وزارة التربية
Ministry of Education
State of Kuwait | دولـــة الكويـــت

تقنية المعلومات 10

الصـف العاشر

الفصل الدراسي الأول

المرحلة الثانوية

وزارة التربية
Ministry of Education
State of Kuwait | دولـــة الكويـت

الصـف العاشر

الفصل الدراسي الأول

الإشراف العام

أ. مـــنى ســـالم عـــوض ســـالم (رئيس اللجنة)

تأليف

أ. منـــى مرزوق مخلـد العـازمـي أ. منـار مصطفى عبدالحميد جمال

أ. إبراهيم عبدالله إبراهيم المياس

تصميم

أ. فـــي دغيـليـب زايـد المـطيـــري

دعم التأليف والمراجعة العلمية

أ. رأفـت صابـر عبـداللاه أحمـد د. أشرف رضوان رضـوان سليمـان

إخراج

د. أشرف رضوان رضـوان سليمـان

الطبعة الثالثة

1448 هـ

2027/2026 م

الطبعة الأولى 2025/2024م
الطبعة الثانية 2026/2025م
الطبعة الثالثة 2027/2026م

الإشراف العام

أ. منى سالم عوض سالم (رئيس اللجنة)

تأليف

أ. رأفت صابر عبداللاه أحمد
أ. حسام الدين علي عبد القادر
أ. محمد السيد محمد إبراهيم
د. أشرف رضوان رضوان سليمان
د. يوسف منصور يوسف الخليفي
أ. منار مصطفى عبد الحميد جمال
أ. إبراهيم عبدالله إبراهيم المياس

تصميم

أ. إيمان عبد العزيز أحمد الفارسي
أ. سنية محمد علي المؤمن
أ. منى مرزوق مخلد العازمي
أ. ساره ياسين عبد الله الأمير

إخراج

د. أشرف رضوان رضوان سليمان

المراجعة اللغوية

أ. أحمد مهدي
أ. ناصر الحسيني
أ. فهد العتيبي

INTERNATIONAL GROUP PRINTING PRESS
مطابع المجموعة الدولية لأعمال الطباعة

أودع في مكتبة الكويت الوطنية - ردمك: 978-9921-33-135-6
أودع في مكتبة وزارة التربية تحت رقم (520) بتاريخ 2026/8/9 م


بسم الله الرحمن الرحيم

حضرة صاحب السمو الشيخ مشعل الأحمد الجابر الصباح

أمير دولة الكويت

H.H. Sheikh Meshal AL-Ahmad Al-Jaber Al-Sabah
Amir Of The State Of Kuwait

سمو الشيخ صباح خالد الحمد الصباح
ولي عهد دولة الكويت

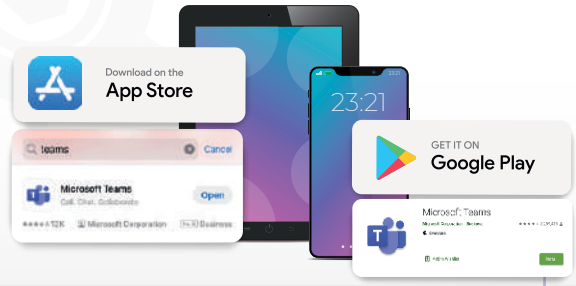
H. H. Sheikh Sabah Khaled Al-Hamad Al-Sabah
Crown Prince Of The State Of Kuwait

الفصل الرقـــمي Digital Classroom

أولًا :تحميل وتثبيت تطبيق Microsoft Teams

الدخول على متجر تطبيقات الأجهزة الرقمية.

من جهاز الحاسوب



من الأجهزة الذكية

ثانياً: تفعيل Microsoft Teams على الجهاز الرقمي

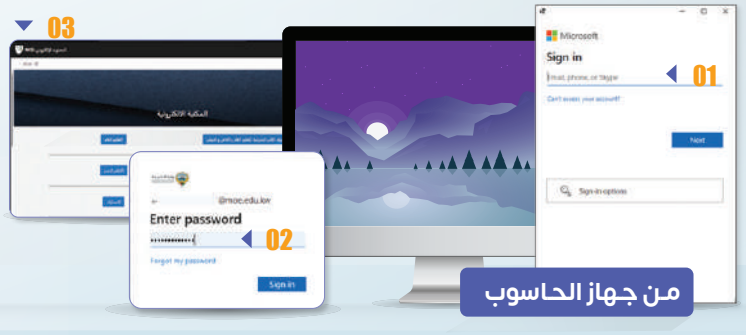
احرص على تحديث تطبيق Microsoft Teams بشكل دوري.

لتشغيل التطبيق اتبع الخطوات التالية:

01 أدخل اسم المستخدم: البريد الإلكتروني الرسمي الخاص بحساب Teams.

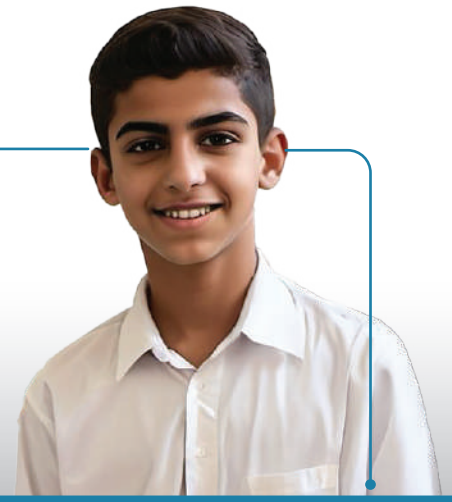
02 أدخل كلمة المرور الخاصة بحساب Teams.

03 تنقّل بين واجهات التطبيق مثل المحتوى الإلكتروني، وفِرَقِ المواد الدراسية.



احتفظ بكلمة المرور في مكان آمن لتسهيل تسجيل الدخول لاحقًا.

لا تشارك بياناتك مع أي شخص غير موثوق به.



مَـنْ حـمد ؟...

هـو مُساعد تعليمـي رقمـي ذكـي يعتمـد علـى تقنيـات الـذكاء الاصطناعـي، مُصمـم لتقديـم الدعـم التعليمـي التفاعلـي لكافـة المناهـج الدراسـية.

حمد دائمًا معك... لتتعلّم بثقة وتنجح بامتياز.

مع حـمد Chat

مـا خـدمـة حـمـد شـات؟

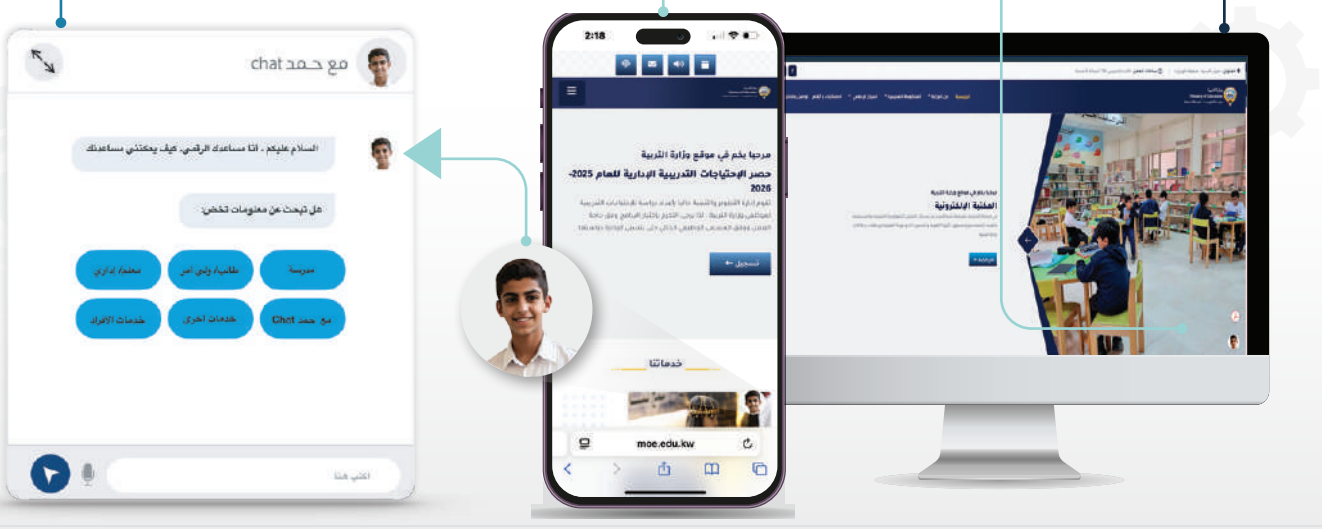
هـي خدمـة المحادثـة الذكيـة المقدمـة مـن وزارة التربيـة فـي دولـة الكويـت، تعتمـد علـى الـذكاء الاصطـناعي، تتمثـل وظيفتـه الأساسـية فـي تسـهيل عمليـة الفهـم والاسـتيعاب مـن خلال تقديـم الشـروحات المبسـطة، والإجابـة علـى الاستفسـارات، وتوجيـه المتعلميـن خلال مسـيرتهم التعليميـة.

أيـــن أجـــده؟

01
- زيارة الموقع الرسمي www.moe.edu.Kw
- أو تطبيق وزارة التربية

02
الضغط عـلى صورة حمد شات

03
اكتب استفساراتك، وستتلقى الإجابات بشكل فوري.



المحـــتوى

العنوان | الصفحة

المقدمة

يشـهد العالَـم اليـوم تحـوّلاً رقميـاً متسـارعاً جعـل التكنولوجيـا جـزءاً أساسـياً مـن حياتنـا اليوميـة، وأصبحـت البيانـات والمعلومـات مـن أهـم الأصـول التـي يجـب حمايتهـا والمحافظـة عليهـا. وفـي ظـل هـذا التطّـور الكبيـر بـرز الأمـن السـيبراني بصفتـه أحـد المجـالات الحيويـة التـي تهـدف إلـى حمايـة الأنظمـة والشـبكات والأجهـزة والمعلومـات مـن التهديـدات والهجمـات الرقميـة، وتعزيـز الوعـي بالممارسـات الآمنـة فـي البيئـة الرقميـة

وفـي السـياق ذاتـه، تُعـد البرمجـة لغة العصـر وأداة الابتكار والإبـداع، حيث تُمكّـن المتعلمين مـن تصميم الحلـول التقنية وتطويـر التطبيقات والأنظمة الذكيـة. وتبرز لغـة Python بصفتهاإحدى لغـات البرمجـة الأكثـر انتشـاراً لمـا تتمّيز بـه من سـهولة التعلم وقـوة الإمكانـات وتنـوّع التطبيقات فـي مجـالات الـذكاء الاصطناعـي وتحليـل البيانـات والأمـن السـيبراني وتطويـر البرمجيـات

يهـدف هـذا الكتـاب إلـى تزويـد المتعلميـن بالمعـارف والمهـارات الأساسـية فـي مجـال الأمـن السـيبراني، وتنميـة قدراتهـم البرمجيـة باسـتخدام لغـة Python مـن خلال أنشـطة عمليـة ومشـروعات تطبيقيـة تُسـهم فـي بنـاء التفكيـر المنطقـي وحـل المشـكلات والابتـكار التقنـي. كمـا يركّـز علـى تنميـة الوعـي الرقمـي والمسـؤولية الأخلاقيـة عنـد اسـتخدام التكنولوجيـا، بمـا ينسـجم مـع متطلبـات القـرن الحـادي والعشـرين ورؤيـة دولـة الكويـت فـي إعـداد جيـل رقمـي قـادر علـى الإبـداع والمنافسـة فـي مجتمـع المعرفـة

وقـد روعـي فـي إعـداد هـذا الكتـاب تقديـم المحتـوى بأسـلوب تدريجـي يجمـع بيـن الجانـب النظـري والتطبيـق العملـي، بحيـث ينتقـل المتعلـم مـن اسـتيعاب المفاهيـم الأساسـية إلـى تنفيذ التطبيقـات والمشـروعات البرمجيـة التـي تعـزّز التعلـم الذاتـي والتعلـم القائـم على المشـروعات نرجـو لأبنائنـا وبناتنـا المتعلميـن رحلـة تعليميـة ممتعـة ومثمـرة، تسـهم فـي بنـاء مهاراتهـم الرقميـة وتمكنهـم مـن توظيـف التكنولوجيـا توظيفـاً آمنـاً وفعّـالاً وإبداعيّـاً

والله ولي التوفيق.

المؤلفون

الوحـدة الأولــى - الأدوات الرقمية

الأمن السيبراني Cyber Security

1. مفاهيم الأمن السيبراني
2. المحـــاور الرئيســـة لأمـــن المعلومـــات والأمـــن الســـيبراني CIA
3. الإجـــراءات الأمنيـــة للمحـــاور الرئيســـة للأمـــن الســـيبراني CIA
4. ممارسات الأمن السيبراني
5. الوظائف في مجال الأمن السيبراني
6. الجوانـب الأخلاقيـة الرئيسـية فـي مجال الأمن الســـيبراني

الأمن السيبراني

Cyber Security

نواتج التعلم

- شرح مفهوم الأمن السيبراني وأمن المعلومات وأهميتهما في العصر الرقمي.
- تفسير عناصر نموذج CIA (السرية، النزاهة، التوفر)، وتوضيح دورها في حماية الأنظمة والمعلومات.
- تطبيق خطوات المصادقة الثنائية، واستخدام تقنيات الحماية والتشفير لتعزيز أمن البيانات.
- توظيف أدوات مثل برامج إدارة كلمات المرور ووسائل التصفح الآمن لحماية المعلومات الشخصية.
- تمييز الأدوار والمسؤوليات المختلفة ضمن مجالات العمل في وظائف الأمن السيبراني.
- إظهار سلوكٍ أخلاقيّ ومسؤولية عالية عند التعامل مع البيانات الرقمية والمعلومات الحساسة.

يُمَثّل رمز الاستجابة السريعة QR رابطاً
لملفات أوراق العمل، ومصادر التعلم

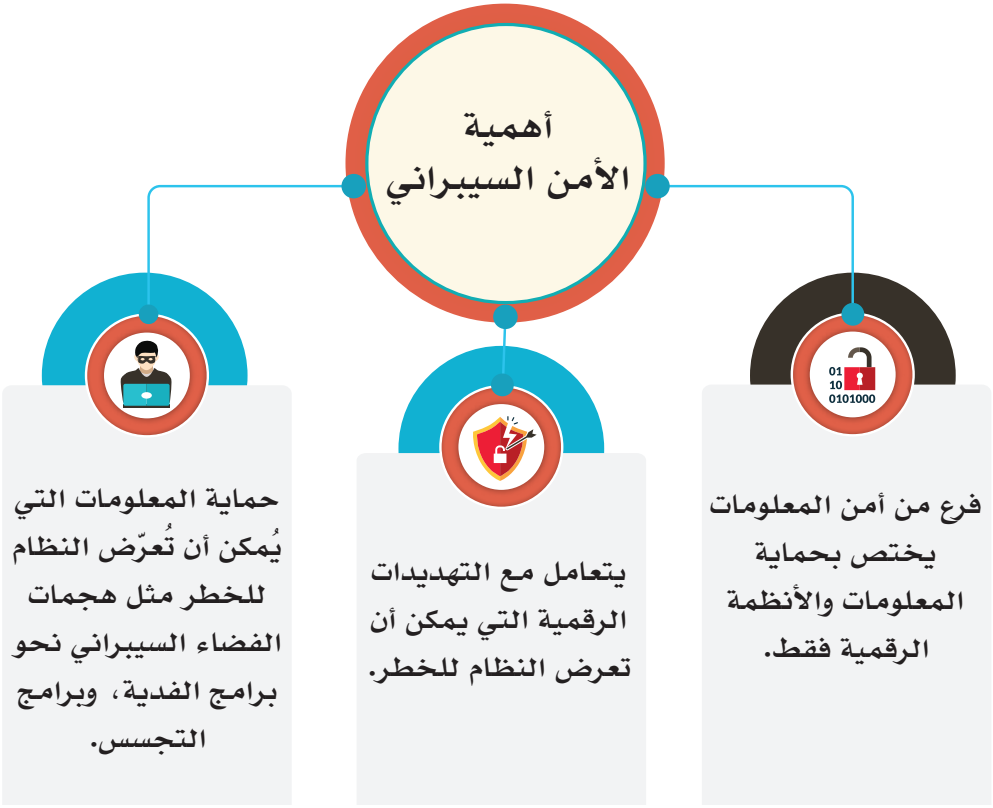
مفاهيم الأمن السيبراني The Concepts of Cyber Security

أمن المعلومات Information Security

الممارسات والتدابير المُتّبعة لحماية البيانات والأنظمة من الوصول غير المصرّح به (لاستخدامها، الاطلاع عليها، تعطيلها، تعديلها، أو تدميرها)، وتشمل حماية المعلومات حمايةً عامةً بغض النظر عن الوسائط المستخدمة سواء كانت رقمية أو غير رقمية.

الأمن السيبراني Cyber Security

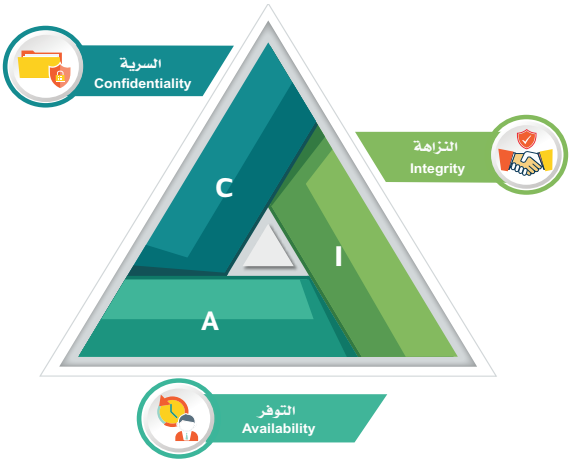
ممارسة حماية الأنظمة والشبكات من التهديدات الإلكترونية، وحماية البيانات الرقمية، والبرامج من الهجمات التي تهدف للوصول إلى المعلومات المهمة أو تغييرها أو تدميرها، وتُطَبّقُ الإجراءات، والضوابط، والعمليات، والتقنيات لتقليل مخاطر الهجمات السيبرانية والحماية من الاستغلال غير المصرّح للأنظمة، والشبكات، والتقنيات.



شكل (1-1) يوضّح أهمية الأمن السيبراني.

المحاور الرئيسة لأمن المعلومات والأمن السيبراني CIA

يُعد نموذج CIA أداة أساسية لضمان أمن المعلومات من خلال التركيز على السرية والنزاهة والتوفر لتمكين المؤسسات من حماية بياناتها من الوصول غير المُصرّح به، وضمان دقتها واكتمالها، وجعلها متاحة للمستخدمين المصرّح لهم عند الحاجة، ومن الآثار المترتبة على انتهاكها (انتهاك الخصوصية - سرقة الهوية - الإضرار بالسمعة - الخسارة المالية - عدم التمكن من الوصول للخدمات الأساسية والطوارئ والتواصل).



شكل (1-2) يوضح محاور CIA.

السرية Confidentiality

التأكد من حماية المعلومات التي لا يمكن الوصول إليها والحفاظ على سريتها إلا من قبل الأفراد المصرح لهم فقط Authorized Individuals مثل (الصور الخاصة، والبيانات المصرفية، وبيانات الدراسة أو العمل، والوثائق الحكومية).

النزاهة Integrity

تشير النزاهة إلى الحفاظ على صحة المعلومات الرقمية وموثوقيتها (Authenticity , Reliability)، وسلامتها. لضمان عدم قيام أي أطراف غير مصرّح لهم Unauthorized بالتلاعب بالمعلومات التي ترسلها وتستقبلها عبر الإنترنت.

التوفر Availability

يقصد بالتوفر التأكّد من أنّ المعلومات والموارد متاحة باستمرار Consistently Available، وقابلة للاستخدام usable عند الحاجة، وتشمل إمكانية الوصول إلى الأنظمة والخدمات وتشغيلها عند الحاجة، وحماية أنظمة المعلومات وبياناتها من الانقطاعات أو التوقّف عن العمل.

شكل (1-3) يُوضح أمثلة على الهجمات السيبرانية.

تذكر أن تطبيق مبادئ CIA في الأمن السيبراني مسؤولية مشتركة. من خلال ممارسة هذه الخطوات البسيطة وتعزيز الوعي، يمكننا إنشاء بيئة رقمية أكثر أمانًا وموثوقية للجميع.

ويجب أن ننتبه أن مبادئ CIA هي أهم ما يحدد آلية تأمين المؤسسات والأفراد في مجال الأمن السيبراني، وتُحَدّدُ أولويات تنفيذ السياسات الأمنية للمبادئ الثلاثة (السرية، النزاهة، التوفر) وفق عملية إدارة المخاطر بالمؤسسة.

مثال: ماكينة الصراف الآلي ATM Automated Teller Machine:

- **السرية Confldentiality:**

تطبيق المصادقة الثنائية مثل: (البطاقة البنكية Bank Card، ورمز المرور PIN)، للسماح للمستخدم بالوصول للبيانات المطلوبة، حفاظًا على سرية المعلومات المصرفية.

- **النزاهة Integrity:**

ضمان سلامة البيانات حيث لا يمكن لأي شخص تغيير معلومات الحسابات المصرفية ما لم يكن مصرّحًا له، من خلال إبلاغ المستخدم عن أي عمليات مصرفية تمّت عبر ماكينة الصراف الآلي.

- **التوفر Availability:**

النظام المصرفي الإلكتروني (ماكينة الصراف الآلي ATM، الموقع Website، والتطبيق Application) متاح في أي وقت إمكانية الاستخدام في الأماكن العامة، والوصول إليه على مدار الساعة.

الإجراءات الأمنية للمحاور الرئيسة للأمن السيبراني CIA

المُصادقة Authentication

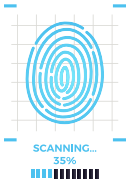
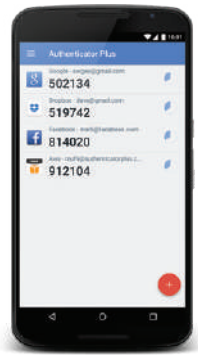
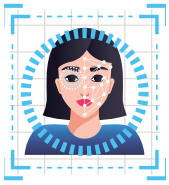
المُصادقة هي طبقة إضافية أو أكثر من الأمان تستخدم لحماية حسابات المستخدمين. بتقديم عدة أشكال منفصلة لتحديد الهوية قبل منح الوصول. وتتمثل في:

1. **عامل المعرفة:** ما تعرفه Something you Know، وهو شيء يعرفه المستخدم.
2. **عامل الامتلاك:** ما تملكه Something you Have، وهو شيء يمتلكه المستخدم.
3. **عامل المقياس الحيوي:** ما أنت عليه Something you Are، وهو شيء لإثبات هوِية الفرد.

ومن أمثلة عامل المقياس الحيوي، **المصادقة البيومترية** Biometric Authentication، وهي تستخدم خصائص بيولوجية فريدة، مثل بصمات الأصابع أو مسح قزحية العين أو التعرف على الوجه، للتحقق من هوية المستخدم. تضيف هذه الطريقة طبقة إضافية من الأمان حيث يصعب تكرار هذه السمات المادية أو تزويرها.

أصبحت المصادقة البيومترية شائعة بتزايد في الهواتف الذكية وأجهزة الكمبيوتر المحمولة، مما يوفر طريقة آمنة لمصادقة المستخدمين وحماية معلوماتهم الشخصية. وعلى سبيل المثال استخدام المصادقة البيومترية (التعرف على الوجه) في تطبيق هويتي.

شكل (1-4) يُوضح أحد شاشات إنشاء حساب على تطبيق هويتي.

1. عامل المعرفة (ما تعرفه) Something you Know	2. عامل الامتلاك (ما تملكه) Something you Have	3. عامل المقياس الحيوي (ما أنت عليه) Something you Are
كلمات المرور Passwords 	مفتاح الأمان المادي Hardware token 	بصمة العين Iris Recognition / Retinal Scan 
مصادقة الرسائل القصيرة OTP - PIN 	الكروت الذكية Smart cards 	بصمة الأصابع Fingerprint Recognition 
المصادقة المستندة إلى التطبيق App-based Authentication 	الأجهزة الذكية Smart devices 	بصمة الوجه Facial Recognition 

جدول (1-1) يُوضح بعض عوامل المصادقة المستخدمة.

الشبكة الافتراضية الخاصة Virtual Private Network (VPN)

تُعد الشبكة الافتراضية الخاصة (VPN) أداة مُهمة لضمان الخصوصية عبر الإنترنت. من خلال تشفير اتصال الإنترنت وتوجيهه عبر خادم Server موجود في موقع Location مختلف.

تُنشئ شبكات VPN اتصالاً آمنا يخفي عنوان IP الخاص بالمستخدم ونشاط التصفح، يساعد هذا في حماية البيانات من المهاجمين وحماية الخصوصية، وبخاصة عند استخدام شبكات Wi-Fi العامة.

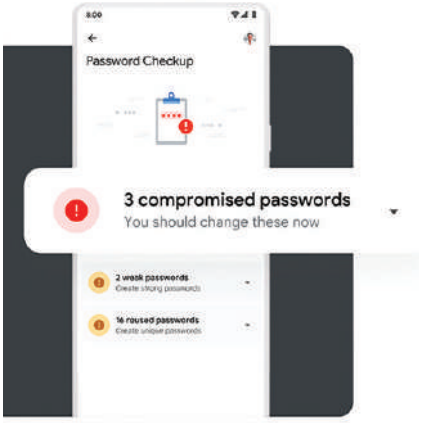
تطبيقات المراسلة الآمنة (التشفير) Secure Messaging Apps

توفّر تطبيقات المراسلة الآمنة تشفيرًا من طرف إلى طرف لحماية محادثات المستخدمين والتأكد من أن المستلمين المقصودين فقط يمكنهم الوصول إلى الرسائل. تمنع هذه التطبيقات أي طرف ثالث، بما في ذلك مزوّدي الخدمة والمتسللين من اعتراض الرسائل أو قراءتها، ويمكن التواصل بسرية دون المساس بالخصوصية.

