وزارة التربية
Ministry of Education
State of Kuwait | دولـــة الكويـت

تقنية المعلومات 10

الصـف العاشر

الفصل الدراسي الأول

المرحلة الثانوية

وزارة التربية
Ministry of Education
State of Kuwait | دولـــة الكويـت

تقنية المعلومات 10

إشراف

منى سالم عوض سالم (رئيس اللجنة)

علي أحمد الكندري (نائب رئيس اللجنة)

تأليف

رأفـت صابـر عبـداللاه أحمـد **حسـام الديـن علي عـبدالقادر**

محمـد السيـد محمد إبراهيم **أشرف رضوان رضـوان سليمـان**

إبراهيم عبدالله إبراهيم المياس

تصميم

إيمان عبدالعزيز أحمد الفارسي **سنيـه محمـد علـي المؤمـن**

إخراج

أشرف رضوان رضوان سليمان

الطبعة الثانية
1447 هـ
2026/2025 م

الطبعة الأولى 2025/2024م

الطبعة الثانية 2026/2025م

لجنة المواءمة

إشراف

منى سالم عوض سالم (رئيس اللجنة)

رأفت صابر عبداللاه أحمد — **حسام الدين علي عبدالقادر**

أشرف رضوان رضوان سليمان — **د.يوسف منصور يوسف الخليفي**

إبراهيم عبدالله إبراهيم المياس — **منار مصطفى عبدالحميد جمال**

تصميم

منى مرزوق مخلد العازمي — **ساره ياسين عبد الله الأمير**

إخراج

أشرف رضوان رضوان سليمان

المراجعة العلمية

أشرف رضوان رضوان سليمان

فاطمة نجم جاسم الهولي

عبدالرحمن محمد مال الله الجزاف

الفريق المُساند

تصميم

سنيه محمد علي المؤمن

أودع بمكتبة الوزارة تحت رقم (73) بتاريخ 2025/9/8م


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

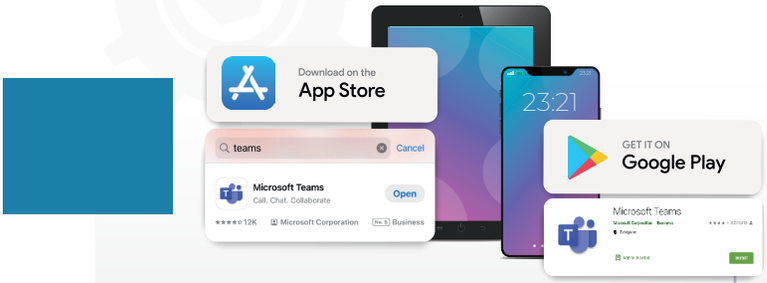
سمو الشيخ صباح خالد الحمد الصباح
ولي عهد دولة الكويت

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 الأجهزة الذكية

من جهاز الحاسوب

احرص على تحديث تطبيق Microsoft Teams بشكل دوري.

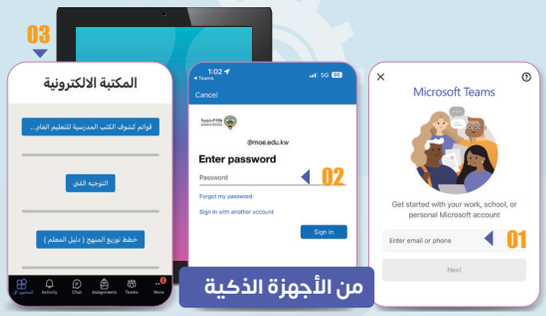
ثانياً: تفعيل Microsoft Teams على الجهاز الرقمي

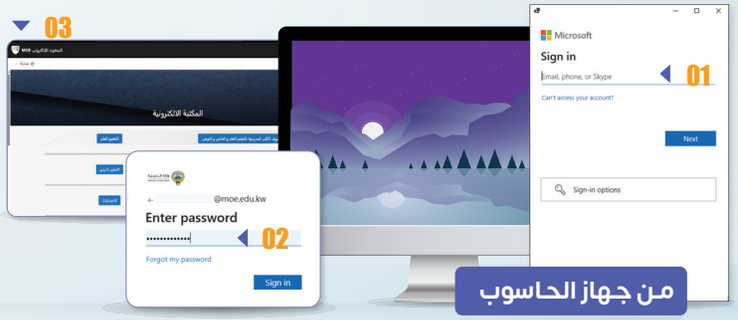
لتشغيل التطبيق اتبع الخطوات التالية:

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

02 أدخل كلمة المرور الخاصة بحساب Teams.

03 تنقل بين واجهات التطبيق مثل المحتوى الإلكتروني، وفرق المواد الدراسية.





احتفظ بكلمة المرور في مكان آمن لتسهيل تسجيل الدخول لاحقًا.

لا تشارك بياناتك مع أي شخص غير موثوق به.

مـن هـو 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 هـي خـدمـة حـمـد شـات؟

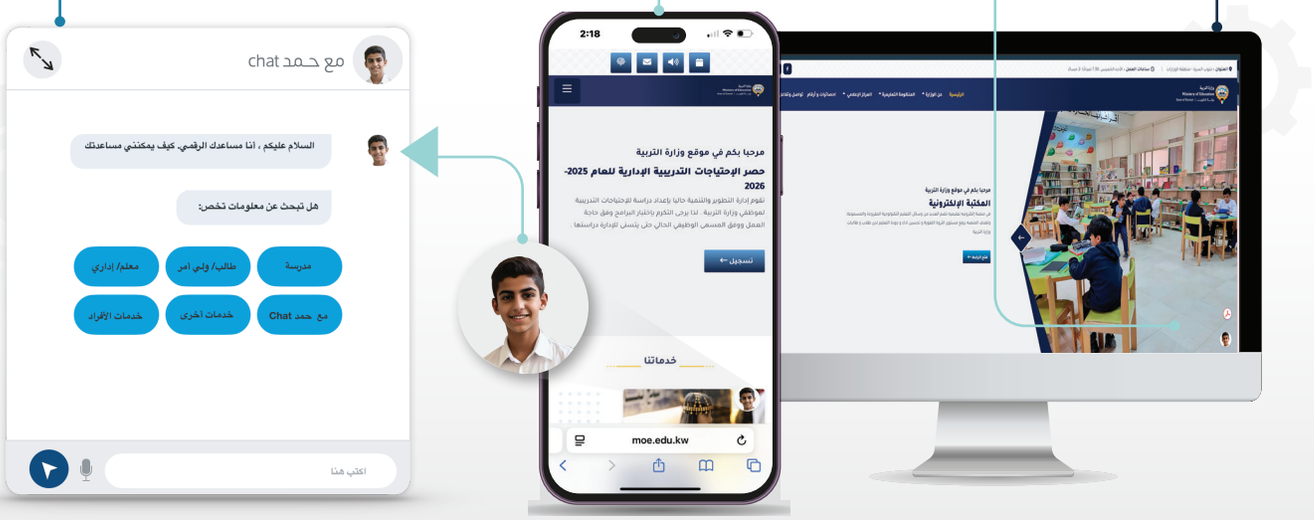
هـي خدمـة المحادثـة الذكيـة المقدمـة مـن وزارة التربيـة فـي دولـة الكويـت، تعتمـد علـى الـذكاء الاصطـناعي، تتمثـل وظيفتـه الأساسـية فـي تسـهيل عمليـة الفهـم والاسـتيعاب مـن خلال تقديـم الشـروحات المبسـطة، والإجابـة علـى الاستفسـارات، وتوجيـه المتعلميـن خلال مسـيرتهم التعليميـة.

أيـــن أجـــده؟

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

02 الضغط عـلى صـورة حمد شات

03 اكتب استفساراتك، وستتلقى الإجابات بشكل فوري.



المحـــتوى

العنوان	الصفحة

المقدمة

في عالم اليوم الذي يعتمد بشكل متزايد على التكنولوجيا، أصبح الأمن السيبراني وبرمجة Python من المجالات ذات الأهمية المتزايدة.

حيث تعد هذه المجالات من المهارات الأساسية التي يحتاجها الطلاب في حياتهم العملية وتنمية الوطن.

كذلك يُستخدم الأمن السيبراني لحماية الأنظمة والشبكات والأجهزة الإلكترونية من الهجمات الإلكترونية. التي قد تُستخدم في سرقة البيانات أو إتلافها أو تعطيل الوصول إلى النظام. والتي من شأنها أن يكون لها عواقب وخيمة على الأفراد والشركات والحكومات.

وتُعتبر برمجة Python لغة برمجة قوية وسهلة التعلم، وهي مفيدة في مجموعة متنوعة من المجالات، بما في ذلك الأمن السيبراني.

ويُمكن أن يساعد تعلم الأمن السيبراني وبرمجة Python الطلاب في:

- اكتساب مهارات مهمة في سوق العمل، حيث أصبحت هذه المهارات أساسية، ومطلوبة في العديد من الوظائف، وخاصة في مجالات التكنولوجيا.
- الاستعداد للمستقبل، حيث يتوقع أن تستمر أهمية هذه المجالات في النمو في المستقبل.
- المساهمة في تنمية الوطن، حيث يمكن للطلاب الاستفادة من هذه المهارات لحماية الوطن من الهجمات الإلكترونية.

كما يهدف هذا الكتاب إلى تزويد الطلاب في الصف العاشر بالمعرفة والمهارات اللازمة في مجال الأمن السيبراني وبرمجة Python.

المؤلفون

الوحـدة الأولــى - الأدوات الرقمية

الأمن السيبراني Cyber Security

1. مفاهيم الأمن السيبراني
2. المحاور الرئيسة لأمن المعلومات و الأمن السيبراني CIA
3. الإجراءات الأمنية للمحاور الرئيسة للأمن السيبراني CIA
4. ممارسات الأمن السيبراني
5. الوظائف في مجال الأمن السيبراني
6. الجوانب الأخلاقية الرئيسية في مجال الأمن السيبراني

الأمن السيبراني

Cyber Security

نواتج التعلم

- شرح مفهوم الأمن السيبراني وأمن المعلومات وأهميتهما في العصر الرقمي.
- تفسير عناصر نموذج CIA (السرية، السلامة، 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 بين الأدوار والمسؤوليات المختلفة ضمن مجالات العمل في وظائف الأمن السيبراني.
- إظهار سلوكًا أخلاقيًا ومسؤولية عالية عند التعامل مع البيانات الرقمية والمعلومات الحساسة.

يمثل رمز الاستجابة السريعة **QR** رابط
لملفات أوراق العمل، ومصادر التعلم

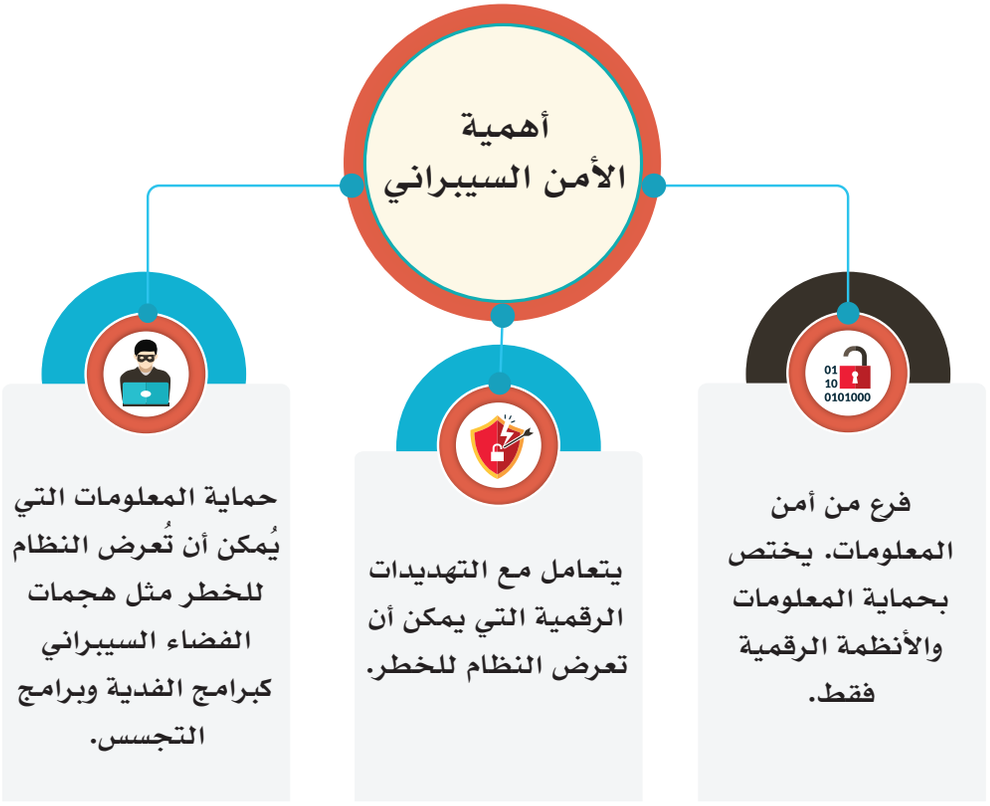
مفاهيم الأمن السيبراني The Concepts of Cyber Security

أمن المعلومات Information Security

الممارسات والتدابير المُتبعة لحماية البيانات والأنظمة من الوصول غير المصرح به (لاستخدامها، الاطلاع عليها، تعطيلها، تعديلها، أو تدميرها)، وتشمل حماية المعلومات بشكل عام بغض النظر عن الوسائط المستخدمة سواء كانت رقمية أو غير رقمية.

الأمن السيبراني Cyber Security

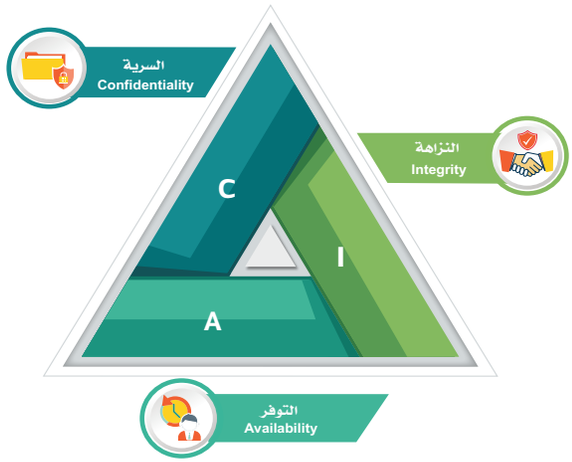
ممارسة حماية الأنظمة والشبكات من التهديدات الإلكترونية، وحماية البيانات الرقمية، والبرامج من الهجمات التي تهدف للوصول إلى المعلومات الهامة أو تغييرها أو تدميرها، ويتم تطبيق الإجراءات، والضوابط، والعمليات، والتقنيات لتقليل مخاطر الهجمات السيبرانية والحماية من الاستغلال غير المصرح للأنظمة والشبكات والتقنيات.



شكل (1-1) يوضح 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 المُصرح به، وضمان دقتها واكتمالها، وجعلها متاحة للمستخدمين المصرح 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 وموثوقية المعلومات الرقمية (Authenticity , Reliability)، وسلامتها. لضمان عدم قيام أي أطراف غير مصرح لهم Unauthorized بالتلاعب بالمعلومات التي ترسلها وتستقبلها عبر الإنترنت.

التوفر Availability

يقصد بالتوفر التأكد من أن 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 التوقف عن العمل.

أمثلة على الهجمات التي تؤثر على المحاور الرئيسية للأمن السيبراني

CIA TRIAD

Confidentiality السرية	Integrity النزاهة	Availability التوفر
		
Password cracking استخدام هجمات مختلفة (خوارزميات وتطبيقات) لتخمين كلمة المرور أو كشفها والوصول إلى حساب المستخدم. **Dumpster diving** يحصل المخترق على المستندات أو البيانات الحساسة التي لم يتم التخلص منها نهائيًا بطريقة آمنة، لشن هجوم سيبراني. **Phishing** محاولة سرقة معلومات حساسة ومهمة، عادةً ما تكون في شكل أسماء مستخدمين أو كلمات مرور أو أرقام بطاقات ائتمان أو معلومات حساب مصرفي أو بيانات هامة أخرى.	**Man In The Middle (MITM) attack** هجوم إلكتروني حيث يقوم المخترق من خلاله بنقل وتغيير الاتصالات بين طرفين يعتقدان أنهما يتواصلان مباشرة مع بعضهما البعض، وأن الاتصال يتم بينهما مباشرة لتبادل المعلومات.	**DoS* (Denial of Service)& DDoS* (Distributed Denial of Service) Attacks** نوع من الهجمات السيبرانية الذي يستهدف مواقع الإنترنت والخوادم Servers، حيث يتم إغراقها بعدد هائل من الطلبات وحركات المرور Traffic، مما يؤدي إلى إضعاف خدمات موقع الإنترنت أو تعطيله تمامًا. • هجمات DoS: تستهدف جهازًا أو خدمة واحدة بغرض إغراقها بطلبات زائفة، ومن السهل تتبع هجماتها. • هجمات DDoS: تُنفذ من خلال أجهزة متعددة ترسل حزمًا من البيانات، وتهاجم من مواقع متعددة. مما يزيد من سرعتها، وقوة الهجوم، مما يصعب التصدي لها وتتبعها. • حركة مرور الويب Traffic: هي كمية البيانات التي يتم إرسالها واستقبالها من قبل زوار ومستخدمي موقع الإنترنت.

شكل (1-3) 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، ويتم تحديد أولويات تنفيذ السياسات الأمنية للمبادئ الثلاثة (السرية، النزاهة، التوفر) وفقًا لعملية 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 الوصول للبيانات المطلوبة، حفاظًا على سرية المعلومات المصرفية.

- **النزاهة Integrity:**

ضمان سلامة البيانات حيث لا يمكن لأي شخص تغيير معلومات الحسابات المصرفية ما لم يكن مصرحًا له، من خلال إبلاغ المستخدم عن أي عمليات مصرفية تمت 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

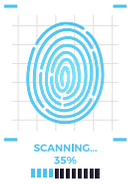
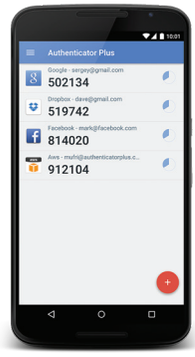
المُصادقة هي طبقة إضافية أو أكثر من الأمان تستخدم لحماية حسابات المستخدمين. بتقديم عدة أشكال منفصلة لتحديد الهوية قبل منح الوصول. وتتمثل في:

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 بشكل متزايد 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) 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.

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

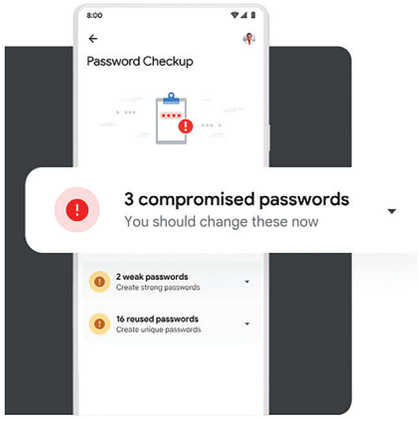
تطبيقات المراسلة الآمنة (التشفير) Secure Messaging Apps

توفر 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 مزودي الخدمة والمتسللين، من اعتراض الرسائل أو قراءتها، يمكن التواصل بسرية دون المساس بالخصوصية.

