّعَة.
   ب. تمكين المِنصّات الإلكترونيـــة المتاجر من التعامل مع عدد كبير من المُســـتهلِكين والعملاء دون حاجة إلى إنشاء بِنية تحتية مادية.

2. **التخصيص والتفاعل الشخصي:**

أ. استخدام البيانات الضخمة وتحليلات العملاء في تقديم توصيات مُتخصِّصة وعروض خاصة بناءً على تفضيلات العملاء وسلوكاتهم.

ب. الإفادة من الدردشة التفاعلية وخدمات العملاء عبر شبكة الإنترنت في تقديم دعم فوري للعملاء.

3. **التسويق الرقمي المُتكامِل:**

أ. استخدام استراتيجيات التسويق الرقمي (مثل: تحسين مُحرِّكات البحث (SEO)، والتسويق عبر البريد الإلكتروني، والتسويق عبر وسائل التواصل الاجتماعي، والإعلانات المدفوعة) في الترويج وجذب مزيد من العملاء.

ب. إسهام التحليلات الرقمية في قياس فاعلية حملات التسويق وضبطها؛ لتحقيق أفضل النتائج.

4. **التجارة الإلكترونية وتكنولوجيا الدفع:**

أ. تمكين المواقع الإلكترونية وتطبيقات التسوُّق العملاء من شراء المُنتَجات بسهولة عبر شبكة الإنترنت.

ب. استخدام تقنيات الدفع الرقمية (مثل: المحافظ الإلكترونية، وبطاقات الائتمان، والتحويلات البنكية) في التعاملات التجارية؛ ما يجعلها أكثر سهولةً وأمانًا.

5. **إدارة سلسلة الإمداد والمخزون:**

أ. استخدام الأنظمة الرقمية في إدارة المخزون وتتبُّع الشحنات؛ ما يزيد من الفاعلية والكفاءة، ويعمل على تخفيض الكُلَف التشغيلية.

ب. تتبُّع الطلبات بصورة مباشرة؛ ما يُمكِّن العملاء من تعرُّف سَيْر الإجراءات التي تمرُّ بها طلباتهم، وتحديد الإجراء الذي وصلت إليه بدِقَّة.

تأثير التحوُّل الرقمي في مجال التسوُّق والتسويق الإلكتروني:

في ما يأتي أبرز آثار التحوُّل الرقمي في مجال التسوُّق والتسويق الإلكتروني:

1. **زيادة المنافسة:** يُسهِّل التحوُّل الرقمي على المُنافِسين الجُدُد دخول السوق، ما يزيد من وتيرة التنافس، ويُحفِّز الشركات على تحسين خدماتها ومُنتَجاتها.
2. **تحسين تجربة العملاء:** يُعزِّز التفاعل الفوري والتخصيص من رضا العملاء، ويزيد من ولائهم للعلامة التجارية.
3. **زيادة الكفاءة:** يؤدّي استخدام الأنظمة الرقمية في إدارة العمليات إلى التقليل من الأخطاء، وزيادة كفاءة العمليات التشغيلية.

أمثلة على التطبيقات والتحوُّلات الرقمية في مجال التسوُّق والتسويق الإلكتروني:

1. **أمازون (Amazon)**: مِنصَّة تجارة إلكترونية وحوسبة سحابية، تُقدِّم حلولاً مُتنوِّعةً؛ للوفاء بحاجات العملاء في مختلف دول العالَم، وتعرض تجربة تسوُّق شاملة، تتضمَّن توصيات مُتخصِّصة، وتعليقات للعملاء، وخيارات شحن مُتنوِّعة.
2. **إعلانات جوجل (Google Ads)**: أدوات تسويق رقمي، تتيح استهداف الجمهور بدِقَّة عن طريق الإعلانات المدفوعة التي تظهر للمُستخدِمين بناءً على اهتماماتهم وسلوكاتهم في شبكة الإنترنت.
3. **نظام إدارة علاقات العملاء (Salesforce: CRM)**: يساعد هذا النظام الشركات على تتبُّع تفاعلات العملاء وتحليلها، وإدارة حملات التسويق والمَبيعات بفاعلية.
4. **مواقع التواصل الاجتماعي**: شاع في الآونة الأخيرة إنشاء التُّجّار – الذين يملكون متاجر حقيقية– صفحات للبيع والشراء الإلكتروني في مواقع التواصل الاجتماعي، مثل: صفحات فيسبوك (Facebook)، وإنستغرام (Instagram)؛ نظرًا إلى سهولة التواصل الدائم مع المُستهلِكين والعملاء في هذه المواقع، علمًا بأنَّ ذلك لا يقتصر فقط على تسويق الملابس والمواد الغذائية، وإنَّما يتعدّاه إلى خدمات النقل، وحجوزات الرحلات، والترفيه، وغير ذلك.
5. **تطبيق السوق المفتوح (OpenSooq)**: يُعَدُّ السوق المفتوح أكبر تطبيق للإعلانات المُبوَّبة باللغة العربية؛ إذ يتيح هذا التطبيق لملايين المُستخدِمين تنفيذ عمليات بيع وشراء للعديد من المُنتَجات والخدمات عبر شبكة الإنترنت من دون وسيط، ويُمكِّن المشترين من مشاهدة السلع والخدمات المعروضة، مثل: السيّارات، والعقارات، والإلكترونيات، والأثاث.