برامج إدارة كلمات المرور Password Managers

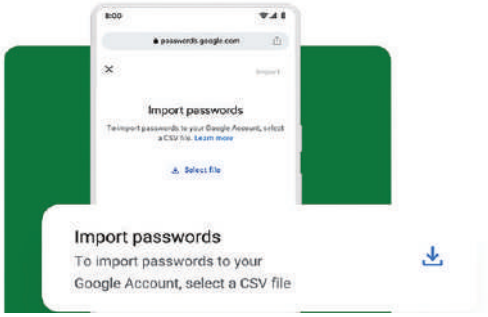
تُعَدّ برامج إدارة كلمات المرور من الأدوات الرقمية التي تساعد على تقديم الحماية والأمان لحسابات المستخدمين، وذلك من خلال المميّزات التالية:

- إنشاء كلمات مرور قوية وفريدة، وتخزينها وإدارتها بصورة آمنة.
- اكتشاف كلمات المرور الضعيفة لتقليل خطر التعرّض للاختراق.



شكل (1-5) يُوضّح اكتشاف برنامج Password Managers لكلمات المرور الضعيفة.

- الوصول إليها تلقائيا عند الحاجة، ودعم خاصية المزامنة بين الأجهزة.



شكل (1-6) يُوضح استيراد برنامج Password Managers لكلمات المرور.

- إنشاء رموز المصادقة الثنائية، ودعم خاصية مفاتيح الأمان U2F (Universal 2nd Factor).
- دعم إدارة بطاقات الائتمان عبر الإنترنت، مع طبقة إضافية من الأمان.
- ومن أشهر تطبيقات إدارة كلمات المرور (Lastpass - 1password - Google Password).

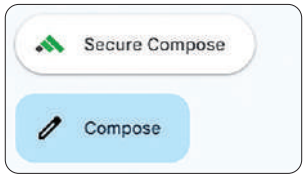
تشفير البريد الإلكتروني Encryption for Email

توفير طبقة إضافية من الحماية للمعلومات التي تتمّ مشاركتها عبر البريد الإلكتروني. حيث يتم تشفيرها بطريقة لا يمكن إلا للمُرسل إليه فكّ تشفيرها وقراءتها.

وهناك العديد من الخدمات الرقمية مثل Pretty Good Privacy (PGP) أو Secure Multipurpose Internet Mail Extension (S/MIME)، التي تتيح تشفير الرسائل الإلكترونية، ونتناول منها أداة FlowCrypt:

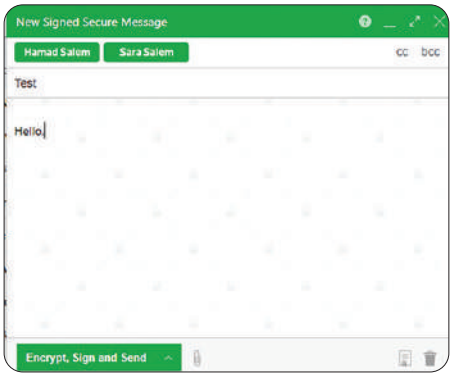
- تثبيت الأداة من موقع Chrome Web Store الخاص بمتصفح Chrome.

- إنشاء حساب جديد، ومنحه الصلاحيات اللازمة لإدارة عملية تشفير البريد Gmail.
- اختيار إرسال رسالة إلكترونية جديدة مشفّرة.



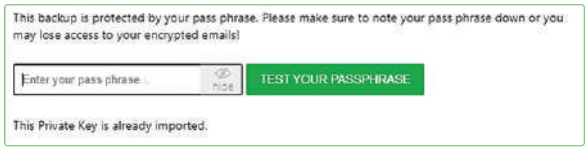
شكل (1-7) يُوضح بداية عملية تشفير الرسالة.

- تحديد جهات الاتصال، وكتابة عنوان ومحتوى الرسالة، وإرسالها.



شكل (1-8) يُوضح شاشة كتابة الرسالة المُشفرة.

- يتم فتح الرسالة من مُستقبل الرسالة بعد إدخال رمز التشفير الصحيح.

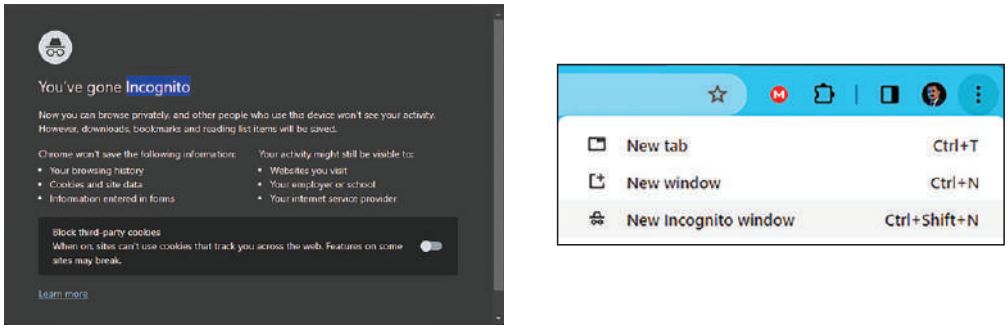


شكل (1-9) يُوضح مكان إدخال رمز التشفير لدى مُستقبل الرسالة المُشفرة.

الخصوصية في متصفحات الإنترنت Privacy-Focused Web Browsers

توفّر متصفّحات الويب الآمنة (مثل Mozilla Firefox أو Brave) تعزيز ميزات الأمان. عن طريق التحكّم بإعدادات وإضافات تتيح للمستخدمين الحفاظ على مستوى أعلى من الخصوصية أثناء تصفّح الإنترنت، ومنها خاصية التصفح المتخفّي Incognito بمتصفّح Google Chrome، حيث توفّر الخصوصية لأي مستخدم لمتصّفح الإنترنت من خلال:

- عدم الاحتفاظ بسجل التصفّح الخاص بك.
- عدم الاحتفاظ بملفات تعريف الارتباط وبيانات الموقع.
- عدم الاحتفاظ بالمعلومات المدخلة في النماذج (اسم المستخدم وكلمات المرور).



شكل (1-10) يُوضح خاصية التصفح المتخفي Incognito بمتصفح Google Chrome.

المحافظ الرقمية وطرق الدفع الآمنة Digital Wallets and Secure Payment Methods

توفّر المحافظ الرقمية (مثل Apple Pay أو Google Wallet) طرق دفع آمنة لحماية المعلومات المالية، للحفاظ على المعلومات، وتقليل مخاطر الوصول إليها.

إدارة التصحيحات الأمنية Patch Management

يُعدّ تحديث الأنظمة، البرامج، والتطبيقات بانتظام أحد أهم الإجراءات الأمنية الأساسية، لتعزيز مبدأي النزاهة والتوفر، ومنع الاختراقات الناتجة عن ثغرات معروفة. فالمطورون يُطلقون تحديثات لسدّ الثغرات الأمنية المكتشفة حديثًا، وفي حال عدم تثبيت هذه التحديثات تظل الأنظمة عرضة للاختراق، وتشتمل إدارة التصحيحات:

- مراقبة التحديثات الرسمية من الشركات.
- اختبار التصحيحات قبل تثبيتها (خاصة في البيئات الكبيرة).
- تطبيق التحديثات في أقرب وقت ممكن.
- تسجيل التحديثات ومراجعتها.

أوراق العمل

ورقة عمل (1) - ورقة عمل لاصفية:

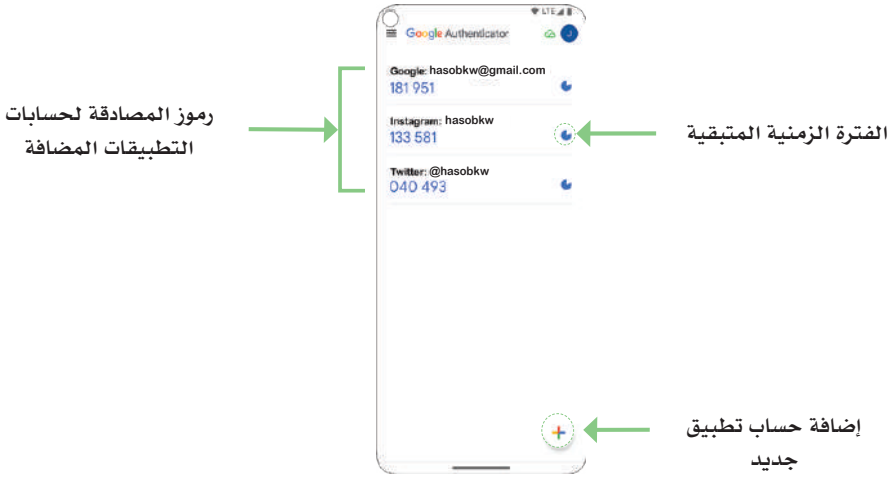
إنشاء مصادقة باستخدام تطبيق Google Authenticator:

1. حمّل التطبيق من متاجر الهواتف الذكية.

Apple store

Andriod Store

2. أنشئ حساباً على التطبيق من خلال حساب Google.
3. أضف التطبيقات المختلفة المراد تأمينها من خلال المصادقة، ومثال على ذلك تطبيق التواصل الاجتماعي Instagram.
4. أدخل Login على تطبيق Instagram، بإدخال كلمة المرور الخاصة بالتطبيق.
5. سيطلب منك تطبيق Instagram إدخال الرمز الذي يتمّ توليده باستخدام تطبيق المصادقة Google Authenticator، كما هو موضح بالشكل.



شكل (1-11) يُوضح قائمة التطبيقات التي تعتمد المصادقة.

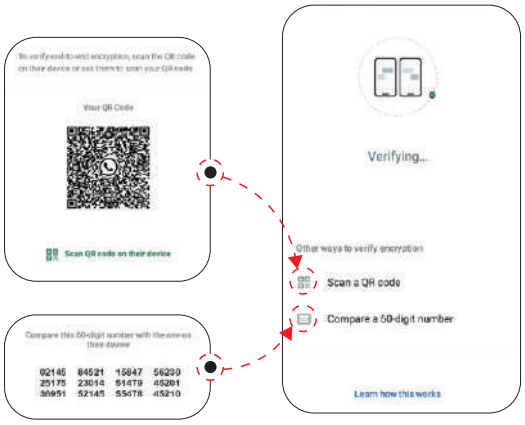
6. انسخ الرمز ثم أضفه في تطبيق Instagram خلال الفترة الزمنية المحددة.
7. شغّل تطبيق Instagram بشكل آمن.

ورقة عمل (2) - ورقة عمل لاصفية:

تتميّز المحادثات المشفّرة بينك وبين شخص آخر في تطبيق WhatsApp برمز أمان خاص بها. يُستخدم هذا الرمز للتحقّق من أنّ المكالمات والرسائل التي ترسلها إلى تلك الدردشة مشفّرة تمامًا.

ويمكن الوصول لهذا الرمز من خلال:

- شاشة معلومات[1] جهة الاتصال Contact Info الخاصة بأحد الأشخاص الذين تتواصل معهم.
- اختيار Encryption.
- التحقّق من رمز الأمان Verify encryption.
- اختيار أيّ من رمز QR أو الرقم المكون من 60 رقماً.



شكل (1-12) يُوضح تشفير المحادثات في تطبيق WhatsApp.

تُعَدّ هذه الرموز فريدة لكل محادثة، ويمكن مقارنتها مع الطرف الآخر للتحقّق من أنّ الرسائل بين الطرفين مشفّرة.

---

1 يجب مراعاة اختلاف الشاشات في أنظمة تشغيل الهواتف الذكية.
يُعدّ التحقق من رمز الأمان عملية اختيارية للمحادثات المشفّرة بين الطرفين.

ورقة عمل (3) لاصفية

- ناقِشْ معلمك وأسرتك كيفية إجراء المصادقة الثنائية بين تطبيق وزارة التربية الرسمي، وتطبيق هويتي، ثم سجّل الخطوات هنا.

الخطوات

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ممارسات الأمن السيبراني Cybersecurity Practices

يُمكن اتخاذ بعض الإجراءات والتدابير الخاصة من أجل الحفاظ على CIA في الأمن السيبراني:

- **كلمات مرور قوية Strong passwords**: استخدم كلمات مرور قوية وفريدة لكل الحسابات عبر الإنترنت وتَجَنّبْ كلمات المرور التي يمكن تخمينها.
- **المُصادقة Authentication**: تفعيل المصادقة الثنائية أو المتعددة لحساباتك المختلفة حال توافرها.
- **حماية الأجهزة Security**: تثبيت برامج مكافحة الفيروسات والبرامج الضارة وتحديث البرامج بانتظام واستخدام تشفير Encryption قوي للبيانات المهمة.
- **مُكافحة التصيّد الاحتيالي Phishing**: عدم فتح روابط Links لا تعرف مصدرها أو تحميل مرفقات من مصادر غير معروفة، وعدم الوقوع فريسة لرسائل البريد الإلكتروني أو الرسائل التي تحاول استدراجك للكشف عن معلومات شخصية.
- **استخدام مواقع الويب والتطبيقات الموثوقة**: التزم بالأنظمة الأساسية ذات الإجراءات الأمنية القوية والمصادر التي لديها موثوقية.
- **الحذر على وسائل التواصل الاجتماعي**: تقليل المعلومات الشخصية التي تشاركها عبر الإنترنت، وتفعيل إعدادات الخصوصية لكل تطبيق.
- **إدارة عملية مشاركة البيانات Data sharing**: التعرف على كيفية مشاركة البيانات عبر الإنترنت، وضبط إعدادات الخصوصية وفقًا لذلك.
- **الإبلاغ عن أي نشاط مشبوه Report**: الإبلاغ عن أي نشاط مشبوه أو انتهاكات مشتبه بها إلى مسئولي النظام الأساسي، أو السلطات المختصة (إدارة مكافحة الجرائم الإلكترونية[2]).
- **التحقق من مصادر المعلومات**: تأكد من صحة المعلومات ومصدرها قبل الوثوق بها.
- **النسخ الاحتياطي للبيانات Data backup**: يجب التأكّد من إجراء عمليات النسخ الاحتياطي للملفات والمستندات المهمة بانتظام لضمان الوصول إليها في حال تمّ اختراق النسخة الأصلية منها.

---

2 إدارة مكافحة الجرائم الإلكترونية تتلقّى جميع البلاغات بخصوص كل ما يخالف القانون وتقوم بإجراء التحريات للتأكّد من صحّة البلاغات وجدية المعلومات واتخاذ الإجراءات القانونية اللازمة حيالها. للبلاغات يرجى الاتصال على رقم الطوارئ 97283939، سيتمّ التعامل معك بسرية تامة

الوظائف في مجال الأمن السيبراني

يقدم مجال الأمن السيبراني مجموعة واسعة من الوظائف، ويركز كل منها على جوانب مختلفة لحماية الأنظمة والمعلومات الرقمية من وصول غير المّصرح به والهجمات والتهديدات، وفيما يلي بعض الوظائف الشائعة للأمن السيبراني:

- **مُحلل الأمن السيبراني Cybersecurity Analyst:**

مراقبة أنظمة الكمبيوتر والشبكات والتطبيقات لاكتشاف انتهاكات الأمان أو الأنشطة المشبوهة. وتحليل مخاطر الأمان، والتحقيق في الحوادث السيبرانية، وتنفيذ تدابير لمنع الهجمات المستقبلية.

- **مُهندس أمن Security Engineer:**

تصميم أنظمة الأمان وتطويرها، وتنفيذها لحماية الشبكات والبنية التحتية للأنظمة، (الجدران النارية وأساليب التشفير وأنظمة الكشف عن الاختراق). وكذلك إجراء تقييمٍ للثغرات باختبارها لتحديد الضعف في الأنظمة ومعالجتها.

- **مُستشار أمني Security Consultant:**

تقديم نصائح وإرشادات متخصصة للمؤسسات حول كيفية تحسين وضعها الأمني. وتقييم المخاطر، وتطوير سياسات الأمان والإجراءات، وإنشاء خطط استجابة للحوادث، وتقديم توصيات لتعزيز ضوابط الأمان.

- **مُخترق أخلاقي Ethical Hacker:**

تحديد الثغرات في أنظمة الكمبيوتر والشبكات، والقيام بمحاولات اختراق مصرح بها لمحاكاة الهجمات الحقيقية ومساعدة المؤسسات في تعزيز دفاعاتها. يحتاج القراصنة الأخلاقيون إلى فهم عميق لتقنيات القرصنة ولغات البرمجة وأدوات الأمان.

- **مُهندس تصميم أنظمة أمنية Security Architect:**

تصميم أنظمة تكنولوجيا المعلومات الآمنة وشبكاتها، وتطوير طرق الأمان وإنشاء سياسات الأمان، وضمان الامتثال للمعايير والتشريعات الصناعية. ويتعاون مهندسو بنية الأمان مع أصحاب المصلحة المختلفين لمواءمة متطلبات الأمان مع أهداف العمل التجارية.

- **المستجيب للحوادث Incident Responder:**

التحقيق في الحوادث، والاستجابة للحوادث الأمنية، مثل اختراقات البيانات أو العدوى بالبرامج الضارة. وتحليل أنماط الحوادث، واحتواء التهديدات، وإجراء تحليل مدعوم بالأدلة لتحديد سبب الحوادث. كذلك تطوير خطط استجابة للحوادث وتقديم توصيات لتحسين قدرات كشف الحوادث واستجابتها

- **مُحلل مركز العمليات الأمنية Security Operations Center Analyst:**

مراقبة الأحداث والتنبيهات الأمنية، والاستجابة للحوادث الأمنية المحتملة. من خلال استخدام أدوات إدارة المعلومات والأحداث الأمنية (SIEM) لتحديد التهديدات، وتحليل بيانات السجل، واتخاذ الإجراءات المناسبة. يحتاج محللو SOC إلى معرفة تقنيات الأمان وإجراءات التعامل مع الحوادث

الجوانب الأخلاقية الرئيسية في مجال الأمن السيبراني

- **احترام خصوصية الآخرين:**

- المعلومات الشخصية ملك للفرد: يجب التعامل مع معلومات الآخرين الشخصية، مثل الاسم والعنوان ورقم الهاتف وصورهم باحترام، وعدم مشاركتها مع أي شخص دون إذن منهم.

- الحذر عند مشاركة المعلومات على الإنترنت: الإنترنت عالم مفتوح، لذا يجب التفكير جيدًا قبل مشاركة أي معلومات شخصية أو صور أو مقاطع فيديو، فقد يراها أشخاص غير مرغوب فيهم.

- عدم التجسّس على الآخرين: التجسّس على حسابات الآخرين أو محاولة الوصول إلى معلوماتهم الخاصة دون إذنهم أمر غير أخلاقي وغير قانوني.

- **استخدام التكنولوجيا بمسؤولية:**

  - **عدم إلحاق الضرر بالآخرين: يجب استخدام الإنترنت لأغراض إيجابية وبناءة، وعدم استخدامه لإيذاء الآخرين أو نشر الشائعات أو التهديد.**

  - **احترام حقوق الملكية الفكرية: يجب احترام حقوق الملكية الفكرية للآخرين، وعدم نسخ البرامج والملفات أو مشاركتها دون إذن.**

  - **حماية الأجهزة من الفيروسات: يجب الحرص على حماية الأجهزة من الفيروسات وبرامج التجسس التي قد تسرق المعلومات أو تدمّر البيانات.**

- **التواصل بأمان واحترام:**

  - **التعامل بلطف واحترام مع الآخرين على الإنترنت: يجب التعامل مع الآخرين على الإنترنت بنفس الطريقة التي نتعامل بها في الحياة الواقعية باستخدام لغة مهذبة ومحترمة.**

  - **عدم التنمر الإلكتروني: التنمّر الإلكتروني هو استخدام الإنترنت لإيذاء شخص ما أو مضايقته، وهو أمر غير مقبول تمامًا.**

  - **الحذر عند التعامل مع الغُرَباء على الإنترنت: يجب عدم مشاركة المعلومات الشخصية مع الغُرَباء على الإنترنت، وعدم قبول طلبات الصداقة من أشخاص غير معروفين.**

- **الصدق والأمانة:**

  - **عدم انتحال شخصية الآخرين: انتحال شخصية شخص آخر على الإنترنت أمر غير أخلاقي وغير قانوني.**

  - **عدم نشر معلومات كاذبة: يجب التحقّق من صحّة المعلومات قبل نشرها على الإنترنت، وعدم نشر الشائعات أو المعلومات المضلّلة.**

  - **الإبلاغ عن أي سلوك غير أخلاقي: إذا شاهدت أي سلوك غير أخلاقي على الإنترنت، مثل التنمّر أو التحرّش أو نشر معلومات كاذبة يجب إبلاغ شخص بالغ موثوق به أو الجهات المختصة.**

الوحـدة الثانية - المُعالجة الرقمية

برمجة Python

1. مدخل إلى البرمجة
   Introduction to Programming
2. المتغيرات Variables
3. السلاسل النصية Strings
4. الشروط Conditions
5. التكرار Loops
6. تصحيح الأخطاء والاستثناءات
   Debugging & Exceptions

برمجة Python

مدخل إلى البرمجة

Introduction to programming

نواتج التعلم

- توضيح مفهوم البرمجة وأهميتها في عالم التكنولوجيا.
- تحديد المراحل الأساسية لتطوير البرامج الحاسوبية.
- كتابة خوارزمية مناسبة لحلّ مشكلة محدّدة بطريقة منظمة.
- تصميم خريطة تدفق Flowchart توضّح خطوات حلّ المشكلة.
- التعرف على لغة البرمجة Python ومجالات استخدامها.
- استخدام بيئة التطوير المتكاملة PyCharm IDE لكتابة التعليمات البرمجية وتنفيذها.
- استخدام الدالة المدمجة print() في طباعة المخرجات.
- التمييز بين أنواع البيانات المختلفة في Python.

يُمَثّل رمز الاستجابة السريعة QR رابطاً
لملفات أوراق العمل، ومصادر التعلم

مُقدمة

تُعَدّ البرمجة فنًا من الفنون التكنولوجية الحديثة، وهي عملية تُوجه الآلة لأداء مهمة معينة في إطار مجموعة من القواعد والتعليمات، وتُنمي البرمجة العديد من المهارات منها:

- **حل المشكلات:**

إنشاء برامج تعالج مشكلات محددة في الحياة العملية، مما يُحسّن من الكفاءة والانتاجية.

- **الإبداع:**

إتاحة الفرصة للإبداع من خلال البرمجة لإنشاء ألعاب، تطبيقات، مواقع إلكترونية، وبرامج أخرى تُعبّر عن الأفكار والرؤية.

- **تحسين المهارات:**

تحسين مهارات حلّ المشكلات، التفكير المنطقي، والإبداع، لتنمية المهارات الأساسية للنجاح في المجالات المختلفة.

- **توفير فرص وظيفية جديدة:**

الحصول على الوظائف في مجال التقنية الرقمية الحديثة، مثل تطوير البرامج، تحليل البيانات، أو الذكاء الاصطناعي، وهي مجالات تُعاني من نقص في الكوادر المؤهلة.

البرمجة

مجموعة من التعليمات والأوامر تُترجم إلى لغة الآلة التي بدورها توجه الحاسوب لتنفيذ مجموعة من المهام لحلّ مشكلة ما.

خطوات إنشاء برنامج

تمرّ عملية إنشاء البرنامج بمجموعة من الخطوات والإجراءات:

1. **تحديد المشكلة:**

تُعرف المشكلة على أنها موقف حياتي يتطلّب حلولاً محددة للوصول لهدف ما.

المشكلة: حساب مساحة المستطيل، ولحلّ أي مشكلة يجب أولاً تحديد المعطيات المتاحة لطول المستطيل وعرضه، بعد ذلك معرفة العمليات والإجراءات اللازمة للوصول إلى الحل، من خلال قانون مساحة المستطيل = الطول x العرض.

2. **إعداد خطة الحل (الخوارزمية Algorithm):**

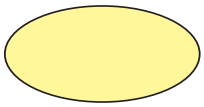
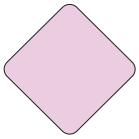
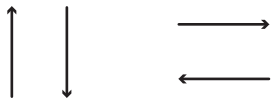
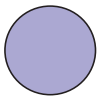
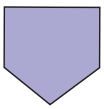
مجموعة من الخطوات المنطقية والمتسلسلة لحل مشكلة ما، وقد أُطلق عليها هذا الاسم نسبةً إلى محمد بن موسى الخوارزمي، عالم الرياضيات المُسلم، وهي طريقة وصفية توضح خطوات حل المشكلة.

كتابة خوارزمية حساب مساحة المُستطيل.

1- بداية البرنامج.
2- إدخال طول المستطيل. ..... المُدخلات
3- إدخال عرض المستطيل. ..... المُدخلات
4- حساب مساحة المستطيل. ..... المُعالجة
5- طباعة مساحة المستطيل. ..... المُخرجات
6- نهاية البرنامج.