برامج إدارة كلمات المرور Password Managers

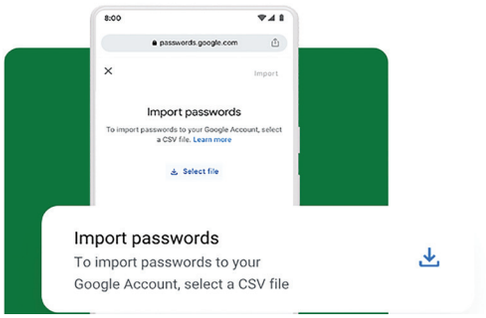
تعتبر برامج إدارة كلمات المرور من الأدوات الرقمية التي تساعد على تقديم الحماية والأمان لحسابات المستخدمين، وذلك من خلال المميزات التالية:

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



شكل (1-5) يوضح 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).

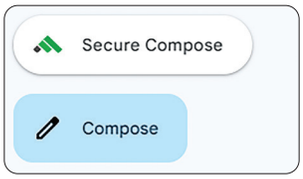
تشفير البريد الإلكتروني Encryption for Email

توفير طبقة إضافية من الحماية للمعلومات التي تتم مشاركتها عبر البريد الإلكتروني. حيث يتم تشفيرها بطريقة لا يمكن إلا للمُرسل إليه فك 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:

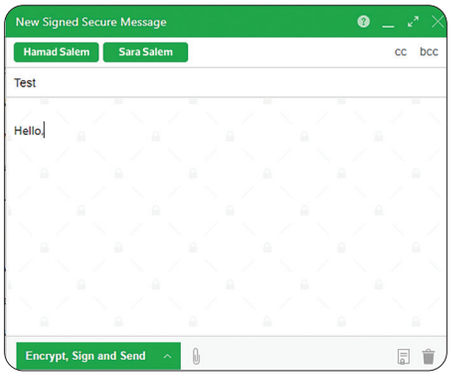
- تثبيت الأداة من موقع Chrome Web Store الخاص بمتصفح Chrome.

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



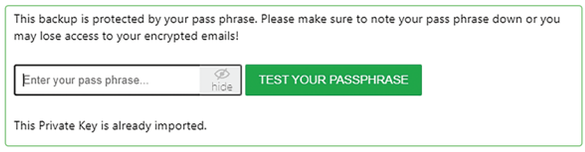
شكل (1-7) يُوضح بداية عملية تشفير الرسالة

- **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.**

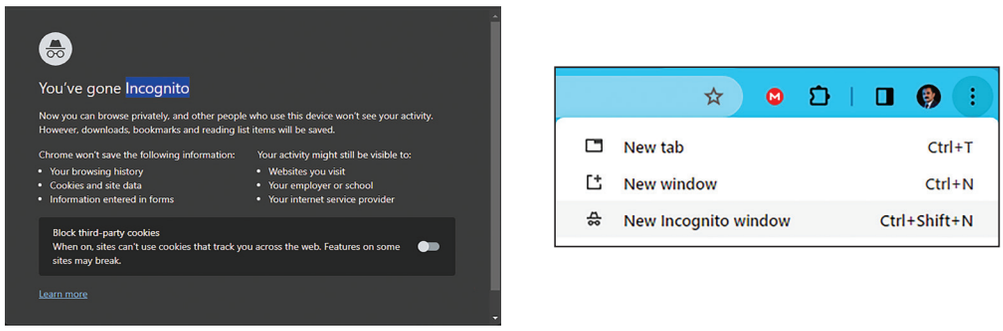


شكل (1-9) يُوضح مكان إدخال رمز التشفير لدى مُستقبل الرسالة المُشفرة

الخصوصية في متصفحات الإنترنت Privacy-Focused Web Browsers

توفر متصفحات الويب الآمنة (مثل Mozilla Firefox أو Brave) تعزيز ميزات الأمان. عن طريق التحكم بإعدادات وإضافات تتيح للمستخدمين الحفاظ على مستوى أعلى من الخصوصية أثناء تصفح الإنترنت، ومنها خاصية التصفح المتخفي Incognito بمتصفح Google Chrome، حيث توفر الخصوصية لأي مستخدم لمتصفح الإنترنت من خلال:

- **عدم الاحتفاظ بسجل التصفح 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

توفر 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، تعزيز مبدأي النزاهة والتوفر، ومنع الاختراقات الناتجة عن ثغرات معروفة. فالمطورون يُطلقون تحديثات لسد الثغرات الأمنية المكتشفة حديثًا، وفي حال عدم تثبيت هذه التحديثات، تظل الأنظمة عرضة للاختراق، تشتمل 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:

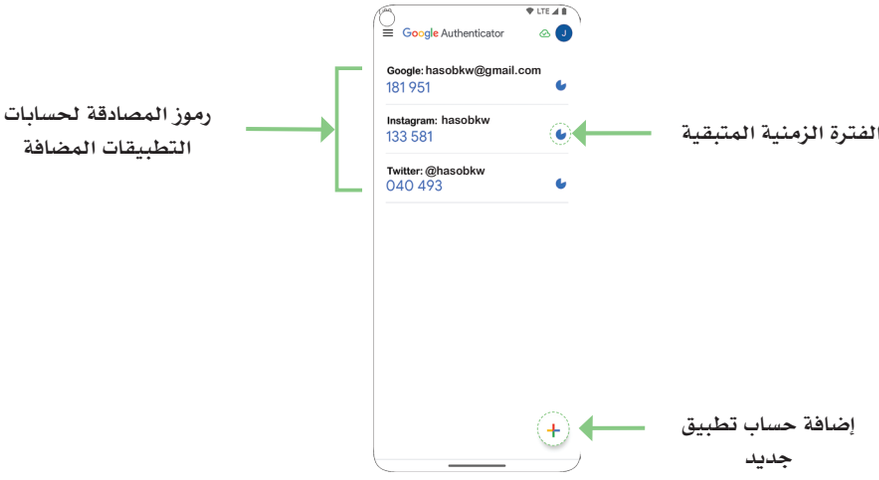
إنشاء مصادقة باستخدام تطبيق Google Authenticator:

1. حمل التطبيق من متاجر الهواتف الذكية.

Apple store

Andriod Store

2. أنشئ حساب على التطبيق من خلال حساب Google.
3. أضف التطبيقات المختلفة المراد تأمينها من خلال المصادقة، مثال 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 يتم توليده باستخدام تطبيق المصادقة Google Authenticator، كما هو موضح بالشكل.



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

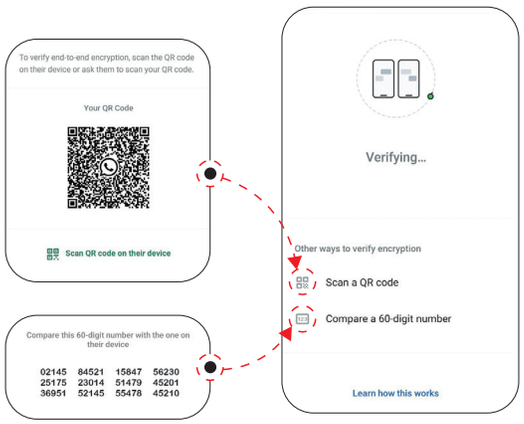
6. انسخ الرمز ثم أضفه في تطبيق Instagram، خلال الفترة الزمنية المحددة.
7. شغل تطبيق Instagram بشكل آمن.

ورقة عمل (2) - ورقة عمل لاصفية:

تتميز المحادثات المشفرة بينك وبين شخص آخر في تطبيق WhatsApp برمز أمان خاص بها. يُستخدم هذا الرمز للتحقق من أن المكالمات والرسائل التي ترسلها إلى تلك الدردشة مشفرة تمامًا.

ويمكن الوصول لهذا الرمز من خلال:

- شاشة معلومات[1] جهة الاتصال Contact Info الخاصة بأحد الأشخاص الذين تتواصل معهم.
- اختيار Encryption.
- التحقق من رمز الأمان Verify encryption.
- اختيار أياً من رمز QR أو الرقم المكون من ٦٠ رقماً.



شكل (1-13) يُوضح تشفير المحادثات في تطبيق WhatsApp

تُعد هذه الرموز فريدة لكل محادثة، ويمكن مقارنتها مع الطرف الآخر للتحقق من أن الرسائل بين الطرفين مشفرة.

---

١ يجب مراعاة اختلاف الشاشات في أنظمة تشغيل الهواتف الذكية.
يعتبر التحقق من رمز الأمان عملية اختيارية للمحادثات المشفرة بين الطرفين.

ورقة عمل (3) ورقة عمل لاصفية

- ناقش مع معلمك وأسرتك كيفية إجراء المصادقة الثنائية بين تطبيق وزارة التربية الرسمي، وتطبيق هويتي، ثم سجل الخطوات هنا.

الخطوات

01. .....
02. .....
03. .....
04. .....
05. .....
06. .....
07. .....
08. .....
09. .....
10. .....
11. .....
12. .....
13. .....
14. .....
15. .....
16. .....
17. .....

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 وتجنب 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 الهامة.
- **مُكافحة التصيد 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*[٢]).
- **التحقق من مصادر المعلومات:** تأكد من صحة المعلومات ومصدرها قبل الوثوق بها.
- **النسخ الاحتياطي للبيانات Data backup:** يجب التأكد من إجراء عمليات النسخ الاحتياطي للملفات والمستندات الهامة بانتظام لضمان الوصول إليها في حال تم اختراق النسخة الأصلية منها.

---

٢ إدارة مكافحة الجرائم الإلكترونية تتلقى جميع البلاغات بخصوص كل ما يخالف القانون وتقوم بإجراء التحريات للتأكد من صحة البلاغات وجدية المعلومات واتخاذ الإجراءات القانونية اللازمة حيالها. للبلاغات يرجى الاتصال على رقم الطوارئ ٩٧٢٨٣٩٣٩، سيتم 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 الغير مصرح 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 وتطوير أنظمة الأمان وتنفيذها لحماية الشبكات والبنية التحتية للأنظمة، (الجدران النارية وأساليب التشفير وأنظمة الكشف عن الاختراق). وكذلك إجراء تقييمًا للثغرات باختبارها لتحديد ومعالجة الضعف في الأنظمة.

- **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 وشبكات تكنولوجيا المعلومات الآمنة، وتطوير أطر الأمان وإنشاء سياسات الأمان، وضمان الامتثال للمعايير والتشريعات الصناعية. يتعاون 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. تحليل أنماط الحوادث، واحتواء التهديدات، وتحليل مدعوم بالأدلة لتحديد سبب الحوادث. كذلك تطوير خطط استجابة للحوادث وتقديم توصيات لتحسين قدرات كشف واستجابة الحوادث

- **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 التجسس على الآخرين: التجسس 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 أو مشاركة البرامج أو الملفات دون إذن.**

  - **حماية الأجهزة من الفيروسات: يجب الحرص على حماية الأجهزة من الفيروسات وبرامج التجسس التي قد تسرق المعلومات أو تدمر 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: التنمر الإلكتروني هو استخدام الإنترنت لإيذاء أو مضايقة شخص ما، وهو أمر غير مقبول تمامًا.**

  - **الحذر عند التعامل مع الغرباء على الإنترنت: يجب عدم مشاركة المعلومات الشخصية مع الغرباء 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 التحقق من صحة المعلومات قبل نشرها على الإنترنت، وعدم نشر الشائعات أو المعلومات المضللة.**

  - **الإبلاغ عن أي سلوك غير أخلاقي: إذا شاهدت أي سلوك غير أخلاقي على الإنترنت، مثل التنمر أو التحرش 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 لحل مشكلة محددة بطريقة منظمة.
- تصميم خريطة تدفق **Flowchart** توضح خطوات حل المشكلة.
- التعرف على لغة البرمجة **Python** ومجالات استخدامها.
- استخدام محرر الأوامر **PyCharm** لكتابة وتنفيذ التعليمات البرمجية.
- استخدام الدالة المدمجة **()print** في طباعة المخرجات.
- التمييز بين هياكل البيانات المختلفة في **Python**، ويحدد أنواعها واستخداماتها.

يمثل رمز الاستجابة السريعة **QR** رابط
لملفات أوراق العمل، ومصادر التعلم

مُقدمة

تُعتبر 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:**

يمكن استخدام البرمجة لإنشاء برامج تعالج مشكلات محددة في حياتك أو عملك، مما يُحسّن من الكفاءة والانتاجية.

- **الإبداع:**

تُتيح البرمجة فرصة الإبداع من خلال إنشاء ألعاب، تطبيقات، مواقع إلكترونية، وبرامج أخرى تُعبر عن الأفكار والرؤية.

- **تحسين المهارات:**

يُساعدك تعلم البرمجة على تحسين مهارات حل المشكلات، التفكير المنطقي، والإبداع، وهي مهارات أساسية للنجاح في أي مجال.

- **توفير فرص وظيفية جديدة:**

يُمكن تعلم البرمجة للحصول على وظيفة في مجال التكنولوجيا، مثل تطوير البرامج، تحليل البيانات، أو الذكاء الاصطناعي، وهي مجالات تُعاني من نقص في الكوادر المؤهلة.

البرمجة

مجموعة من التعليمات والأوامر تترجم إلى لغة الآلة والتي بدورها توجه الحاسوب لتنفيذ مجموعة من القواعد، والتعليمات لحل مشكلة ما.

خطوات إنشاء برنامج

تمر عملية إنشاء برنامج حاسوبي بمجموعة من الخطوات والإجراءات:

أ- تحديد المشكلة

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

مثلاً إذا كان كنا نرغب في إيجاد مساحة المستطيل، فأول سؤال يتبادر للذهن، ماهي (المعطيات) المتاحة؟، حيث نحتاج لمعرفة طول وعرض المستطيل، بعد ذلك نحتاج لمعرفة العمليات والإجراءات اللازمة للوصول إلى الحل، حيث سيكون القانون، المساحة = الطول X العرض الذي من خلاله سيتم إيجاد الناتج النهائي المطلوب.

ب- إعداد خطة الحل (الخوارزمية Algorithm)

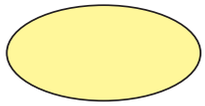
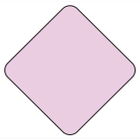
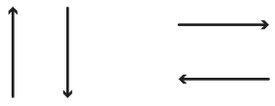
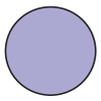
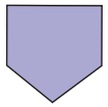
الخوارزمية عبارة عن مجموعة من الخطوات المنطقية والمتسلسلة لحل مشكلة ما، وقد أُطلق عليها هذا الاسم نسبةً إلى محمد بن موسى الخوارزمي، عالم الرياضيات المسلم، وهي طريقة وصفية توضح خطوات حل المشكلة.

كتابة خوارزمية لجمع عددين وطباعة ناتج الجمع.

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.

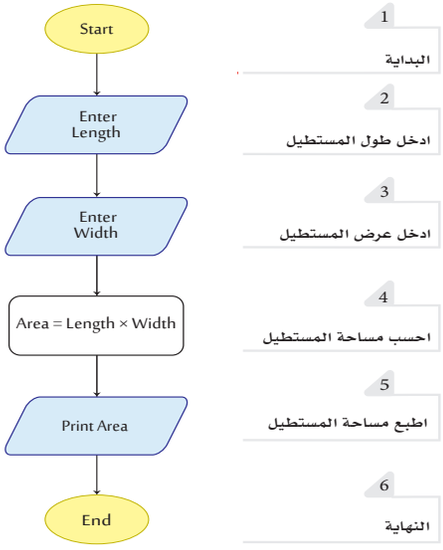
ج- خرائط التدفق (Flowchart)

إحدى أنواع المخططات الرسومية التي توضح ترتيب العمليات اللازمة لحل مشكلة ما، حيث تستخدم أشكال قياسية متفق عليها للدلالة على العمليات المختلفة، والتي تُسهل على المبرمج فهمها لتحويلها إلى برنامج مكتوب بإحدى لغات البرمجة، والجدول رقم (1-1) يوضح وظيفة كل شكل:

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

جدول (1-1) أشكال ووظائف المخططات الرسومية

- التخطيط التالي يُمثل خريطة تدفق حساب مساحة المُستطيل:



شكل (1-2) خريطة التدفق / حساب مساحة المُستطيل

د- كتابة البرنامج

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

هـ- اختبار البرنامج

يتم تشغيل البرنامج وإدخال بيانات ومقارنة النتائج لاختبار صحته وعمله بصورة سليمة، لتفادي ظهور بعض الأخطاء أثناء كتابة البرنامج أو أثناء التنفيذ.

و- إصدار البرنامج

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

ز- توثيق البرنامج

عملية تسجيل وتنظيم المعلومات المتعلقة بالبرنامج، بما في ذلك تصميمه، ووظائفه، وكيفية استخدامه، وآلية صيانته، لتيسير تطويره من قبل المطورين فضلاً عن تسهيل استخدامه من قبل 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 صفر، ورسم خريطة تدفق تُوضح ذلك

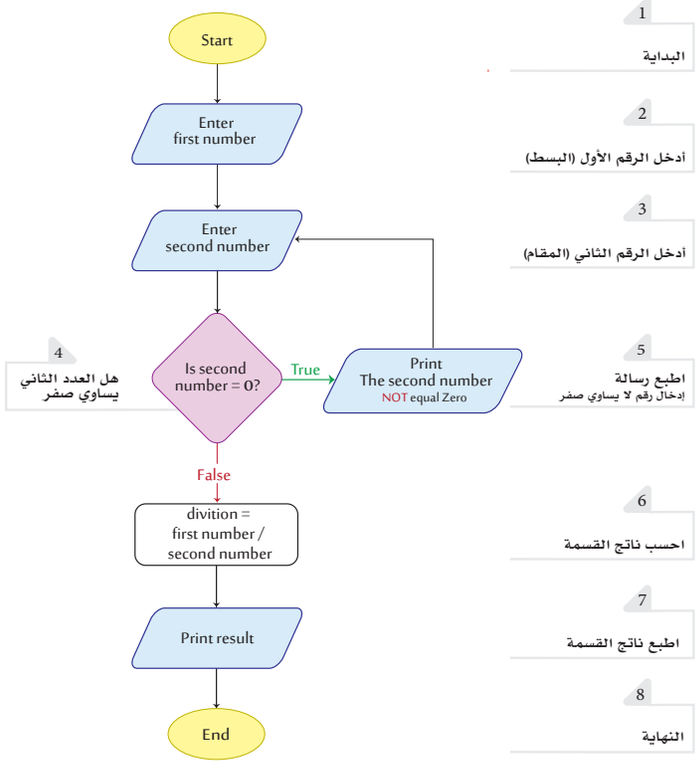
خوارزمية البرنامج:

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 The second number NOT equal Zero، ثم الانتقال للخطوة 3.
6. حساب ناتج القسمة.
7. طباعة ناتج القسمة.
8. نهاية البرنامج.

خريطة التدفق:



شكل (1-3) خريطة التدفق / لإيجاد ناتج القسمة

لغة البرمجة Python

تُعتبر 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، برامج سطح المكتب، أدوات تحكم الأتمتة (مجموعة من الإجراءات لإنجاز مهمة ما بأقل قدر من التدخل البشري)، تطبيقات الشبكة، علم البيانات، الذكاء الاصطناعي، والتعلم الآلي، تحليل البيانات الضخمة، وإنشاء نماذج تنبؤية، وذلك لسهولة استخدامها و وجود العديد من المكتبات المتخصصة. كذلك يتمتع مجتمع Python بدعم كبير من المطورين والمبرمجين، مما يسهل الحصول على المساعدة والدعم المستمر.