نشاط

أزور الموقع الإلكتروني للسوق المفتوح عن طريق الرابط الإلكتروني الآتي:

https://jo.opensooq.com/ar

أو مسح الرمز سريع الاستجابة (QR Code) المجاور، ثمَّ أستعرض السلع المتوافرة في الموقع، وأستكشف أهم مزايا الموقع في ما يخصُّ مجال التسوُّق، والفئات التي يستهدفها.

أُوْلى الأردن عملية التحـــوُّل الرقمي اهتمامًا كبيرًا، وتمثَّل ذلك فـــي أتمتة الخدمات الحكومية المُقدَّمة للمواطنين، بالإعلان عن برنامج الحكومـــة الإلكترونية عام 2001م، الذي أُطلِق برعاية مَلَكية ســـامية، وكُلِّفت وزارة الاتصالات وتكنولوجيا المعلومات بتنفيذه وقتئذٍ، ثمَّ تولَّت إكماله اليوم وزارة الاقتصاد الرقمي والريادة، بتوفيرها عددًا من التطبيقات للهواتف الذكية، ومجموعةً من القنوات الرقمية عبر شبكة الإنترنت؛ بُغْيَةَ إنجاز المعاملات الحكومية التي تخصُّ المواطنين، أنظر الشكل (3-1) الذي يُبيِّن أهداف برنامج الحكومة الإلكترونية.

الشكل (3-1): أهداف برنامج الحكومة الإلكترونية.

في ما يأتي أبرز الخدمات التي تُقدِّمها الحكومة الإلكترونية للمواطنين:

1. **إصدار شـــهادة عدم محكومية:** تتيح هذه الخدمة للمواطنين إصدار شهادة عدم المحكومية إلكترونيًّا، وإمكانية تقديم الطلب والدفع بصورة إلكترونية.
2. **الاستعلام عن دفع المخالفات:** تُوفِّر هذه الخدمة قناة إلكترونية تُمكِّن المواطنين والمقيمين من الاســـتعلام عن مخالفات المَركَبات، ودفع قِيَمها إلكترونيًّا، والاطِّلاع على تفاصيل كل مخالفة منها.
3. **الاستعلام عن ضريبة الأبنية (المُســـقَّفات):** تُوفِّر هذه الخدمة الاستعلام عن ضريبة الأبنية، ودفع قيمتها إلكترونيًّا.
4. **تجديد رخصة المهن ولوحة الإعلانات إلكترونيًّا:** تُوفِّر هـــذه الخدمة قناة إلكترونية تُمكِّن أصحاب رخص المهن والأعمال الحُرَّة من تجديد رخصهم، ودفع رسومها إلكترونيًّا، إضافةً إلى تسلُّم هذه الرخص إمّا عن طريق البريد الأردني، وإمّا شخصيًّا.
5. خدمة إصدار شهادة الميلاد المُسجَّلة مُسبَّقًا.