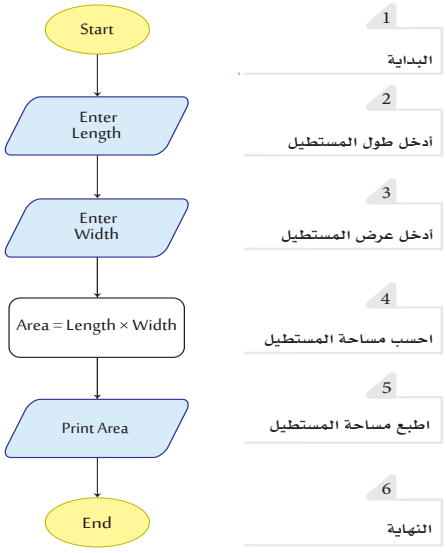
3. خرائط التدفق Flowcharts:

إحدى أنواع المخططات الرسومية التي توضّح ترتيب العمليات اللازمة لحل مشكلة ما، حيث تستخدم أشكال قياسية متفق عليها للدلالة على العمليات المختلفة التي تُسهل على المبرمج فهمها لتحويلها إلى برنامج مكتوب بإحدى لغات البرمجة، والجدول التالي يُوضح وظيفة كل شكل:

الشكل	الوظيفة
	بداية البرنامج أو نهايته End / Start
	عمليات الإدخال أو الإخراج Input / Output
	عملية المعالجة Process
	اتخاذ قرار (التفرع) Decision
	خطوط الاتجاه والتوصيل بين الأشكال Arrows
	الواصلة في الصفحة نفسها On-page connector
	الواصلة بين الصفحات Off-page connector

جدول (2-1) يُوضح أشكال ووظائف عناصر خرائط التدفق.

- التخطيط التالي يُمثل خريطة تدفق حساب مساحة المُستطيل:



شكل (2-1) يُوضح خريطة التدفق لحساب مساحة المُستطيل.

4. كتابة البرنامج:

تحويل خريطة التدفق لمجموعة من التعليمات بإحدى لغات البرمجة، حيث تختلف لغات البرمجة في قواعد كتابة أوامرها واستخداماتها.

5. اختبار البرنامج:

التأكد من أن البرنامج يعمل وفق آلية صحيحة من خلال تشغيله واختبار نتائجه ومقارنتها بالنتائج المتوقعة.

6. إصدار البرنامج:

بناء نسخة تنفيذية من البرنامج أو التطبيق، وعند تصميم المواقع يتم النشر على شبكة الإنترنت.

7. توثيق البرنامج:

عملية تسجيل المعلومات المتعلقة بالبرنامج وتنظيمها، بما في ذلك تصميمه، ووظائفه، وكيفية استخدامه، وآلية صيانته، لتيسير تطويره من المطورين فضلاً عن تسهيل استخدامه من المستخدم.

مثال: تصميم خوارزمية وخريطة تدفق لقسمة عددين.

مشكلة البرنامج:

تصميم خوارزمية لحساب ناتج قسمة عددين، مع الأخذ في الاعتبار بأنه لا يُمكن القسمة على العدد (صفر)، ورسم خريطة تدفق تُوضح ذلك.

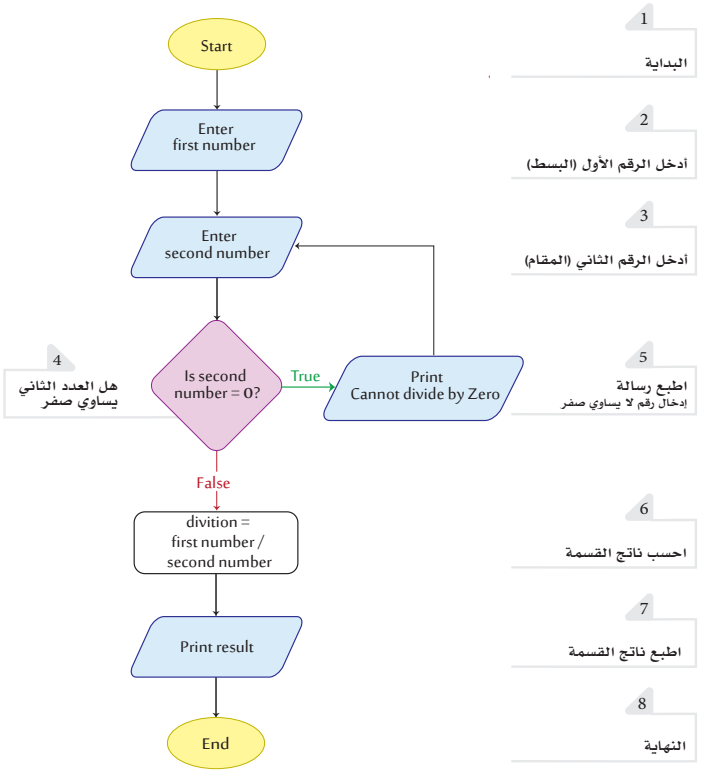
خوارزمية البرنامج:

1. بداية البرنامج.
2. إدخال العدد الأول (البسط).
3. إدخال العدد الثاني (المقام).
4. إذا كان العدد الثاني يساوي صفر:

   نعم: الانتقال إلى الخطوة 5.

   لا: الانتقال إلى الخطوة 6.
5. طباعة رسالة Cannot divide by Zero، ثم الانتقال للخطوة 3.
6. حساب ناتج القسمة.
7. طباعة ناتج القسمة.
8. نهاية البرنامج.

خريطة التدفق:



شكل (2-2) يُوضح خريطة التدفق لحساب ناتج القسمة.

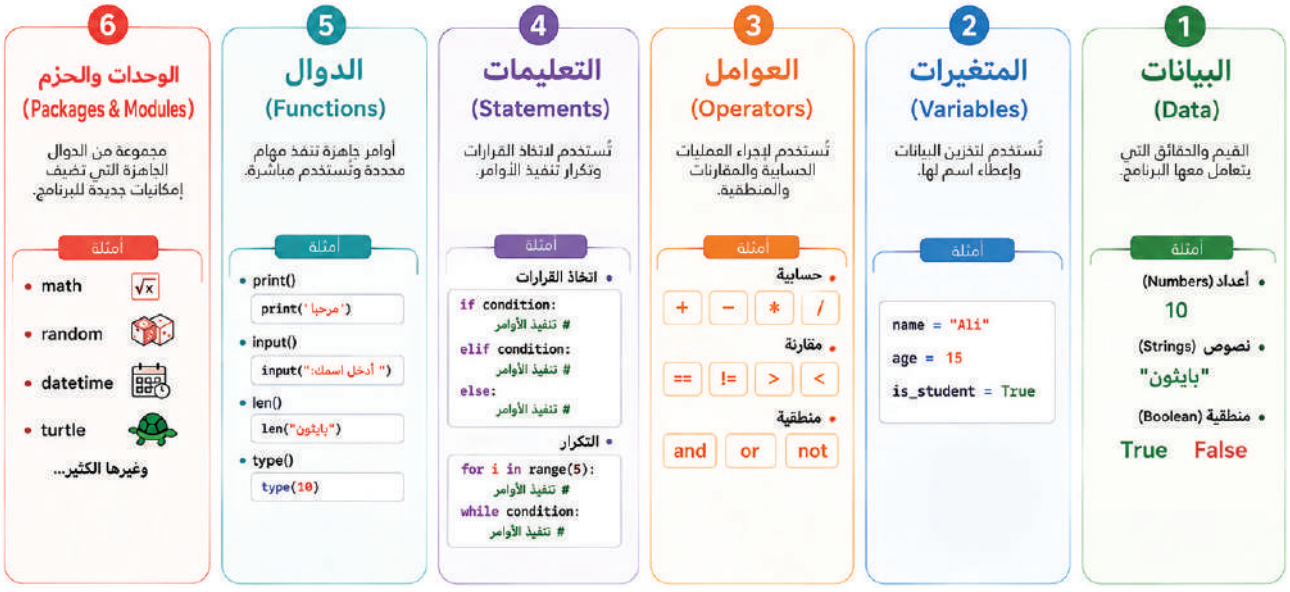
لغة البرمجة Python

تُعَدّ لغة Python من لغات البرمجة عالية المستوى، مفتوحة المصدر، قابلة للتوسع، سهلة التعلم وقوية الاستخدام. تُستخدم على نطاق واسع في مختلف المجالات، منها تطوير تطبيقات الويب، وبرامج سطح المكتب،وأدوات تحكم الأتمتة[1]، وتطبيقات الشبكة، وعلم البيانات، والذكاء الاصطناعي، والتعلم الآلي، وتحليل البيانات الضخمة، وإنشاء نماذج تنبؤية، وذلك لسهولة استخدامها و وجود العديد من الوحدات والحزم المتخصصة. كذلك يتمتع مجتمع Python بدعم كبير من المطورين والمبرمجين، مما يسهل الحصول على المساعدة والدعم المستمر.

كذلك تُعَدّ Python لغة كائنية التوجه Object-Oriented Programming، وتُمثل: (الأرقام، النصوص، القوائم، القواميس، الدوال، وغير ذلك) ككائنات Objects.

المفاهيم الأساسية في لغة Python

تُمثل هذه المفاهيم اللبنات الأساسية في لغة Python، حيث تتكامل لإنشاء البرامج والتطبيقات ومعالجة البيانات بكفاءة.



شكل (2-3) يُوضح شكل توضيحي يمثل المفاهيم الأساسية في لغة Python.

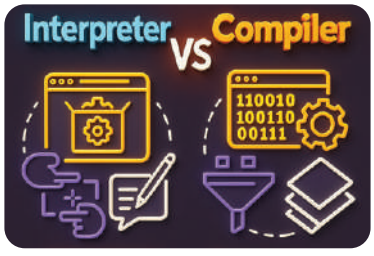
1 مجموعة من الإجراءات لإنجاز مهمة ما بأقل قدر من التدخل البشري.

الفرق بين Compiler و Interpreter

تعتمد لغات البرمجة على مبدأ تحويل الأوامر البرمجية من تمثيلٍ عالي المستوى إلى تعليماتٍ تنفيذية بلغة الآلة يستطيع المعالج فهمها، والتعامل معها وتنفيذها من خلال نظام معالجات لغات البرمجة Programming languages processing system الذي يشير إلى الخطوات المطلوب تنفيذها من أجل تحويل الأوامر البرمجية إلى لغة الآلة، ويتمّ عبر برامج وسيطة كالمترجم Compiler، والمُفسر Interpreter، ومن أهم ما يميزهما ما يلي:

العنصر	Compiler	Interpreter
طريقة العمل	يُترجم البرنامج بالكامل دفعة واحدة قبل التنفيذ.	يُترجم البرنامج سطرًا بسطر أثناء التنفيذ.
سرعة الترجمة	أبطأ في الترجمة (يأخذ وقتًا أكبر في البداية).	أسرع في الترجمة (فوري لكل سطر).
سرعة التنفيذ	أسرع في التنفيذ (بعد الترجمة).	أبطأ في التنفيذ (لأن الترجمة تتمّ أثناء التشغيل).
اكتشاف الأخطاء	تظهر جميع الأخطاء بعد الترجمة الكاملة.	تظهر الأخطاء مباشرة عند السطر الذي يحتويها.
سهولة التصحيح Debug	أصعب؛ لأن الأخطاء تظهر دفعة واحدة بعد الترجمة.	أسهل؛ لأن البرنامج يتوقف عند السطر الخاطئ.
مخرجات الترجمة	ملف تنفيذي مستقل.	لا يُنتج ملف تنفيذي، بل ينفذ التعليمات البرمجية مباشرة.

جدول (2-2) يُوضح الفرق بين Compiler و Interpreter.



مع العلم بأن لُغة Python تجمع بين المترجم والمفسر، حيث تُترجم التعليمات البرمجية أولًا إلى Byte Code[2]، ثُمّ تُفسّرها وتنفذها.

2 Byte Code تمثيل وسيط للبرنامج يتمّ إنشاؤه بعد ترجمة التعليمات البرمجية، وتكون جاهزة للتنفيذ من آلة افتراضية.

كتابة البرامج في Python

- يُمكن استخدام تطبيقات معالجات النصوص مثل Notepad، لكتابة التعليمات البرمجية بلغة Python ومِنْ ثَمّ حفظ الملف بامتداد py.
- يُمكن استخدام إحدى بيئات التطوير المتكاملة Integrated Development Environment (IDE) التي توفر الأدوات اللازمة لكتابة التعليمات وتحريرها واختبارها وتصحيح أخطائها في مكان واحد مثل:
  - العديد من مواقع تحرير التعليمات البرمجية على الإنترنت مثل: onlinegdb.com و replit.com و online-python.com
  - مُحرر Visual Studio Code: الخاص بشركة Microsoft.
  - مُحرر PyCharm: الذي سيستخدم خلال هذا الكتاب.

بيئة التطوير المتكاملة PyCharm

- تحميل البرنامج.

يُمكن الحصول على آخر إصدار من PyCharm من الموقع الرسمي jetbrains.com/pycharm

تحميل برنامج PyCharm

- اختيار نظام التشغيل المناسب. Windows macOS Linux
- تثبيت البرنامج.

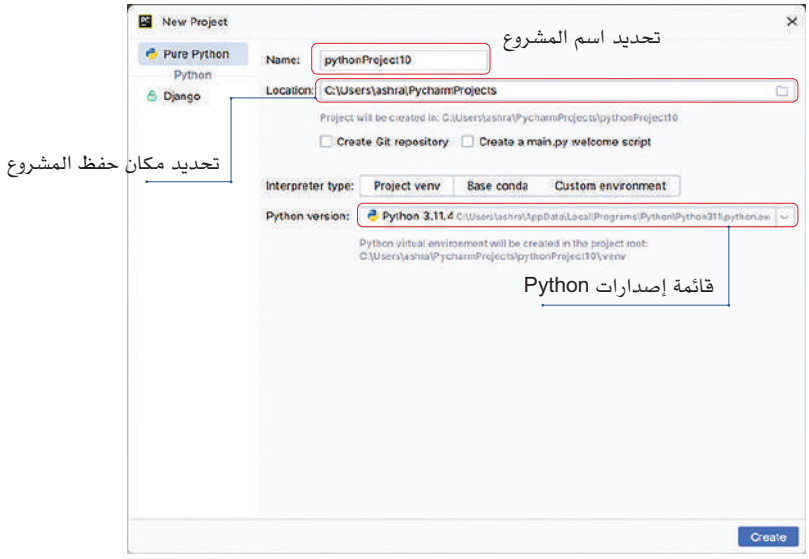
نافذة بيئة التطوير المتكاملة PyCharm

شكل (2-4) يُوضح شاشة البدء لبرنامج PyCharm.

- **القوائم الرئيسية Main Menu: إظهار شريط قوائم البرنامج.**
- **شجرة المشروع والملفات Project: عرض مسار المشروع الحالي وملفات Python التي يحتويها.**
- **شاشة البدء Welcome Screen: عرض مفاتيح روابط لبعض العمليات الشائعة.**

إنشاء مشروع جديد

يتمّ إنشاء مشروع جديد، من قائمة File نختر الأمر new project.

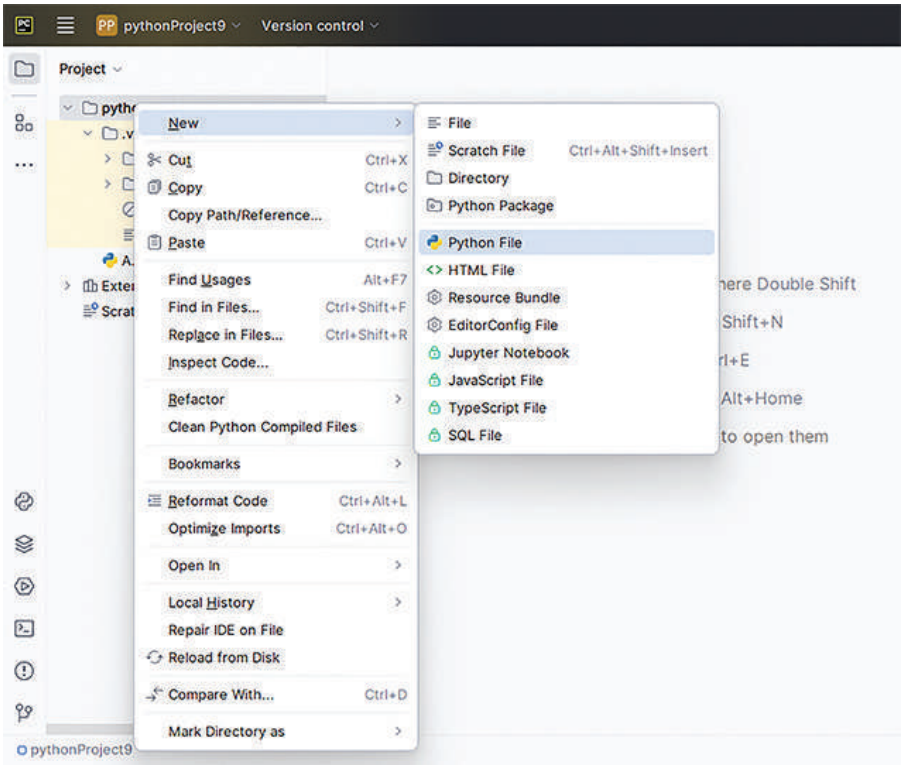


شكل (2-5) يُوضح خطوات إنشاء مشروع جديد.

إنشاء ملف Python جديد

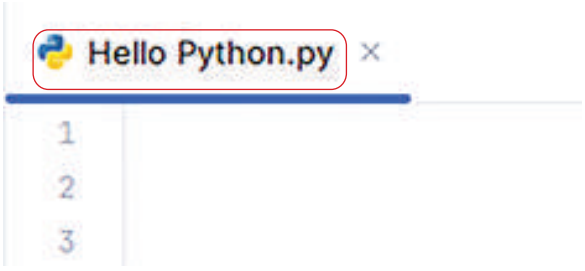
يُمكن أن يحتوي المشروع على العديد من ملفات التعليمات، ولإنشاء ملف Python جديد:

1. يتم الضغط بالزر الأيمن على اسم المشروع في منطقة Project (شجرة المشروع والملفات).
2. اختيار الأمر Python File من قائمة New.



شكل (2-6) يُوضح خطوات إنشاء ملف جديد.

3. كتابة اسم الملف، ثُمّ الضغط على مفتاح Enter.



شكل (2-7) يُوضح اسم الملف وامتداده.

تظهر منطقة تحرير التعليمات البرمجية، ويظهر في الأعلى علامة تبويب باسم الملف الحالي، ويظهر المؤشر في السطر البرمجي الأول

شكل (2-8) يُوضح شاشة كتابة التعليمات البرمجية وتحريرها Code Editor.

كتابة البرنامج الأول

في نافذة كتابة التعليمات البرمجية، وفي السطر الأول اكتب التعليمات البرمجية التالية، مع مراعاة الدقة.

```
1    print('I am a Python Developer')
```

تشغيل البرنامج Run

يمكن تشغيل البرنامج بإحدى الطرق التالية:

- القائمة المختصرة لمنطقة كتابة التعليمات البرمجية واختيار الأمر RUN.
- استخدام مفاتيح الاختصار Ctrl + Shift + F10 (تشغيل الملف الحالي).
- منطقة شريط الأدوات Toolbar، حيث يتمّ اختيار الملف المراد تنفيذه، أو اختيار Current File لتشغيل الملف الحالي، ثُمّ الضغط على الأداة RUN الموضحة في الشكل التالي:

شكل (2-9) يُوضح شريط أدوات تشغيل البرنامج.

تظهر نافذة جديدة وبها نتيجة تنفيذ البرنامج كالتالي:

شكل (2-10) يُوضح شاشة ناتج تشغيل البرنامج.

مفهوم البيانات Data Concept

مجموعة من الحقائق Facts، أو القيم الأولية Raw Data التي يتعامل معها البرنامج أثناء التنفيذ، وتُستخدم كأساس لتشغيل الأنظمة الرقمية ودعم عمليات المعالجة واتخاذ القرار. وقد تكون البيانات نصوصًا أو قيمًا عددية أو منطقية، حيث تعتمد عليها البرامج والتطبيقات الرقمية لتنفيذ العمليات المختلفة، والحصول على النتائج المطلوبة بكفاءة ودقة.

دورة معالجة البيانات Data Processing Cycle

تعتمد البرامج والأنظمة الرقمية على دورة معالجة البيانات لتنفيذ العمليات المختلفة، حيث يتم إدخال البيانات وتخزينها ومعالجتها ثم عرض النتائج على شكل معلومات مفيدة تساعد على تشغيل الأنظمة واتخاذ القرارات بكفاءة ودقة.

شكل (2-11) يُوضح دورة معالجة البيانات.

أنواع البيانات Data Types

تُوفر لغة Python مجموعة متنوعة من أنواع البيانات التي تُستخدم لتخزين القيم المختلفة والتعامل معها داخل البرامج، ويساعد تحديد نوع البيانات المناسب على تنفيذ العمليات البرمجية تنفيذاً صحيحاً ومنظماً، حيث تختلف طريقة تخزين البيانات ومعالجتها حسب طبيعتها واستخدامها داخل النظام، وتنقسم أنواع البيانات إلى عدة فئات رئيسية، وهي:

شكل (2-12) يُوضح أنواع البيانات في لغة البرمجة Python.

تُعَدُّ البيانات أساس عمل البرامج والأنظمة الرقمية، فعلى سبيل المثال في نظام حجز الطيران يتمّ إدخال بيانات المسافر والرحلة، ثُمّ تخزينها ومعالجتها للتحقق من الحجز وحساب سعر التذكرة، وبعد ذلك تُعرض النتائج مثل تأكيد الحجز وإصدار تذكرة السفر.

أوراق العمل

ورقة عمل (4): **التاريخ:**

برنامج حساب مساحة الدائرة

مشكلة البرنامج

من خلال دراستك لقوانين مساحة الأشكال الهندسية، احسب مساحة الدائرة بمعلومية نصف القطر.

الخوارزمية

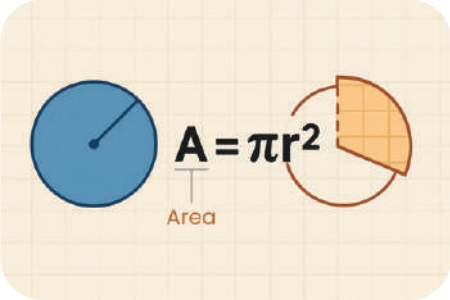
1. بداية البرنامج.
2. استقبال من المستخدم قيمة نصف قطر الدائرة.
3. حساب مساحة الدائرة مستعينًا بالمعادلة التالية:

   مساحة الدائرة = πr^2، حيث (π = 3.14، و r نصف قطر الدائرة).
4. عرض مساحة الدائرة على الشاشة.
5. نهاية البرنامج.

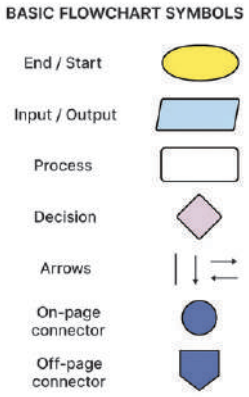
المطلوب:

خريطة التدفق

- ارسم خريطة التدفق المناسبة.



خريطة التدفق



ورقة عمل (5): **التاريخ:**

برنامج تحويل وحدات قياس درجات الحرارة.

مشكلة البرنامج

التحويل ما بين وحدات القياس المختلفة لدرجة الحرارة.

الخوارزمية

1. **بداية البرنامج.**
2. **استقبال من المستخدم درجة حرارة الطقس.**
3. **استقبال من المستخدم وحدة قياس درجة الحرارة المُدخلة (C أو F).**
4. **استخدام المعادلة التالية لتحويل درجة الحرارة من فهرنهايت إلى سيليزية:**

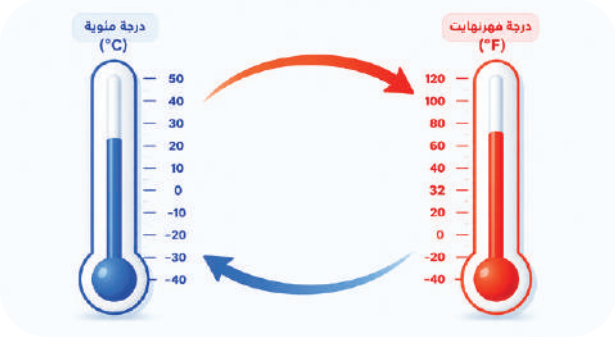
   $°C = (°F - 32) / 1.8$.
5. **استخدام المعادلة التالية لتحويل درجة الحرارة من سيليزية إلى فهرنهايت:**

   $°F = (°C \times 1.8) + 32$.
6. **عرض درجة الحرارة بعد التحويل لوحدة القياس المطلوبة.**
7. **نهاية البرنامج.**

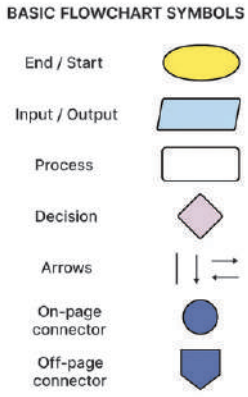
المطلوب:

خريطة التدفق

- **ارسم خريطة التدفق المناسبة.**



خريطة التدفق



برمجة Python

المتغيرات Variables

نواتج التعلم

- توضيح مفهوم المتغيرات ودورها في تخزين البيانات داخل البرنامج.
- الإعلان عن متغيرات باستخدام الصيغة الصحيحة وفق قواعد لغة Python.
- شرح قواعد تسمية المتغيرات وتطبيقها بشكل سليم عند كتابة التعليمات البرمجية.
- استخدام المتغيرات بفاعلية في تنفيذ العمليات البرمجية.
- توظيف الدالة المدمجة ()print في طباعة القيم والمخرجات على الشاشة.
- استخدام الدّالة ()input لاستقبال بيانات من المستخدم.
- تحويل البيانات بين الأنواع المختلفة بما يتناسب مع متطلبات البرنامج.

يُمَثّل رمز الاستجابة السريعة QR رابطاً
لملفات أوراق العمل، ومصادر التعلم

المتغيرات Variables

المُتغيّر هو مكان يتم حجزه في الذاكرة لتخزين البيانات بصورة مؤقتة أثناء تنفيذ البرنامج، وله اسم، وقيمة، ونوع: (الأعداد والنصوص والقوائم والقواميس، ...)، وتُمكن لغة Python من تغيير نوع وقيمة المُتغيّر أثناء تنفيذ البرنامج.