كذلك تُعتبر Python لغة كائنية التوجه Object-Oriented Programming، وتُمثل (الأرقام، النصوص، القوائم، القواميس، الدوال، وغير ذلك ككائنات Objects. كما تمتاز لغة Python بالعديد من المميزات التي جعلتها شائعة الاستخدام وواسعة الانتشار منها:

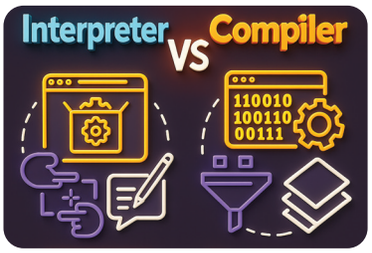
- سهولة التعلم والاستخدام، مما يجعلها اللغة المفضلة لتعلم البرمجة.
- يُمكن تطوير برمجيات معقدة من خلالها.
- تُستخدم في مجالات متعددة.
- تحتوي على العديد من المكتبات الجاهزة.
- تتوافق مع العديد من أنظمة التشغيل.
- يتم تحديثها بصورة مستمرة.

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، والذي يشير إلى الخطوات المطلوب تنفيذها من أجل تحويل الأوامر البرمجية إلى لغة الآلة، ويتم عبر برامج وسيطة كالمترجم Compiler، والمفسر 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 تتم 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 (1-1) يُوضح الفرق بين Compiler و Interpreter



مع العلم بأن لُغة **Python** تجمع بين المترجم والمفسر، حيث تُترجم التعليمات البرمجية أولًا إلى **Byte Code**[1]، ثم تُفسّرها وتنفذها.

١ Byte Code تمثيل وسيط للبرنامج يتم إنشاؤه بعد ترجمة التعليمات البرمجية، وتكون جاهزة للتنفيذ من قبل آلة افتراضية.

كتابة البرامج في Python

- يُمكن استخدام تطبيقات معالجات النصوص مثل **Notepad**، لكتابة التعليمات البرمجية بلغة **Python** ومن ثم حفظ الملف بامتداد **py.**
- يُمكن استخدام إحدى بيئات التطوير المتكاملة **Integrated Development Environment (IDE)**، والتي 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 سيتم استخدامه خلال هذا الكتاب.

بيئة التطوير المتكاملة PyCharm

- تحميل البرنامج.

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

تحميل برنامج PyCharm

- اختيار نظام التشغيل المناسب.

Windows macOS Linux

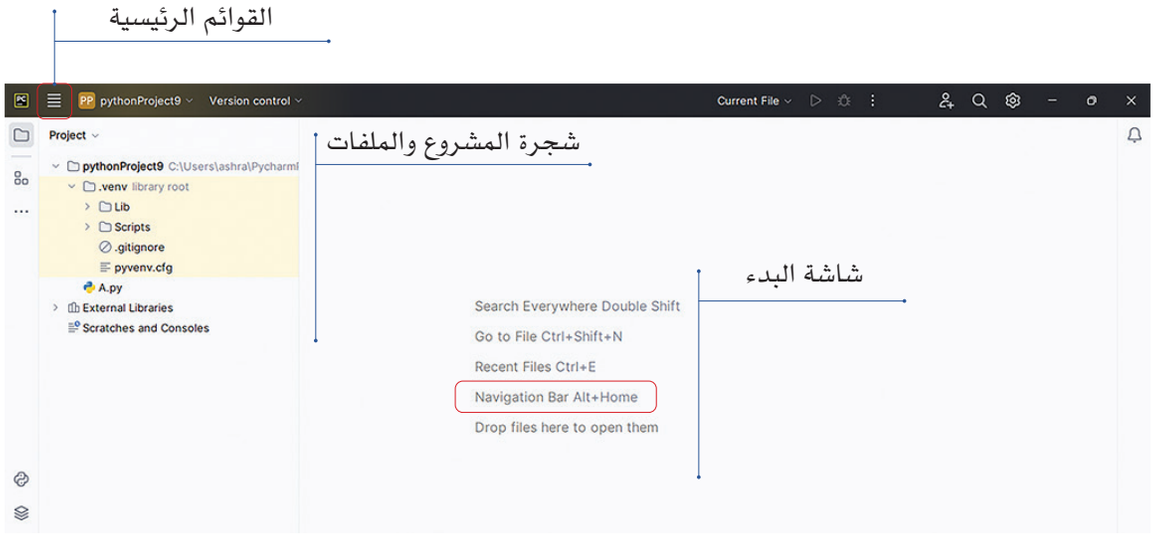
- اختيار إحدى النسخ التالية:

الأولى: PyCharm Professional وهي نسخة غير مجانية؛ متقدمة للمحترفين.

الثانية: PyCharm Community Edition وهي نسخة مجانية؛ تُستخدم للتعليم.

- تثبيت البرنامج.

نافذة بيئة التطوير المتكاملة PyCharm

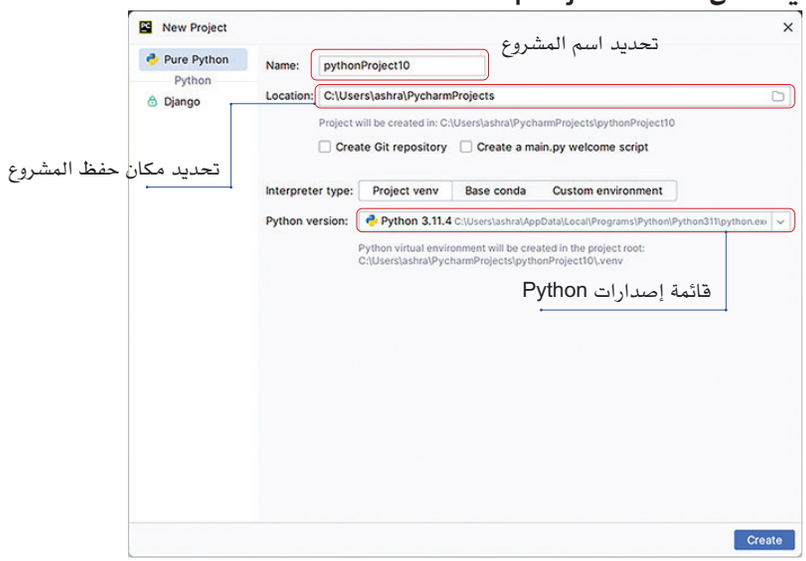


شكل (1-4) PyCharm Editor - شاشة البدء

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

إنشاء مشروع جديد

لإنشاء مشروع جديد، من قائمة File > new project

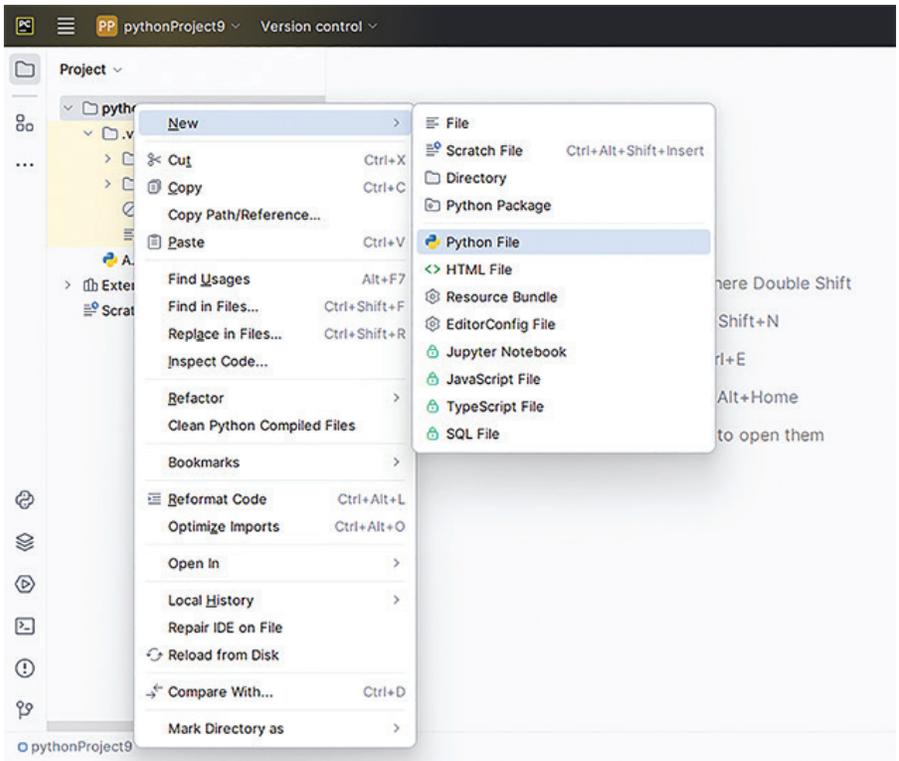


شكل (1-5) شاشة PyCharm Editor - New Project

إنشاء ملف Python جديد

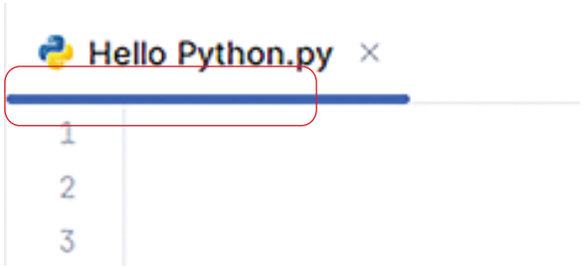
يُمكن أن يحتوي المشروع على العديد من ملفات التعليمات، ولإنشاء ملف Python جديد:

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



شكل (1-6) شاشة PyCharm Editor - New Python File

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



شكل (1-7) شاشة PyCharm Editor - File Name

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

شكل (1-8) شاشة PyCharm - 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")
```

حيث التعليمة print هي من الدوال المدمجة في لغة بايثون والتي عند تنفيذ البرنامج تطبع المحتويات بين القوسين في نافذة التشغيل RUN.
لاحظ كتابة النص بين علامتي تنصيص، والتي سوف نوضح وظيفتها لاحقًا.

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، يتم اختيار الملف المراد تنفيذه، أو اختيار Current File لتشغيل الملف الحالي، ثم الضغط على الأداة RUN الموضحة في الشكل التالي:

تظهر نافذة جديدة وبها نتيجة تنفيذ البرنامج كالتالي:

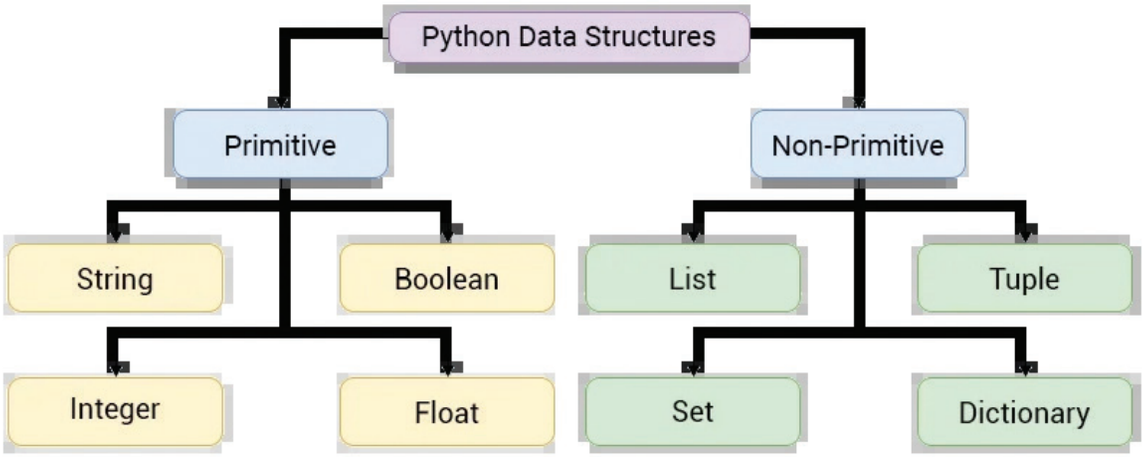
شكل (1-9) شاشة PyCharm Editor - RUN

هياكل البيانات Data Structure

طريقة لتنظيم وتخزين البيانات بشكل يسمح بالوصول إليها ومعالجتها بكفاءة. تُستخدم هياكل البيانات في جميع أنواع البرامج، من أبسطها إلى أكثرها تعقيدًا.

وتنقسم هياكل البيانات إلى قسمين أساسيين:

- أولًا: هياكل البيانات الأولية (Primitive)
- ثانيًا: هياكل البيانات غير الأولية (Non-Primitive)



الشكل (1-10) يُمثل بعض أنواع هياكل البيانات

هياكل البيانات الأولية (Primitive):

- البنية التي تسمح بتخزين قيم لنوع واحد فقط من البيانات.
- يعتمد حجم هياكل البيانات الأولية على نوعها.
- تُعد الأعداد الصحيحة والعشرية والسلاسل النصية (سلسلة من الحروف والأرقام والرموز الخاصة)، والبيانات المنطقية (True - False) من هياكل البيانات الأولية.

أنواع هياكل البيانات الأولية:

- الأعداد الصحيحة Integer.
- الأعداد العشرية Float.
- السلاسل النصية String.
- البيانات المنطقية Boolean.

هياكل البيانات غير الأولية Non-Primitive:

- البنية التي تسمح بتخزين قيم أنواع بيانات متعددة.
- حجم بنية البيانات غير ثابت.
- تعد القائمة والمجموعة والقاموس والصف والمصفوفة من هياكل البيانات غير الأولية.

أنواع هياكل البيانات غير الأولية

أ- القائمة List : تستخدم لتخزين مجموعة من القيم. يمكن أن تحتوي على أي نوع من البيانات، بما في ذلك الأعداد والسلاسل النصية، والقوائم الأخرى.

ب- المجموعة Set: تستخدم لتخزين مجموعة من القيم الفريدة **Unique** values. يمكن أن تحتوي عناصرها على أي نوع من البيانات.

ج- القاموس Dictionary: تستخدم لتخزين مجموعة من المفاتيح والقيم. ويمكن أن تكون القيم من أي نوع من البيانات.

د- الصف Tuple: تستخدم لتخزين مجموعة من القيم الغير قابلة للتعديل، حيث يمكن أن تحتوي على عناصر من أنواع مختلفة.

أوراق العمل

ورقة عمل (4):

برنامج حساب مساحة الدائرة

مشكلة البرنامج

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

الخوارزمية

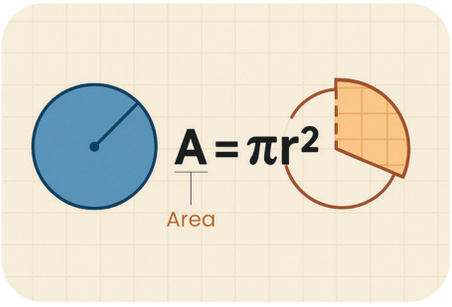
1. بداية البرنامج.
2. استقبال من المستخدم قيمة نصف قطر الدائرة.
3. حساب مساحة الدائرة مستعينًا بالمعادلة التالية:

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

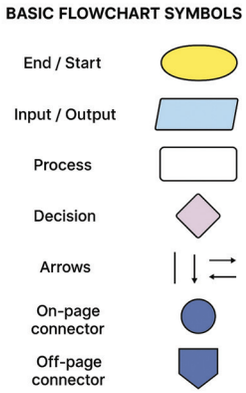
المطلوب:

خريطة التدفق

- ارسم خريطة التدفق المناسبة.



خريطة التدفق



ورقة عمل (5):

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.**
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:**

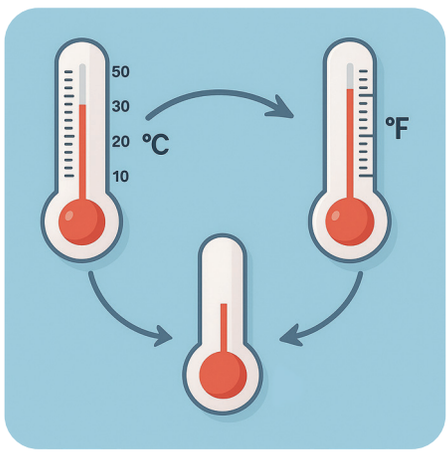
$$°F = (°C \times 1.8) + 32$$

6. **عرض درجة الحرارة بعد التحويل لوحدة القياس المطلوبة.**
7. **نهاية البرنامج**

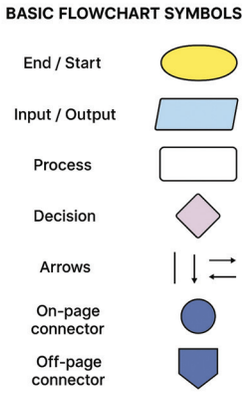
المطلوب:

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 ويطبّقها بشكل سليم عند كتابة الشيفرة.
- استخدام المتغيرات بفاعلية في تنفيذ العمليات البرمجية.
- توظيف الدالة المدمجة ()print في طباعة القيم والمخرجات على الشاشة
- استخدام الدالة ()input لاستقبال بيانات من المستخدم ضمن البرنامج.
- تحويل البيانات بين الأنواع المختلفة (مثل النصوص والأعداد الصحيحة والعشرية) بما يتناسب مع متطلبات البرنامج.

يمثل رمز الاستجابة السريعة QR رابط لملفات أوراق العمل، ومصادر التعلم

المتغيرات Variables

المتغير 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 المتغير أثناء تنفيذ البرنامج.

الإعلان عن المتغيرات Declaration

تتميز Python بالتعرف على نوع المتغير بشكل مباشر حسب القيمة المخصصة 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**	عددي عشري	تخزين قيم تحتوي على كسور 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.	var_num1 = 10
			var_num2 = var_num1 > 5 #True
			var_num3 = var_num1 < 5 #False

جدول (2-1) أمثلة على أنواع البيانات

كما يوجد أنواع أخرى من البيانات سيتم التعرف عليها في وقتها مثل القوائم والصفوف.

ملاحظة: تُستخدم # لإضافة التعليقات، ولا يُنفذها البرنامج كتعليمة برمجية.

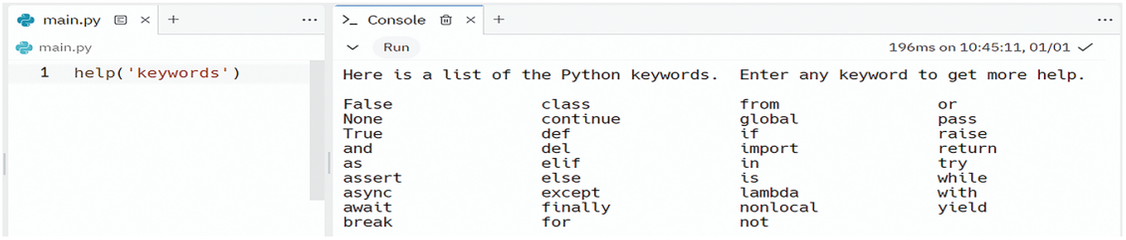
قواعد تسمية المتغيرات Naming Variables

يجب أن تتبع المتغيرات في Python قواعد معينة للتسمية:

- يجب أن يبدأ اسم المتغير (بحرف أو شرطة سفلية)، 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 السفلية (_).
- لا يمكن استخدام الكلمات المحجوزة keywords reserved لتسمية المتغيرات.

ملاحظات:

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



شكل (2-1) يوضح الكلمات المحجوزة في Python

دالة type():

تستخدم للتعرف على نوع المتغير.

مثال 1:

```
1  var_name = "Abdulla Rahman Al-Smait"
2  var_count = 29
3  print(type(var_name))
4  print(type(var_count))
```

Program output

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

ملاحظات:

المتغير الأول :var_name من نوع نصي<class 'str'> .

المتغير الثاني var_count : من نوع عدد صحيح<class 'int'>.

دالة الإخراج print

دالة مدمجة في Python، تُستخدم لعرض البيانات والمخرجات على الشاشة. يمكن لها التعامل مع مختلف أنواع البيانات، كالنصوص، والأرقام، والقوائم، 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 قيم المتغيرات

```
1    x = "Free"
2    y = "Kuwait"
3    print(x,y)
```

Program output

```
Free Kuwait
```

مثال 3:

ربط النصوص، وقيم المتغيرات.