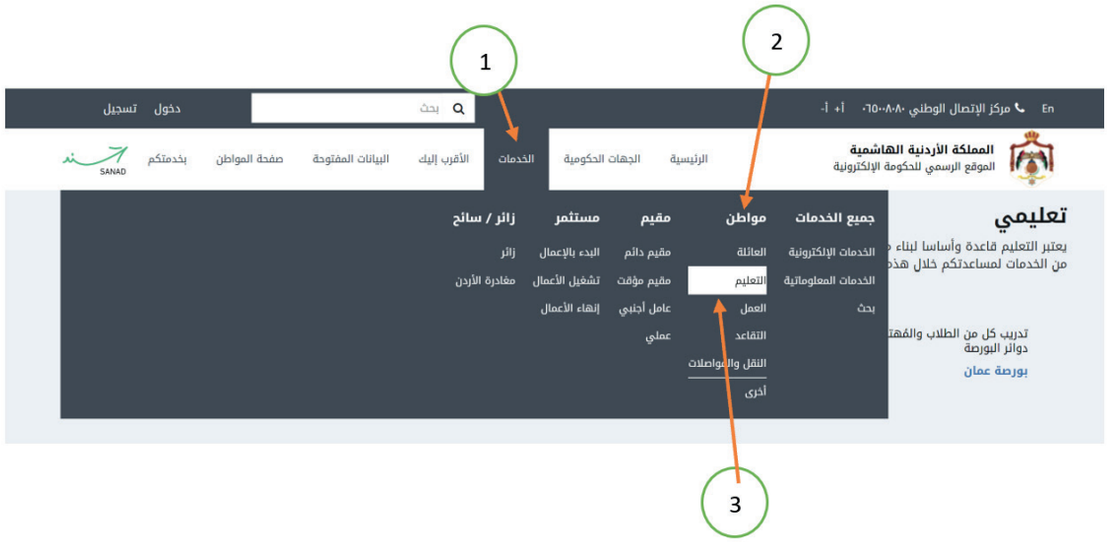
نشاط

أزور الموقع الإلكتروني الرسمي للحكومة الإلكترونية:

/https://portal.jordan.gov.jo/wps/portal/Home

ثمَّ أُجيب عن الأسئلة الآتية بعد تصفُّح الموقع:

1. هل يستطيع السائح أو المُستثمِر استخدام هذا الموقع؟ أُوضِّح ذلك.
2. ما أهم الخدمات التي تُقدِّمها صفحة المواطن في الموقع؟
3. أستكشف الخدمات التي يُمكِن للحكومة الإلكترونية أنْ تُقدِّمها لي - بوصفي طالبًا- بعد دخولي على صفحة الخدمات، ثمَّ الضغط على خيار مواطن، ومنه على خيار التعليم كما في الشكل الآتي:



تطبيقات حاسوبية للوسائط المُتعدِّدة

يوجد العديد من التطبيقات الحاسوبية الخاصة بإعداد الوسائط المُتعدِّدة وتطويرها وطباعتها. وهذه أبرزها:

صناعة الأفلام

تُعَدُّ برامج صناعة الأفلام وتحريرها (Movie Maker) أحد أكثر التطبيقات الحاسوبية انتشارًا في الهواتف المحمولة وأجهزة الحاسوب. أمّا الجديد في هذا المجال فهو تدخُّل الذكاء الاصطناعي

في صناعة الأفلام؛ إذ توجد تطبيقات كثيرة للذكاء الاصطناعي سـهَّلت عملية صناعة الأفلام من دون حاجة إلى اسـتخدام آلة تصوير (كاميرا) عالية الجودة، ومن دون حاجة إلى تصوير أيِّ شيء أساسًا، أو إرفاق صوت أو موسيقى مع الفيلم. فكل ما هو مطلوب تزويد برنامج الذكاء الاصطناعي بالموضـوع المطلوب ونبذة عنه، ليقوم البرنامج بإعداد فيلم فائق الجودة، ومُكتمِل العناصر؛ من: صوت، وموسيقى تصويرية، وصور مُتحرِّكة.

من تطبيقات الذكاء الاصطناعي في صناعة الأفلام: تطبيق (Invideo AI) الذي يتطلَّب اسـتخدامه إنشاء حسـاب في الموقع، وإدخال كلمات مفتاحية عن موضوع الفيلم، فيتولّى الموقع صنع فيلم كامل.

نشاط عملي

أزور الموقع الإلكتروني لتطبيق (Invodeo AI) عن طريق الرابط الإلكتروني الآتي:

https://ai.invideo.io/login

أو مسح الرمز سريع الاسـتجابة (QR Code) المجاور، ثمَّ أُنفِّذ الإجراءات الآتية:

1. إنشاء حساب خاص بي في الموقع.
2. إعداد مقطع فيديو عن الحوسبة الخضراء باستخدام الذكاء الاصطناعي.
3. عرض مقطع الفيديو أمام طلبة الصف.