إنشاء المتغيرات Create Variables

تتميز Python بالتعرف على نوع المُتغيّر بشكل مباشر حسب القيمة المخصصة له تلقائيًا، ويمكن تخصيص القيم للمتغيرات باستخدام رمز التخصيص (=).

أنواع المتغيرات Data Types

نوع المتغير	نوع البيانات	استخداماته	مثال
String وتُكتب str	نص	تخزين سلسلة من الأحرف والأرقام التي يتعامل معها كنصوص، ولابد أن تكتب بين علامتي التنصيص المزدوجة " " أو المفردة ' '.	my_name = 'Ali' my_country = "Kuwait" my_address = '15 Beirut street'
Integer وتُكتب int	عدد صحيح	تخزين قيم عددية صحيحة.	months_count = 12
Float وتُكتب float	عدد عشري	تخزين قيم عددية عشرية.	my_salary = 1930.750 my_weight = 50.00
Boolean وتُكتب bool	منطقية	يأخذ إحدى القيمتين True (صحيح) أو False (خطأ)، وغالبًا ما تنتج عن عمليات المقارنة.	wifi_connected = True battery_low = False

جدول (3-1) يوضح أمثلة على أنواع البيانات.

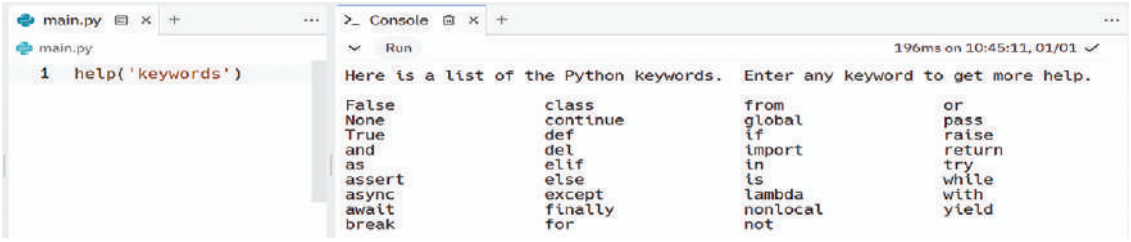
قواعد تسمية المتغيرات Naming Variables

يجب أن تتبع المتغيرات في Python قواعد معينة للتسمية:

- يجب أن يبدأ اسم المتغير (بحرف أو شرطة سُفلية)، ولا يبدأ برقم.
- يمكن أن يحتوي اسم المتغير على أحرف أبجدية وأرقام.
- لا يمكن أن يحتوي اسم المتغير على مسافات أو رموز خاصة (!@#$%^&*) باستثناء الشرطة السُفلية (_).
- لا يمكن استخدام الكلمات المحجوزة keywords reserved لتسمية المتغيرات.

ملاحظات:

- يراعى في تسمية المتغير حالة الأحرف (Small / Capital) sensitive for letters case.
- استخدام أسماء وصفية للمتغيرات ذات دلالة ومعنى.
- يُمكن التعرف على الكلمات المحجوزة في Python باستخدام الدالة help('keywords').



شكل (3-1) يوُضح الكلمات المحجوزة في Python.

دالة type():

دالة مُدمجة[1] في Python، تُستخدم للتعرف على نوع المتغير.

مثال 1:

```
1  # Determining the type of variable.
2  var_name = 'Abdul-Rahman Al-Sumait'
3  var_year = 1947
4  print(type(var_name))
5  print(type(var_year))
```

Program output

```
<class 'str'>
<class 'int'>
```

1 الدوال المُدمجة في لغة Python: دوال جاهزة تُستخدم لتنفيذ مهام محددة دون الحاجة إلى إنشائها أو تعريفها مسبقًا.

ملاحظات:

- يُستخدم الرمز # لإضافة تعليق Comment داخل البرنامج لتوثيق أو شرح التعليمات البرمجية، ولا يُنفّذ أثناء تشغيل البرنامج.
- المتغير الأول var_name: من نوع نصي <class 'str'>.
- المتغير الثاني var_year: من نوع عدد صحيح <class 'int'>.

دالة print()

دالة مُدمجة تُستخدم لعرض البيانات والمُخرجات على الشاشة، يمكن لها التعامل مع مختلف أنواع البيانات كالنصوص والأرقام، وغيرها.

استخدام الفاصلة (,) comma مع دالة print()

تُستخدم لتنسيق المخرجات حيث تُضيف مسافات بشكل تلقائي بين العناصر، مما يجعل النص أكثر وضوحًا وتنظيمًا دون الحاجة إلى إدخال المسافات يدويًا.

مثال 2:

الربط بين المتغيرات.

```
1  var_x = 'Free'
2  var_y = 'Kuwait'
3  print(var_x, var_y)
```

Program output

```
Free Kuwait
```

مثال 3:

ربط النصوص والمتغيرات.

```
1  var_age = 18
2  print('Your age = ', var_age)
```

Program output

```
Your age = 18
```

العمليات الحسابية

تستخدم Python العمليات الحسابية الأساسية، وفي الجدول التالي رموز العمليات الحسابية ووظيفتها.

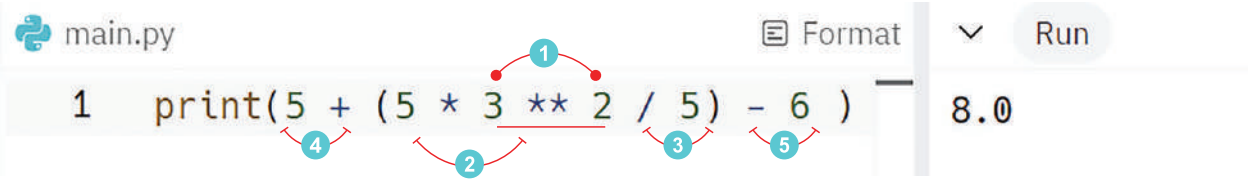
الرمز	العمليات الحسابية	مثال
+	الجمع Addition	1 num_1 = 23 2 num_2 = 5 3 print(num_1 + num_2) **28**
-	الطرح Subtraction	1 num_1 = 23 2 num_2 = 5 3 print(num_1 - num_2) **18**
*	الضرب Multiplication	1 num_1 = 23 2 num_2 = 5 3 print(num_1 * num_2) **115**
/	القسمة Division	1 num_1 = 23 2 num_2 = 5 3 print(num_1 / num_2) **4.6**
//	ناتج القسمة الصحيح بدون الكسور Floor division	1 num_1 = 23 2 num_2 = 5 3 print(num_1 // num_2) **4**
%	باقي القسمة Modulus	1 num_1 = 23 2 num_2 = 5 3 print(num_1 % num_2) **3**
**	رفع العدد إلى أُس محدد Exponentiation	1 num_2 = 5 2 print(num_2 ** 2) 3 print(num_2 ** 3) **25** **125**

جدول (3-2) يوُضح رموز العمليات الحسابية ووظيفتها.

أولويات تنفيذ العمليات الحسابية

1. العمليات داخل الأقواس.
2. رفع الأسس.
3. الضرب والقسمة.
4. الجمع والطرح.
5. يتم تنفيذ العمليات المتكافئة من اليسار إلى اليمين.

مثال 4:



دالة input()

دالة مُدمجة في Python تُستخدم لاستقبال البيانات المُدخلة من المستخدم، مثل أسماء المستخدمين وعناوين البريد الإلكتروني وأرقام الهواتف.

صيغة الدالة Syntax:

تكتب صيغة الدالة input() كالتالي:

input([prompt])

- تُستخدم لعرض رسالة إلى المستخدم [اختياري].
- الدالة input() تُرجع أي قيمة مُدخلة من المستخدم كسلسلة نصية String.

مثال 5:

```
1  name = input('Enter your name: ')
2  print('Nice to meet you', name)
```

Program output

```
Enter your name: Ahmed
Nice to meet you Ahmed
```

تفسير النص البرمجي:

- يعرض رسالة للمستخدم :Enter your name.
- ينتظر البرنامج حتى يُدخل المستخدم اسمًا ويضغط على مفتاح Enter.
- تخصيص القيمة المُدخلة إلى المتغير name.
- طباعة العبارة [Name] Nice to meet you.

ملاحظة: يجب الانتباه إلى أن دالة input() تُرجع سلسلة string. ويمكن تحويلها إلى عدد صحيح أو عدد عشري إذا كانت المُدخلات عددية، باستخدام الدالة int() أو الدالة float().

مثال 6:

إجراء عملية حسابية بجمع المتغيرين الأول والثاني كأرقام.

```
1    num_1 = input('Enter first number: ')
2    num_2 = input('Enter second number: ')
3    num_1 = int(num_1)
4    num_2 = int(num_2)
5    print(num_1 + num_2)
```

Program output

```
Enter first number: 10
Enter second number: 12
22
```

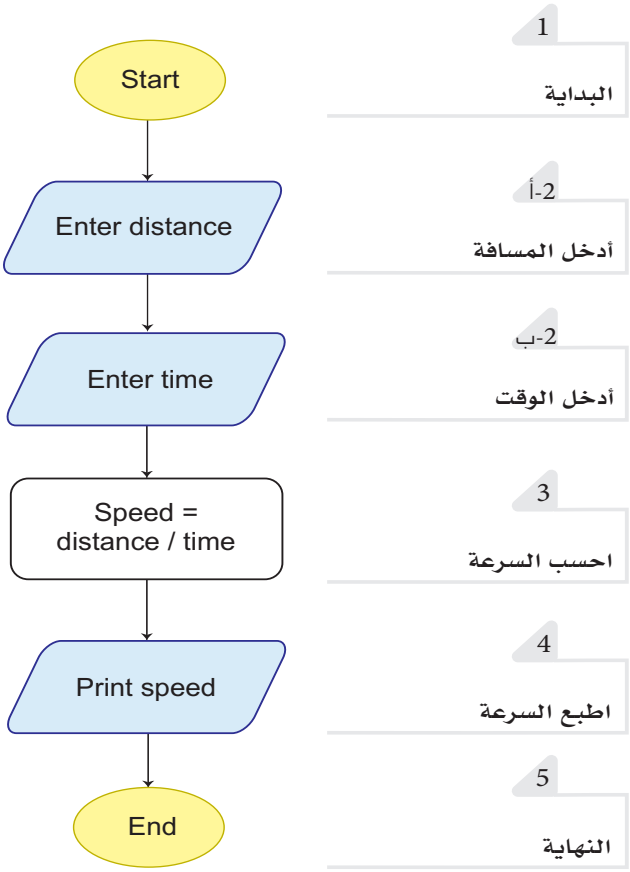
كما يمكن التحويل مباشرة أثناء عملية الإدخال كالتالي:

```
score = int(input('Enter player score: '))
```

ناتج استخدام الدالة input()، والدالة int() تُعيد العدد المدخل إلى عدد صحيح.

مثال تطبيقي 7:

من خلال مصادر التعلم المتاحة، ادرس خريطة التدفق، التي تمثل قانون حساب (السرعة العددية) الذي درسته في منهج الفيزياء، ثم ناقش مع معلمك وزملائك كيفية تحويل خريطة التدفق إلى برنامج مكتوب بلغة Python.



شكل (3-2) يُوضح خريطة تدفق مثال تطبيقي.

1. بداية البرنامج.
2. يطلب من المستخدم إدخال قيمتيْ المسافة والزمن باستخدام دالة input().

أ. التعليمة البرمجية لإدخال المسافة distance.

```
var_distance = input('Enter the distance (km): ')
```

ب. التعليمة البرمجية اللازمة لإدخال قيمة الزمن time.

```
var_time = input('Enter the time (hours): ')
```

- تظهر رسالة للمستخدم لإدخال قيمتي المسافة والزمن.
- تستقبل دالة input() جميع القيم كسلاسل نصية.

3. حساب السرعة القياسية باستخدام المعادلة التالية:

$$speed = distance / time$$

- التعليمة البرمجية اللازمة لحساب السرعة:

```
var_speed = float(var_distance) / float(var_time)
```

- إسناد ناتج العملية الحسابية للمتغير speed.
- المعامل (/) يمثل عملية القسمة.
- استخدام دالة float() لتحويل قيمتي المسافة والزمن من نصية إلى عشرية.

4. طباعة السرعة.
5. نهاية البرنامج.

التعليمات البرمجية:

```
1  var_distance = input('Enter the distance (km): ')
2  var_time = input('Enter the time (hours): ')
3  var_speed = float(var_distance) / float(var_time)
4  print('Speed = ', var_speed, 'km/h')
```

Program output

```
Enter the distance (km): 151
Enter the time (hours): 2
Speed = 75.5 km/h
```

أوراق العمل

ورقة عمل (6): **التاريخ:**

برنامج: رحلة إلى أبراج الكويت

مشكلة البرنامج

حساب تكلفة رحلة مدرسية لأبراج الكويت بمعلومية التكلفة: المواصلات، دخول الأبراج، الوجبة.

الخوارزمية

1. بداية البرنامج.
2. استقبال من المستخدم:
   أ. سعر دخول أبراج الكويت.
   ب. سعر الوجبة.
   ج. سعر المواصلات.
3. حساب إجمالي تكلفة الرحلة.
4. طباعة إجمالي تكلفة الرحلة.
5. نهاية البرنامج.

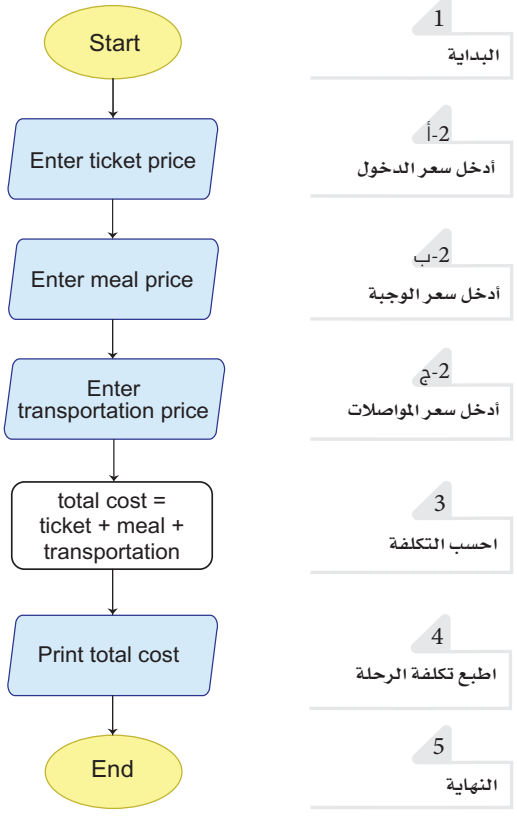
المطلوب:

خريطة التدفق

- ادرس خريطة التدفق المقابلة.

البرنامج

- أنشئ الملف School_Trip.py.
- اكتب التعليمات البرمجية اللازمة.
- اختبر البرنامج، وتأكد خلوه من الأخطاء.



شكل (3-3) يُوضح خريطة تدفق حساب تكلفة رحلة أبراج الكويت.

اكتب التعليمات البرمجية

01. .....

02. .....

03. .....

04. .....

05. .....

06. .....

07. .....

08. .....

09. .....

10. .....

11. .....

12. .....

13. .....

14. .....

15. .....

16. .....

17. .....

18. .....

ورقة عمل (7): **التاريخ:**

برنامج حساب المبلَغ المُستحق لحقائب السفر

مشكلة البرنامج

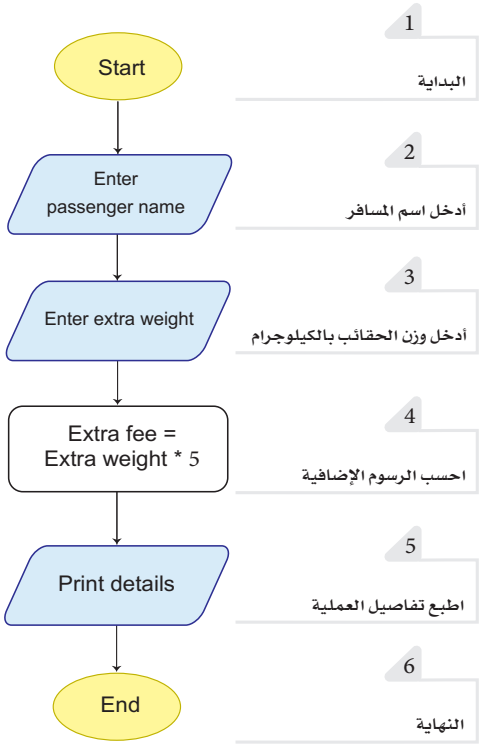
حساب المبلغ المستحق لحقائب السفر التي تخطت الوزن المسموح به لرحلة الطيران.

الخوارزمية

1. **بداية البرنامج.**
2. **استقبال من المُستخدم قيمة: اسم المسافر.**
3. **استقبال من المُستخدم قيمة: الوزن الإضافي بالكيلو جرام.**
4. **حساب المبلغ المستحق بضرب الوزن في خمسة دنانير كويتية.**
5. **طباعة التفاصيل.**
6. **نهاية البرنامج.**

المطلوب:

- **افتح الملف weight _calculator.py.**
- **ادرس خريطة التدفق المقابلة.**



شكل (3-4) يُوضح خريطة تدفق حساب الوزن الإضافي.

البرنامج

- أكمل التعليمات البرمجية اللازمة ليعمل البرنامج بشكل صحيح.

علمًا بأن:

- المتغير passenger_name يمثل اسم المسافر.
- المتغير extra_weight يمثل الوزن الإضافي.
- المتغير extra_fee يمثل المبلغ المستحق بضرب قيمة المتغير extra_weight في خمسة دنانير كويتية.

استكمل التعليمات البرمجية

```
1  # Get passenger name.
2  passenger_name = input('Enter passenger name: ')
3  # Get extra bag weight in kg (float).
4  extra_weight = .....
5  # Calculate extra fee.
6  extra_fee =.....
7  # Print details
8  .....
```

- اختبر البرنامج، وتأكد من خلوه من الأخطاء.

برمجة Python

السلاسل النصية Strings

نواتج التعلم

- التمييز بين علامات الاقتباس المستخدمة في إنشاء السلاسل النصية.
- استخدام عمليات التقطيع للوصول إلى أجزاء محددة من السلسلة النصية.
- تطبيق دوال شائعة لمعالجة السلاسل النصية.
- تنفيذ عمليات المقارنة بين السلاسل النصية باستخدام المعاملات المنطقية.
- معالجة مدخلات المستخدم النصية، وتحويلها حسب الحاجة لتنفيذ تعليمات برمجية.
- إنشاء برامج بسيطة تتضمن مدخلات ومخرجات تعتمد على معالجة السلاسل النصية.

يُمَثّل رمز الاستجابة السريعة QR رابطاً
لملفات أوراق العمل، ومصادر التعلم

السلاسل النصية String

مجموعة من الأحرف والأرقام والرموز تُميَز بعلامات اقتباس، وتُخزن في متغيرات يتم التعامل معها كقيمة نصية. تُعد السلاسل النصية قابلة للتكرار Iterable، كما أنها غير قابلة للتغيير Immutable، لذا فإن جميع العمليات التي تُجرى عليها لا تُؤثر على السلسلة الأصلية، بل تُنتج نسخة جديدة معدّلة تحتوي على التغييرات المطلوبة.

العمليات الأساسية على السلاسل النصية:

إنشاء السلاسل النصية:

يتم إنشاء أو تخصيص السلاسل النصية بعلامات اقتباس فردية ' ' أو مزدوجة " ".

مثال 1:

إنشاء سلسلة نصية باستخدام علامتيْ اقتباس فردية.

```
1    var_x = 'Free Kuwait'
2    print(var_x)
```

Program output

```
Free Kuwait
```

مثال 2:

إنشاء سلسلة نصية باستخدام علامتيْ اقتباس مزدوجة.

```
1    var_x = "Free Kuwait"
2    print(var_x)
```

Program output

```
Free Kuwait
```

الفهرسة [ ]Indexing :

الوصول إلى موقع الأحرف في سلسلة نصية [x]، حيث تبدأ عملية الفهرسة من رقم صفر.

Index Number	0	1	2	3	4	5	6	7	8	9	10
String	F	r	e	e		K	u	w	a	i	t

جدول (4-1) يوضح طريقة الوصول إلى موقع الأحرف في السلسلة النصية.

مثال 3 :

```
1  var_x = 'Free Kuwait'
2  print(var_x[0])
3  print(var_x[1])
```

Program output

```
F
r
```

تبدأ عملية الفهرسة بترتيب عكسي من نهاية السلسلة النصية برقم 1- والرقم الذي يسبقه 2- وهكذا.

Index Number	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
String	F	r	e	e		K	u	w	a	i	t

جدول (4-2) يوضح عملية الفهرسة بترتيب عكسي.

مثال 4 :

```
1  var_x = 'Free Kuwait'
2  print(var_x[-1])
3  print(var_x[-4])
```

Program output

```
t
w
```

القطع Slicing:

يمكن الحصول على جزء من السلسلة النصية بأرقام الفهارس [start : stop : step].

الصيغة	الوصف
str[x : y :1]	تُعيد النص بداية من الموقع x وحتى الموقع ما قبل y، والتحرك بمقدار خطوة واحدة من اليسار إلى اليمين.
str[x : y :-1]	تُعيد النص بداية من الموقع x وحتى الموقع ما قبل y، والتحرك بمقدار خطوة واحدة من اليمين إلى اليسار.
str[: y : 1]	تُعيد النص بداية من الموقع 0 وحتى الموقع ما قبل y، والتحرك بمقدار خطوة واحدة من اليسار إلى اليمين.
str[x : :1]	تُعيد النص بداية من الموقع x وحتى نهاية السلسلة النصية بمقدار خطوة واحدة من اليسار إلى اليمين.

جدول (4-3) يوضح أمثلة على عملية القطع.

ملاحظة: عند عدم تحديد مقدار واتجاه الخطوة يكون الاتجاه تلقائيًا من اليسار إلى اليمين بمقدار خطوة واحدة.

مثال 5:

استخدام القطع Slicing لتجزئة الاسم الكامل (Full Name) إلى أجزاء (الاسم الأول - الاسم الأوسط - الاسم الأخير).

```
1  var_name = 'Abdul-Rahman Hammoud Al-Sumait'
2  print('First Name: ', var_name[ : 12])
3  print('Middle Name: ',var_name[13 : 20])
4  print('Last Name: ',var_name[21 : ])
5  var_org='diA tceriD'
6  print('Reverse org. name:', var_org[ : : -1])
```

Program output

```
First Name:  Abdul-Rahman
Middle Name:  Hammoud
Last Name:  Al-Sumait
Reverse org. name: Direct Aid
```

مثال 6:

استخدام القطع Slicing للحصول على بيانات محددة (تقسيم الرقم المدني للحصول على [شهر، يوم] من تاريخ الميلاد).

Index Number	0	1	2	3	4	5	6	7	8	9	10	11
Civil ID	3	1	6	0	1	2	2	0	0	6	4	1

جدول (4-4) يوضح مثال على عملية الفهرسة.

```
1  civil_id = '316012200641'
2  month = civil_id[3 : 5]
3  day = civil_id[5 : 7]
4  print('Month = ',month)
5  print('Day = ',day)
```

Program output

```
Month = 01
Day =  22
```

جمع السلاسل النصية Concatenation:

يُستخدم معامل عملية الجمع (+) لربط سلسلتينْ نصيتينْ أو أكثر معًا لتكوين سلسلة نصية واحدة جديدة.

مثال 7:

```
1  print('Liberation ' + 'Day: ' + '26' +  ' ' + 'Feb')
```

Program output

```
Liberation Day: 26 Feb
```

ملاحظات:

- تم إضافة مسافة بعد كلمة Liberation، وكلمة Day:.
- تُمثل ' ' سلسلة نصية تحتوي على مسافة واحدة، وتُستخدم للفصل بين الكلمات أو العناصر النصية عند دمج السلاسل النصية.

مثال 8:

عند إدخال الأعداد باستخدام دالة input() دون تحويلها إلى أعداد رقمية، يتم التعامل معها كنصوص، لذلك يؤدي استخدام عملية الربط Concatenation إلى دمج القيم معًا بدلاً من جمعها حسابيًا.

```
1    num_1 = input('Enter  first number: ')
2    num_2 = input('Enter  second number: ')
3    print(num_1 + num_2)
```

Program output

```
Enter first number: 10
Enter second number: 12
1012
```

تكرار السلاسل النصية:

تكرار سلسلة نصية لأي عدد باستخدام معامل الضرب (*).

مثال 9:

```
1    print('#' * 10)
```

Program output

```
##########
```

دالة ( )len:

دالة مُدمجة تُستخدم للحصول على طول السلسلة النصية.

مثال 10:

```
1    var_x = 'Free Kuwait'
2    print(len(var_x))
```

Program output

```
11
```

دالة ( )capitalize:

أحد الأساليب المُدمجة Built-in Method[1]، تُستخدم لتحويل الحرف الأول من السلسلة النصية إلى حرف كبير Capital Letter، وبقية أحرف السلسلة النصية إلى أحرف صغيرة Small Letter.

Method Syntax: string.capitalize( )

مثال 11:

```
var_x = 'hello, cybersecurity!'
print(var_x.capitalize())
print(var_x)
```

Program output

```
Hello, cybersecurity!
hello, cybersecurity!
```

دالة ( )upper:

تُحول جميع الأحرف في السلسلة النصية إلى أحرف كبيرة.

Method Syntax: string.upper( )

مثال 12:

```
1  var_message = 'Hello, Cybersecurity!'
2  print(var_message .upper())
```

Program output

```
HELLO, CYBERSECURITY!
```

دالة ( )lower:

تُحول جميع الأحرف في السلسلة النصية إلى أحرف صغيرة.

Method Syntax: string.lower( )

1 الأساليب المُدمجة Built-in Method: دوال تُنفذ مهامًا معينةً، مُرتبطة بكائن object، ويفصل بينهما (.) عند الاستدعاء.