```
1    age= 18
2    print("Your age = ", age)
```

Program output

```
Your age = 18
```

العمليات الحسابية

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

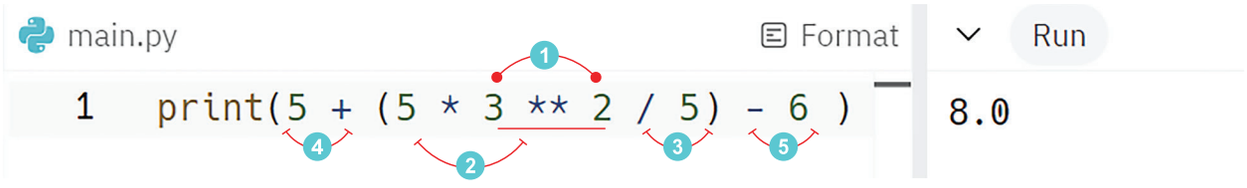
الرمز	العمليات الحسابية	مثال	
+	الجمع Addition	var1 = 23 var2 = 5 print(var1+var2)	28
-	الطرح Subtraction	var1 = 23 var2 = 5 print(var1-var2)	18
*	الضرب Multiplication	var1 = 23 var2 = 5 print(var1*var2)	115
/	القسمة Division	var1 = 23 var2 = 5 print(var1/var2)	4.6
//	ناتج القسمة الصحيح بدون الكسور Floor division	var1 = 23 var2 = 5 print(var1 // var2)	4
%	باقي القسمة Modulus	var1 = 23 var2 = 5 print(var1 % var2)	3
**	رفع العدد إلى أُس محدد Exponentiation	var2 = 5 print(var2 ** 2) print(var2 ** 3)	25 125

جدول (2-2) 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 مدمجة 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])

سلسلة نصية 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("Hello, "+ name + "!")
```

Program output

```
Enter your name: Ahmed
Hello, Ahmed!
```

تفسير النص البرمجي:

- يعرض رسالة للمستخدم: Enter your name.
- ينتظر البرنامج حتى يُدخل المستخدم اسمًا ويضغط على مفتاح Enter.
- تخصيص القيمة المدخلة إلى المتغير name.
- طباعة العبارة Hello, [Name].

ملاحظات:

- 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:

إدخال قيم عددية لمتغيرين باستخدام دالة input، وإجراء عملية الجمع.

```
1  num1 = input("Enter  First number: ")
2  num2 = input("Enter  Second number: ")
3  print(num1 + num2)
```

Program output

```
Enter First number: 10
Enter First number: 12
1012
```

لاحظ أن عملية جمع الأعداد لم تُنفذ بشكل صحيح، حيث تم التعامل مع القيم المُدخلة على أنها نصوص، وتم ربطها بجوار بعضها البعض بدلاً من جمعها حسابيًا.

مثال 7:

إجراء عملية حسابية بجمع المتغيرين الأول والثاني كأرقام.

```
1    num1 = input("Enter first number: ")
2    num1 = input("Enter second number: ")
3    num1 = int(num1)
4    num1 = int(num2)
5    print(num1 + num2)
```

Program output

Enter First number: 10

Enter First number: 12

22

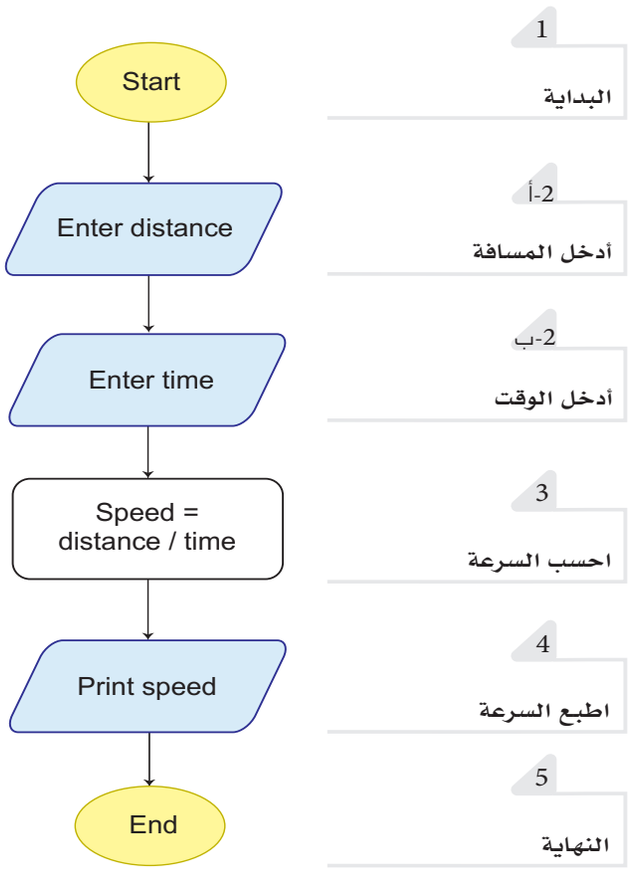
كما يمكن التحويل مباشرة أثناء عملية الإدخال كالتالي:

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

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

مثال تطبيقي 8:

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 (2-2) خريطة تدفق مثال تطبيقي

1- يُمثل بداية البرنامج.
2- يطلب من المستخدم إدخال قيمتي المسافة والزمن، ولتنفيذ ذلك باستخدام دالة **input()**.

أ. التعليمة البرمجية لإدخال المسافة distance.

```
distance = input("Enter the distance (km): ")
```

ب. التعليمة البرمجية اللازمة لإدخال قيمة الزمن time.

```
time = input("Enter the time (hours): ")
```

- تظهر رسالة للمستخدم عند تشغيل البرنامج تطلب منه إدخال قيمة المسافة، ثم رسالة أخرى تطلب منه إدخال قيمة الزمن.
- تستقبل دالة input() جميع القيم كسلاسل نصية.

3- حساب السرعة تُستخدم معادلة السرعة القياسية التالية:

speed = distance / time

- **التعليمة البرمجية اللازمة لحساب السرعة:**

```
speed = float(distance) / float(time)
```

- يتم الإعلان عن متغير جديد باسم **speed**، بتخصيص قيمه له وهي(ناتج قسمة المسافة على الزمن).
- المعامل (/) يمثل عملية القسمة، راجع جدول العمليات الحسابية.

تم استخدام دالة **float()** لتحويل قيمة المسافة والزمن المُدخلة من نصية إلى عشرية

4- طباعة السرعة.
5- نهاية البرنامج.

التعليمات البرمجية:

```
1  distance = input("Enter the distance (km): ")
2  time = input("Enter the time (hours): ")
3  speed = float(distance) / float(time)
4  print("Speed = ", speed, "km/h")
```

Program output

```
Enter the distance (km): 151
Enter the time (hours): 2
Speed = 7.5 km/h
```

أوراق العمل

ورقة عمل (6):

برنامج رحلة إلى أبراج الكويت

مشكلة البرنامج

حساب تكلفة رحلة مدرسية لأبراج الكويت بمعلومية تكلفة المواصلات، دخول الأبراج، والوجبة.

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.

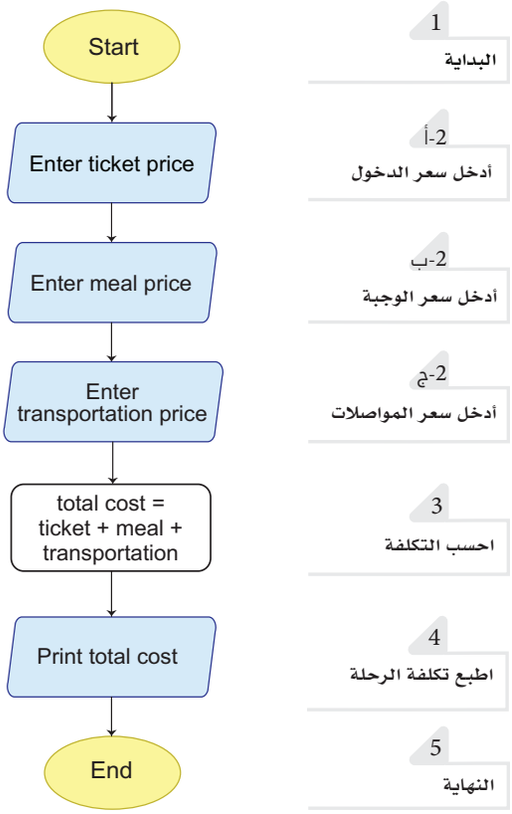
المطلوب:

خريطة التدفق

- ادرس خريطة التدفق المقابلة.

البرنامج

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



شكل (2-3) خريطة تدفق 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):

برنامج حساب قوة الشغل المبذول

مشكلة البرنامج

حساب الشغل المبذول بمعلومية القوى المؤثرة، والإزاحة في اتجاهها.

الخوارزمية

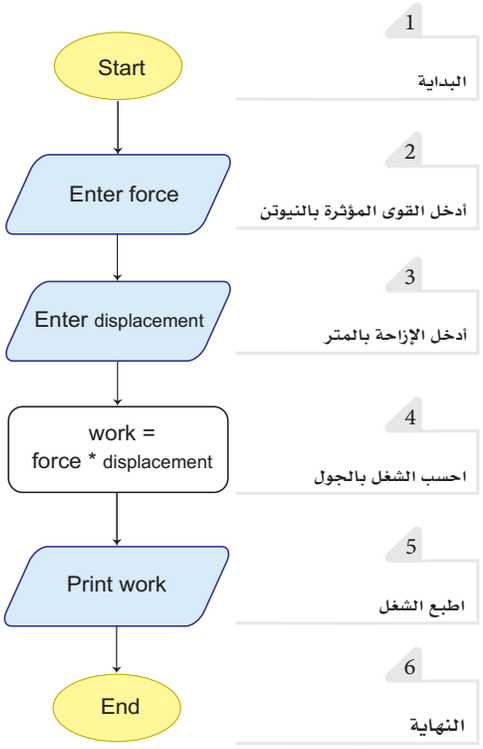
1. بداية البرنامج.
2. استقبال من المستخدم قيمة القوى المؤثرة Force بوحدة النيوتن.
3. استقبال من المستخدم قيمة الإزاحة Displacement في اتجاهها بوحدة المتر.
4. حساب قيمة الشغل Work بوحدة الجول، باستخدام المعادلة:

الشغل W = القوة F * الإزاحة D

5. طباعة ناتج المعادلة (الشغل) على الشاشة.
6. نهاية البرنامج.

المطلوب:

- افتح الملف work.py.
- ادرس خريطة التدفق المقابلة.



شكل (2-4) خريطة تدفق الشغل

البرنامج

- أكمل التعليمات البرمجية اللازمة ليعمل البرنامج بشكل صحيح.

علمًا بأن:

- المتغير force يمثل قيمة القوى المؤثرة.
- المتغير displacement يمثل قيمة الإزاحة في اتجاه القوة.
- المتغير work يمثل الشغل ويساوي حاصل ضرب قيمة المتغيرين force و displacement.

استكمل التعليمات البرمجية

```
1  # Enter Force value in Newtons.
2  force = float(input("Enter Force in Newtons: "))
3  # Enter Displacement value in Meters.
4  displacement = .....
5  # Calculate Work in Joules.
6  work =.....
7  # Print Work Value
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 ويحوّلها حسب الحاجة لتنفيذ تعليمات برمجية.
- إنشاء برامج بسيطة تتضمن مدخلات ومخرجات تعتمد على معالجة السلاسل النصية.

يمثل رمز الاستجابة السريعة QR رابط
لملفات أوراق العمل، ومصادر التعلم

السلاسل النصية String

هي مجموعة من الأحرف والأرقام والرموز تُميز بعلامات اقتباس، تخزن السلاسل في متغيرات يتم التعامل معها كقيمة نصية، وهي 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.

(1) العمليات الأساسية على السلاسل النصية:

تعريف، تخصيص، وطباعة السلاسل النصية:

يتم تعريف، وتخصيص السلاسل النصية بعلامات اقتباس فردية ' ' أو مزدوجة " ".

مثال 1:

إنشاء سلسلة نصية باستخدام علامتي اقتباس فردية.

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

Program output

```
Free Kuwait
```

مثال 2:

إنشاء سلسلة نصية باستخدام علامتي 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 (3-1) 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 (3-2) 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 [x : y].

الصيغة	الوصف
str[x : y]	تُعيد النص بداية من الموقع x وحتى الموقع ما قبل y
str[: y]	تُعيد النص بداية من الموقع 0 وحتى الموقع ما قبل y
str[x :]	تُعيد النص بداية من الموقع x وحتى نهاية السلسة النصية

جدول (3-3) يوضح أمثلة على عملية القطع

مثال 5:

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

```
1  var_x = 'Abdul-Rahman Hamoud Al-Sumait'
2  print('First Name: ', var_x[ : 12])
3  print('Middle Name: ',var_x[13 : 19])
4  print('Last Name: ',var_x[20 : ])
```

Program output

```
First Name: Abdul-Rahman
Middle Name: Hamoud
Last Name: Al-Sumait
```

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 (3-4) 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:

يستخدم معامل عملية الجمع (+) لإضافة سلسلتين نصيتين لإنشاء سلسلة نصية جديدة.

مثال 7:

```
1  var_x='Free'
2  var_y = 'kuwait'
3  print(var_x + ' ' + var_y)
```

Program output

```
Free Kuwait
```

لاحظ:

يُمكن استخدام ' '، لإضافة مسافة بين المتغيرين.

مثال 8:

```
print("Liberation" + "Day: " + "26" + " Feb")
```

Program output

LiberationDay: 26 Feb

لاحظ:

تم إضافة مسافة بعد كلمة **Day:** وقبل كلمة **Feb:**.

تكرار السلاسل النصية:

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

مثال 9:

```
print("#" * 10)
```

Program output

##########

المقارنة Comparison:

مقارنة السلاسل النصية باستخدام معامل المقارنة == ، != .

مثال 10:

```
prophet = 'Muhammad'
prophet_input = input('Who is the last prophet? ')
print(prophet_input == prophet)
```

Program output

في حال تساوي القيمة المُدخلة للمتغير prophet_input، مع قيمة المُتغير prophet يتم طباعة True، وفي حال عدم التساوي يتم طباعة False.

مثال 11 :

```
1  prophet = 'Muhammad'
2  prophet_input = input('Who is the last prophet? ')
3  print('Correct answer!' * (prophet_input == prophet) + 'Wrong answer!' * (prophet_input != prophet))
```

Program output

إذا كانت المقارنة صحيحة، (prophet_input == prophet) يُرجع True، الذي يُحوّل إلى 1،

```
print("Correct answer!‘ * (1) + "Wrong answer!" * (0))   # Correct answer!
```

إذا كانت المقارنة خاطئة، (prophet_input != prophet) يُرجع False، الذي يُحوّل إلى 0.

```
print("Correct answer!‘ * (0) + "Wrong answer!" * (1))   # Wrong answer!
```

(2) دوال السلاسل النصية:

الدوال هي مجموعة من الأوامر الجاهزة تُنفذ مهامًا معينة عند استدعائها.

- الدوال المدمجة (Built-in Functions):

هي دوال تُنفذ مهامًا معينة وتتوفر في Python بشكل افتراضي، ومن أمثلتها:

دالة len( ):

تُستخدم للحصول على طول السلسلة النصية.

مثال 12 :

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

Program output

```
11
```

- **الأساليب المُدمجة (Built-in Method):**

هي دوال تُنفذ مهامًا معينة، مُرتبطة بكائن object، ويفصل بينهما (.) عند الاستدعاء، ومن أمثلتها :

دالة ( )capitalize:

تُستخدم لتحويل الحرف الأول إلى حرف كبير . string.capitalize( )

مثال 13 :

```
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. string.upper( )

مثال 14 :

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

Program output

```
HELLO, CYBERSECURITY!
```

دالة ( )lower:

تُحول جميع الأحرف في السلسلة النصية إلى أحرف صغيرة. string.lower()

مثال 15:

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

Program output

```
computer science
```

دالة ( )find:

تُستخدم لإرجاع رقم فهرس لأول ظهور لسلسلة فرعية في السلسلة المحددة. 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 16:

```
1  text = 'Free Kuwait'
2  index_chr = text.find('K')
3  index_word = text.find('Kuwait')
4  index = text.find('Emirates')
5  print(index_chr)
6  print(index_word)
7  print(index)
```

Program output

```
5
5
-1
```

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

دالة ( )replace:

دالة تستخدم لاستبدال نص بآخر في السلسلة النصية.

string.replace(oldvalue, newvalue)

مثال 17:

```
text = 'The computter is new. I like my computter.'
new_text = text.replace('computter', 'computer')
print(new_text)
```

Program output

The computer is new. I like my computer.

التعليمة f-string:

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

مثال 18:

```
name = 'Muhammed'
age = 63
info = f'Name: {name} and Age: {age}'
print(info)
```

Program output

Name: Muhammed and Age: 63

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

```
print(f'Name: {name} and Age: {age}')
```

يُمكن التحكم في تنسيق الأرقام العشرية.

مثال 19:

```
x = 11
y = 3
z = x / y
print(f'{x} / {y} = {z}')
print(f'{x} / {y} = {z:.3f}')
```

Program output

```
11 / 3 = 3.6666666666666665
11 / 3 =  3.667
```

دالة ( )int:

تُستخدم لتحويل السلاسل النصية إلى أعداد صحيحة:

درست أن:
دالة ()input تُرجع البيانات المُدخلة كسلسلة نصية string، ولتحويلها إلى integer تستخدم دالة ()int.

مثال 20:

```
n1 = input('Enter First Number: ')
n2 = input('Enter Second Number: ')
print(int(n1) + int(n2))
```

Program output

```
Enter First Number: 4
Enter Second Number: 6
10
```

ويُمكن استخدام الطريقة التالية:

مثال 21:

```
1  n1 = int(input('Enter First Number: '))
2  n2 = int( input('Enter Second Number: '))
3  print(n1 + n2)
```

Program output

```
Enter First Number: 4
Enter Second Number: 6
10
```

لاحظ: مثال 7 جمع السلاسل النصية.

دالة float( ):

تُستخدم لتحويل السلاسل النصية (أرقام **numeric**)، أو الأعداد الصحيحة إلى أعداد عشرية.

مثال 22:

```
1  n1 = float(input('Enter First Number: '))
2  n2 = float( input('Enter Second Number: '))
3  print(n1 + n2)
```

Program output

```
Enter First Number: 5.6
Enter Second Number: 4
9.6
```

دالة count( ):

تُستخدم لإيجاد عدد مرات ظهور سلسلة نصية.