أُفكِّر في المزايا والسلبيات لاستخدام الذكاء الاصطناعي في صناعة الأفلام، ثمَّ أُناقِش ذلك مع زملائي/ زميلاتي في الصف.

الطباعة ثلاثية الأبعاد (3D Printing):

الطباعة ثلاثية الأبعاد هي عملية إنشـاء كائن ثلاثي الأبعاد من ملف رقمي. ولكنْ، ما مبدأ العمل الذي تقوم عليه الطباعة ثلاثية الأبعاد؟

تتمثَّل هذه العملية أوَّلًا في بناء نموذج ثلاثي الأبعاد باسـتخدام برامج خاصة لهذا الغرض، مثل برنامج (Tinker CAD) الذي يمتاز بأنَّه مجّاني، ولا يَلزم تحميله في جهاز الحاسـوب الخاص بي، فضلًا عن إعطائه المُبتدِئين دروسًا في هذا المجال، واحتوائه على مزية تصدير النموذج الذي يُنشَأ بوصفه ملفًّا قابلًا للطباعة بامتداد (OBJ).

نشاط عملي

أفتح برنامج (Tinker CAD) عن طريق الرابط الإلكتروني الآتي:

https://www.tinkercad.com/things/ehBCM23h:Ro4-super-esboo/edit

أو مسح الرمز سريع الاستجابة (QR Code) المجاور، ثمَّ أُنفِّذ الإجراءات الآتية:

- إنشاء حساب خاص بي في الموقع.
- إنشاء نموذج ثلاثي الأبعاد خاص بي.
- تصدير النموذج إلى ملف امتداده (OBJ).

بعد أنْ أصبح لديَّ نموذج ثلاثي الأبعاد، قابل للطباعة باستخدام برنامج خاص، فإنَّني أُعِدُّه للطباعة ثلاثية الأبعاد عن طريق عملية تُسمّى التقطيع (Slicing)؛ أيْ تقطيع النموذج ثلاثي الأبعاد إلى المئات أو الآلاف من الطبقات، ليصبح جاهزًا للطباعة في طابعة ثلاثية الأبعاد، طبقةً تلو الأُخرى، علمًا بأنَّه توجد برامج خاصة للتقطيع.

إثراءٌ

مصنع الأفكار (TechWorks): إحدى مبادرات مؤسسة ولي العهد التي تدعم الشباب الأردني، وتُحفِّزهم على الابتكار والإبداع.

يُعَدُّ مصنع الأفكار مختبر تصنيع رقميًّا (FabLab)، يضمُّ عددًا من الأجهزة والمعدّات الحديثة، ويهدف إلى تشجيع التعليم التقني، وتعزيز منظومة الابتكار في الأردن؛ ليكون مِنصَّة للابتكار، تخدم الشباب ورُوّاد الأعمال والقطاع الصناعي والمجتمعات المحلية، وتُمكِّنهم من تطوير أفكارهم إلى مُنتَجات، وتسجيل براءات اختراع لها.

يضمُّ مصنع الأفكار عددًا من الطابعات ثلاثية الأبعاد، ومحطَّة الأعمال الخشبية، ومحطَّة أعمال الحديد، والمَرافق المُخصَّصة للإلكترونيات ومشاغل الخياطة، وهي تمتاز جميعًا باستخدام التكنولوجيا الحديثة، وتُمكِّن مُستخدِميها من تحويل أفكارهم وتصاميمهم إلى مُنتَجات حقيقية فائقة الدِّقَّة والجودة.

نشاط جماعي

استكشاف تطبيقات حاسوبية في مجال الرسوم المُتحرِّكة والتصميم ثلاثي الأبعاد.

أبحث – بالتعاون مع أفراد مجموعتي– في المواقع الإلكترونية الموثوقة في شبكة الإنترنت عن تطبيقات حاسوبية تُستخدَم في الرسوم المُتحرِّكة والتصميم ثلاثي الأبعاد، ثمَّ أتعرَّف هذه التطبيقات، وأتعلَّم أساسيات استخدامها، ثمَّ أكتب مُلخَّصًا عنها، وأُشارِكه أفراد المجموعات الأُخرى عن طريق اللوح الإلكتروني (Padlet).

بعد ذلك أتصفَّح مُقترَحات زملائي/ زميلاتي في المجموعات الأُخرى، ثمَّ أُبدي رأيي فيها.