مثال 13:

```
1  var_subject = 'COMPUTER SCIENCE'
2  print(var_subject.lower())
```

Program output

```
computer science
```

دالة ( )find:

تُستخدم لإرجاع رقم فهرس **لأول ظهور** لسلسلة فرعية في السلسلة المحددة.

Method Syntax: string.find(sub, start, end)

sub: النص أو الحرف الذي تبحث عنه.
start: الموضع الذي يبدأ منه البحث (اختياري).
end: الموضع الذي ينتهي عنده البحث (اختياري).

مثال 14:

```
1  var_text = 'Free Kuwait'
2  index_chr1 = var_text.find('K')
3  index_chr2 = var_text.find('k')
4  index _word = var_text.find('Kuwait')
5  print(index_chr1)
6  print(index_chr2)
7  print(index_word)
```

Program output

```
5
-1
5
```

لاحظ: إذا لم يتم العثور على النص، فستُرجع دالة ( )find العدد 1-.

دالة ( )replace:

دالة تستخدم لاستبدال نص بنص آخر في السلسلة النصية.

Method Syntax: string.replace(old, new, count)

مثال 15:

```
1  old_text = 'The device is new. I like my device.'
2  new_text = old_text.replace('device', 'computer')
3  print(new_text)
```

Program output

```
The computer is new. I like my computer.
```

التعليمة f-string: Built-in Method

تُستخدم لكتابة النصوص مع القيم و المتغيرات ، بحيث يُكتب حرف f قبل النص و تكتب المتغيرات داخل الأقواس المعقوفة Curly Brackets { }.

مثال 16:

```
1  prophet_name = 'Muhammed'
2  prophet_age = 63
3  prophet_info = f'Name: {prophet_name} and Age: {prophet_age}'
4  print(prophet_info)
```

Program output

```
Name: Muhammed and Age: 63
```

ويُمكن استخدام التعليمة البرمجية f-string مباشرة مع دالة print.

```
1  print(f 'Name: {prophet_name} and Age: {prophet_age}')
```

التحكم في تنسيق الأرقام العشرية.

مثال 17 :

```
1   math_pi = 22 / 7
2   print(f'PI = {math_pi}')
3   print(f'PI = {math_pi :.2f}')
```

Program output

```
PI = 3.142857142857143
PI = 3.14
```

دالة ( )count :

تُستخدم لإيجاد عدد مرات ظهور سلسلة نصية.

Method Syntax: string.count(sub, start, end)

مثال 18 :

```
1   sura_alnas = '''
2   بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ
3   قُلْ أَعُوذُ بِرَبِّ النَّاسِ
4   مَلِكِ النَّاسِ
5   إِلَهِ النَّاسِ
6   مِن شَرِّ الْوَسْوَاسِ الْخَنَّاسِ
7   الَّذِي يُوَسْوِسُ فِي صُدُورِ النَّاسِ
8   مِنَ الْجِنَّةِ وَالنَّاسِ
9   '''
10  count = sura_alnas.count('النَّاس')
11  print(f 'The word النَّاس is repeated {count} times.')
```

Program output

```
The word النَّاس is repeated 5 times.
```

ملاحظة: تُستخدم علامات اقتباس الثلاثية ''' ''' لكتابة السلاسل النصية متعددة الأسطر.

دالة ( )center:

تُستخدم لتوسيط النص داخل سلسلة نصية بطول مُحدد، حيث يُمثل الطول العدد الكلي للسلسلة النصية، وذلك بإضافة فراغات تلقائيًا (الوضع الافتراضي)، أو استخدام رموز وأحرف على جانبي النص.

Method Syntax: string.center(width, fillchar)

مثال 19:

```
1    var_x = 'Security'
2    print(var_x.center(30))
3    print(var_x.center(30, '#'))
```

Program output

```
          Security
##########Security##########
```

ملاحظة: إذا كانت القيمة المحددة لطول السلسلة أقل من طول النص الأصلي أو تساويه، فلن تتم إضافة أي رموز للتوسيط وسيُعرض النص كما هو.

شكل (4-1) يُوضح مثال على توسيط النص باستخدام دالة ( )center.

أوراق العمل

ورقة عمل (8): **التاريخ:**

تأمين كلمة المرور

مشكلة البرنامج

عكس ترتيب الأحرف في السلسلة النصية لتصبح مؤّمنة (كلمة السر).

الخوارزمية

1. بداية البرنامج.
2. استقبال كلمة المرور من المستخدم.
3. تخزين الكلمة في متغير (password).
4. إنشاء المُستخدم متغيرًا جديدًا لتخزين الكلمة المعكوسة (reversed_password).
5. استخدام التقطيع Slicing لعكس الكلمة [1-::].
6. طباعة الكلمة المعكوسة على أنها كلمة مرور محمية.
7. نهاية البرنامج.

ملاحظة: هذا لا يُعد تشفيرًا حقيقيًا، لكنه مثال بسيط لشرح فكرة (إخفاء شكل كلمة المرور).

المطلوب:

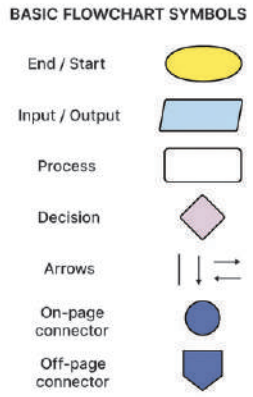
خريطة التدفق

- ارسم خريطة التدفق المناسبة.

البرنامج

- أنشئ ملف Python باسم reverse_string.py.
- اكتب التعليمات البرمجية اللازمة.
- اختبر البرنامج، وتأكد خلوه من الأخطاء.

خريطة التدفق



ورقة عمل (9): **التاريخ:**

تقطيع السلسلة النصية

مشكلة البرنامج

استخراج الجزء الصحيح من السلسلة (دينار)، والجزء العشري (فلس) من مبلغ مالي عددي مُدخل من المستخدم.

الخوارزمية

1. بداية البرنامج.
2. استقبال من المستخدم المبلغ المالي مثال: (500.750).
3. تحديد موضع الفاصلة العشرية للسلسلة النصية المُدخلة.
4. تحديد المبلغ بالدينار (الجزء قبل الفاصلة).
5. تحديد المبلغ بالفلس (الجزء بعد الفاصلة).
6. طباعة المبلغ بالتنسيق التالي: 500 KD and 750 fils.
7. نهاية البرنامج.

المطلوب:

خريطة التدفق

- ارسم خريطة التدفق المناسبة.

البرنامج

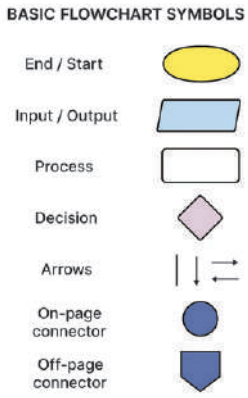
- استدع الملف: amount.py.
- أكمل التعليمة البرمجية اللازمة ليعمل البرنامج بشكل صحيح.

استكمل التعليمات البرمجية

```
1  var_amount = input('Enter amount e.g (500.750) : ')
2  var_position = .....
3  var_kd = .....
4  var_fils = .....
5  print(f '{var_kd} KD And {var_fils} fils')
```

- اختبر البرنامج، وتأكد خلوه من الأخطاء.

خريطة التدفق



ورقة عمل (10): **التاريخ:**

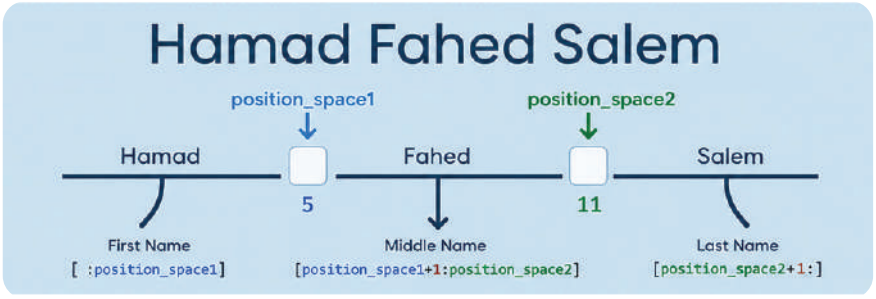
تقطيع السلسلة النصية

مشكلة البرنامج

تقسيم الاسم الكامل full name (باللغة الإنجليزية) للحصول على: الاسم الأول، اللاسم الأوسط، والاسم الأخير.

الخوارزمية

1. بداية البرنامج.
2. استقبال الاسم الكامل (ثلاثي) باللغة الإنجليزية من المُستخدم، وحفظه في متغير full_name.
3. حساب طول السلسلة النصية للاسم الكامل وطباعته باستخدام دالة ()len.
4. استخدام دالة ()find لإيجاد موضع (Position) المسافة الأولى بين الاسم الأول والثاني، وحفظه في متغير position_space1.
5. استخدام دالة ()find لإيجاد موضع (Position) المسافة الثانية بين الاسم الثاني والثالث، وحفظه في متغير position_space2.
6. استخدام الفهرسة والتقطيع للحصول على الاسم الأول، والثاني، والثالث وحفظهم في متغيرات.
7. استخدم الدالتين ()print و ()capitalize لطباعة الاسم الأول والاسم الثاني والاسم الثالث، مع تحويل أول حرف فقط من كل جزء من أجزاء الاسم الكامل إلى حرف كبير (Capital Letter).
8. نهاية البرنامج.



المطلوب:

خريطة التدفق

- ارسم خريطة التدفق المناسبة.

البرنامج

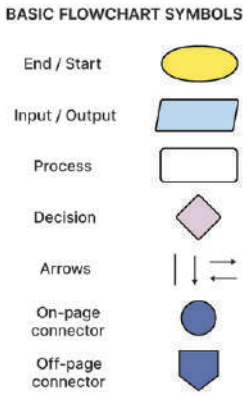
- استدع الملف full_name.py.
- أكمل التعليمات البرمجية اللازمة ليعمل البرنامج بشكل صحيح.

استكمل التعليمات البرمجية

```
1   full_name = input('Enter your full name (Three Parts): ')
2   print('Characters in your name: ', ..... (full_name))
3   position_space1 = full_name.find(' ')
4   position_space2 = full_name.find(" ", position_space1 + 1)
5   first_name = full_name[ : ..... ]
6   second_name = full_name[position_space1 + 1:position_space2]
7   third_name = full_name[ ..... : ]
8   print(first_name.capitalize())
9   print(second_name..... )
10  print(third_name..... )
```

- اختبر البرنامج، وتأكد خلوه من الأخطاء.

خريطة التدفق



برمجة Python

الشروط Conditions

نواتج التعلم

- توضيح مفهوم الشروط ودورها في اتخاذ القرارات داخل البرامج.
- التمييز بين عمليات المقارنة (مثل: == ، > ، <)، والعمليات المنطقية (and، or، not) وتحديد استخداماتها.
- توظيف التعليمات الشرطية if، else، و elif في كتابة برامج تتخذ قرارات مختلفة حسب المدخلات.
- تصميم برامج تعتمد على شروط متعددة لتنفيذ أوامر مختلفة.
- استخدام الشروط المنطقية المتداخلة في مواقف تتطلب أكثر من شرط.
- التحقق من صحة الشروط، وتنفيذ التصحيح اللازم عند ظهور نتائج غير متوقعة.
- تقدير أهمية التفكير المنطقي في تصميم البرامج واتخاذ القرارات البرمجية الصحيحة.

يُمَثّل رمز الاستجابة السريعة QR رابطاً
لملفات أوراق العمل، ومصادر التعلم

الشروط Conditions

تعبيرات منطقية تُستخدم لاختبار حالة معينة داخل البرنامج، وتُرجع قيمة من نوع Boolean تتمثل في True أو False، وفق عوامل المقارنة Comparison Operators، والعوامل المنطقية Logical Operators، والتعليمات الشرطية Conditional Statements.

عوامل المقارنة Comparison Operators

تُمثل العمليات التي تقارن بين عاملين (متغيرات، قيم، ...)، وناتج المقارنة يكون من نوع boolean إما True أو False.

عامل المقارنة	الوصف
==	يساوي
!=	لا يساوي
<	أصغر من
>	أكبر من
<=	أصغر من أو يساوي
>=	أكبر من أو يساوي

جدول (5-1) عوامل المقارنة.

لاحظ:

- عامل التخصيص Assignment = يُستخدم لتخصيص قيمة للمتغير.
- عامل المقارنة Comparison == يُستخدم للمقارنة بين عنصرين.

بعد التعرّف على عوامل المقارنة ووصف كل عامل، الجدول التالي يوضح ناتج تطبيق هذه العمليات على قيمتيْن عدديتيْن للمتغيرين y و x لإرجاع القيم المنطقية False أو True.

x	y	x = = y	x ! = y	x > y	x < y	x > = y	x < = y
1	1	True	False	False	False	True	True
1	2	False	True	False	True	False	True
2	1	False	True	True	False	True	False

جدول (5-2) يُوضح نتائج عمليات المقارنة.

مثال 1:

```
1  prophet_name = input('Who is the last prophet? ')
2  prophet_age = int(input('Enter prophet age: '))
3  print(prophet_name == 'Muhammad')
4  print(prophet_age == 63)
```

Program output

```
Who is the last prophet? Muhammad
Enter prophet age: 40
True
False
```

لاحظ: إذا كانت المقارنة صحيحة يُرجع True، أما إذا كانت المقارنة خاطئة يُرجع False.

العوامل المنطقية Logical Operators

عمليات تُستخدم لربط عوامل المقارنة وتكوين شروط مركبة من أكثر من عامل، بحيث تساعد البرنامج على اتخاذ القرار المناسب، وتكون النتيجة إما True أو False وهي:

1. العملية المنطقية and: تُرجع True إذا كانت كلتا القيمتيْن المنطقيتيْن صحيحتين، وإلا تُرجع False.
2. العملية المنطقية or: تُرجع True إذا كانت أي من القيمتيْن المنطقيتيْن صحيحة، وإلا تُرجع False.
3. العملية المنطقية not: تُرجع False إذا كانت القيمة المنطقية صحيحة، وإلا تُرجع True.

مثال 2:

استخدام العمليات المنطقية للتحقق من درجات الطالب واتخاذ القرار المناسب بناءً على شروط محددة.

```
1  var_math = 85
2  var_science = 70
3  print(var_math >= 50 and var_science >= 50)
4  print(var_math >= 90 or var_science >= 90)
5  print(not var_math < 50)
```

Program output

```
True
False
True
```

بعد التعرّف على العمليات المنطقية ووظيفة كل عملية، يوضح الجدول التالي ناتج تطبيق العمليات المنطقية عمليًا على القيم المنطقية للمتغيرين x و y لإرجاع القيم True أو False.

x	y	x and y	x or y	not x
True	True	True	True	False
True	False	False	True	
False	True	False	True	True
False	False	False	False	

جدول (5-3) ناتج العمليات المنطقية.

التعليمات الشرطية Conditional Statements

بُنية تحكم برمجية تعتمد على نتائج شرط منطقي لاتخاذ القرارات وتحديد مسار تنفيذ البرنامج، ويعتمد بناء التعليمة الشرطية على البنود التالية:

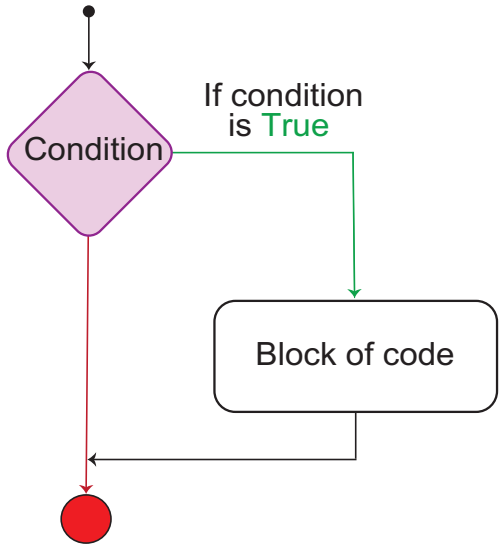
- الكتلة البرمجية Block of Code، وهي تتكوّن من سطر أو أكثر من التعليمات البرمجية التي تشترك في نفس المسافة البادئة Indentation.
  - تبدأ الكتلة البرمجية بعد علامة النقطتين الرأسيتين (:) Colon التي تُوضع في نهاية سطر التعليمة الشرطية.
  - تنتهي الكتلة البرمجية عند انتهاء المسافة البادئة Indentation.
- المسافة البادئة Indentation تُحدد الأوامر التابعة للشرط والتي تُنفذ كوحدة واحدة عند تحقق الشرط.
  - يجب الالتزام بالمسافات البادئة بصورة صحيحة لتجنب الأخطاء البرمجية.

الشرط البسيط Simple if statement

تعليمة شرطية تُستخدم لتنفيذ مجموعة أوامر عند تحقق شرط واحد فقط.

صيغة كتابة الشرط البسيط Simple Conditional Syntax :

```
if condition:
    Block of Code
```



شكل (5-1) خريطة تدفق صيغة if.

مثال 3 :

برنامج يُظهر تقدير الطالب Excellent عند حُصوله على درجة الامتياز في أحد الاختبارات.

```
1    student_score = float(input('Enter student score: '))
2    if student_score >= 90 :
3        print('Excellent')
```

Program output

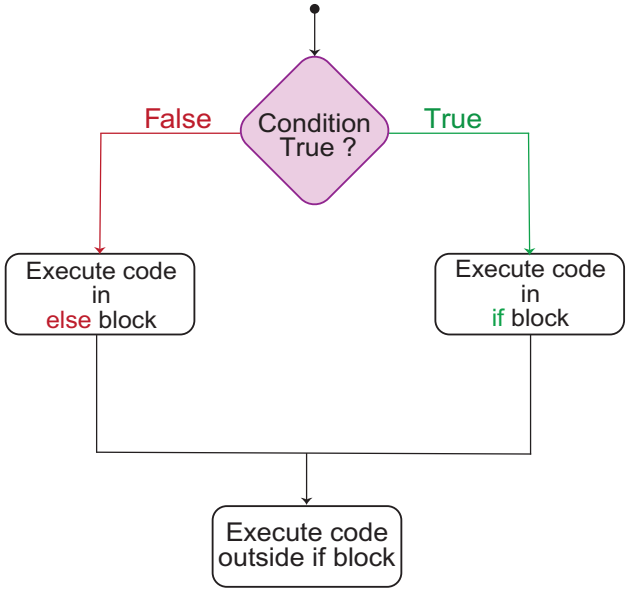
```
Enter student score: 60.5
```

الشرط البديل if-else statement

تعليمة شرطية تُستخدم لتنفيذ كتلة برمجية عند تحقق الشرط، وتنفيذ كتلة بديلة عند عدم تحققه.

صيغة كتابة Alternative Conditional Syntax:

```
if condition:
    Block of Code
else:
    Block of Code
```



شكل (5-2) خريطة تدفق صيغة if else.

مثال 4:

برنامج يُظهر رسالة (You got an excellent grade) إذا درجة الطالب أكبر من أو يساوي العدد (تسعين)، ورسالة (Work hard to get an excellent grade) إذا درجة الطالب أصغر من العدد (تسعين).

```
1    score = float(input('Enter exam score' ))
2    if score >= 90:
3        print('You got an excellent grade')
4    else:
5        print('Work hard to get an excellent grade')
```

Program output

```
Enter exam score 60
Work hard to get an excellent grade
```

مثال 5:

برنامج يُظهر رسالة محددة في حال تحقق الشرط، وإظهار رسالة أُخرى في حال عدم تحققه.

```
user_name = input('Enter user name: ')
user_password = input('Enter password: ')
if user_name == 'admin' and user_password == '123@q8':
    print('Login successful. Welcome admin!')
else:
    print('Invalid username or password.')
```

Program output

```
Enter user name: admin
Enter password: 123@q8
Login successful. Welcome admin!
```

الشرط المُتداخل Nested if statement

تعليمة شرطية تحتوي على تعليمة شرطية أخرى داخلها، وتُستخدم لفحص أكثر من شرط داخل البرنامج.

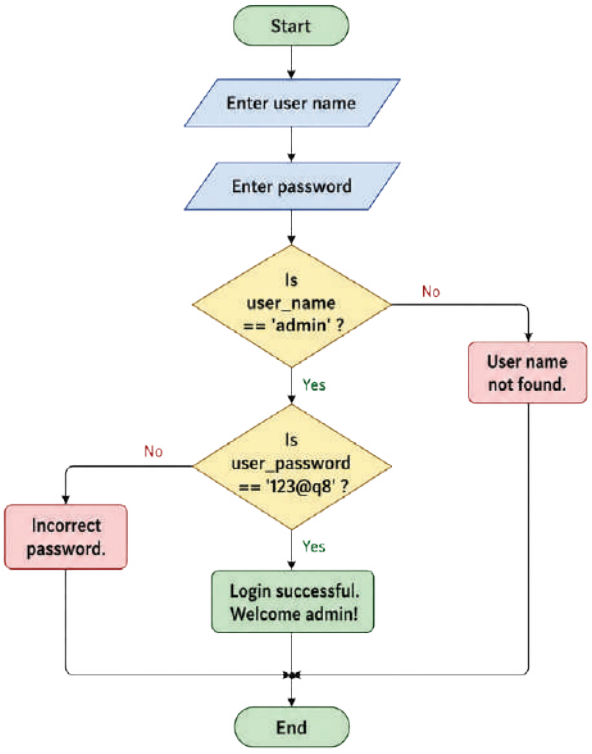
مثال 6:

التحقق من صحة اسم المستخدم أولاً، ثم فحص كلمة المرور باستخدام Nested if، وبعد ذلك يعرض رسالة توضح نجاح عملية تسجيل الدخول أو سبب فشلها.

```
user_name = input('Enter user name: ')
user_password = input('Enter password: ')
if user_name == 'admin':
    if user_password == '123@q8':
        print('Login successful. Welcome admin!')
    else:
        print('Incorrect password.')
else:
        print('User name not found.')
```

Program output

```
Enter user name: admin
Enter password: 123@q8
Login successful. Welcome admin!
```



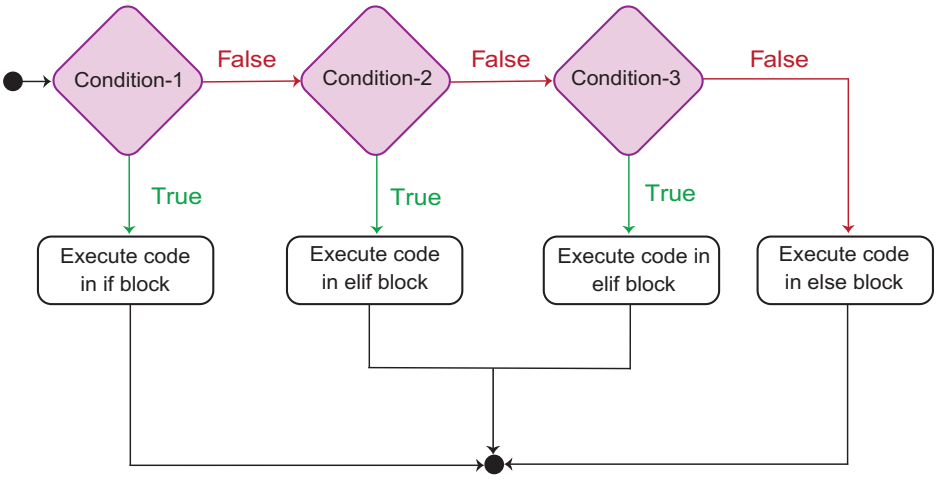
شكل (5-3) يُوضح خريطة برنامج التحقق من اسم المستخدم وكلمة المرور.

الشرط المتسلسل if-elif-else statement

تعليمة شرطية تُستخدم لفحص عدة شروط مترابطة ومتتابعة، بحيث يُنفّذ أول شرط يتحقق ويتم تجاهل بقية الشروط.

صيغة كتابة Chained Conditional Syntax:

```
if condition1:
    Block of code
elif condition2:
    Block of Code
elif condition3:
    Block of Code
...
else:
    Block of Code
```



شكل (5-4) خريطة تدفق صيغة if elif else.

ترتيب الشروط في الشرط المتسلسل:

يُراعى عند كتابة الشرط المتسلسل ترتيب الشروط بصورة منطقية ومناسبة، لأن البرنامج يفحص الشروط من الأعلى إلى الأسفل، وعند تحقق أول شرط يتم تنفيذ التعليمات المرتبطة به وتجاهل بقية الشروط.

في التعليمات البرمجية التالية:

- درجة الطالب 95.
- وضع الشرط student_score >= 50 قبل الشرط student_score >= 90.
- تنفيذ الشرط الأول وطباعة النتيجة Acceptable grade بدلاً من Excellent grade.