مثال 23:

```
sura_alnas = '''
بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ
قُلْ أَعُوذُ بِرَبِّ النَّاسِ
مَلِكِ النَّاسِ
إِلَهِ النَّاسِ
مِن شَرِّ الْوَسْوَاسِ الْخَنَّاسِ
الَّذِي يُوَسْوِسُ فِي صُدُورِ النَّاسِ
مِنَ الْجِنَّةِ وَالنَّاسِ
'''
count = sura_alnas.count('النَّاس')
print(f"The word 'النَّاس' appears {count} times.")
```

Program output

The word 'النَّاس' appears 5 times.

ملاحظة

تستخدم علامات اقتباس ثلاثية """ """ لكتابة سلاسل نصية متعددة الأسطر.

دالة center( ):

تُعيد أحرف السلسلة مزاحة نحو الوسط ضمن سلسلة ذات طول محدد.

مثال 24:

```
var_x = 'Security'
print(var_x.center(16, '#'))
```

Program output

####Security####

عند استخدام الدالة center()، إذا كانت القيمة المحددة لطول السلسلة المحددة أقل من أو تساوي طول النص الأصلي، فلن تتم إضافة أي رموز للتوسيط وسيُعرض النص كما هو.

أوراق العمل

ورقة عمل (8)

تأمين كلمة المرور

مشكلة البرنامج

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

الخوارزمية

1. بداية البرنامج.
2. طلب من المستخدم إدخال كلمة المرور.
3. تخزين الكلمة في متغير (مثلاً: password).
4. إنشاء متغيرًا جديدًا لتخزين الكلمة المعكوسة (مثلاً: 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:

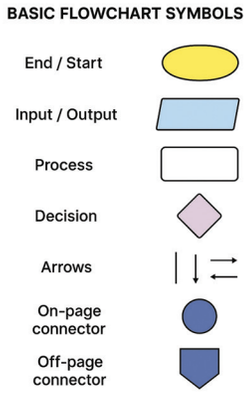
خريطة التدفق

- ارسم خريطة التدفق المناسبة.

البرنامج

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

خريطة التدفق



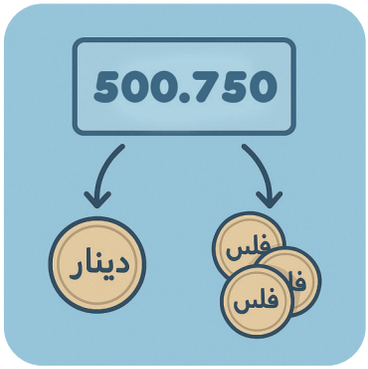
ورقة عمل (9)

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

المطلوب:

خريطة التدفق

- ارسم خريطة التدفق المناسبة.

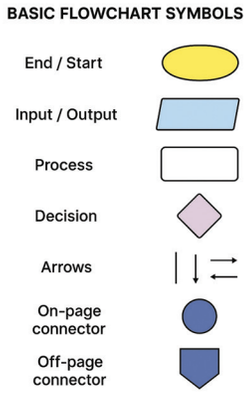
البرنامج

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

```
1  amount = input("Enter amount e.g (500.750) : ")
2  position= .....
3  kd = .....
4  fils = .....
5  print(f"{kd} KD And {fils} fils")
```

- اختبر البرنامج، وتأكد من خلوه من الأخطاء.

خريطة التدفق



ورقة عمل (10)

تقطيع السلسلة النصية

مشكلة البرنامج

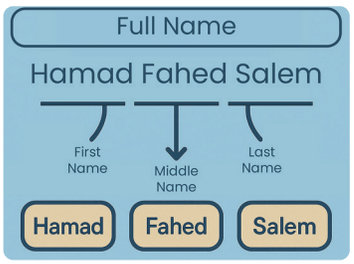
تقسيم الاسم الكامل full name (باللغة الانجليزية) للحصول على الاسم الأول، الأوسط، والأخير.

الخوارزمية

1. بداية البرنامج.
2. استقبال الاسم الكامل (ثلاثي) باللغة الإنجليزية من المستخدم، وحفظه في متغير full_name.
3. طباعة عدد أحرف الاسم الكامل 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، الثاني، والثالث وحفظهم في متغيرات.
7. طباعة:
   - الاسم الأول.
   - الاسم الثاني.
   - الاسم الثالث.

   مع تحويل أول حرف فقط من كل جزء في الاسم الكامل إلى حرف كبير Capital letter.
8. نهاية البرنامج.

المطلوب:



خريطة التدفق

- ارسم خريطة التدفق المناسبة.

البرنامج

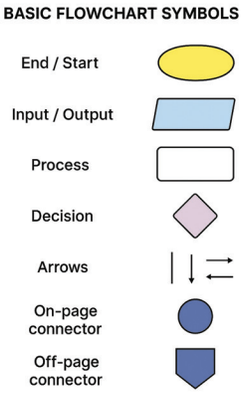
- افتح الملف name_entry_system.py.
- أكمل التعليمات البرمجية اللازمة ليعمل البرنامج بشكل صحيح.

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)
9   print(second_name)
10  print(third_name)
11  .....
```

- اختبر البرنامج، وتأكد من خلوه من الأخطاء.

خريطة التدفق



برمجة Python

الشروط Conditions

نواتج التعلم

- توضيح مفهوم الشروط ودورها في اتخاذ القرارات داخل البرامج.
- تمييز بين عمليات المقارنة (مثل: == ، > ، <) والعمليات المنطقية (and، or، not) ويحدد استخداماتها.
- توظيف التعليمات الشرطية else ،if، 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.

يمثل رمز الاستجابة السريعة QR رابط
لملفات أوراق العمل، ومصادر التعلم

الشروط Conditions

بُنية تَحكم تستخدم لاتخاذ القرارات وتنفيذ التعليمات بناءً على تحقق شرط محدد، والتحكم في مسار البرنامج، وفق العمليات التالية:

عمليات المقارنة Comparison Operators

العمليات التي تقارن بين عاملين (متغيرات، قيم، ...)، والنتيجة تكون من نوع boolean إما True أو False .

المعامل	الوصف
==	يساوي
!=	لا يساوي
<	أصغر من
>	أكبر من
<=	أصغر من أو يساوي
>=	أكبر من أو يساوي

جدول (4-1) معاملات المقارنة

لاحظ:

معامل = يُستخدم لتخصيص قيمة للمتغير Assignment.

معامل == يُستخدم للمقارنة بين عنصرين Comparison.

مثال 1:

```
1  # Assignment
2  var_number = 50
3  # Comparison
4  print(var_number == 70) # false
5  print(var_number == 50) # 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 (4-2) جدول نتائج عمليات المقارنة

العمليات المنطقية Logical Operators

تُستخدم لإجراء عمليات منطقية على قيم المتغيرات:

- **العملية المنطقية and:** تُرجع **True** إذا كانت كلتا القيمتين المنطقيتين صحيحتين، وإلا تُرجع **False**.

مثال 2:

```
1  x = True
2  y = True
3  print(x and y)
```

Program output

```
True
```

مثال 3:

```
1  x = True
2  y = False
3  print(x and y)
```

Program output

```
False
```

- **العملية المنطقية or:** تُرجع **True** إذا كانت أي من القيمتين المنطقيتين صحيحتان، وإلا تُرجع **False**.

مثال 4:

```
1  x = True
2  y = False
3  print(x or y)
```

Program output

```
True
```

• **العملية المنطقية NOT:** تُرجع **False** إذا كانت القيمة المنطقية صحيحة، وإلا تُرجع **True**.

مثال 5:

```
1   x = True
2   y = False
3   print(not x)
4   print(not y)
```

Program output

False

True

مثال 6:

لكتابة شرط يتحقق إذا كانت درجة الطالب تتراوح بين 50 و 100.

```
1   score = 80
2   print(score >= 50 and score <= 100)  #print(50 <= score <= 100)
```

Program output

True

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 (3-4) العمليات المنطقية

التعليمات الشرطية Conditional Statements

يمكن استخدام التعليمات الشرطية **if** و **elif** و **else** لاتخاذ القرارات وتنفيذ كتلة من التعليمات البرمجية بناءً على نتائج تلك القرارات.

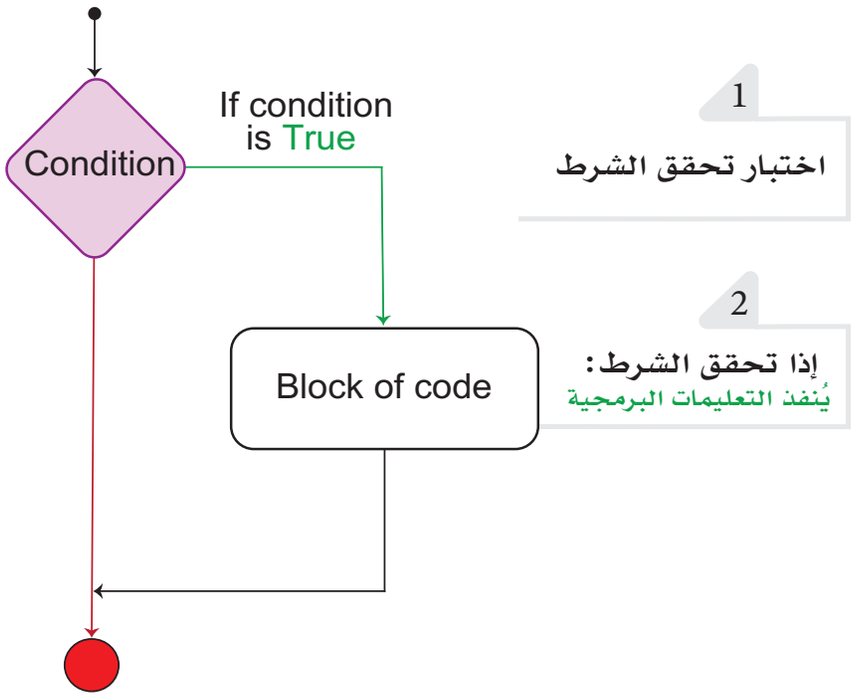
الكتلة البرمجية **(Block of code)**: هي سطر أو أكثر من التعليمات البرمجية، لها نفس المسافة البادئة **(indentation)**.

التعليمة الشرطية if

تُستخدم التعليمة الشرطية **if** لاختبار شرط منطقي، وتنفيذ تعليمات برمجية إذا تحقق الشرط.

الصيغة العامة Syntax:

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



شكل **(4-1)** خريطة تدفق الصيغة العامة **if**

لاحظ :

يستخدم Python المسافة البادئة بينما تستخدم بعض لغات البرمجة الأخرى الأقواس والعبارات للدلالة على بداية ونهاية الكتل البرمجية
تبدأ الكتلة البرمجية بعد علامة (:) colon وتنتهي عندما تنتهي المسافة البادئة Indentation.

مثال 7 :

برنامج يختبر ما إذا كانت الإجابة المُدخلة صحيحة بإظهار رسالة محددة.

```
1  password = input('Enter your password: ')
2  if password == 'admin' :
3      print('Correct Password')
```

مثال 8 :

برنامج يختبر ما إذا كانت الإجابة المُدخلة غير صحيحة بإظهار رسالة محددة.

```
1  prog_lang = input('What is your favorite programming language? ')
2  if prog_lang != "python" :
3      print("Try Python , you'll love it")
```

مثال 9 :

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

```
1  student_score= int(input('Enter student score: '))
2  if student_score >= 90 :
3      print('Excellent')
4  print('Good luck')
```

ملاحظات :

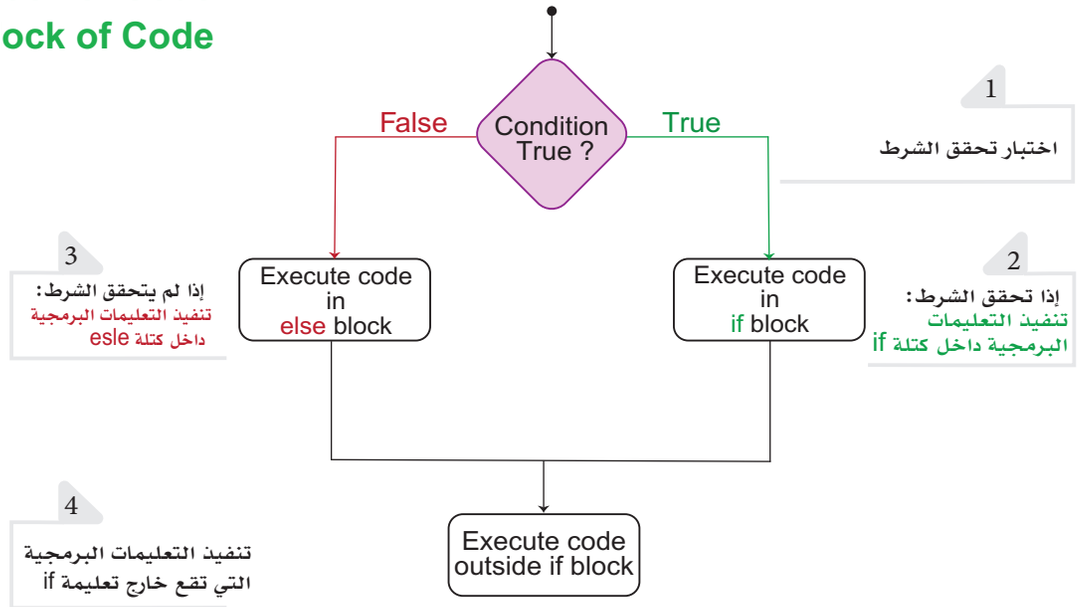
- عدم وجود مسافة بادئة للتعليمة البرمجية للسطر 4 يدل على أنها خارج نطاق الكتلة البرمجية للتعليمة الشرطية if.
- إذا تحقق الشرط سيطبع عبارة Excellent ثم عبارة Good luck.
- إذا لم يتحقق الشرط سيتجاهل تعليمة if ، وينتقل للتعليمة البرمجية التالية print ليطبع عبارة Good luck.

التعليمة الشرطية else

تستخدم لتحديد ما يجب تنفيذه إذا لم يتحقق الشرط.

الصيغة العامة Syntax:

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



شكل (2-4) خريطة تدفق الصيغة العامة else

مثال 10:

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

```
1  password = input('Enter your password: ')
2  if password == 'admin':
3      print('Correct Password')
4  else:
5      print('Worng Password')
```

مثال 11 :

برنامج يُظهر تقدير الطالب (Excellent) إذا الدرجة أكبر أو يساوي 90.

```
score = int(input('Enter exam score' ))
if score >= 90:
    print('You got an excellent grade')
else:
    print('Work hard to get an excellent grade')
```

Program output

```
Enter exam score 90
You got an excellent grade
```

مثال 12 :

برنامج يُظهر نتيجة الطالب (ناجح / راسب) بناءاً على الدرجة المدخلة.

```
score= int(input('Enter student score: ' ))
if score >= 50:
    print(' Congratulations, you succeeded ! ')
else:
    print('Sorry, you failed ')
```

Program output

```
Enter student score: 49
Sorry, you failed
```

مثال 13 : إثرائي (nested if)

تطوير البرنامج السابق للتأكد من أن الدرجة المُدخلة تتراوح بين صفر و 100 ، وتُظهر رسالة خطأ عند إدخال درجة غير ذلك

```
score = int(input('Enter student score: '))
if score >= 0 and score <= 100:
    if score >= 50:
        print('Congratulations, you succeeded!')
    else:
        print('Sorry, you failed')
else:
    print('Wrong number!')
```

Program output

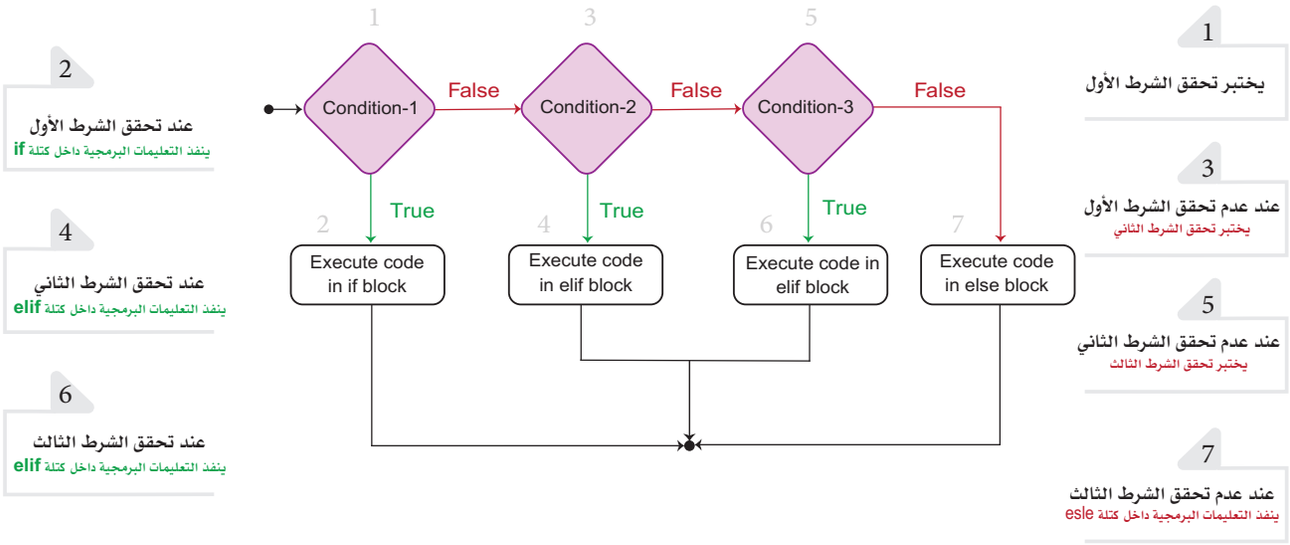
```
Enter student score: 98
Congratulations, you succeeded!
```

التعليمة الشرطية elif

وهي اختصار لجملة (else if) وتستخدم لإضافة شروط متعددة إلى التعليمة الشرطية if. ويتم تنفيذ النص البرمجي الموجود داخل كتلة التعليمات البرمجية إذا تحقق أحد هذه الشروط.