المواطنةُ الرقميةُ:

أُراعي ما يأتي بعد دراسة موضوع (تطبيقات الحاسوب في الحياة):

- **الخصوصية والأمان**: أتجنَّب مشـــاركة الآخرين في معلوماتي الشـــخصية بمِنصّات التعليم الإلكتروني والتطبيقات الصحية، وأتحقَّق من أمانها قبل التفكير في مشاركتها.
- **استخدام كلمات مرور مُعقَّدة ومُحْكَمة**: أتأكَّد أنَّ كلمات المرور الخاصة بي مُعقَّدة، وأحرص على تغييرها بصورة دورية.
- **الموارد التعليمية**: أســـتخدِم الموارد التعليمية الإلكترونية على نحوٍ مسؤول، وأحترم حقوق المُلْكية الفكرية.
- **التفاعل الرقمي**: أتعامل باحترام مع الآخرين في البيئات التعليمية والبيئات الصحية الرقمية.
- **التحقُّق من المصادر**: أتحقَّق من موثوقية المصادر التي أســـتخدمها، ومن المواقع الإلكترونية التي أتصفَّحها قبل إدخال بياناتي الشخصية الخاصة.

مشروع

المشروع: تنفيذ مشروع ريادي رقمي يتناول القضايا البيئية والاجتماعية المُتعلِّقة بالحوسبة، باستخدام إحدى تطبيقات الحاسوب/ المهمة (3).

أعمل ـ بالتعاون مع أفراد مجموعتي – على تنفيذ المرحلة الثانية من المشـــروع، وهي مرحلة **التنفيذ والإنتاج:**

1. أبدأ بإنتاج المحتوى الرئيس للمشـــروع وَفقًا للخُطَّة الموضوعة، مثل كتابة السيناريوهات وتحريرها؛ سواء كان المشروع مقطع فيديو، أو ملصقًا، أو رسومًا مُتحرِّكةً، أو تصميمًا ثلاثي الأبعاد.
2. أتحقَّق من تجهيز البرنامج المناسب، ثمَّ أبدأ العمل بإنتاج المشروع في نسخته الأوَّلية.
3. أتعاون مع أفراد مجموعتـــي، وأعقد معهم اجتماعات دورية؛ لمتابعة سَـــيْر العمل في المشروع، وتبادل الأفكار.

الاختبار والتحسين:

1. أعرِض المشروع – في نسخته الأوَّلية– على مجموعة محدودة من الزملاء/ الزميلات، أو المُعلِّمين/ المُعلِّمات؛ للحصول على التغذية الراجعة اللازمة.
2. أستخدِم ملاحظات المجموعة في إجراء التحسينات اللازمة.

النشر:

1. أُجهِّز المشروع للنشر في المِنصّات المناسبة، مثل يوتيوب YouTube))، أو مواقع التواصل الاجتماعي، أو مِنصَّة المدرسة.
2. أكتب وصفًا موجزًا للمشروع وهدفه؛ لجذب الانتباه.
3. أنشر المشروع في المِنصّات المُحدَّدة، وأستعمل الرسوم المناسبة لجذب مزيد من الزُّوّار.
4. أُحفِّز زملائي وأصدقائي على مشاهدة المشروع ومشاركته.

التقييم والتحسين:

1. أستخدِم نماذج جوجل (Google Forms) في إنشاء استبانة لجمع التغذية الراجعة من الزُّوّار.
2. أُحلِّل الردود والملاحظات، ثمَّ أُحدِّد مَواطن القوَّة ومَواطن الضعف في المشروع.
3. أُجري – بناءً على التغذية الراجعة– التعديلات النهائية لتحسين المشروع.
4. أعرِض النسخة المُعدَّلة مَرَّة أُخرى (عند الحاجة)، وأُشارِكها من جديد.

أُقيِّمُ تعلُّمي

المعرفة: أُوظِّف في هذا الدرس ما تعلَّمْتُه من معارف في الإجابة عن الأسئلة الآتية:

السؤال الأوَّل: أُوضِّح المقصود بكلٍّ ممّا يأتي:

1. التعلُّم الإلكتروني (e-learning).

2. الحكومة الإلكترونية (e-government).

3. الطباعة ثلاثية الأبعاد (3D-printint).