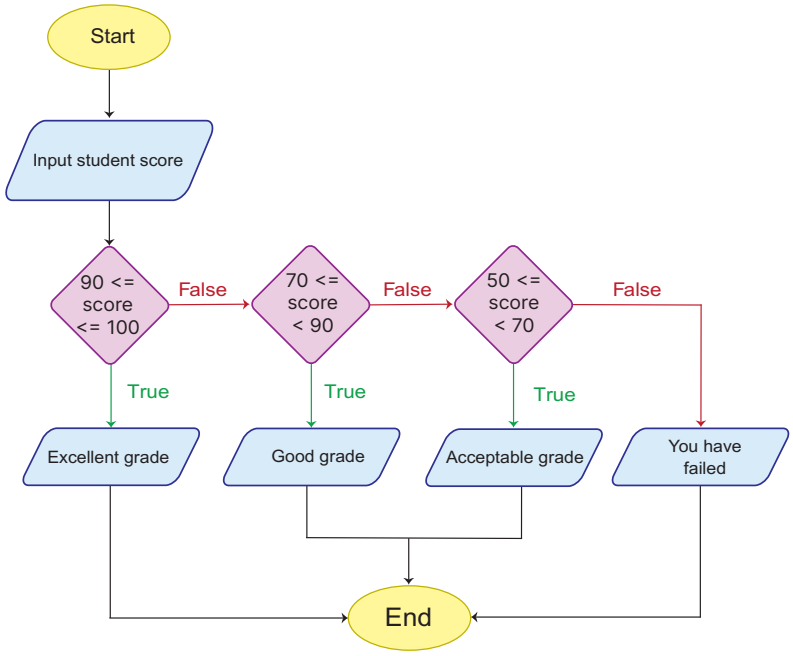
```
1    student_score = 95
2    if student_score >= 50:
3        print('Acceptable grade')
4    elif student_score >= 90:
5        print('Excellent grade')
6    ...
7
```

مثال 7:

برنامج يظهر تقدير الطالب في أحد الاختبارات بناءً على الدرجة المدخلة.

```
1    student_score = float(input('Enter student score: '))
2    if student_score >= 90 and student_score <= 100:
3        print('Excellent grade')
4    elif student_score >= 70 and student_score < 90:
5        print('Good grade')
6    elif student_score >= 50 and student_score < 70:
7        print('Acceptable grade')
8    else:
9        print('You have failed')
```

إذا تحقق الشرط الأول سينفذ البرنامج التعليمات الخاصة به ويتخطى اختبار باقي الشروط، وإذا لم يتحقق سيختبر الشرط الثاني، فإذا تحقق سينفذ التعليمات الخاصة به، وهكذا لبقية الشروط.



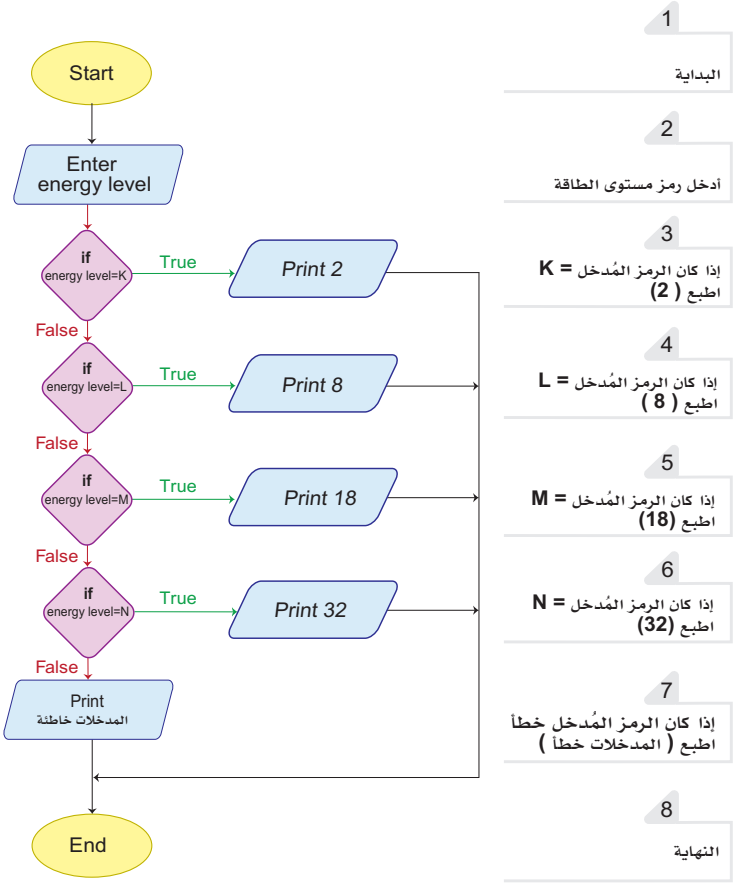
شكل (5-5) خريطة تدفق برنامج تقدير الطالب.

مثال 8:

من خلال دراستك لمادة الكيمياء، لمعرفة العدد الأقصى من الإلكترونات التي توجد في كل مستوى طاقة في الذرّة الجدول التالي:

رقم مستوى الطاقة	الأول	الثاني	الثالث	الرابع
الرمز	K	L	M	N
أقصى عدد من الإلكترونات	2	8	18	32

جدول (5-4) العدد الأقصى من الإلكترونات التي توجد في كل مستوى طاقة في الذرة.



شكل (5-6) خريطة تدفق برنامج التعرف على العدد الأقصى للإلكترونات.

```
energy_level = input('Enter energy level code: ').upper()
if energy_level == 'K':
    print('Energy level K -> ',2)
elif energy_level == 'L':
    print('Energy level L -> ',8)
elif energy_level == 'M':
    print('Energy level M -> ',18)
elif energy_level == 'N':
    print('Energy level N -> ',32)
else:
    print('Invalid input.')
```

ملاحظة: تم استخدام دالة upper() لتحويل السلسلة النصية المُدخلة إلى أحرف كبيرة.

أوراق العمل

ورقة عمل (11): **التاريخ:**

برنامج حساب الخصومات على المنتجات

مشكلة البرنامج

تصميم برنامج يحسب التكلفة النهائية لشراء منتجات، وفقًا لقائمة الخصومات المُعلنة.

الخوارزمية

1. بداية البرنامج.
2. استقبال من المستخدم السعر الإجمالي للمنتجات.
3. التحقق من قيمة السعر الإجمالي:
   - إذا كان سعر المُنتج أكبر من 1000:
     - احسب الخصم بنسبة 25%.
   - إذا كان السعر الإجمالي يتراوح ما بين 500 و 1000:
     - احسب الخصم بنسبة 15%.
   - إذا كان السعر الإجمالي أقل من 500:
     - لا يوجد خصم.
4. حساب السعر النهائي بعد تطبيق الخصم.
5. طباعة السعر النهائي.
6. نهاية البرنامج.

المطلوب:

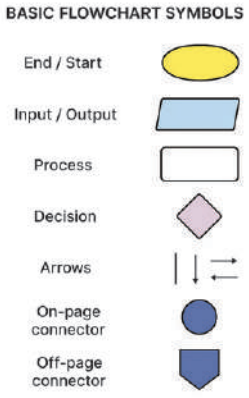
خريطة التدفق

- ارسم خريطة التدفق المناسبة.

البرنامج

- أنشئ الملف calculate_flnal_price.py.
- اكتب التعليمات البرمجية اللازمة.
- اختبر البرنامج، وتأكد من خلوه الأخطاء.

خريطة التدفق



ورقة عمل (12): **التاريخ:**

برنامج تصميم آلة حاسبة

مشكلة البرنامج

استخدام المعاملات الحسابية (+ ، - ، * ، /) في تنفيذ العمليات الحسابية (الجمع، الطرح، الضرب، القسمة).

الخوارزمية

1. بداية البرنامج.
2. استقبال من المستخدم العدد الأول، ثم العدد الثاني.
3. استقبال من المستخدم المعامل الحسابي المطلوب.
4. استخدام الشروط Conditions:

- إذا كان المعامل المُدخل +، يجمع العددين الأول والثاني.
- إذا كان المعامل المُدخل -، يطرح العدد الثاني من العدد الأول.
- إذا كان المعامل المُدخل *، يضرب العددين الأول والثاني.
- إذا كان المعامل المُدخل /، يقسم العدد الأول على العدد الثاني.

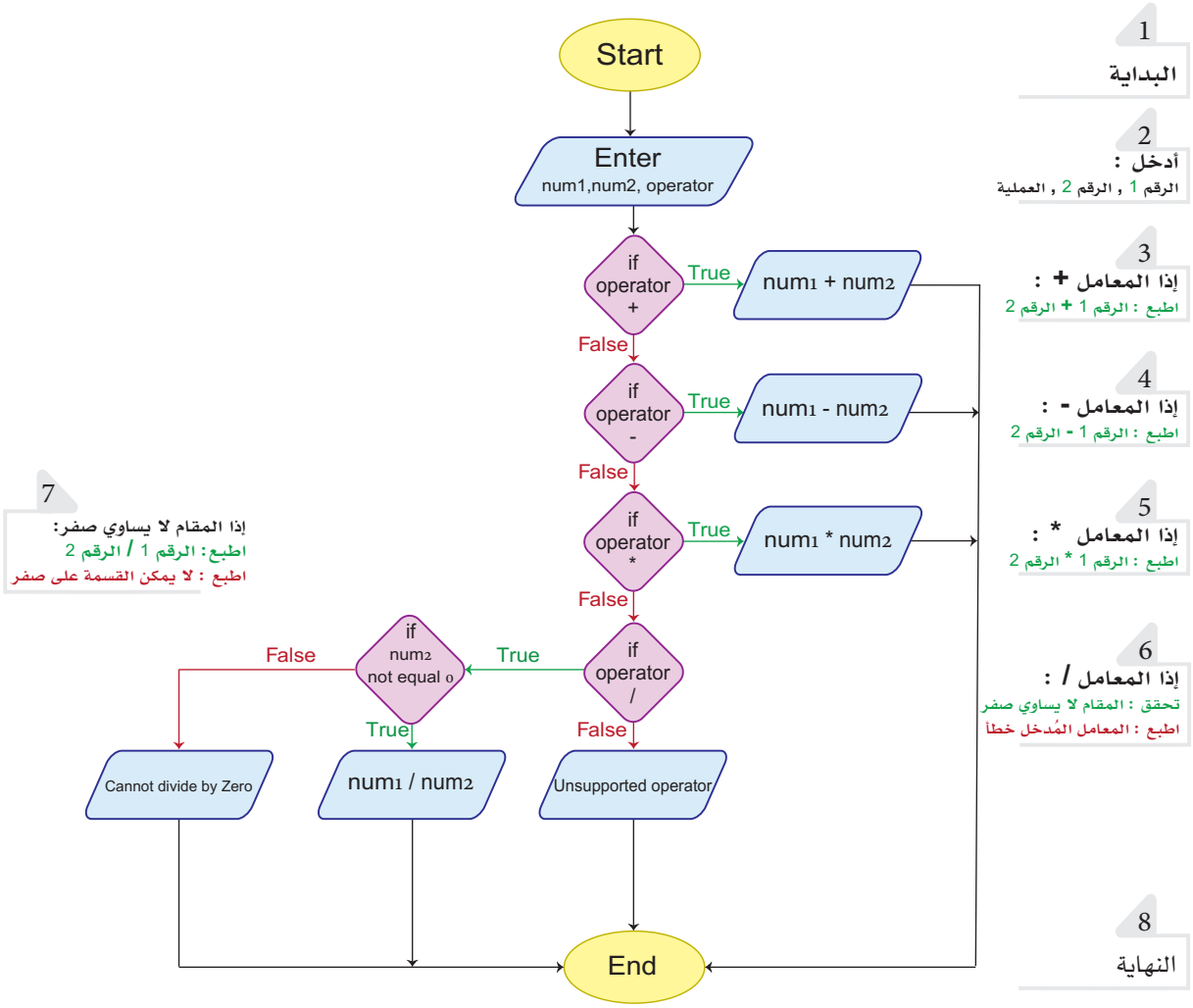
(التأكد من العدد الثاني لا يساوي صفر، وطباعة رسالة It cannot be divided by zero).

5. طباعة ناتج العملية الحسابية.
6. نهاية البرنامج.

المطلوب:

خريطة التدفق

- ادرس خريطة التدفق التالية، والتي صممت لعمل آلة حاسبة بسيطة للعمليات الأساسية.



شكل (5-7) تُوضح خريطة تدفق برنامج آلة حاسبة بسيطة للعمليات الحسابية.

البرنامج

- افتح الملف Calculator.py، نفذ التعديلات المناسبة ليعمل البرنامج بشكل صحيح.

استكمل التعليمات البرمجية

```
1   # Conditions and math. operator
2   num1 = float(input('Enter First Number: '))
3   num2 = float(input('Enter Second Number: '))
4   operator = input('Enter Math. Operator +,-,*,/: ')
5   if operator == '+':
6       print(num1 + num2)
7   ..... # Check the math. operator = -
8       print(num1 - num2)
9   ..... # Check the math. operator = *
10      print(num1 * num2)
11  elif operator == '/':
12      if num2 != 0:
13      .....
14      else:
15      .....
16  else:
17      print('Unsupported operator')
```

- اختبر البرنامج، وتأكد من خلوه الأخطاء.

برمجة Python

التكرار Loops

نواتج التعلم

- توضيح مفهوم التكرار وأهميته في تقليل التكرار اليدوي للأوامر داخل البرنامج.
- استخدام الحلقة التكرارية while لتنفيذ أوامر طالما الشروط متحققة.
- تطبيق الحلقة التكرارية for للتكرار على عناصر مثل السلاسل النصية والنطاقات الرقمية.
- تنفيذ تكرارات متداخلة Nested Loops لحل مشكلات معقدة تتطلب أكثر من مستوى من التكرار.
- استخدام التعليمة البرمجية break للخروج من الحلقة التكرارية عند تحقق شرط معين.
- توظيف التعليمة continue لتخطي تكرار معين دون إنهاء الحلقة بالكامل.
- شرح البنية العامة واستخدامها while...else لفهم كيفية تنفيذ التعليمات بعد انتهاء التكرار.

يُمَثّل رمز الاستجابة السريعة QR رابطاً
لملفات أوراق العمل، ومصادر التعلم

التكرار Loops

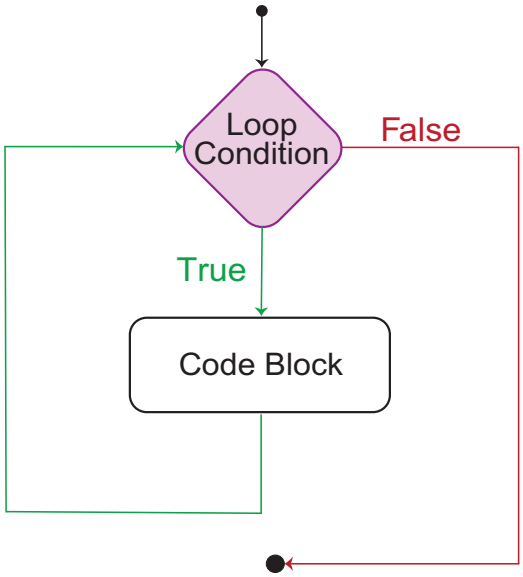
عملية تُستخدم لتكرار تنفيذ مجموعة من التعليمات البرمجية وفق عدد محدد من المرات أو عند تحقّق شرط معيّن، وهناك نوعان رئيسيان من التكرار وهما الحلقة التكرارية while، والحلقة التكرارية for.

1. الحلقة التكرارية while

تُستخدم لتكرار تنفيذ كتلة برمجية طالما تَحَقّقَ شرط مُعين.

صيغة كتابة While Syntax:

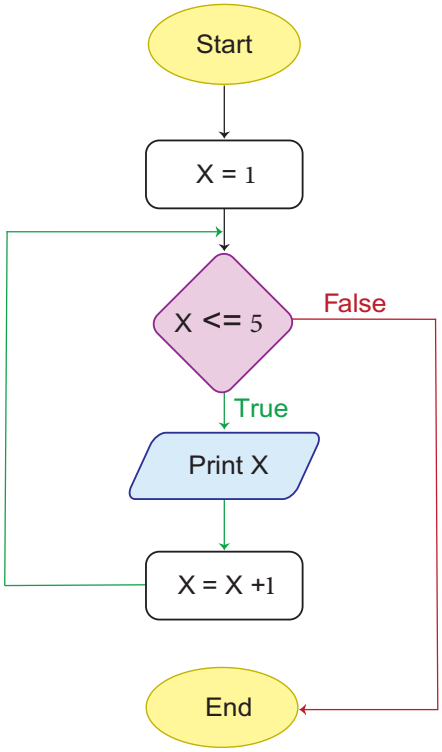
```
while condition:
    code block to be executed until the condition becomes False.
```



شكل (6-1) يُوضح خريطة تدفق صيغة كتابة while.

مثال 1:

استخدام الحلقة التكرارية while لطباعة الأعداد من 1 إلى 5:



شكل (2-6) يُوضح خريطة تدفق طباعة الأعداد.

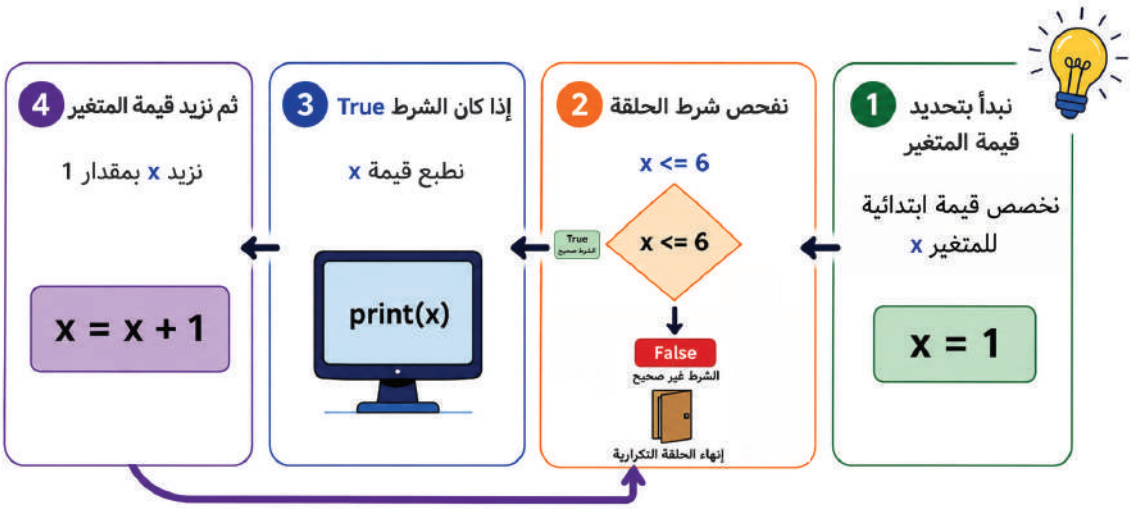
```
1    X = 1
2    while X <= 5:
3      print(x)
4      X = X + 1
```

Program output

```
1
2
3
4
5
```

ملاحظات:

- يُمكن استخدام تعليمة x = x + 1 أو تعليمة x += 1 لزيادة قيمة المتغير بمقدار واحد.
- عند استخدام الحلقة التكرارية while، غالبًا ما يتضمن الشرط متغيرًا يُعطى قيمة ابتدائية Initial Value.
- تستمر الحلقة في التنفيذ طالما أنّ الشرط صحيح True.
- تنتهي الحلقة التكرارية عند عدم تحقق الشرط False بعد تعديل قيمة متغير الشرط أثناء التكرار.



شكل (3-6) يُوضح مخطط تنفيذ الحلقة التكرارية while.

تكرارات While المتداخلة Nested While Loops

حلقات تكرارية تُكتب داخل حلقات تكرارية أخرى، بحيث تُنفّذ الحلقة الداخلية بالكامل في كل مرة تُنفّذ فيها الحلقة الخارجية.

مثال 2:

يستخدم البرنامج التكرارات المتداخلة لإدخال عدد الفصول (الحلقة الخارجية)، وعدد الطلاب في كل فصل (الحلقة الداخلية).

```
1   total_classrooms = int(input('Enter number of classrooms: '))
2   current_classroom = 1                                          الحلقة الخارجية
3   while current_classroom <= total_classrooms:
4       print(f'\nClassroom {current_classroom}')
5       total_students = int(input('Enter number of students in each classroom: '))
6       current_student = 1                                        الحلقة الداخلية
7       while current_student <= total_students:
8           print(f'Student {current_student}')
9           current_student+= 1
10      current_classroom += 1
11  print('\nProgram completed successfully.')
```

Program output

```
Enter number of classrooms: 2

Classroom  1
Enter number of students in each classroom: 3
Student  1
Student  2
Student  3

Classroom  2
Enter number of students in each classroom: 2
Student  1
Student  2

Program completed successfully.
```

ملاحظات:

- كل مستوى جديد من التداخل Nested Loop يحتاج إلى مسافة بادئة إضافية مقارنة بالمستوى السابق.
- تُستخدم \n [1] للانتقال إلى سطر جديد new line عند تنفيذ البرنامج.

1 تسلسلات الهروب Escape Sequences هي مجموعة من الرموز الخاصة التي تبدأ بالشرطة المائلة العكسية \، وتُستخدم داخل السلاسل النصية لتمثيل أوامر أو أحرف خاصة لا يمكن كتابتها مباشرة.

استخدام التعليمة البرمجية break و continue مع الحلقة التكرارية while

تُستخدم للتحكم في عملية التكرار داخل الحلقات التكرارية، ممّا يساعد على تنفيذ الشروط المطلوبة بكفاءة أكبر، وجَعل البرامج أكثر مرونة وسهولة في التحكم.

التعليمة البرمجية break

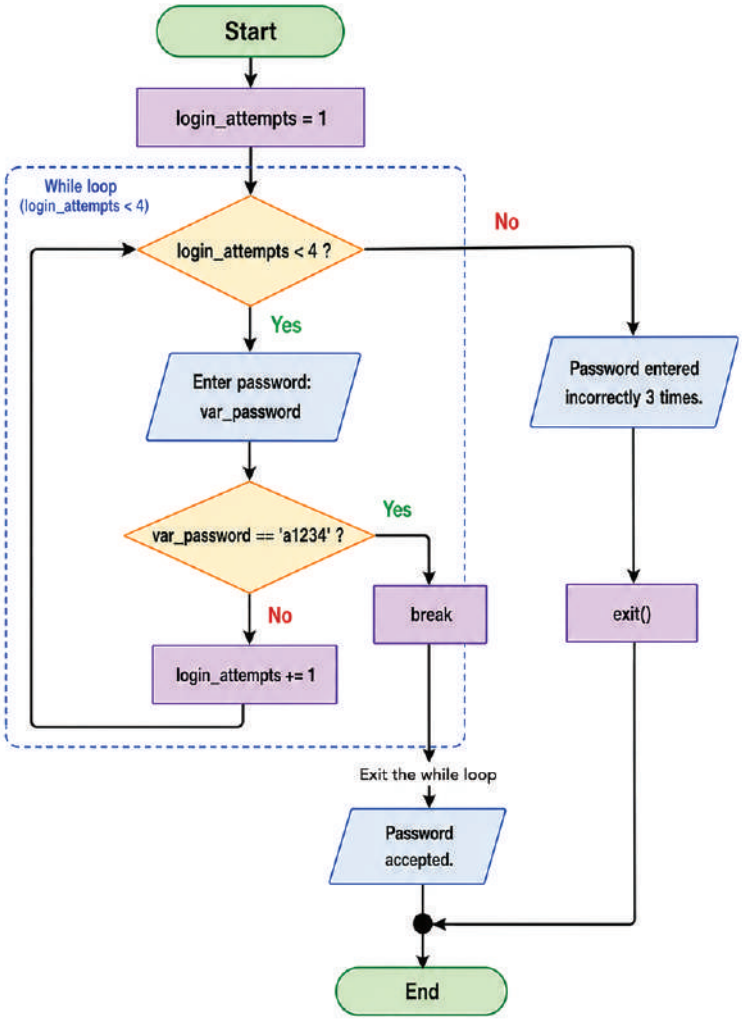
تُستخدم لإيقاف الحلقة التكرارية، والخروج منها مباشرة عند تحقق شرط معين.

مثال 3:

برنامج للتحقق من كلمة المرور يسمح بثلاث محاولات لتسجيل الدخول، ويستخدم تعليمة break للخروج من الحلقة التكرارية عند إدخال كلمة المرور الصحيحة.

خوارزمية البرنامج:

1. بداية البرنامج.
2. تعيين قيمة ابتدائية للمتغير login_attempts = 1.
3. بداية الحلقة التكرارية: تستمر الحلقة التكرارية مادامت قيمة المتغير login_attempts أقل من 4:
   - استقبال كلمة المرور من المستخدم.
   - إذا كانت كلمة المرور صحيحة تساوي a1234:
     - الخروج من الحلقة التكرارية (الانتقال إلى 5).
   - وإلا (إذا كانت كلمة المرور غير صحيحة):
     - زيادة قيمة login_attempts بمقدار 1.
4. إذا انتهت الحلقة التكرارية دون إدخال كلمة المرور الصحيحة:
   - عرض الرسالة Password entered incorrectly 3 times.
   - الخروج من البرنامج (الانتقال إلى 6).
5. عرض الرسالة Password accepted.
6. نهاية البرنامج.



شكل (4-6) يُوضح خريطة تدفق برنامج التحقق من كلمة المرور.

```
1   login_attempts = 1
2   while login_attempts < 4:
3       var_password = input('Enter password: ')
4       if var_password == 'a1234':
5           break
6       login_attempts += 1
7   else:
8       print('Password entered incorrectly 3 times.')
9       exit()
10  print('Password accepted.')
```

Program output

```
Enter password: 123
Enter password: a123
Enter password: a234
Password entered incorrectly 3 times.
```

ملاحظات:

- else: تُنفّذ بعد انتهاء الحلقة التكرارية إذا لم يتم إدخال كلمة المرور الصحيحة خلال جميع المحاولات.
- exit(): تُستخدم لإنهاء البرنامج.

مثال 4:

تستخدم while True لتكرار تنفيذ التعليمات البرمجية إلى ما لا نهاية، على النحو التالي:

```
1  while True:
2      var_password = input('Enter password: ')
3      if var_password == 'a1234':
4          break
5  print('Password accepted.')
```

Program output

```
Enter password: 321
Enter password: b123
Enter password: a1234
Password accepted.
```

ملاحظات:

- استخدمت while True الحلقة التكرارية اللانهائية infinity Loop للاستمرار في إدخال كلمة المرور دون توقف، حيث تستمر في التنفيذ استمراراً متواصلاً لأن الشرط (True) يكون صحيحًا دائمًا.
- استخدمت تعليمة break للخروج من الحلقة التكرارية عند إدخال كلمة مرور صحيحة، حيث تُنهي تنفيذ الحلقة التكرارية الحالية، وتنتقل مباشرة إلى أول تعليمة بعد الحلقة.