الصيغة العامة Syntax :

```
if condition1:
    Block of code condition1 is true.
elif condition2:
    Block of Code condition2 is true.
elif condition3:
    Block of Code condition3 is true.
...
else:
    Block of Code all conditions are false.
```



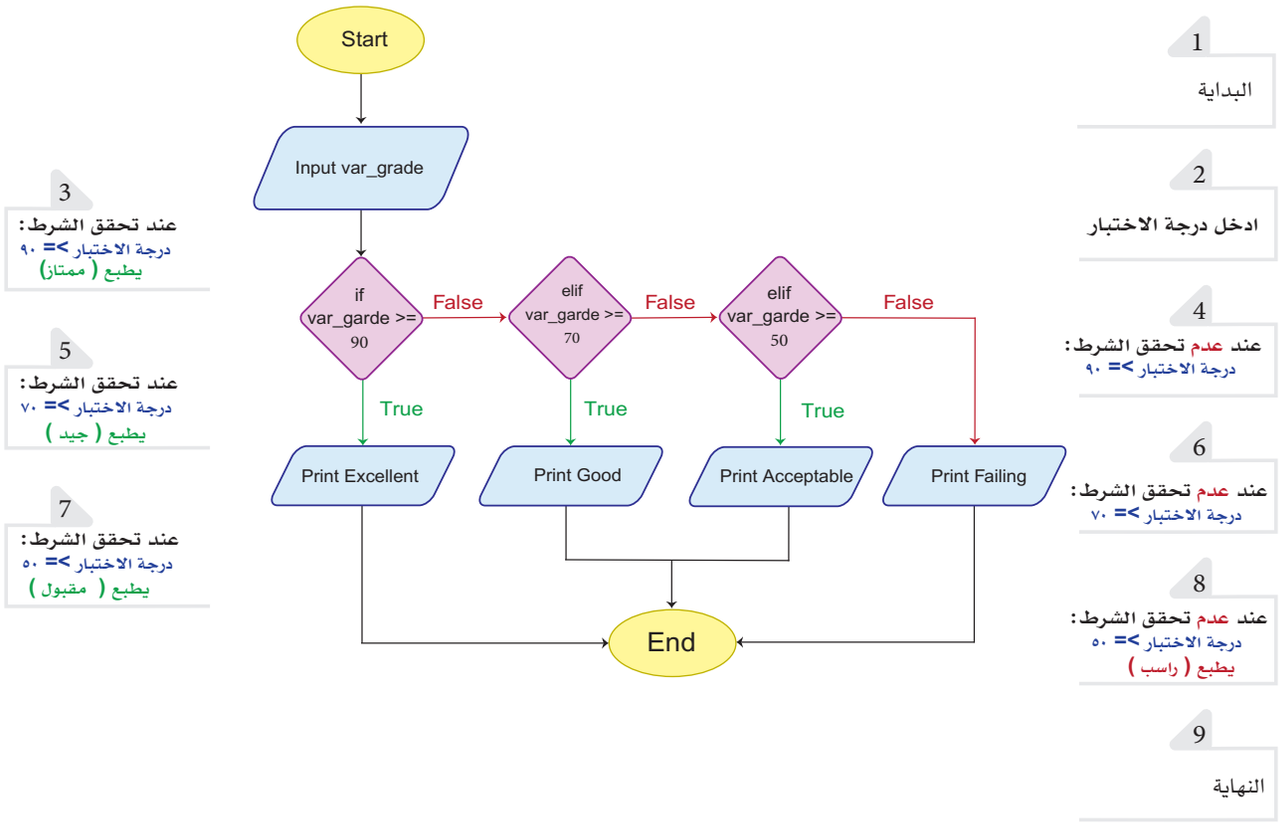
شكل (3-4) خريطة تدفق الصيغة العامة if ... elif

مثال 14 :

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

```
student_score = int(input('Enter student score: '))
if student_score >= 90 :    #student_score >= 90 and student_score <= 100
    print('Excellent grade !')
elif student_score >= 70: #student_score >= 70 and student_score <= 89
    print('Good grade !')
elif student_score >= 50:    #student_score >= 50 and student_score <= 69
    print('Acceptable grade !')
else:
    print('You have failed !')
```

في المثال السابق 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، وإذا كان ترتيب الشروط غير صحيح، يؤدي إلى نتائج غير صحيحة



شكل (4-4) خريطة تدفق برنامج تقدير الطالب

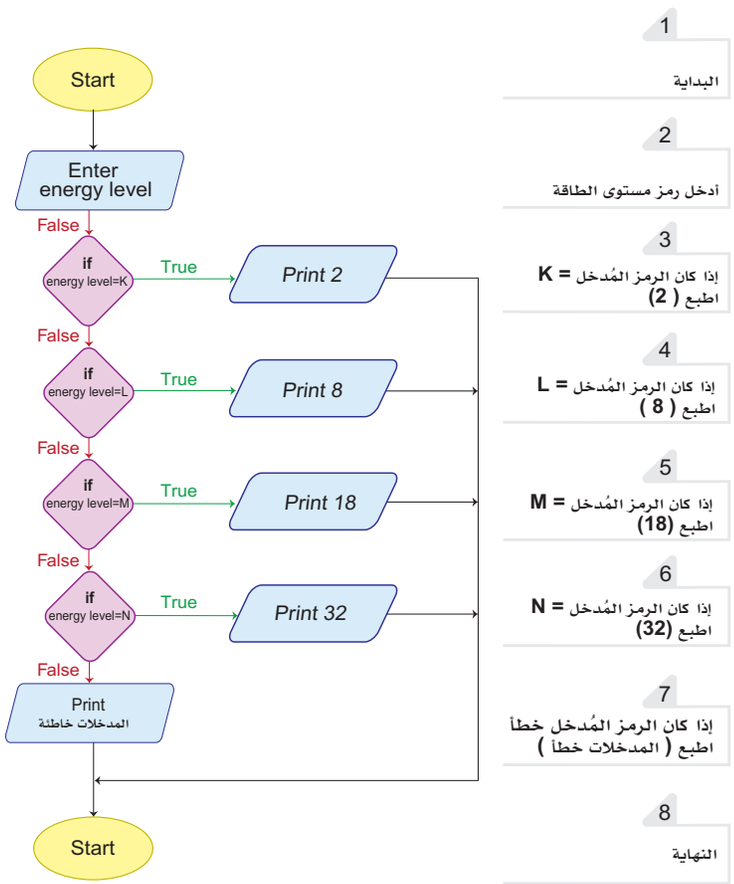
مثال تطبيقي 15 :

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

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

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

ادرس خريطة التدفق التالية ثم افتح الملف (energylevel.py)، واستكمل البرنامج المقترح والذي يُظهر العدد الأقصى للإلكترونات حسب رمز مستوى الطاقة، ليعمل بشكل صحيح.



شكل (4-5) خريطة تدفق برنامج التعرف على العدد الأقصى للإلكترونات

energylevel.py

```
1   energylevel = input("Enter energy level code: ")
2   if energylevel == "K":
3       print('Energy level K -> ',2)
4   elif energylevel == "L":
5   ..........
6   elif energylevel == "M":
7       print('Energy level M -> ',18)
8   ..........
9       print('Energy level N -> ',32)
10  ..........
11      print("Invalid input.")
```

طور البرنامج ليستقبل الحرف الأول من النص المُدخل ويحوله إلى حرف كبير.

.....

أوراق العمل

ورقة عمل (11):

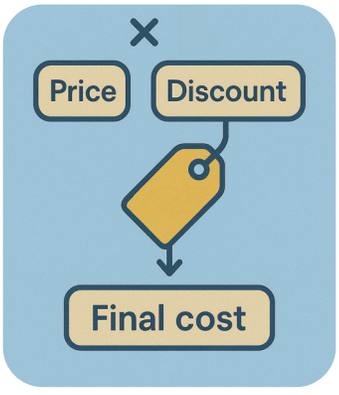
برنامج حساب الخصومات على المنتجات

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:

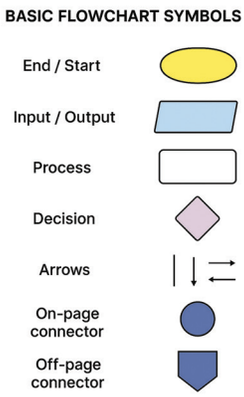
خريطة التدفق

- ارسم خريطة التدفق المناسبة.

البرنامج

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

خريطة التدفق



ورقة عمل (12):

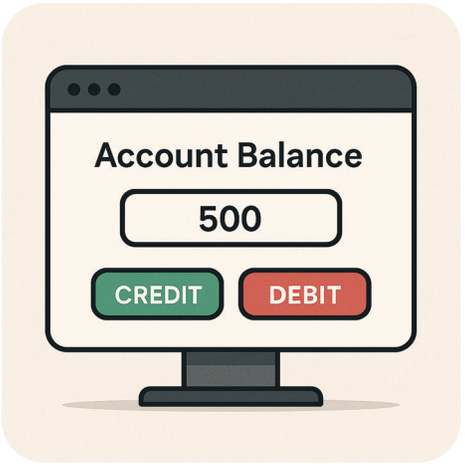
برنامج لمعرفة حالة الرصيد الائتماني

مشكلة البرنامج

تصميم برنامج يتحقق من حالة رصيد حساب المستخدم.

الخوارزمية

1. بداية البرنامج.
2. استقبال من المستخدم قيمة الرصيد كرقم عشري.
3. التحقق من قيمة الرصيد:
   - إذا كان الرصيد يساوي صفرًا:
     - طباعة "لا يوجد رصيد."
   - إذا كان الرصيد أكبر من صفر:
     - طباعة "رصيد دائن [الرصيد]."
   - إذا كان الرصيد أقل من صفر:
     - طباعة "رصيد مدين [الرصيد]."
4. نهاية البرنامج.



المطلوب:

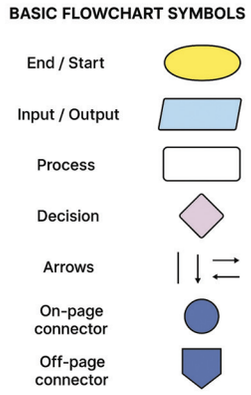
خريطة التدفق

- ارسم خريطة التدفق المناسبة.

البرنامج

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

خريطة التدفق



ورقة عمل (13):

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.

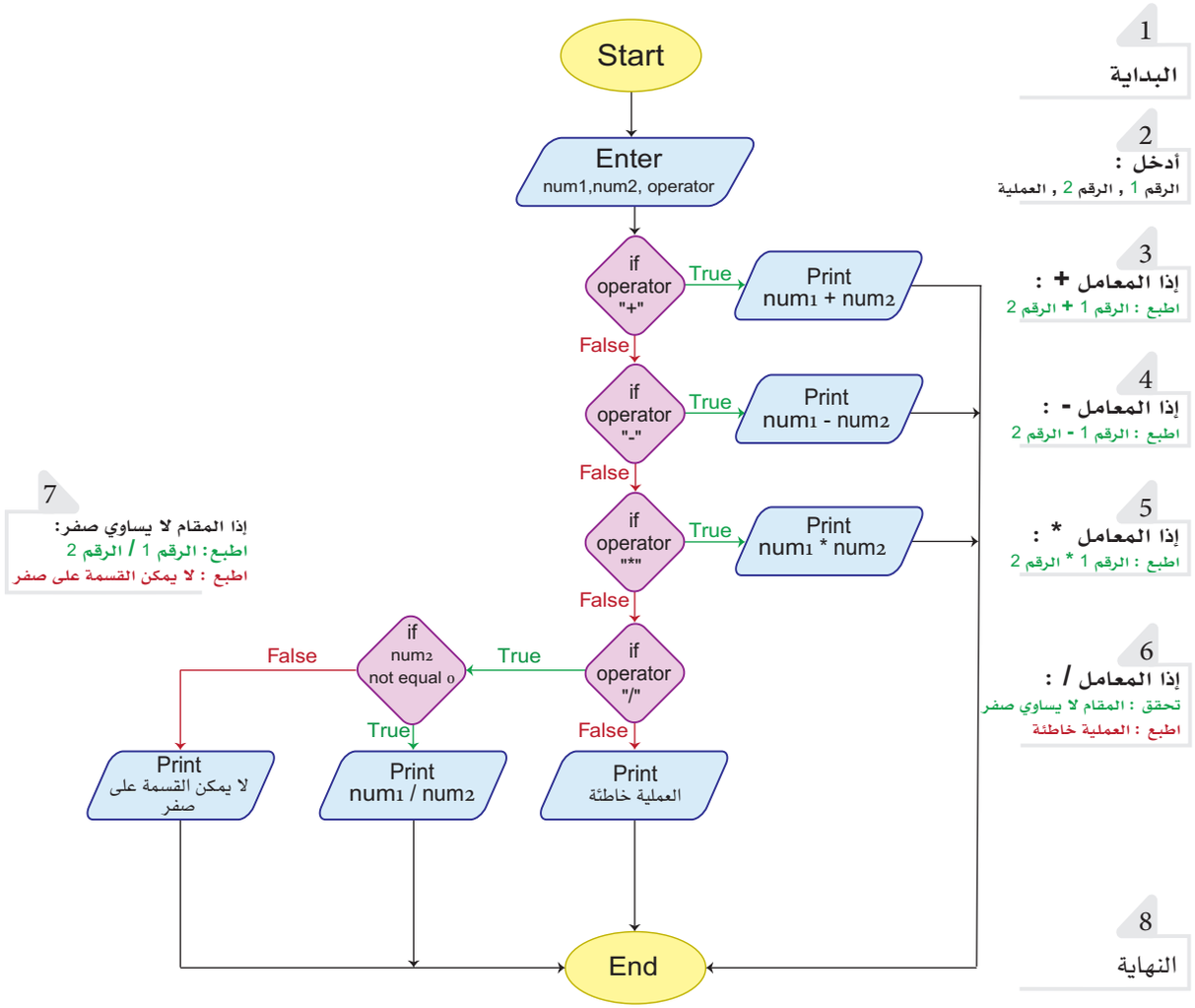
(التأكد من العدد الثاني لا يساوي صفر، وطباعة رسالة تفيد ذلك).

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

المطلوب:

خريطة التدفق

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



شكل (4-6) 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("Invalid operator")
```

- اختبر البرنامج، وتأكد من خلوه من 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 القوائم 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 واستخدام البنية العامة **while...else** لفهم كيفية تنفيذ التعليمات بعد انتهاء التكرار.

يمثل رمز الاستجابة السريعة **QR** رابط
لملفات أوراق العمل، ومصادر التعلم

التكرار

هو تنفيذ مجموعة من التعليمات البرمجية بشكل متكرر.

وهناك نوعان رئيسيان من التكرار وهما :

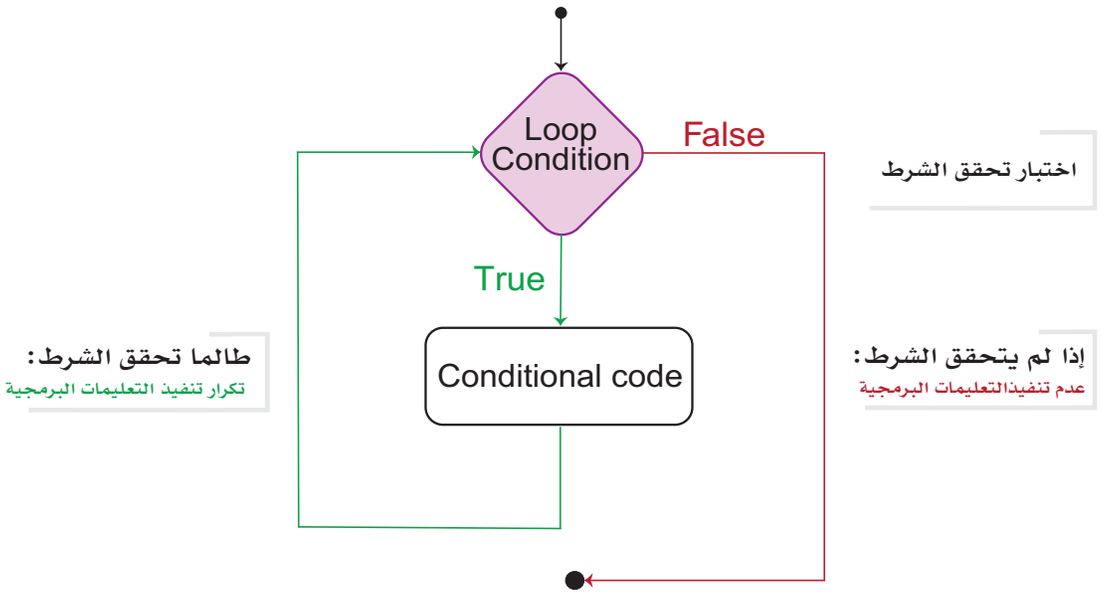
- الحلقة التكرارية : **while**.
- الحلقة التكرارية : **for**.

الحلقة التكرارية while

تُستخدم لتكرار تنفيذ كتلة برمجية طالما تحقق شرط مُعين.

الصيغة العامة Syntax :

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

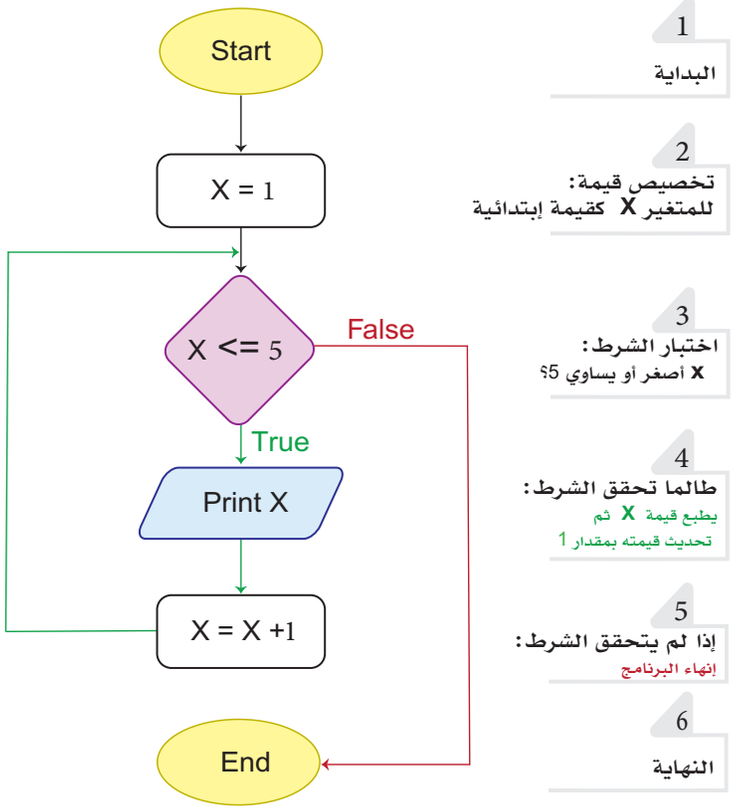


شكل (1-5) خريطة تدفق الصيغة العامة **while**

مثال 1:

استخدام الحلقة التكرارية while لطباعة الأعداد من 1 إلى 5:

أ. خريطة التدفق



شكل (2-5) خريطة تدفق طباعة الأعداد

ب. التعليمة البرمجية

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

Program output

```
1
2
3
4
5
```

لاحظ: كل من x = x + 1 و x += 1 يؤديان نفس الوظيفة، وهي زيادة قيمة المتغير بمقدار واحد.

عند استخدام الحلقة التكرارية while هناك نقاط يجب الانتباه لها:

- الشرط غالباً ما يتضمن متغير تم الإعلان عنه مُسبقاً، ويخصص له قيمة ابتدائية initial value.
- الخروج من التكرار: يتم الخروج من التكرار في حالتين:
  1. عدم تحقق الشرط (تعديل قيمة متغير الشرط).
  2. استخدام التعليمة البرمجية break.

لاحظ :

التكرارات اللانهائية Infinite loops

مثال 2:

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

```
1    while True:
2        print("Hello Word !")
```

Program output

```
Hello Word !
Hello Word !
Hello Word !
Hello Word !
Hello Word !
Hello Word !
Hello Word !
Hello Word !
Hello Word !
Hello Word !
```

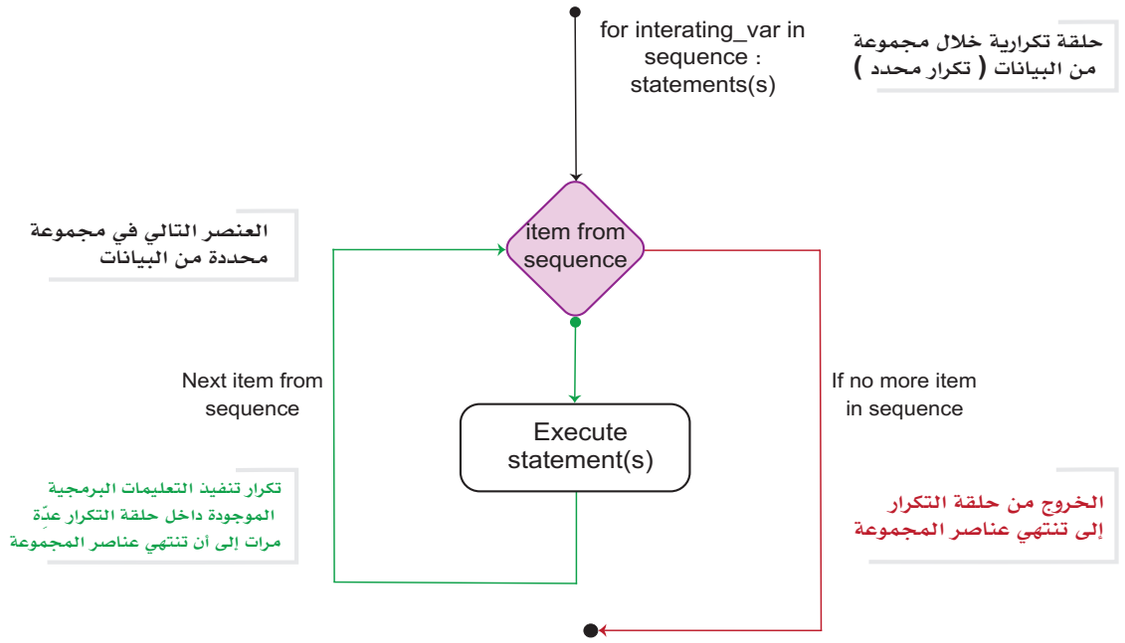
- ستؤدي هذه التعليمات البرمجية إلى طباعة **Hello, world** دون توقف.
- يمكن إيقاف تنفيذ التعليمة البرمجية باستخدام أداة **Stop** من شريط الأدوات.
- عدل التعليمات البرمجية في مثال 1 لطباعة قيمة المتغير x بصورة لا نهائية.

الحلقة التكرارية for

تستخدم الحلقة التكرارية for للتكرار خلال مجموعة من البيانات (تكرار محدد).

الصيغة العامة Syntax:

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



شكل (3-5) خريطة تدفق الصيغة العامة for

قابل للتكرار iterable: كائن يمكن الانتقال عبر عناصره بشكل متعاقب.

السلاسل النصية Strings تعتبر كائنات قابلة للتكرار.

مثال 3:

استخدام دالة input لإدخال الاسم ثم طباعة كل حرف من السلسلة النصية للمتغير في أسطر مستقلة.