السؤال الثاني: ما الخدمات التي تُقدِّمها الحكومة الإلكترونية للمواطنين؟

السؤال الثالث: كيف تُسهِم التكنولوجيا في تحسين جودة التعليم؟

المهارات: أُوظِّف مهارات التفكير الناقد والتواصل الرقمي والبحث الرقمي في الإجابة عن الأسئلة الآتية:

السؤال الأوَّل: أَصِف كيف يعمل تطبيق السجل الصحي الإلكتروني على تحسين خدمات الرعاية الصحية.

السؤال الثاني: أُوضِّح تأثير التحوُّل الرقمي في التسوُّق والتسويق الإلكتروني في الاقتصاد المحلي، وأذكر أمثلة على ذلك، وأُحلِّل بيانات ذات صلة بالموضوع.

السؤال الثالث: أُعِدُّ – بالتعاون مع أفراد مجموعتي – فيلمًا قصيرًا عن تدوير النفايات الإلكترونية والحوسبة الخضراء في منطقتي المحلية، وأستعين لذلك بشبكة الانترنت، ومهارات البحث الرقمي، وتطبيقات صناعة الأفلام التي تعتمد الذكاء الاصطناعي، ثمَّ أُشارِك الفيلم في الصفحة الإلكترونية الرسمية للمدرسة.

القِيَمُ والاتجاهاتُ

أُنشِئ محتوًى مرئيًّا (إنفوجرافيك) باستخدام برمجية (CANVA)، أو أيِّ موقع لتصميم الإنفوجرافيك في شبكة الإنترنت، وأُضمِّنه وثيقة سلوك للآداب العامة والسلوكات التي يجب أنْ ألتزمها بوصفي مُستخدِمًا للتطبيقات الحاسوبية المختلفة.

مُلخَّصُ الوحدةِ

في ما يأتي أبرز الجوانب التي تناولتها هذه الوحدة:

1. الحوسبة الخضراء هي الاستخدام البيئي المسؤول لأجهزة الحاسوب والموارد التكنولوجية ذات الصلة، الذي يَحُدُّ من التأثير السلبي لتكنولوجيا المعلومات والاتصالات في البيئة. .
2. إسهام الحوسبة الخضراء في ترشيد استهلاك الطاقة، والحدِّ من انتشار النفايات الإلكترونية؛ ما يؤدّي إلى خفض الكُلَف التشغيلية، وتعزيز الاستدامة البيئية. وهي تعتمد أساسًا على تحسين كفاءة الطاقة، واستخدام مصادر الطاقة المُتجدِّدة، والإدارة الصحيحة للنفايات الإلكترونية.
3. وجود تحدِّيات وعقبات كثيرة تَحول دون تطبيق مبدأ الحوسبة الخضراء، أبرزها: التكلفة العالية، والحاجة إلى التدريب والتحديث المستمر، والبِنية التحتية غير المُلائِمة، وقِلَّة الوعي.
4. النفايات الإلكترونية وهي أجهزة إلكترونية قديمة أوشك عمرها الافتراضي على الانتهاء، واستُبدِل بها أجهزة أُخرى جديدة أو حديثة، مثل: أجهزة الحاسوب، والهواتف المحمولة. تحتوي النفايات الإلكترونية على مواد سامَّة، مثل الرصاص والزئبق؛ ما يُمثِّل خطرًا كبيرًا على الصحة العامة وسلامة البيئة..
5. إدارة النفايات الإلكترونية تشمل اتِّخاذ عدد من الخطوات الرئيسة، وهي: جمع النفايات ثمَّ فصل بعضها عن بعض لتحليل مُكوِّناتها، وتنظيف البيانات لضمان عدم استخدامها، وإعادة تدوير النفايات الإلكترونية لفصل المواد القابلة لإعادة الاستخدام، وتجديد الأجزاء ذات الجودة العالية لإطالة عمرها.
6. الاستراتيجيات الفاعلة في إدارة النفايات الإلكترونية تشمل التوعية العامة، وسَنَّ التشريعات، وتعزيز التعاون بين مختلف الجهات المَعْنِيَّة، ودعم البحوث الخاصة بتطوير تقنيات إعادة التدوير، وتقليل استخدام (استهلاك) الأجهزة الإلكترونية، وإعادة استخدام هذه الأجهزة ما أمكن، وتعزيز ممارسات إعادة التدوير المسؤولة.
7. التطبيقات الحاسوبية تؤدّي دورًا فاعلًا في تحسين جودة الحياة وزيادة الإنتاجية في مجالات مُتعدِّدة. ففي مجال التعليم، توفر بعض الأدوات مرونة كبيرة في التعلُّم، مثل: أداة (Google Classroom)، وأداة (Coursera)؛ ما يُسهِّل الوصول إلى المعلومات، ويُعزِّز المهارات التقنية. أمّا في مجال الصحة، فتُسهِم التطبيقات الحاسوبية (مثل برنامج حكيم HAKEEM) في تحسين خدمات الرعاية الصحية، بما تُوفِّره من وصول سريع إلى البيانات، ودِقَّة في التشخيص. في حين تؤدّي الحكومة الإلكترونية دورًا مُهِمًّا في تسهيل المعاملات والوصول إلى الخدمات الحكومية بسرعة وكفاءة.