التعليمة البرمجية continue

تُستخدم لتجاوز التكرار الحالي والانتقال مباشرة إلى التكرار التالي، حيث يتم تجاهل الأوامر المتبقية في الدورة الحالية، وتبدأ دورة تكرارية جديدة.

مثال 5:

طباعة الأعداد الزوجية باستخدام الحلقة التكرارية while.

```
1    x = 0
2    while x < 10:
3        x += 1
4        if x % 2 == 1:
5            continue
6        print(x)
```

Program output

```
2
4
6
8
10
```

2. الحلقة التكرارية for

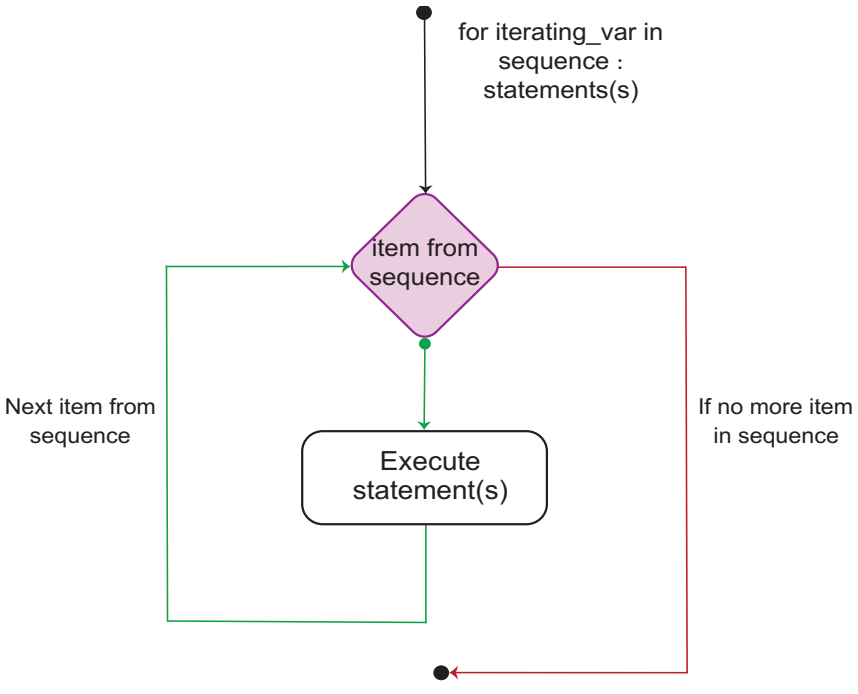
تُستخدم لتكرار تنفيذ كتلة من التعليمات البرمجية لعدد محدد من المرات.

صيغة كتابة For Syntax:

```
for variable in iterable[2]:
        code block to be executed for each element.
```

---

2 قابل للتكرار iterable: كائن يمكن الانتقال عبر عناصره انتقالاً متعاقباً، مثل السلاسل النصية Strings.



شكل (6-5) يُوضح خريطة تدفق صيغة الحلقة التكرارية for.

مثال 6:

استخدام دالة input لإدخال الاسم ثم طباعة كل حرف من السلسلة النصية للمتغير في سطر مستقل.

```
1    var_name = input('Enter your country name: ')
2    for ch in var_name :
3        print(ch)
```

Program output

```
Enter your country name: Kuwait
K
u
w
a
i
t
```

الدالة range()

تُستخدم لتوليد تسلسل من الأعداد الصحيحة ضمن مدى محدد، لتحديد عدد مرات التكرار.

الصيغة	الوصف	عناصر السلسلة	مثال
range(start,stop,step)	تُستخدم لإنشاء سلسلة من الأعداد الصحيحة تبدأ من start وحتى العدد السابق stop، مع تغير بمقدار قيمة step في كل مرة	1, 3, 5, 7, 9 5, 4, 3, 2	range(1,11,2) range(5,1,-1)
range(start,stop)	تُستخدم لإنشاء سلسلة من الأعداد الصحيحة تبدأ من start وحتى العدد السابق stop وتزيد بمقدار واحد.	3, 4, 5, 6 -4, -3, -2	range(3,7) range(-4,-1)
range(stop)	تُستخدم لإنشاء سلسلة من الأعداد الصحيحة تبدأ من صفر وحتى العدد السابق stop وتزيد بمقدار واحد	0, 1, 2, 3, 4	range(5)

جدول (6-1) استخدام الدالة range() لإنشاء سلسلة من الأعداد الصحيحة.

مثال 7:

إدخال أسعار ثلاثة عناصر من المُستخدم باستخدام الحلقة التكرارية for، وإضافة إلى المتغير total_price أثناء كل تكرار، ثم طباعة المجموع الكلي للأسعار

```
1  total_price = 0
2  for x in range(3):
3      item_price = float(input('Enter item price: '))
4      total_price += item_price
5  print(total_price)
```

Program output

```
Enter item price: 5.5
Enter item price: 3
Enter item price: 8
16.5
```

تكرارات for المتداخلة Nested For Loops

حلقات تكرارية تُكتب داخل حلقات تكرارية أخرى، بحيث تُنفّذ الحلقة الداخلية بالكامل في كل مرة تُنفّذ فيها الحلقة الخارجية.

مثال 8:

يَعرض البرنامج بيانات مقاعد الطائرة باستخدام التكرارات المتداخلة، حيث تمثل الحلقة الخارجية أرقام الصفوف وتمثّل الحلقة الداخلية أحرف المقاعد داخل كل صف.

```
1  airplane_seats = 'ABCD'
2  for airplane_row in range(1, 4):
3      print(f'\nRow: {airplane_row}')
4
5      for airplane_seat in airplane_seats:
6          print('Seat:' ,airplane_seat, airplane_row  )
```

الحلقة الخارجية

الحلقة الداخلية

Program output

```
Row: 1
Seat: A 1
Seat: B 1
Seat: C 1
Seat: D 1

Row: 2
Seat: A 2
Seat: B 2
Seat: C 2
Seat: D 2

Row: 3
Seat: A 3
Seat: B 3
Seat: C 3
Seat: D 3
```

ملاحظة:

كل مستوى جديد من التداخل Nested Loop يحتاج إلى مسافة بادئة إضافية مقارنة بالمستوى السابق.

استخدام التعليمة البرمجية break و continue مع الحلقة التكرارية for

تُستخدم التعليمات البرمجية break و continue للتحكم في عملية التكرار.

التعليمة البرمجية break

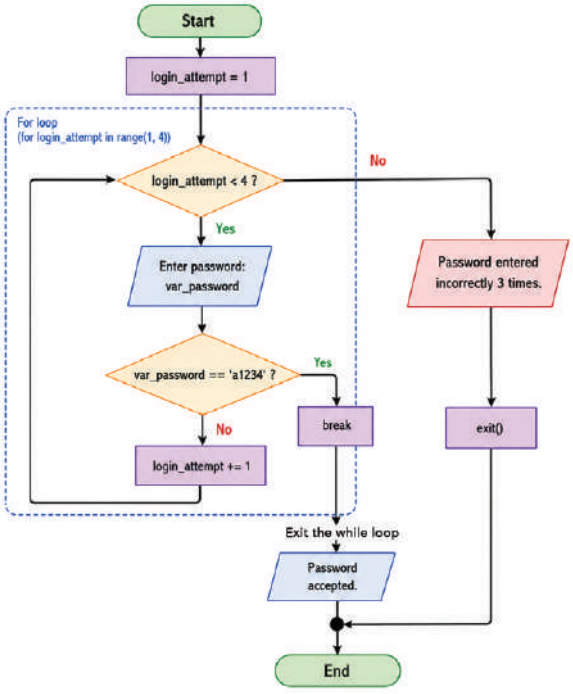
تُستخدم تعليمة break للخروج من الحلقة التكرارية.

مثال 9:

برنامج للتحقق من كلمة المرور يسمح بثلاث محاولات لتسجيل الدخول، ويستخدم تعليمة break للخروج من الحلقة عند إدخال كلمة المرور الصحيحة.

خوارزمية البرنامج:

1. بداية البرنامج.
2. تنفيذ حلقة تكرارية ثلاث مرات:
   - استقبال كلمة المرور من المستخدم.
   - إذا كانت كلمة المرور صحيحة تساوي a1234:
     - الخروج من الحلقة التكرارية (الانتقال إلى 4).
3. إذا انتهت الحلقة التكرارية بعد ثلاث محاولات دون إدخال كلمة المرور الصحيحة:
   - عرض الرسالة Password entered incorrectly 3 times.
   - الخروج من البرنامج (الانتقال إلى 5).
4. عرض الرسالة Password accepted.
5. نهاية البرنامج.



شكل (6-6) يُوضح خريطة تدفق برنامج التحقق من كلمة المرور.

```
1  for x in range(1,4):
2      var_password = input('Enter password: ')
3      if var_password == 'a1234':
4          break
5  else:
6      print('Password entered incorrectly 3 times.')
7      exit()
8  print('Password accepted.')
```

Program output

Enter password: 123
Enter password: a123
Enter password: a234
Password entered incorrectly 3 times.

Program output

Enter password: 123
Enter password: a1234
Password accepted.

ملاحظات:

- else: تُنفّذ بعد انتهاء الحلقة التكرارية إذا لم تُدخل كلمة المرور الصحيحة خلال جميع المحاولات المتاحة.
- exit(): تُستخدم لإنهاء البرنامج.
- تُستخدم تعليمة break للخروج من الحلقة التكرارية عند إدخال كلمة مرور صحيحة، حيث تُنهي تنفيذ الحلقة التكرارية الحالية وتنتقل مباشرة إلى أول تعليمة بعد الحلقة.

التعليمة البرمجية continue

تُستخدم لتجاوز التكرار الحالي، والانتقال مباشرة إلى التكرار التالي، حيث تتجاهل الأوامر المتبقية في الدورة الحالية، وتبدأ دورة تكرارية جديدة.

مثال 10:

طباعة الأعداد الزوجية باستخدام الحلقة التكرارية for.

```
1  for x in range(1 , 11):
2      if x % 2 == 1:
3          continue
4      print(x)
```

Program output

```
2
4
6
8
10
```

أوراق العمل

ورقة عمل (13): **التاريخ:**

برنامج تخمين عدد

مشكلة البرنامج

إدخال أعداد صحيحة، والتأكد من مدى مطابقتها للعدد المُعرف في البرنامج.

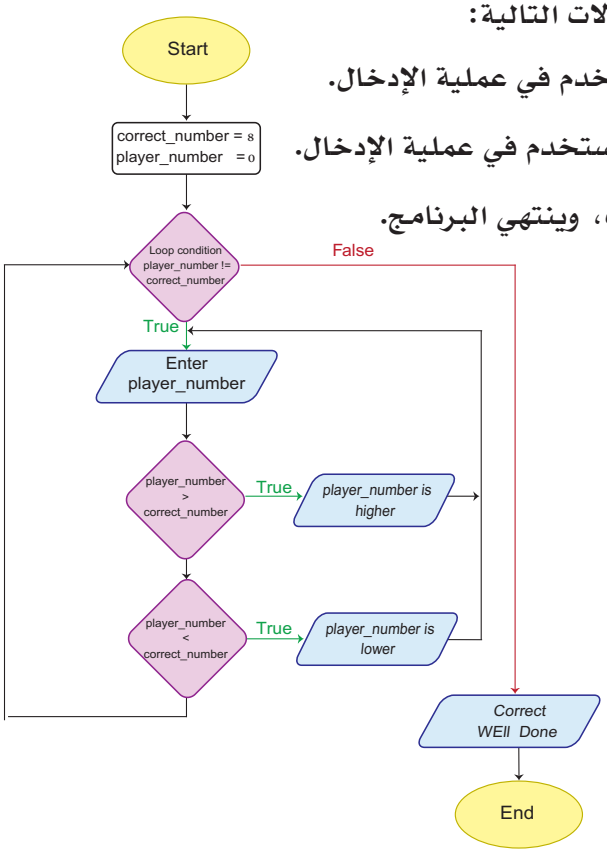
الخوارزمية

1. بداية البرنامج.
2. الإعلان عن متغير يساوي العدد المطلوب تخمينه.
3. استقبال عدد صحيح من المستخدم.
4. مطابقة العدد المُدخل مع العدد المحدّد في الحالات التالية:
   - إذا كان أكبر يطبع رسالة العدد أكبر ويستمر المستخدم في عملية الإدخال.
   - إذا كان أصغر يطبع رسالة العدد أصغر ويستمر المستخدم في عملية الإدخال.
   - طباعة رسالة العدد المُدخل مطابق للعدد المُعرّف، وينتهي البرنامج.
5. نهاية البرنامج.

المطلوب:

خريطة التدفّق

- ادرس خريطة التدفق المقابِلة.



شكل (6-7) خريطة تدفق برنامج تخمين عدد صحيح.

البرنامج

- افتح الملف correct_number.py.

```
correct_number = 8
player_number = 0
..........                    # Looping
    player_number = int(input('Guess! what's the number in my mind: '))
    if player_number > correct_number:
        print('The player number is higher.')
    ..........                # if the number less
    ..........                # show message
print('correct! well done.')
```

- أكمل التعليمات البرمجية اللازمة ليعمل البرنامج باستمرار حتى يتطابق العدد المُدخل مع العدد المُعرف.
- اختبر البرنامج، وتأكد من خلوه من الأخطاء.

إثرائي: ناقش مع معلمك التعليمات البرمجية لتطوير البرنامج السابق لتوليد رقم عشوائي:

```
# لتوليد رقم عشوائي من 1 إلى 30
import random
correct_number = random.randint(1,30)
# --------------------
player_number = 0
while player_number != correct_number:
  player_number = int(input("Enter a number between 1 and 30: "))
  if player_number > 30 or player_number < 1:
    print("Please enter a number between 1 and 30")
    continue
  if player_number < correct_number:
    print("The number is low. Try again.")
  elif player_number > correct_number:
    print("The number is high. Try again.")
  else:
    print("Congratulations! You guessed the correct number.")
```

ورقة عمل (14): التاريخ:

برنامج جدول الضرب

مشكلة البرنامج

برنامج يطبع جدول الضرب لعدد محدد.

الخوارزمية

1. بداية البرنامج.
2. استقبال العدد المطلوب من المستخدم وطباعة جدول الضرب الخاص به.
3. طباعة رسالة (Multiplication table for)، ثُمّ قيمة العدد السابق.
4. إنشاء حلقة تكرارية لطباعة ضرب العدد المُدخل في الأعداد من 1 إلى 12.
   - كتابة التعليمات البرمجية لحساب :

     الناتج = العدد المُدخَل * قيمة متغير التكرار.

   - طباعة جدول الضرب.
5. نهاية البرنامج.

المطلوب:

Program output

```
Please enter an integer number: 3
Multiplication table for  3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
3 x 11 = 33
3 x 12 = 36
```

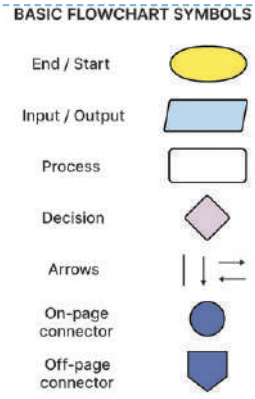
خريطة التدفّق

- ارسم خريطة التدفق المناسبة.

البرنامج

- أنشئ الملف Multiplication_table.py.
- اكتب التعليمات البرمجية اللازمة لطباعة جدول الضرب لعدد محدد، (مثال العدد المحدد 3).
- اختبر البرنامج، وتأكد من عدم وجود أخطاء.

خريطة التدفق



برمجة Python

تصحيح الأخطاء والاستثناءات

Debugging and Exceptions

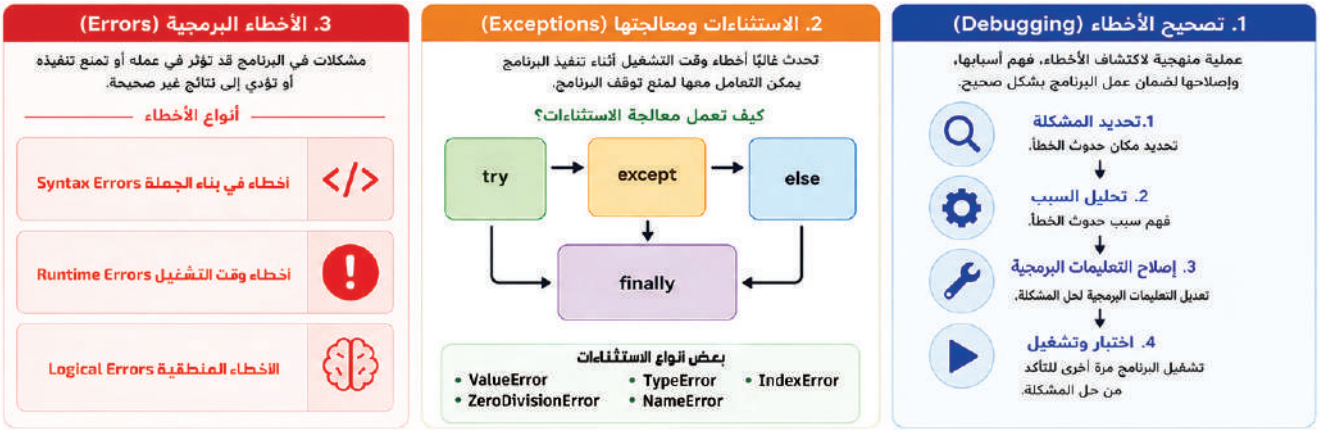
نواتج التعلم

- شرح مفهوم الأخطاء Errors، والاستثناءات Exceptions في البرامج.
- التمييز بين أنواع الأخطاء الشائعة مثل: أخطاء بناء الجملة Syntax Errors، وأخطاء وقت التشغيل Runtime Errors، والأخطاء المنطقية Logical Errors.
- شرح أهمية التعامل مع الاستثناءات لضمان استقرار البرنامج ومنع توقفه المفاجئ.
- استخدام التعليمة try / except لمعالجة الأخطاء المتوقعة وتقديم حلول بديلة أو رسائل توضيحية.
- تطبيق مهارات تصحيح الأخطاء Debugging في تتبع مصدر الخطأ وتحليله ومعالجته بكفاءة.

يُمَثّل رمز الاستجابة السريعة QR رابطاً
لملفات أوراق العمل، ومصادر التعلم

معالجة الأخطاء والاستثناءات Debugging and Exceptions

تُعَدّ معالجة الأخطاء والاستثناءات من المفاهيم الأساسية في البرمجة، حيث تهدف إلى التعامل مع الأخطاء البرمجية Errors التي قد تؤثّر في عمل البرنامج أو تؤدي إلى توقفه. وتُستخدم مهارات تصحيح الأخطاء Debugging واكتشاف الاستثناءات ومعالجتها Exception Handling لتحديد المشكلات والتعامل معها بطريقة منظمة، مما يُسهم في تحسين استقرار البرنامج ورفع جودة تجربة المستخدم User Experience [1] من خلال تقليل الأعطال وتوفير رسائل واضحة تساعد المستخدم على فهم المشكلة، والتعامل معها بفعاليّة.



شكل (7-1) يُوضح أنواع معالجة الأخطاء والاستثناءات.

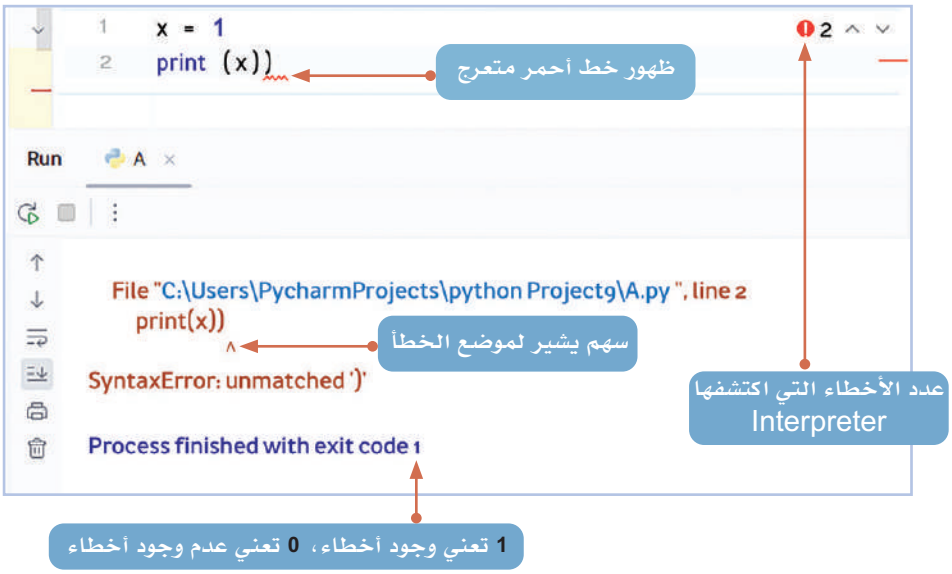
الأخطاء البرمجية Errors

عند التعامل مع التعليمات البرمجية بلغة Python قد نواجه بعض الأخطاء التي تؤثر على سير البرنامج. ولمعالجة هذه الأخطاء يمكن استخدام مجموعة من التقنيات التي تساعد على اكتشافها وتصحيحها لضمان عمل البرنامج بشكل صحيح وتنقسم الأخطاء في Python إلى ثلاثة أنواع رئيسة، وهي:

1 تجربة المستخدم UX هي مدى سهولة وفاعلية المُستخدم ورضاه أثناء التعامل مع البرنامج.

1. أخطاء في بناء الجُملة Syntax Errors

أخطاء تَحدُث بسبب عدم الالتزام بقواعد كتابة التعليمات البرمجية، فلا يستطيع (المفسّر) Interpreter فهمها، ممّا يمنع البرنامج من التنفيذ. وتَظْهر هذه الأخطاء قبل بدء تشغيل البرنامج، ويُوضح الشكل التالي كيفية اكتشاف المفسّر لأخطاء بناء الجملة، حيث يحدد موضع الخطأ ويَعرض رسالة بنوع الخطأ.



شكل (7-2) يوضح اكتشاف خطأ في بناء الجملة Syntax Error، وتحديد موضعه في بيئة البرمجة.

عند حدوث خطأ في بناء الجملة، يَعرض المفسّر Interpreter رسالة تساعد المبرمج على تحديد المشكلة وتصحيحها، ويوضح الجدول التالي بعض هذه الأخطاء ورسائلها الشائعة.

الخطأ	مثال	رسالة الخطأ
نسيان النقطتين (:) Colon بعد الجملة الشرطية.	if x > 0 print(x)	SyntaxError: expected ':'
عدم إغلاق الأقواس.	print ("Hello, Kuwait!"	SyntaxError: '(' was never closed
عدم ترك المسافة البادئة لتحديد الكتلة البرمجية للحلقة التكرارية.	for i in range (5): print (i)	IndentationError: expected an indented block after 'for' statement on line 1
الكلمة المفتاحية while كُتبت كتابة غير صحيحة.	whlie x < 10: print(x)	SyntaxError: invalid syntax

جدول (7-1) يوضح أمثلة على بعض الأخطاء في بناء الجملة.

2. أخطاء وقت التشغيل Runtime Errors

أخطاء تَحدُث أثناء تنفيذ البرنامج، نتيجة تنفيذ عملية غير صحيحة أو التعامل مع بيانات غير مناسبة. تؤدي هذه الأخطاء إلى توقف البرنامج أو مقاطعة سير تنفيذه ما لم يُتعامل معها باستخدام آليات معالجة الاستثناءات، ويوضح الجدول التالي بعض أخطاء وقت التشغيل الشائعة ورسائل الخطأ المرتبطة بها.

الخطأ	مثال	رسالة الخطأ
قسمة عدد على صفر (ZeroDivisionError)	print(5 / 0)	ZeroDivisionError: division by zero
استخدام متغير غير مُعرّف (NameError)	x = 5 print(z)	NameError: name 'z' is not defined
محاولة جمع قيمة نصية مع قيمة عددية، مما يؤدي إلى حدوث خطأ من النوع (TypeError)	print('5' + 10)	TypeError: can only concatenate str (not 'int') to str
تحويل قيمة نوع بيانات غير مناسبة (ValueError)	number = int('abc')	ValueError: invalid literal for int() with base 10: 'abc'

جدول (2-7) يوضّح أمثلة على بعض أخطاء وقت التشغيل Runtime Errors، ورسائل الخطأ المرتبطة بها.

3.الأخطاء المنطقية Logical Errors

الأخطاء التي قد تؤدي إلى نتائج غير متوقعة أو غير صحيحة؛ بمعنى أنّ التعليمات البرمجية تعمل عملاً صحيحًا، لكن نتيجة تنفيذها غير صحيحة، ولا يتعرّف عليها Python Interpreter، و منها :

- استخدام شرط غير صحيح.
- استخدام العمليات الحسابية استخدامًا غير صحيح.

الخطأ	مثال خاطئ	تصحيح الخطأ المقترح
استخدام عملية حسابية غير مناسبة، ممّا يؤدي إلى إنتاج نتائج غير صحيحة رَغْم عمل البرنامج دون ظهور رسائل خطأ.	length = 5 width = 4 rectangle_area = length + width print(rectangle_area) # output 9	length = 5 width = 4 rectangle_area = length * width print(rectangle_area) # output 20
عدم استخدام الأقواس أو أولوية العمليات الحسابية بصورة صحيحة، مما يؤدي إلى نتائج غير صحيحة.	a = 10 b = 10 average = a +(b / 2) print (average) # output 15	a = 10 b = 10 average = (a + b) / 2 print (average) # output 10
استخدام شرط غير مناسب، ممّا يؤدي إلى تنفيذ تعليمات لا تتوافق مع النتيجة المتوقعة وإظهار مخرجات غير صحيحة.	if score < 50: print('Successful') # No output	if score >= 50: print('Successful') # output Successful

جدول (7-3) يوضح أمثلة على بعض الأخطاء المنطقية.