```
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
```

يمكن استخدام الدالة range() لإنشاء سلسلة من الأعداد الصحيحة

عناصر السلسلة	مثال	الوصف	الصيغة
0, 1, 2, 3, 4, 5, 6, 7, 8, 9	range(10)	الأعداد الصحيحة من صفر إلى أقل من stop بواحد	range(stop)
3, 4, 5, 6, 7	range(3,8)	الأعداد الصحيحة من start إلى أقل من stop بواحد.	range(start,stop)
1, 3, 5, 7, 9	range(1,11,2)	سلسلة من الأعداد الصحيحة من start إلى أقل من stop بواحد بزيادة step في كل مرة.	range(start, stop,-step)

جدول (5-1) استخدام الدالة range() لإنشاء سلسلة من الأعداد الصحيحة

أكمل الجدول التالي:

السلسلة	المثال
.....	range(7)
.....	range(2,10)
.....	range(-10,-1)
.....	range(10,2,-1)
.....	range(2,10,2)

جدول (5-2) تدريب على استخدام الدالة range()

استخدام الحلقة التكرارية for مع range

مثال 4:

استخدام الحلقة التكرارية for لطباعة الأعداد من 0 إلى 3 على النحو التالي:

```
1  for x in range( 4 ):
2      print(x)
```

Program output

```
0
1
2
3
```

مثال 5:

استخدام الحلقة التكرارية for لطباعة الأعداد من 6 إلى 10 على النحو التالي:

```
1  for x in range( 6 , 11):
2      print(x)
```

Program output

```
6
7
8
9
10
```

مثال 6:

لدينا مجموعة من الأرقام (من 1 إلى 10)، المطلوب استخدام التكرار لطباعة الأرقام أكبر من 5، ويمكننا القيام بذلك على النحو التالي:

```
for number in range(1,11):
    if number > 5:
        print(number)
```

Program output

```
6
7
8
9
10
```

مثال 7:

يطلب البرنامج إدخال كلمة المرور حتى يتم مطابقتها مع كلمة المرور الصحيحة، ثم يعرض رسالة «تم إدخال كلمة المرور بشكل صحيح».

```
correctPassword = '1234'
password = ''
while (password != correctPassword):
  password = input("Enter password: ")
print("Password Correct")
```

Program output

```
Enter password: 12
Enter password: 1234
Password Correct
```

طور البرنامج ليتم تحديد عدد محاولات الإدخال بثلاث فقط.

التكرارات المتداخلة nested loops

مما سبق تعلمت أن التكرار يُكرر مجموعة من التعليمات البرمجية، والتي يُمكن أن تحتوي بداخلها على تكرارات أخرى.

مثال 8:

```
var_x = 0
var_y = 0
for r in range(2):
    var_x += 1
    for c in range(3):
        var_y += 1
print(var_x)
print(var_y)
```

ناقش مع معلمك عدد مرات تخصيص المتغير var_x والمتغير var_y.

التعليمة البرمجية break و continue.

تُستخدم التعليمات البرمجية break و continue للتحكم في عملية التكرار.

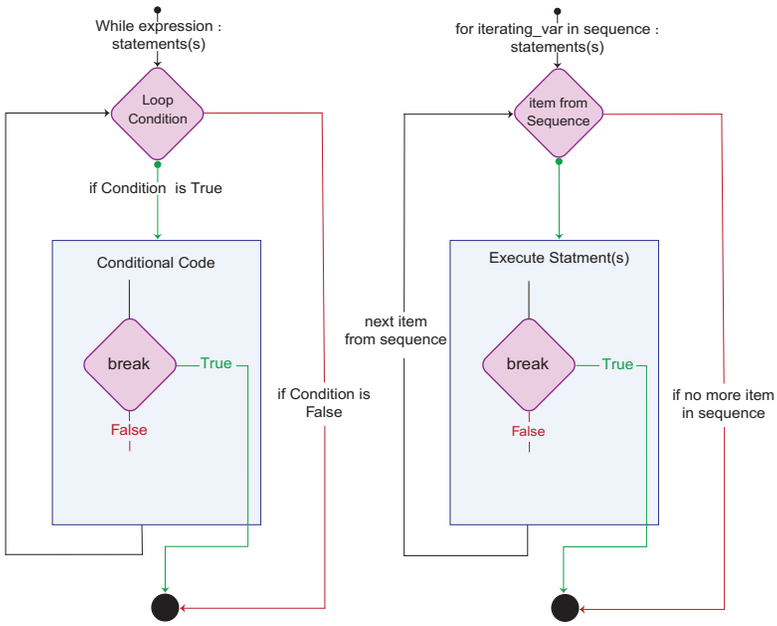
التعليمة البرمجية break

تُستخدم تعليمة break للخروج من الحلقة فورًا.

الصيغة العامة Syntax:

```
for variable in iterable:
    loop code
    if condition_to_break:
        break
    more loop code
```

```
while condition:
    loop code
    if condition_to_break:
        break
    more loop code
```



شكل (4-5) خريطة تدفق الصيغة العامة break

مثال 9:

إذا كانت لدينا الحلقة التكرارية while تتكرر من 0 إلى 10، ونريد الخروج من الحلقة عندما تصل إلى 5، يمكننا استخدام تعليمات break على النحو التالي:

```
1    i = 0
2    while i <= 10:
3        print(i)
4        if i == 5 :
5            break
6        i += 1
```

Program output

```
0
1
2
3
4
5
```

سيؤدي هذا النص البرمجي إلى طباعة الأرقام من 0 إلى 5. عند الوصول إلى 5، ستؤدي تعليمة break إلى خروج الحلقة، ولن يتم طباعة أي شيء بعد ذلك.

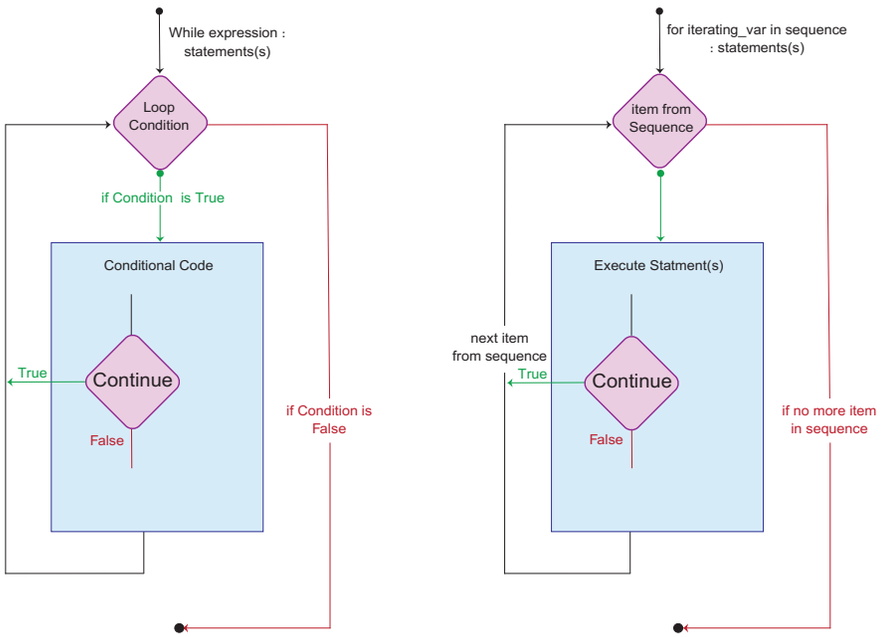
التعليمة البرمجية continue

تُستخدم تعليمة continue لتجاوز بقية النص البرمجي في الحلقة الحالية والانتقال إلى التكرار التالي.

الصيغة العامة Syntax:

```
while condition:
    loop code
    if condition_to_ continue:
        continue
    more loop code
```

```
for variable in iterable:
    loop code
    if condition_to_continue:
        continue
    more loop code
```



شكل (5-5) خريطة تدفق continue

مثال 10 :

طباعة الأعداد الزوجية باستخدام الحلقة التكرارية while

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

Program output

```
2
4
6
8
10
```

مثال 11 :

طباعة الأعداد الزوجية باستخدام الحلقة التكرارية for

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

Program output

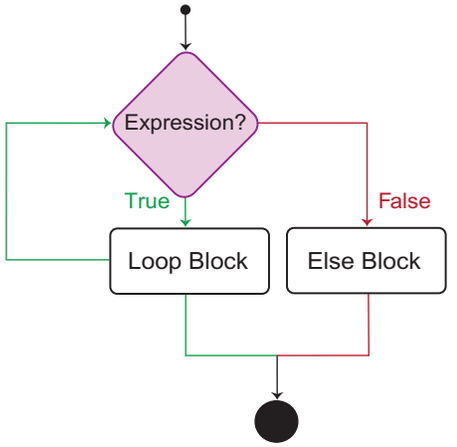
```
2
4
6
8
10
```

الحلقة التكرارية while ... else

يمكن تنفيذ الكتلة البرمجية **else** مع الحلقة التكرارية **while**.

الصيغة العامة Syntax:

```
while condition:
    loop body
else:
    code to be executed when loop finishes-
    normally
```



شكل (6-5) خريطة تدفق الصيغة العامة **while ... else**

مثال 12:

برنامج بسيط يُظهر رسالة خطأ عند محاولة إدخال المستخدم كلمة المرور 3 مرات بصورة خاطئة.

```
1   attempts = 1
2   correct_password = 'a1234'
3   while attempts < 4:
4       password = input("Enter password: ")
5       if password == correct_password:
6           break
7       attempts = attempts + 1
8   else:
9       print("You have entered the wrong password 3 times.")
10      exit()          #Exit the program
```

Program output

```
Enter password: 123
Enter password: a123
Enter password: a234
You have entred the wrong password 3 times.
```

أوراق العمل

ورقة عمل (14):

برنامج تخمين عدد

مشكلة البرنامج

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

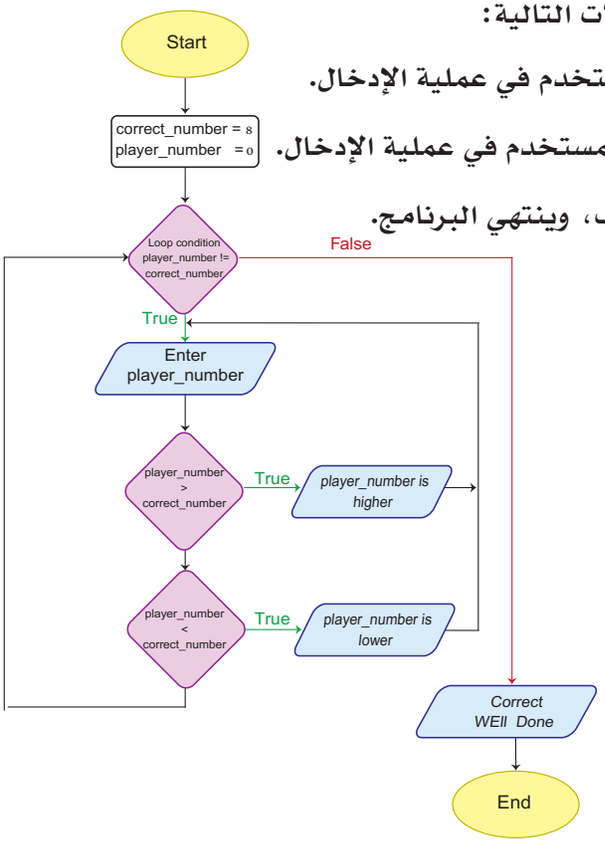
الخوارزمية

1. بداية البرنامج.
2. الإعلان عن متغير يساوي العدد المطلوب تخمينه.
3. استقبال عدد صحيح من المستخدم.
4. مطابقة العدد المُدخل مع العدد المحدد 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 المُعرف، وينتهي البرنامج.
5. نهاية البرنامج.

المطلوب:

خريطة التدفق

- ادرس خريطة التدفق المقابلة.



شكل (5-7) 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 ١ إلى ٣٠
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 (15):

برنامج جدول الضرب

مشكلة البرنامج

برنامج يطبع جدول الضرب لعدد محدد.

الخوارزمية

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

     الناتج = العدد المُدخل * 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
```

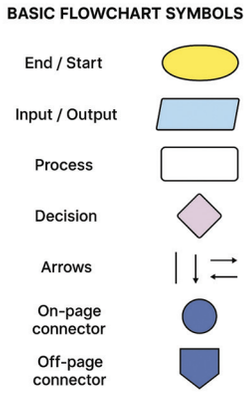
خريطة التدفق

- 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

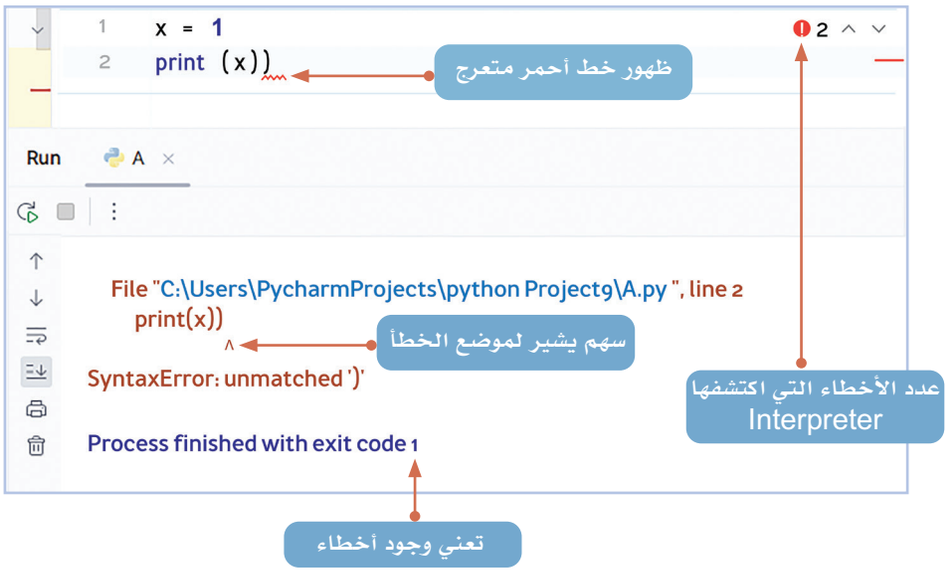
- شرح مفهوم الأخطاء Errors والاستثناءات Exceptions في البرامج البرمجية.
- تمييز بين أنواع الأخطاء الشائعة مثل: أخطاء نحوية Syntax Errors، وأخطاء وقت التشغيل Runtime Errors، وأخطاء المنطق Logic Errors.
- شرح أهمية التعامل مع الاستثناءات لضمان استقرار البرنامج ومنع توقفه المفاجئ.
- استخدام التعليمة try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

يمثل رمز الاستجابة السريعة QR رابط
لملفات أوراق العمل، ومصادر التعلم

مدخل إلى تصحيح الأخطاء والاستثناءات

تعتبر الأخطاء والاستثناءات مفهوم أساسي في البرمجة، وتهدف إلى التعامل مع الأخطاء التي قد تحدث أثناء كتابة وتشغيل البرنامج بطريقة منظمة تمنع توقفه، وتتيح آلية معالجتها للمبرمجين التحقق من الأخطاء والتعامل معها بمرونة. كما تساعد في تحسين استقرار البرنامج وتقديم تجربة مستخدم User Experience أفضل من خلال عرض رسائل تنبيهية أو اتخاذ إجراءات بديلة دون تعطيل البرنامج بالكامل.