أسئلةُ الوحدةِ

السؤال الأوَّل: أكتب المصطلح المناسب بجانب كل عبارة من العبارات الآتية:

العبارة	المصطلح
تصميم أجهزة الحاسوب والأجهزة الإلكترونية الأُخرى، أو تصنيعها، أو استخدامها بصورة تَحُدُّ من آثارها الضارَّة بالبيئة، مثل: انبعاثات الكربون، واستهلاك الطاقة .	
أدوات تحتوي على مقابس وأسلاك ومُكوِّنات إلكترونية، مثل: أجهزة التلفاز، وأجهزة الحاسوب، والهواتف المحمولة، ومُكيِّفات الهواء، وألعاب الأطفال الإلكترونية.	
إنشاء كائن ثلاثي الأبعاد من ملف رقمي.	
برنامج وطني مُهِمٌّ لحوسبة قطاع الصحة في الأردن، أُطلِق عام 2009م.	
جهاز حاسوب لا يحوي شاشة أو لوحة مفاتيح، ويُشبِه جهاز العرض (Projector) في مبدأ عمله.	

السؤال الثاني: ما التحدِّيات التي تقف حائلًا دون تطبيق مبدأ الحوسبة الخضراء؟

السؤال الثالث: أُعلِّل ما يأتي:

1. استبدال مادة الخيزران (Bamboo) بمادة البلاستيك التي تدخل في صناعة ملحقات أجهزة الحاسوب.

2. أجهزة الحاســـوب من نوع (Notebooks) أقل تأثيرًا في البيئة من أجهزة الحاسوب المحمولة (Laptops)، وأجهزة الحاســـوب المحمولة أقل تأثيرًا في البيئة من أجهزة الحاسوب المكتبية (Desktop).

3. التخلُّص من النفايات الإلكترونية بالحرق يُؤثِّر سلبًا في البيئة.

الســـؤال الرابع: ما الخطوات التي يجب عليَّ تنفيذها للتخلُّص مـــن النفايات الإلكترونية بصورة صحيحة؟

السؤال الخامس: أُوضِّح أثر التطبيقات الحاسوبية الآتية في الحياة اليومية:

1. تطبيق حكيم (Hakeem) في القطاع الصحي.

2. مِنصَّة التعلُّم الأردنية في قطاع التعليم.

3. تطبيق السوق المفتوح (Opensooq) في مجال التسويق الإلكتروني.

السؤال السادس: لماذا يُعَدُّ التخلُّص الآمن والصحيح من النفايات الإلكترونية مطلبًا ضروريًّا؟

المهارات:

السؤال الأوَّل: كيف يُمكِن تطبيق مبادئ الحوسبة الخضراء في المنزل؟

السؤال الثاني: أُبيِّن مزايا تطبيق مبدأ الحوسبة الخضراء في الشركات، والتحدِّيات التي تعترض ذلك.

السؤال الثالث: أُحلِّل تأثير التطبيقات الحاسوبية في الاقتصاد المحلي، مُمثِّلًا لذلك تطبيق السوق المفتوح.

السؤال الرابع: ما التحدِّيات التي يُواجِهها تطبيق مبدأ الحوسبة الخضراء على مستوى المجتمع؟

السؤال الخامس: أستخدِم أمثلة من النص لشرح كيف يُمكِن أنْ تُسهِم التطبيقات الحاسوبية في تحسين عملية التعليم عن بُعْد.