معالجة الاستثناءات Exceptions Handling

أسلوب برمجي يُستخدم لاكتشاف الأخطاء غير المتوقعة التي قد تحدث أثناء تشغيل البرنامج ومعالجتها.

التعليمات البرمجية لمعالجة الاستثناءات

تُستخدم لتحديد الجزء من البرنامج المُتوقع حدوث استثناء فيه والتعامل معه عند حدوثه، وتتكون من كتل التعليمات البرمجية التالية:

- try: تُستخدم لاحتواء التعليمات البرمجية التي يُحتمل أنّ يُنْتج عنها استثناء أثناء التنفيذ. فإذا حدث استثناء داخل هذه الكتلة، ينتقل البرنامج إلى كتلة except لمعالجته، أمّا إذا لم يَحدث أي استثناء يستمر تنفيذ البرنامج.
- except: تُستخدم لاكتشاف الاستثناءات التي قد تَحدث أثناء تنفيذ التعليمات البرمجية داخل كتلة try ومعالجتها، ممّا يساعد على منع توقف البرنامج واستمرار تنفيذه استمرارًا آمناً ومُنظماً.
- else: تُستخدم لتنفيذ تعليمات برمجية محددة بعد الانتهاء من تنفيذ كتلة try بنجاح.
- finally: تُستخدم لتنفيذ أوامر يتم تنفيذها دائمًا في نهاية البرنامج، سواء حدث خطأ في كتلة try أو لم يحدث.

صيغة كتابة معالجة الاستثناءات Exceptions Syntax

```
try:
    # Code that might cause an error
except ErrorType:
    # Code to execute if an error occurs
else:
    # Code to execute if no error occurs
finally:
    # Code that always executes, whether an error occurred or not
```

مثال 1:

استخدام استثناءات محددة Specific Exceptions:

التعليمات البرمجية التالية تستخدم لحساب قيمة مشاركة كل طالب في تكلفة الرحلة المدرسية؛ وفق التالي:

الكتلة try:

- **استقبال تكلفة الرحلة.**
- **استقبال عدد الطلاب.**
- **حساب قيمة المشاركة.**

الكتلة except:

- **عند حدوث خطأ من نوع ValueError، تُعرَض رسالة تُوضح أنّ القيَم المدخلة غير رقمية.**
- **عند حدوث خطأ من نوع ZeroDivisionError، (عدد الطلاب يساوي صفرًا) تُعَرض رسالة تُوضح أنّه لا يمكن إجراء القسمة على صفر.**

الكتلة else:

- **عند نجاح جميع العمليات دون وقوع استثناء، تُطبَع قيمة المشاركة.**

الكتلة finally:

- **تُنفّذ دائماً بغض النظر عن وقوع أخطاء أم لا.**
- **تَطْبع رسالة تؤكد انتهاء تنفيذ البرنامج.**

```
try:
    print("School Trip Cost Sharing Calculator")
    cost_total = float(input("Enter total trip cost (KWD): "))
    students_count = int(input("Number of participating students: "))
    share_amount = cost_total / students_count
except ValueError:
    print("Input error: Please enter valid numbers only!")
except ZeroDivisionError:
    print("Error: Cannot divide by zero - please check student count")
else:
    print(f"Each student's share: {share_amount : .2f} KWD")
finally:
    print("program ended!")
```

Program output

```
School Trip Cost Sharing Calculator
Enter total trip cost (KWD): 1.5
Number of participating students: 0
Error: Cannot divide by zero - please check student count
program ended!
```

مثال 2:

استخدام استثناء شامل Catch-all Exception:

التعليمات البرمجية التالية تستخدم لحساب قيمة مشاركة كل طالب في تكلفة الرحلة المدرسية وفق التالي:

الكتلة try:

- استقبال تكلفة الرحلة.
- استقبال عدد الطلاب.
- حساب قيمة المشاركة.
- طباعة قيمة المشاركة مباشرة.

الكتلة except:

- تلتقط أي استثناء يحدث من أي نوع عبر (Exception as e).
- تَعرض رسالة عامة تحتوي على وصف الخطأ المخزن في المُتغيّر e.

الكتلة finally:

- تُنفّذ دائماً بغض النظر عن وقوع أخطاء أم لا.
- تَطْبع رسالة تؤكد انتهاء تنفيذ البرنامج.

```python
try:
    print("School Trip Cost Sharing Calculator")
    total_cost = float(input("Enter total trip cost (KWD): "))
    students = int(input("Number of participating students: "))
    share = total_cost / students
    print(f"Each student's share: {share: .2f} KWD")
except Exception as e:
    print('Error is:', e)
finally:
    print("Program ended!")
```

Program output

```
School Trip Cost Sharing Calculator
Enter total trip cost (KWD): 10
Number of participating students: 0
Error is: float division by zero
Program ended!
```

ملاحظة:

1. التعامل مع الاستثناءات Exceptions.

- يُفضل استخدام الاستثناءات المحددة (Specific Exceptions)

مثل ValueError و ZeroDivisionError؛ لأنها تساعد في تحديد الخطأ ومعالجته معالجة واضحة ومحددة.

- يمكن استخدام الاستثناء الشامل (Catch-all Exception).

except Exception as e:

- تُستخدم لالتقاط جميع الأخطاء المحتملة ومعالجتها، ويُفضل استخدام الاستثناء المحدد؛ لأنه يسهّل تتبع الأخطاء وفهمها.

2. التعامل مع كتلة else.

- يُستغنى عن الكتلة else، ونُقلت التعليمة التي كانت بداخلها إلى نهاية الكتلة try.

مثال 3:

برنامج يَسْتقبل أعدادًا صحيحة، ويُظهر رسالة تُحدد ما إذا كان العدد المُدخَل زوجيًا أم فرديًا، وعند إدخال بيانات نصية، يُستثنى الخطأ وطباعة الرسالة You did not enter a valid number، ويستمر البرنامج دون توقف.

```
1   while True:
2     try:
3       number = int(input('Enter a number:'))
4       if number % 2 == 0:
5           print('The number is even.')
6       else:
7           print('The number is odd.')
8     except ValueError:
9       print('You did not enter a valid number.')
```

Program output

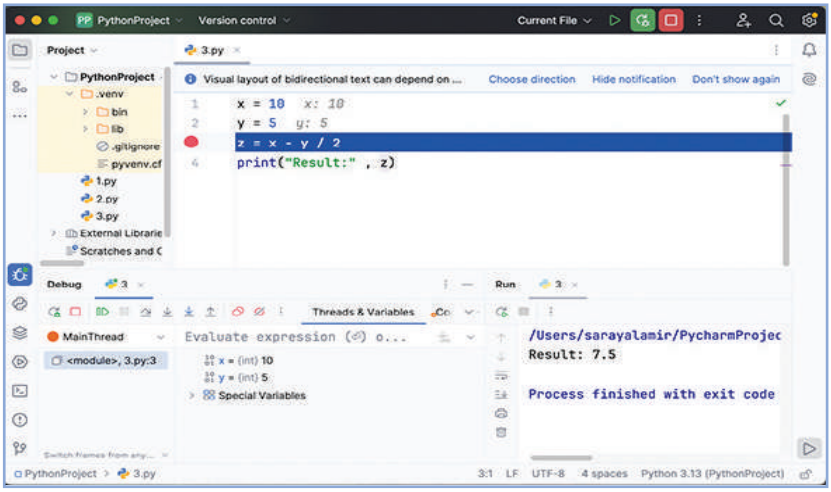
```
Enter a number:56
The number is even.
Enter a number:8.9
You did not enter a valid number.
Enter a number:GR11
You did not enter a valid number.
Enter a number:
```

تصحيح الأخطاء Debugging

عملية تتبع التعليمات البرمجية وتحليلها خطوة بخطوة؛ لاكتشاف الأخطاء، فَهْم سَيْرِ التنفيذ، بهدف تصحيح النتائج غير المتوقعة.

خطوات تتبع الأخطاء في PyCharm

تُنفّذ الخطوات التالية بهدف تتبع الأخطاء البرمجية وتصحيحها وتحليلها.



شكل (3-7) يوضح بعض أدوات تتبع الأخطاء في PyCharm.

1. إضافة نقطة توقف (Breakpoint).

الضغط بجانب رقم السطر المطلوب إيقاف التنفيذ عنده؛ لتظهر دائرة حمراء (●)، في محرر التعليمات البرمجية PyCharm.

لاحظ: Breakpoint : هي أداة نستخدمها لإيقاف البرنامج مؤقتًا عند سطر معين أثناء تشغيله.

2. الضغط على أداة Debug.

- تنفيذ التعليمات البرمجية.
- توقف تنفيذ التعليمات البرمجية عند أول Breakpoint.
- ظهور شريط أدوات التصحيح (Debug Toolbar).
- ظهور لوحة Threads Panel & Variables للقيام بالعديد من العمليات منها مراقبة المتغيرات وقيمها أثناء التصحيح.

3. التعامل مع أدوات التصحيح .

تُستخدم أدوات التصحيح لمتابعة تنفيذ التعليمات البرمجية خُطْوة بخُطْوة، أو للانتقال إلى السطر التالي، أو لإيقاف عملية التصحيح في أي وقت.

وهِي مجموعة من الأدوات تُستخدم لتتبع تنفيذ البرنامج خُطْوة بخُطوة، وتساعد على مراقبة القِيم، واكتشاف الأخطاء المنطقية أو أخطاء أثناء التشغيل.

وتُمكّن هذه الأدوات من إيقاف البرنامج مؤقتًا عند سطر معين (Breakpoint)، أو تنفيذ الأسطر واحدًا تلو الآخر (Step Over/Into)، أو الدخول والخروج من الدوال، أو إعادة التشغيل، أو إنهاء التصحيح.

-  Resume Program: لاستكمال تشغيل البرنامج حتى نقطة التوقف التالية أو حتى النهاية.
-  Pause Program: لإيقاف تشغيل البرنامج مؤقتًا في أي لحظة.
-  Step Over: لتنفيذ السطر الحالي دون الدخول إلى داخل الدوال.
-  Step Into: للدخول داخل الدالّة التي تُستدعى في السطر الحالي.
-  Step Out: للخروج من الدالّة الحالية والعودة للسطر التالي في الدالة التي استدعتها.
-  Rerun : لإعادة تشغيل البرنامج من البداية بنفس إعدادات التصحيح.
-  Stop : لإيقاف جلسة التصحيح نهائيًا.
-  View Breakpoints : لعرض جميع نقاط التوقف وإدارتها.
-  Mute Breakpoints : لتعطيل جميع نقاط التوقف مؤقتًا دون حذفها.

4. ظهور النتائج في Console :

بعد الانتهاء من تنفيذ البرنامج، تَظْهر النتائج أو المُخَرجات في نافذة Console.

أوراق العمل

ورقة عمل (15): **التاريخ:**

حساب قيمة فاتورة المشتريات بعد الخصم .

مشكلة البرنامج:

حساب المستخدم السعر بعد تطبيق خصم الفاتورة، من خلال:

- **إدخال قيمة الفاتورة (أكبر من صفر).**
- **إدخال نسبة خصم (تتراوح بين 0 و 100).**

خوارزمية البرنامج:

1. **بداية البرنامج.**
2. **استخدام كتلة try لمحاولة تنفيذ الخطوات التالية:**
   - **استقبال من المستخدم price (السعر الأصلي).**
   - **استقبال من المستخدم discount (نسبة الخصم).**
   - **حساب السعر النهائي باستخدام:**

     net_price = price - (price * discount / 100)

   - **طباعة** 'Net price after discount is: net_price'
3. **استخدام كتلة except بحيث إذا حدث خطأ في نوع البيانات (ValueError):**

   طباعة: 'Error: Please enter valid numeric values .'
4. **في جميع الأحوال، طباعة الرسالة النهائية:**

   'Program has ended' **داخل** finally.
5. **نهاية البرنامج.**

المطلوب:

اكتب برنامجًا ينفذ التالي:

1. إدخال السعر الأصلي للفاتورة (بالدينار الكويتي).
2. إدخال نسبة الخصم (%).
3. التحقق من أنّ المُدخَلات عددية باستخدام التعليمة البرمجية (try — except).
4. استخدم التعليمة البرمجية الشرطية (if - else) للتأكد من أنّ:
   - السعر أكبر من 0.
   - نسبة الخصم بين 0 و100.
5. حساب السعر النهائي بعد الخصم.
6. طباعة 'Program has ended' في جميع الحالات (وجود أو عدم وجود خطأ).
7. اختبر البرنامج، وتأكد من عدم وجود أخطاء.

تطوير البرنامج:

تحقق:

- إذا كان price <= 0:

  اطبع رسالة خطأ ' Error: Price must be greater than zero '

- إذا كانت discount < 0 أو > 100:

  اطبع رسالة خطأ ' Error: Discount must be between 0% and 100% '

ورقة عمل (16): **التاريخ:**

مؤشّر كتلة الجسم BMI (Body mass index).

مشكلة البرنامج:

اكتشاف الأخطاء وتجاوزها (استكمال البرنامج ليعمل بصورة صحيحة).

خوارزمية البرنامج:

1. بداية البرنامج.
2. بداية التكرار اللانهائي وتنفيذ التالي:
   - استخدام كتلة try لمحاولة تنفيذ الخطوات التالية:
     - استقبال من المستخدم الوزن بالكيلوجرام وحفظه كونه عددًا عشريًا (float).
     - استقبال من المستخدم الطول بالمتر وحفظه كونه عددًا عشريًا (float).
     - حساب مؤشر كتلة الجسم BMI باستخدام المعادلة:
       - قانون كتلة الجسم BMI = الوزن (الكيلوجرام) ÷ الطول (المتر^2).
     - طباعة قيمة BMI بتنسيق عشري من خانتين.
     - تحديد الفئة الصحية بناءً على قيمة BMI
       - إذا كان BMI أقل من 18.5، طباعة 'You are underweight.'.
       - إذا كان BMI بين 18.5 و 24.9، طباعة 'You have a normal weight.'.
       - إذا كان BMI بين 25 و 29.9، طباعة 'You are overweight.'.
       - إذا كان BMI غير ذلك طباعة 'You are obese.'.
     - استخدام الدالة input() لطلب اختيار المستخدم بين الاستمرار في تشغيل البرنامج أو إنهائه.
       - استخدام جملة if للتحقق من إدخال الحرف Q والخروج من الحلقة باستخدام break.
   - استخدام كتلة except بحيث إذا أدخل المستخدم قيمة غير رقمية: (ValueError).
     - طباعة رسالة: 'Error: Please enter valid numeric values for weight and height.'.
   - استخدام كتلة except بحيث إذا حدث خطأ وأدخل المستخدم الطول صفراً (لايمكن القسمة على صفر (:ZeroDivisionError).
     - طباعة رسالة: 'Error: You cannot divide by zero!.'.
3. الرجوع إلى الخطوة 2.
4. نهاية البرنامج.

المطلوب:

1. افتح الملف BMI.py، الذي يَحسِب مؤشّر كتلة الجسم BMI، بناءً على طول الشخص (بالأمتار) ووزنه (بالكيلوجرامات).
2. استخدم التعليمة Try / except البرمجية للتأكد من عدم توقف البرنامج في حال إدخال قيم / نوع بيانات غير متوقع.

استكمل التعليمات البرمجية

```
# Program to calculate BMI with error handling
while True:
    .....:
        # Get user input for weight and height
        weight_kg = float(input('Enter your weight in kilograms: '))
        height_m = float(input('Enter your height in meters: '))

        # Calculate BMI
        bmi_value = weight_kg / (height_m ** 2)

        # Display the BMI
        print(f'Your BMI is: {bmi_value:.2f}')

        # Determine BMI category
        if bmi_value < 18.5:
            print('You are underweight.')
        elif bmi_value >= 18.5 and bmi_value < 25:
            print('You have a normal weight.')
        .....:
            print('You are overweight.')
        else:
            print('You are obese.')
        stop_program = input('Press Enter to continue or Q to quit.').capitalize()
            if stop_program == 'Q':
                break

    .....:
        print('Error: Please enter valid numbers for weight and height.')
    .....:
        print('Error: You cannot divide by zero!')
```

3. اختبر البرنامج، وتأكّد من عدم وجود أخطاء.

الوحـدة الثالثة

المُنتجات الرقمية

1 أمثلة على المنتجات الرقمية

2 مدخل إلى المشروع

3 إعداد فريق العمل

4 خطوات إعداد مشروع

برمجة Python

المنتجات الرقمية Digital products

نواتج التعلم

- توظيف المهارات البرمجية المكتسبة في Python توظيفًا فعالًا لإنتاج برامج متنوعة.
- تنمية القدرة على الابتكار والإبداع من خلال تحليل المشكلات وتصميم حلول برمجية مناسبة.
- تطبيق المهارات البرمجّية في تصميم برامج وتطويرها لتعالج احتياجات من واقع الحياة اليومية.
- المُشاركة بفاعليّة ضمن فرق العمل البرمجية، وإظهار روح التعاون وتبادل المعرفة.
- عرض الأفكار البرمجية أمام الآخرين، ومناقشة الحلول المختلفة مع تقبّل وجهات النظر المتنوعة.
- التفاعل مع التغذية الراجعة؛ لتحسين جودة المشروع، وتطوير الأداء البرمجي.

يُمَثّل رمز الاستجابة السريعة QR رابطاً
لملفات أوراق العمل، ومصادر التعلم

أمثلة على المنتجات الرقمية

استعرضنا فيما سبق المفاهيم الأساسية للبرمجة بلغة Python مثل المُتَغّيرات، الحلقات التَكرارية، الشروط والتعامل مع الأخطاء ...، ومِن ثَمّ يمكننا تصميم برامج متكاملة وتطويرها ؛ لتخدم مواقف حياتية:

أفكار مشاريع Python للصف العاشر:

1

لعبة حجر ورقة مقص:

لعبة تفاعلّية مسلّية بين اللاعب والحاسوب، حيث يختار كلّ طرف إحدى الخيارات الثلاثة: حجر، ورقة، أو مقص، ثُمّ يقوم البرنامج بمقارنة اختيار اللاعب مع اختيار الحاسوب؛ لتحديد الفائز وَفْقًا لقواعد اللعبة المعروفة أو حَسْب الخطوات التالية:

- يستقبل من اللاعب أحد الخيارات: حجر، ورقة، مقص.
- يختار الحاسوب عشوائياً أحد الخيارات: حجر، ورقة، مقص.
- تحديد الفائز وَفقًا لقواعد اللعبة المعروفة.

2

تحدي العواصم:

يُعرَضُ على اللاعب اسم دولة عشوائية ويُطلب منه إدخال عاصمتها الصحيحة، ثم يُقيّم البرنامج صحة الإجابة ويجمع النقاط لكل إجابة صحيحة وَفْق الخُطْوات التالية:

- يُعَرضُ للاعب عشوائياً اسم دولة.
- يَستقبل من اللاعب اسم العاصمة.
- يَختَبِر صحة الإجابة وتزداد عدد النقاط بمقدار واحد عند صحتها.
- تُكرِر الخطوات السابقة لعدد محدد من الأسئلة، ويَعرِضُ المجموع النهائيّ للنقاط.

3

حساب الوقت المتبقي:

يساعد في حساب الوقت المتبقي حتى موعد معيّن (مثل موعد الحصة أو الحافلة أو النوم) يُدخل المستخدم الوقت الحاليّ والوقت المستهدف ويقوم البرنامج بطرح القيمتيْن؛ لإظهار الفرق بالساعات والدقائق وفق الخطوات التالية:

- يَستقبِل من المستخدم الوقت الحالي (الساعة والدقيقة).
- يَستقبِل من المستخدم وقت الموعد (الساعة والدقيقة).
- يَعرِض الوقت المتبقي للمستخدم بالساعة والدقيقة ما لم يكون وقت الموعد قد مضى.

مدخل إلى المشروع

من خلال دراستك لوحدة البرمجة بلغة Python، والاطّلاع على التطبيقات (المنتجات الرقمية) المساعدة المتوافرة على الرابط الإلكتروني المتمثّل في رمز الاستجابة السريع QR بداية وحدة المُنتَجات الرقمية، والمتوافِر على أجهزة الحاسوب بالمختبر المدرسيّ، صَمّم المشروع الخاص بك، والذي يتضمّن كلاً من:

- تحديد مشكلة المشروع.
- إعداد الخوارزمية.
- تصميم خريطة التدفّق.
- كتابة الرموز البرمجية.
- اختبار البرنامج، تصحيح البرنامج إنْ وجد.
- عرض المشروع على زملائك، ومعلمك.
- تحسين البرنامج وَفْق التغذية الراجعة.
- بناء البرنامج.

إعدد فريق العمل

يتكـوّن الفريــق مــن 4 إلــى 5 أعضــاء، تُقسّــم المهــام بينهــم علــى النحــو التالــي:

- قائد الفريق: المسؤول عن إدارة عمل الفريق وتجميع الملفات والمستندات المطلوبة وتقديمها للمعلم.
- مُعِد الخوارزمية: المسؤول عن إعداد الخطوات الأساسية لعمل المشروع.
- مصمم خريطة التدفّق: المسؤول عن ترجمة الخوارزميّة إلى خريطة التدفّق.
- المُبرمِج: المسؤول عن ترجمة خريطة التدفّق لتعليمات برمجية واختبارها والتأكد من سلامتها.
- مُعِد العرض التقديمي: المسؤول عن إعداد وتصميم العرض التقديميّ للمناقشة.

اكتب بيانات أعضاء الفريق موضحاً مهام كل عضو:

اعتماد المعلم: التاريخ:

خُطوات إعداد المشروع

1. فكرة المشروع

يُناقش أعضاء الفريق ويحدد موضوعاً يثير اهتمامهم، سواءً كان تطبيقاً أو لعبة أو أداة تحليل البيانات وغيرها، على أن تكون مُطابقة للشروط التالية:

- أن تكون الفكرة واضحة وقابلة للتطبيق (شرح المشكلة والحلول).
- تحديد الفئة المستفادة وتوضيح كيفية الاستفادة.
- أن تكون الفكرة قابلة لتصميم نموذج برمجيّ.
- أن تحتوي على عدة مصادر خارجية يسهل الرجوع إليها.

اكتب فكرة المشروع بعد مناقشة أعضاء الفريق:

اعتماد المعلم: التاريخ:

2. إعداد بيئة العمل

تثبيت ما يحتاج إليه الفريق من برمجيات وأدوات لإعداد المشروع، ومنها:

- PyCharm: لتصميم النموذج البرمجي واستيراد المكاتب اللازمة.
- PowerPoint: لإعداد العرض التقديمي.
- Windows screen recorder: لتسجيل الشاشة إستعدادًا لتوثيق المشروع.
- Draw.io: لرسم خريطة التدفّق للمشروع.
- Microsoft Edge: للبحث عن المعلومات.
- Copilot: للمساعدة في كتابة الأكواد واقتراح حلول برمجية ذكية.
- GitHub (أو Git): لإدارة نسخ التعليمات البرمجية ومشاركة الملفات بين أعضاء الفريق.

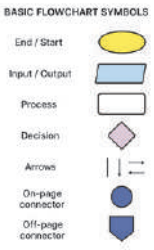
و أي مستلزمات أخرى تخدم المشروع وتوزيع مهام أعضاء الفريق.

3. إعداد هيكل المشروع

يُفَضّل الاستغلال الأمثل للمهارات البرمجية التالية:

- استخدام لغة Python.
- استخدام أنواع متعددة من المتغيرات.
- استخدام التعليمات البرمجية للشروط Conditions.
- معالجة الأخطاء باستخدام الاستثناءات try/ except.

ارسم خريطة التدفق / الخوارزمية



اعتماد المعلم: التاريخ:

تابع/ ارسم خريطة التدفق / الخوارزمية

اعتماد المعلم: التاريخ:

4. كتابة التعليمات البرمجية

يَكتب المبرمجُ التعليمات البرمجية استناداً لما خُططّ له سابقاً.

5. اختبار المشروع

يَختبرِ المبرمجُ البرنامج، ويتأكد من خلوه من الأخطاء البرمجية، اللُغوية، والعلمية.

6. تطوير المشروع

يُبسِّط المبرمجُ ويرتب التعليمات البرمجية إذا أصبحت صعبة القراءة، ويضيف التعليقات والبيانات الإرشادات التي توضح مهمة التعليمات البرمجية.

7. توثيق المشروع

يُنشئ مصمم العرض التقديمي عرضًا شاملاً للجوانب التالية:

- عَرْض بيانات المدرسة وأعضاء الفريق.
- الترحيبُ بالمعلم وزملائه المتعلمين.
- عَرْض مقدمة عن فكرة المشروع؛ لتوضيح الهدف والمشكلة والفئة المستفادة والحل.
- عَرْض خريطة التدفق.
- عَرْض النموذج البرمجي.
- عَرْض المصادر.
- فتح باب الأسئلة والمناقشة.
- الختام.

وتسليم مجلد مُجمّع لجميع ملفات المشروع.

8. مشاركة المشروع

عَرْض الفريق للمشروع، ومناقشته مع المعلم والمتعلمين.

9. الاستمرار في التعلم

التوسّع في الاطّلاع على مستجدِّات الذكاء الاصطناعي وتنمية قدرات المتعلم البرمجية من خلال الدورات التدريبية والمنتديات الحاسوبية.

المَراجِع

- مؤسسة بايثون للبرمجيات. (2024). دليل بايثون الرسمي: إصدار 7. https://www.python.org/doc.
- إطار عمل الأمن السيبراني الوطني الأمريكي. (2024). https://www.nist.gov/cyberframework.

الكتاب كاملاً

قيِّم مناهجنا