الأخطاء في التعليمات البرمجية في Python داخل بيئة PyCharm



شكل (6-1) يوضح الأخطاء في Python من خلال بيئة PyCharm

ملاحظة: يشير الرقم (2) إلى وجود خطأين في الكود:

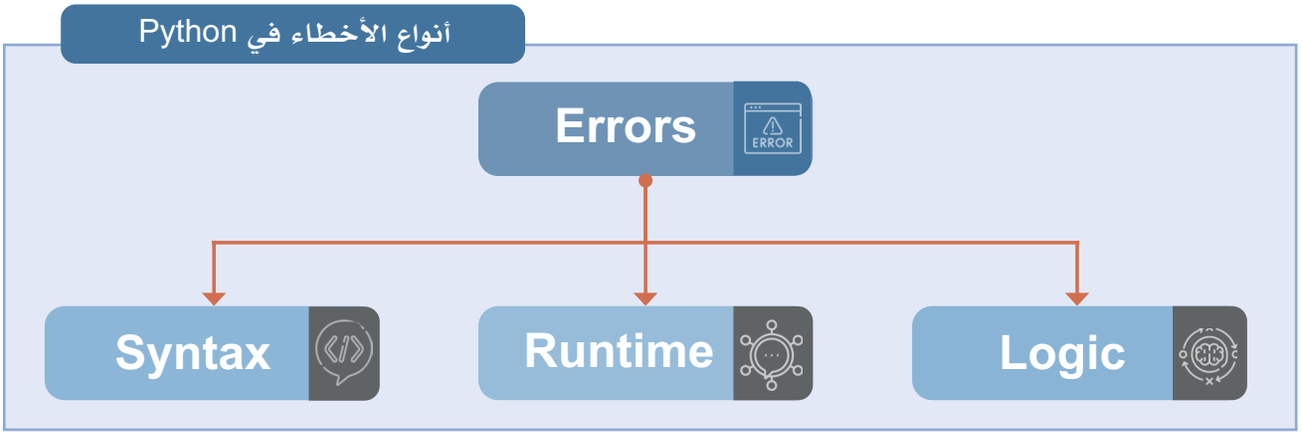
- وجود قوس زائد غير مطابق (في السطر ((x)print، وهو ما تسبب في ظهور رسالة الخطأ SyntaxError: unmatched.
- خطأ في بناء الجملة البرمجية نتيجة القوس الزائد، مما جعل المفسر غير قادر على فهم تركيب الجملة بشكل صحيح.

تصحيح الخطأ

شكل (2-6) يوضح تصحيح الأخطاء في Python من خلال بيئة PyCharm

تصحيح الأخطاء Debugging

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:



شكل (3-6) أنواع الأخطاء في Python

1. الأخطاء اللغوية Syntax Errors

أخطاء تحدث بسبب عدم الالتزام بقواعد كتابة التعليمات البرمجية، ولا يستطيع المفسر Interpreter فهم تلك التعليمات، مما يمنع البرنامج من العمل، (وهذه الأخطاء تظهر قبل بدء التنفيذ).

أخطاء في بناء الجملة (Syntax Errors):

- تحدث عند مخالفة قواعد كتابة التعليمة البرمجية، مثل نسيان الأقواس، أو النقطتين، أو علامات التنصيص، أو كتابة الكلمات المفتاحية بشكل خاطئ.

نوع الخطأ	مثال	نوع رسالة الخطأ
نسيان النقطتين (:) Colon بعد الجملة الشرطية، أو مع الحلقات التكرارية For / While.	if x > 0 print(x)	SyntaxError
عدم إغلاق الأقواس.	print ("Hello, Ku wait!"	SyntaxError
أخطاء المسافات البادئة (Indentation Errors): بايثون تعتمد على المسافات البادئة لتحديد الكتل البرمجية. أي خطأ في ترتيب المسافات أو المزج بين المسافات والـ tab يؤدي إلى ظهور خطأ نحوي.	for i in range (5): print (i)	IndentationError

جدول (6-1) يوضح أمثلة على بعض الأخطاء في بناء الجملة

2. أخطاء وقت التشغيل Runtime Errors

- تظهر هذه الأخطاء أثناء تنفيذ البرنامج، رغم أن كتابته اللُغوية سليمة، وغالبًا ما تحدث بسبب ظروف غير متوقعة.
- تظهر هذه الأخطاء أثناء تنفيذ البرنامج، بسبب مدخلات أو عمليات رغم أن التعليمة البرمجية مكتوبة بشكل صحيح.

من أشهر أخطاء وقت التشغيل Runtime Errors:

نوع الخطأ	مثال	رسالة الخطأ
محاولة قسمة عدد على صفر (ZeroDivisionError)	print(5 / 0)	ZeroDivisionError: division by zero
محاولة استخدام متغير غير مُعرف (NameError)	x = 5 print(z)	NameError: name 'z' is not defined
إدخال نوع بيانات غير صحيحة (TypeError)	print('5' + 10)	TypeError: can only concatenate str (not "int") to str
إدخال قيمة غير مناسبة لنوع البيانات (ValueError)	number = int('abc')	ValueError: invalid literal for int() with base 10: 'abc'

جدول (6-2) يوضح أمثلة على بعض أخطاء وقت التشغيل

3.الأخطاء المنطقية Logic Errors

الأخطاء التي لا يتعرف عليها Python Interprerter، ولكنها قد تؤدي إلى نتائج غير متوقعة أو غير صحيحة (بمعنى أن التعليمات البرمجية تعمل بشكل صحيح، لكن نتيجة تنفيذها غير صحيحة). على سبيل المثال:

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

نوع الخطأ	مثال خاطئ	رسالة الخطأ	تصحيح الخطأ المقترح
خطأ في العمليات الحسابية استخدام عملية رياضية غير مناسبة للهدف	length = 5 width = 4 rectangle_area = length + width print(rectangle_area)	لا تظهر رسالة خطأ وتظهر النتيجة العدد 9 التفسير: يقوم بجمع الطول والعرض، وهذا غير صحيح رياضيًا لحساب مساحة المستطيل.	length = 5 width = 4 rectangle_area = length * width print(rectangle_area)
خلط في ترتيب العمليات الحسابية عدم استخدام الأقواس مما يسبب تنفيذ العمليات بترتيب غير متوقع	a= b= c= 10 average = a + b + c / 3 print (average)	لا تظهر رسالة خطأ وتظهر النتيجة العدد 23.333333333333332 التفسير: العمليات الحسابية لم تتبع أولوية التنفيذ لحساب المتوسط الحسابي.	a= b= c= 10 average = (a + b + c) / 3 print (average)

مقارنة منطقية خاطئة
 	
خطأ في صياغة شرط if يؤدي لنتائج عكسية
	# Maximum final score = 100
	score = 45
	if score < 50:
	print(”Excellent”)
		لا تظهر رسالة خطأ
		وتظهر النتيجة Excellent
		
		التفسير:
		
		الشرط score < 50 يعني أن الطالب راسب (أقل من 50)، ولكن الجملة التي ستُطبع هي: ”Excellent” (ممتاز)
		وهذا يتناقض مع نظام التقييم التعليمي.
			# Maximum final score = 100
			score = 45
			if score < 50:print(”Fail”)

جدول (3-6) يوضح أمثلة على بعض الأخطاء المنطقية

الاستثناءات Exceptions

يمكن استخدام التعليمات البرمجية (try/except) لتحديد الجزء من البرنامج المتوقع حدوث خطأ به، لاستمرار تنفيذ البرنامج دون توقف.

تنقسم التعليمات البرمجية try/except إلى :

- try: تُستخدم كتلة try لكتابة التعليمة البرمجية الذي قد يسبب خطأ أثناء التنفيذ. إذا حدث خطأ في هذا الجزء، ينتقل البرنامج إلى كتلة except، أما إذا لم يحدث خطأ، فيستكمل تنفيذ البرنامج
- except: تُستخدم كتلة except لمعالجة الأخطاء التي قد تحدث داخل try.
- else: تُنفذ كتلة else فقط إذا لم يحدث أي خطأ داخل try.
- finally: تُستخدم كتلة finally لتنفيذ أوامر يتم تنفيذها دائمًا في نهاية البرنامج، سواء حدث خطأ في كتلة try أو لم يحدث.

الصيغة العامة للاستثناءات 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، تُعرض رسالة تُوضح بأن القيم المدخلة غير رقمية.
- عند حدوث خطأ من نوع ZeroDivisionError، (عدد الطلاب يساوي صفر) تُعرض رسالة تُوضح بأنه لا يمكن إجراء القسمة على صفر.

الكتلة else:

- عند نجاح جميع العمليات دون وقوع استثناء، يتم طباعة قيمة المشاركة.

الكتلة finally:

- تنفّذ دائماً بغض النظر عن وقوع أخطاء أم لا.
- تطبع رسالة تؤكد انتهاء تنفيذ البرنامج.

```
1   try:
2       print("School Trip Cost Sharing Calculator")
3       total_cost = float(input("Enter total trip cost (KWD): "))
4       students = int(input("Number of participating students: "))
5       share = total_cost / students
6   except ValueError:
7       print("Input error: Please enter valid numbers only!")
8   except ZeroDivisionError:
9       print("Error: Cannot divide by zero - please check student count")
10  else:
11      print(f"Each student's share: {share: .2f} KWD")
12  finally:
13      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).
- تعرض رسالة عامة تحتوي على وصف الخطأ المخزن في المتغير e.

الكتلة else:

- عند نجاح جميع العمليات دون وقوع استثناء، يتم طباعة قيمة المشاركة.

الكتلة finally:

- تنفّذ دائماً بغض النظر عن وقوع أخطاء أم لا.
- تطبع رسالة تؤكد انتهاء تنفيذ البرنامج.

```
1   try:
2       print("School Trip Cost Sharing Calculator")
3       total_cost = float(input("Enter total trip cost (KWD): "))
4       students = int(input("Number of participating students: "))
5       share = total_cost / students
6       print(f"Each student's share: {share: .2f} KWD")
7   except Exception as e:
8       print('Error is:', e)
9   finally:
10      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!
```

ملاحظات:

1. التعامل مع الاستثناءات Exceptions.

- يُفضل استخدام الاستثناءات المحددة (Specific Exceptions)

مثل ValueError و ZeroDivisionError؛ لأنها تساعد في تحديد ومعالجة الخطأ بشكل واضح ومحدد

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

except Exception as e:

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

2. التعامل مع كتلة else.

- تم الاستغناء عن الكتلة else، ونقل التعليمة التي كانت بداخلها إلى نهاية الكتلة try.

مثال 3:

برنامج يستقبل أعداد صحيحة، ويُظهر رسالة تُحدد ما إذا كان العدد المدخل زوجيًا أم فرديًا، وعند إدخال بيانات نصية، يتم استثناء الخطأ وطباعة رسالة You did not enter a valid number، ويستمر البرنامج دون توقف.

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

Program output

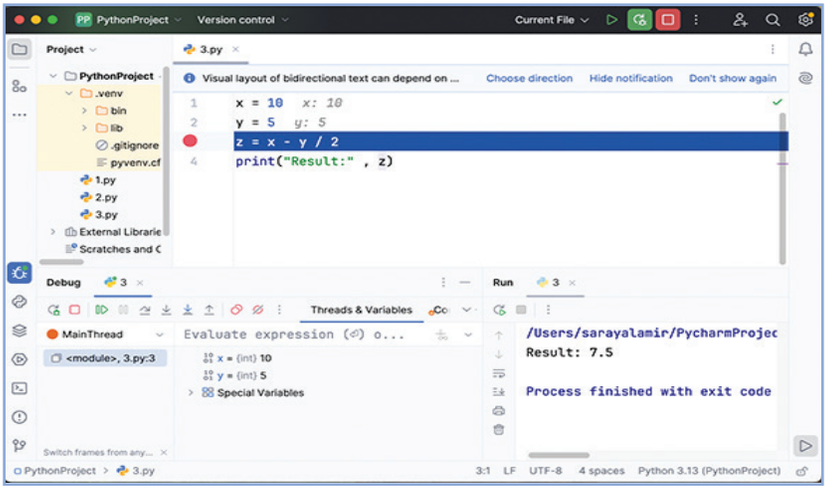
```
Enter a number:56
The number is even.
Enter a number:8.9
You did not enter a valid number.
Enter a number:GR11
You did not enter a valid number.
Enter a number:
```

تصحيح الأخطاء Debugging

عملية تتبع وتحليل التعليمات البرمجية خطوة بخطوة لاكتشاف الأخطاء، فهم سير التنفيذ، بهدف تصحيح النتائج غير المتوقعة.

خطوات تتبع الأخطاء في PyCharm

يتم تنفيذ الخطوات التالية بهدف تتبع وتحليل الأخطاء البرمجية وتصحيحها.



شكل (6-4) 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 خطوة بخطوة، أو للانتقال إلى السطر التالي، أو لإيقاف عملية التصحيح في أي وقت

وهي مجموعة من الأدوات تُستخدم لتتبع تنفيذ البرنامج خطوة بخطوة، وتساعد على مراقبة القيم، واكتشاف الأخطاء المنطقية أو أخطاء أثناء التشغيل

وتمكّن 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 الدالة التي يتم استدعاؤها في السطر الحالي.
- Step Out: للخروج من الدالة 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، تظهر النتائج أو المخرجات في نافذة Console.

أوراق العمل

ورقة عمل (16):

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:**
   - **طلب من المستخدم إدخال price (السعر الأصلي).**
   - **طلب من المستخدم إدخال discount (نسبة الخصم).**
   - **حساب السعر النهائي باستخدام:**

     net_price = price - (price * discount / 10)

   - **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 أن المدخلات عددية باستخدام التعليمة البرمجية (try — except).
4. استخدم التعليمة البرمجية الشرطية (if - else) للتأكد أن:
   - 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 (17):

مؤشر 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:
     - طلب من المستخدم إدخال الوزن بالكيلوغرام وحفظه كعدد عشري (float).
     - طلب من المستخدم إدخال الطول بالمتر وحفظه كعدد عشري (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 كتلة except بحيث إذا ادخل 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 canfit divide by zero!.".
3. الرجوع إلى الخطوة 2.
4. نهاية البرنامج.

المطلوب:

1. افتح الملف BMI.py، الذي يحسب مؤشر 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 = float(input("Enter your weight in kilograms: "))
        height = float(input("Enter your height in meters: "))
        # Calculate BMI
        bmi = weight / (height ** 2)
        # Display the BMI
        print(f"Your BMI is: {bmi:.2f}")
        # Determine BMI category
        if bmi < 18.5:
            print("You are underweight.")
        elif 18.5 <= bmi <= 24.9:
            print("You have a normal weight.")
        elif 25 <= bmi <= 29.9:
            print("You are overweight.")
        else:
            print("You are obese.")
    .....
        print("Error: Please enter valid numeric values for weight and height.")
    .....
        print("Error: You can't divide by zero!")

```

3. اختبر البرنامج، وتأكد 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

نواتج التعلم

- توظيف المهارات البرمجية المكتسبة في Python بشكل فعّال لإنتاج برامج متنوعة.
- تنمية القدرة على الابتكار والإبداع من خلال تحليل المشكلات وتصميم حلول برمجية مناسبة.
- تطبيق المهارات البرمجية في تصميم وتطوير برامج تعالج احتياجات من واقع الحياة اليومية.
- المُشاركة بفاعلية 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.

يمثل رمز الاستجابة السريعة QR رابط لملفات أوراق العمل، ومصادر التعلم

أمثلة على المنتجات الرقمية

استعرضنا فيما سبق المفاهيم الأساسية للبرمجة بلغة Python مثل المتغيرات، الحلقات التكرارية، الشروط والتعامل مع الأخطاء ...، ومن ثم يمكننا تصميم وتطوير برامج متكاملة تخدم مواقف حياتية:

أفكار مشاريع Python للصف العاشر:

1

لعبة حجر ورقة مقص:

هي لعبة تفاعلية مسلية بين اللاعب والحاسوب، حيث يختار كل طرف إحدى الخيارات الثلاثة: حجر، ورقة، أو مقص، ثم يقوم البرنامج بمقارنة اختيار اللاعب مع اختيار الحاسوب لتحديد الفائز وفقًا لقواعد اللعبة المعروفة وفق الخطوات التالية:

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

2

تحدي العواصم:

في هذا التحدي يُعرض على اللاعب اسم دولة عشوائية ويُطلب منه إدخال عاصمتها الصحيحة، ثم يُقيّم البرنامج صحة الإجابة ويجمع النقاط لكل إجابة صحيحة وفق الخطوات التالية:

- يعرض للاعب عشوائياً اسم دولة.
- يستقبل من اللاعب اسم العاصمة.
- يختبر صحة الإجابة وتزداد عدد النقاط بمقدار واحد عند صحتها.
- يُكرر الخطوات السابقة لعدد محدد من الأسئلة، ويعرض المجموع النهائي للنقاط.

3

آلة الوقت المتبقي:

يساعد هذا البرنامج المستخدم في حساب الوقت المتبقي حتى موعد معين (مثل موعد الحصة أو الحافلة أو النوم) يدخل المستخدم الوقت الحالي والوقت المستهدف ويقوم البرنامج بطرح القيمتين لإظهار الفرق بالساعات والدقائق وفق الخطوات التالية:

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

مدخل إلى المشروع

من خلال دراستك لوحدة البرمجة بلغة Python، والاطلاع على التطبيقات (المنتجات الرقمية) المساعدة المتوافرة على الرابط الإلكتروني المتمثل في رمز الاستجابة السريع QR بداية وحدة المُنتجات الرقمية، والمتوافر على أجهزة الحاسوب بالمختبر المدرسي، صمم المشروع الخاص بك، والذي يتضمن كلاً من:

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

إعدد فريق العمل

يتكـون الفريـق مـن 4 إلـى 5 أعضـاء، تُقسـم المهـام بينهـم علـى النحـو التالـي:

- **قائد الفريق: المسؤول عن إدارة عمل الفريق وتجميع الملفات والمستندات المطلوبة وتقديمها للمعلم.**
- **معد الخوارزمية: المسؤول عن إعداد الخطوات الأساسية لعمل المشروع.**
- **مصمم خريطة التدفق: المسؤول عن ترجمة الخوارزمية إلى خريطة التدفق.**
- **المبرمج: المسؤول عن ترجمة خريطة التدفق لتعليمات برمجية واختبارها والتأكد من سلامتها.**
- **معد العرض التقديمي: المسؤول عن إعداد وتصميم العرض التقديمي للمناقشة.**

اكتب بيانات أعضاء الفريق موضحاً مهام كل عضو:

اعتماد المعلم: **التاريخ:**

خطوات إعداد المشروع

1. فكرة المشروع

يناقش أعضاء الفريق ويحدد موضوع 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 برمجي.
- 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 إستعداداً لتوثيق المشروع.
- **Draw.io**: لرسم خريطة التدفق للمشروع.
- **Microsoft Edge**: للبحث عن المعلومات.
- **Copilot**: للمساعدة في كتابة الأكواد واقتراح حلول برمجية ذكية.
- **GitHub (أو Git)**: لإدارة نسخ الكود ومشاركة الملفات بين أعضاء الفريق.

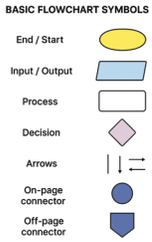
و أي مستلزمات أخرى تخدم المشروع وتوزيع مهام أعضاء الفريق.

3. إعداد هيكل المشروع

يُفضل 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

يكتب المبرمج التعليمات البرمجية استناداً لما تم تخطيطه سابقاً.

5. اختبار المشروع

يختبر المبرمج البرنامج، ويتأكد من خلوه من الأخطاء البرمجية، اللغوية، والعلمية.

6. تطوير المشروع

يُبسط المبرمج ويرتب التعليمات البرمجية إذا أصبحت صعبة القراءة، وإضافة التعليقات والبيانات الإرشادات التي توضح مهمة التعليمات البرمجية.

7. توثيق المشروع

يُنشئ مصمم العرض التقديمي عرضًا شاملاً للجوانب التالية:

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

و تسليم مجلد مجمع لجميع ملفات المشروع.

8. مشاركة المشروع

عرض الفريق للمشروع، ومناقشته مع المعلم والمتعلمين.

9. الاستمرار في التعلم

التوسع في الاطلاع على مستجدات الذكاء الاصطناعي وتنمية قدرات المتعلم البرمجية من خلال الدورات التدريبية والمنتديات الحاسوبية.

المراجع

- **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.

10