القِيَم والاتجاهات:

أختار واحدًا من الموضوعات الآتية لتطبيقه:

1. أُصمِّم - باستخدام أحد تطبيقات التصميم- تعهُّدًا (Pledge) بطريقة جاذبة، أتعهَّد فيه أنا وطلبة الصف بالمحافظة على البيئة، وتحقيق أهداف التنمية المستدامة؛ بالتخلُّص الآمن والصحيح من النفايات الإلكترونية، وأترك في التعهُّد مساحة فارغة لكي يُوقِّع فيها كل مَنْ يقرأ التعهُّد، ثمَّ أضعه في مكان بارز داخل المدرسة .
2. أقترح تصميمًا لتطبيق حاسوبي يساعد على إدارة النفايات الإلكترونية في مجتمعي.
3. أبتكر فكرة تطبيق حاسوبي جديد في مجال الصحة الرقمية، وأُبيِّن المزايا والعيوب فيه، وكيف يُمكِن قياس فاعلية التطبيق.

تقويمٌ ذاتيٌّ (Self Evaluation)

بعدَ دراستي هذهِ الوحدةَ، اقرأُ الفقراتِ الواردةَ في الجدولِ الآتي، ثمَّ أضعُ إشارةَ (✔) في العمودِ المناسبِ:

مؤشرات الأداء	نعمْ	لا	لستُ مُتأكِّدًا
أُعرِّف مفهوم الحوسبة الخضراء.	○	○	○
أُبيِّن أهمية الحوسبة الخضراء وفوائدها ومزاياها.	○	○	○
أُسهِم في تطبيق الحوسبة الخضراء في حياتي اليومية.	○	○	○
أُعرِّف مفهوم النفايات الإلكترونية.	○	○	○
أُوضِّح طرائق التخلُّص الآمنة من النفايات الإلكترونية.	○	○	○
أذكر أدوات حاسوبية صديقة للبيئة.	○	○	○
أذكر تطبيقات حاسوبية في مجال التعلُّم الإلكتروني.	○	○	○
أستخدِم تطبيقات حاسوبية في مجال التعلُّم عن بُعْد.	○	○	○
أذكر تطبيقات حاسوبية في مجال الصحة، وأُبيِّن أهميتها.	○	○	○
أستخدِم تطبيقات حاسوبية في مجال التسوُّق والتسويق الإلكتروني	○	○	○
أُبيِّن أهمية تطبيقات الحكومة الإلكترونية في تسهيل المعاملات	○	○	○
أُوضِّح أهمية بعض تطبيقات الحاسوب في الحياة، مثل: صناعة الأفلام، والتصميم ثلاثي الأبعاد، والطباعة ثلاثية	○	○	○

تعليماتٌ للمراجعةِ والتحسينِ: إذا اخترْتُ (لا) أوْ (لسْتُ مُتأكِّدًا) لأيٍّ منَ الفقراتِ السابقةِ، فأَتَّبعُ الخطواتِ الآتيةَ لتجنُّبِ ذلكَ:

- أُراجِعُ المادةَ الدراسيةَ؛ بأنْ أُعيدَ قراءةَ المحتوى المُتعلِّقِ بالمعيارِ.
- أطلبُ المساعدةَ؛ بأنْ أُناقِشَ مُعلِّمي/ مُعلِّمتي أوْ زملائي/ زميلاتي في ما تعذَّر عليَّ فهمُهُ.
- أستخدِمُ مراجعَ إضافيةٍ؛ بأنْ أبحثَ عنْ مراجعَ أُخرى مثلِ الكتبِ، أوْ أستعينَ بالمواقعِ الإلكترونيةِ الموثوقةِ التي تُقدِّمُ شرحًا وافيًا للموضوعاتِ التي أجدُ صعوبةً في فهمِها.

تأمُّلاتٌ ذاتيةٌ

عزيزي الطالبَ/ عزيزتي الطالبةَ:
التأمُّلاتُ الذاتيةُ هيَ فرصةٌ لتقييمِ عمليةِ التعلُّمِ، وفهمِ التحدِّياتِ، وتطويرِ استراتيجياتٍ لتحسينِ عمليةِ التعلُّمِ مستقبلًا. أملأُ الفراغَ في ما يأتي بالأفكارِ والتأمُّلاتِ الشخصيةِ التي يُمكِنُ بها تحقيقُ أفضلِ استفادةٍ منَ التجربةِ التعليميةِ:

تعلَّمْتُ في هذهِ الوحدةِ:

يُمكِنُني أنْ أُطبِّقَ ما تعلَّمْتُهُ في:

الصعوباتُ التي واجهْتُها أثناءَ عمليةِ التعلُّمِ:

ذلَّلْتُ هذهِ الصعوباتِ عنْ طريقِ:

يُمكِنُني مستقبلًا تحسينُ: