

UULA.COM



UULA

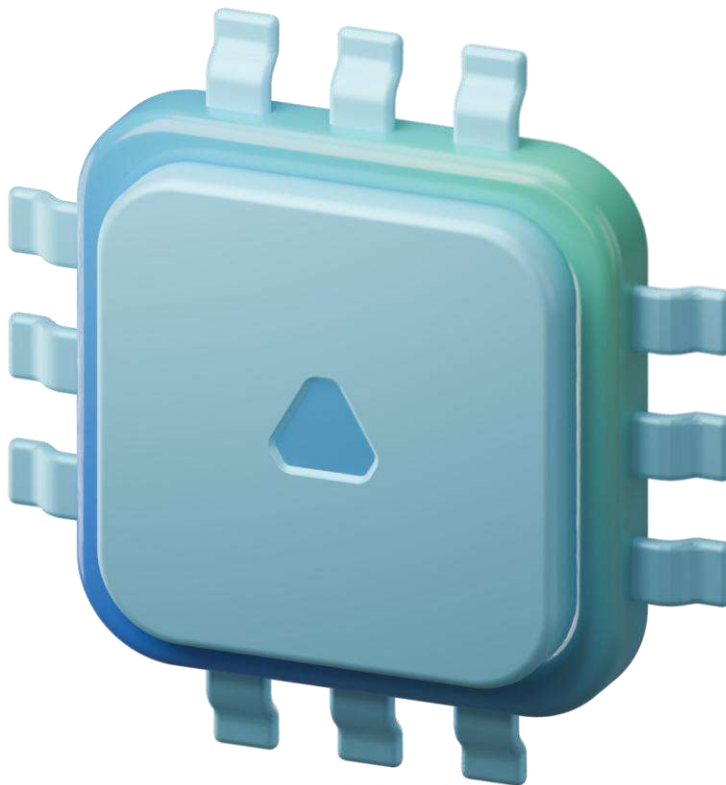
الحاسوب



الكورس الأول ✦ 2026 – 2027

10

UULA.COM



صَفْوَةُ مَعَالِمِ الْكُوَيْتِ

الحاسوب

الكورس الأول ✦ 2026 – 2027

10

حقق هدفك الدراسي

ريح بالك وارفع مستوى دراستك مع المذكرة الشاملة والفيديوهات التي تشرحها والاختبارات التي تدربك في منصة علا



نخبة المعلمين يجابونك بأسرع وقت

ما فهمت؟ تواصل مع أقوى
المعلمين واحصل على شرح
لسؤالك

دروس يشرحها أقوى معلمي الكويت

فيديوهات مبسطة قصيرة تشرح
لك كل شيء خطوة بخطوة

تفوق في القصير والفايل مع نماذج اختبارات سابقة

نماذج اختبارات سابقة مشروحة
بالكامل تجهزك لاختبارك



اكتشف عالم التفوق مع منصة علا

لتشارك بالمادة وتستمتع بالشرح المميز صور
أو اضغط على رمز الQR

المعلق



هذه المذكرة تغطي المادة كاملة.

في حال وجود أي تغيير للمنهج أو تعليق جزء منه يمكنكم مسح رمز QR للتأكد من المقرر.



المنقذ



أول ما تحتاج مساعدة بالمادة ، المنقذ موجودا!

صور ال QR بكاميرا التلفون أو اضغط عليه إذا كنت تستخدم المذكرة من جهازك ويطلع لك فيديو يشرح لك.



قائمة المحتوي

01 تثبيت برنامج Pycharm خطوات تثبيت Pycharm

1

02 برمجة Python

02

مدخل إلى البرمجة

Variables المتغيرات

Strings السلاسل النصية

Conditions الشروط

Loops التكرار

Debugging and Exceptions تصحيح الأخطاء والاستثناءات

2

6

10

13

17

21

03 المشروع والاختبارات

03

إعداد المشروع ونماذج الاختبارات

24

صِفْوَةٌ مَعَ الْكُتُبِ



تثبيت برنامج PyCharm

يمكن الحصول على آخر إصدار من محرر **PyCharm** من الرابط أو مسح QR-Code :

[رابط تنزيل البرنامج](#)



خطوات تثبيت PyCharm

يمكنك اتباع الخطوات لتثبيت البرنامج كما هو في الفيديو .

تنزيل الملفات

[رابط تنزيل الملفات](#)

تذكر :

- لتعديل لون الواجهة للبرنامج اختيار **File** ثم **Settings** ثم **Appearance** ثم تحديد **Theme** المطلوب .
- للتحكم بحجم الخط من خلال اختيار **File** ثم **Settings** واختيار Editor ثم **Font** ثم تحديد حجم الخط المطلوب .

ملاحظات عامة :

- كل درس من الدروس يوجد له : **فيديو شرح** - **فيديو حل ورقة العمل** .
- فيديوهات إعداد المشروع عبارة عن مراجعة شاملة للمادة .
- ضمن اختصار المنهج هناك **مسار خاص** لحل جميع أوراق العمل .





مدخل إلى البرمجة

مفهوم البرمجة

كتابة مجموعة من التعليمات والأوامر التي يترجمها الحاسوب إلى لغة الآلة لتنفيذ مهمة أو حل مشكلة.
- غير مطلوب الحفظ -

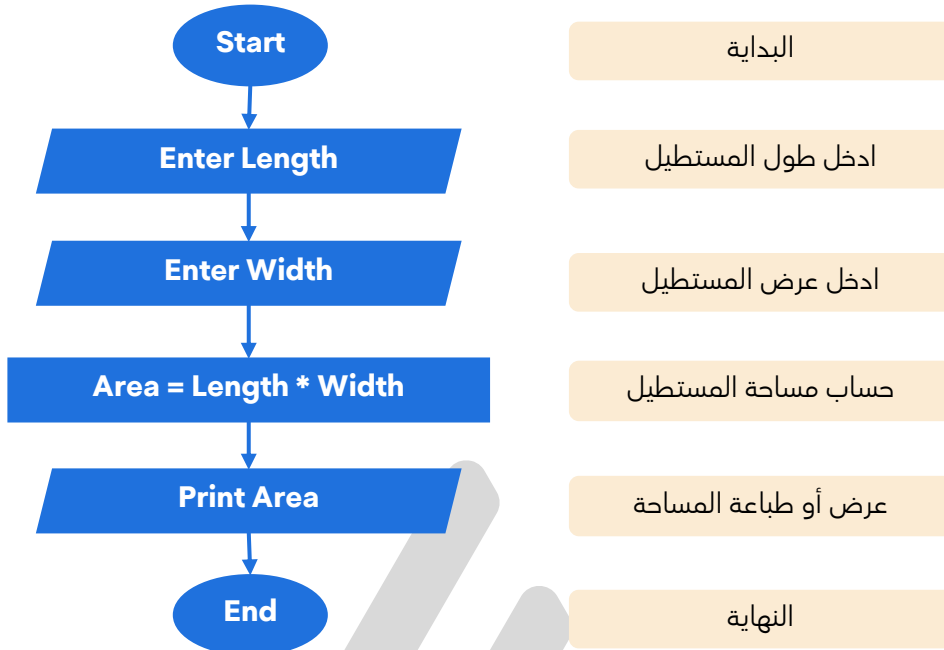
خطوات إنشاء برنامج

1. تحديد المشكلة .
2. إعداد خطة الحل (الخوارزمية Algorithm) .
3. خرائط التدفق (Flowchart) : مخططات رسومية توضح ترتيب العمليات اللازمة لحل المشكلة

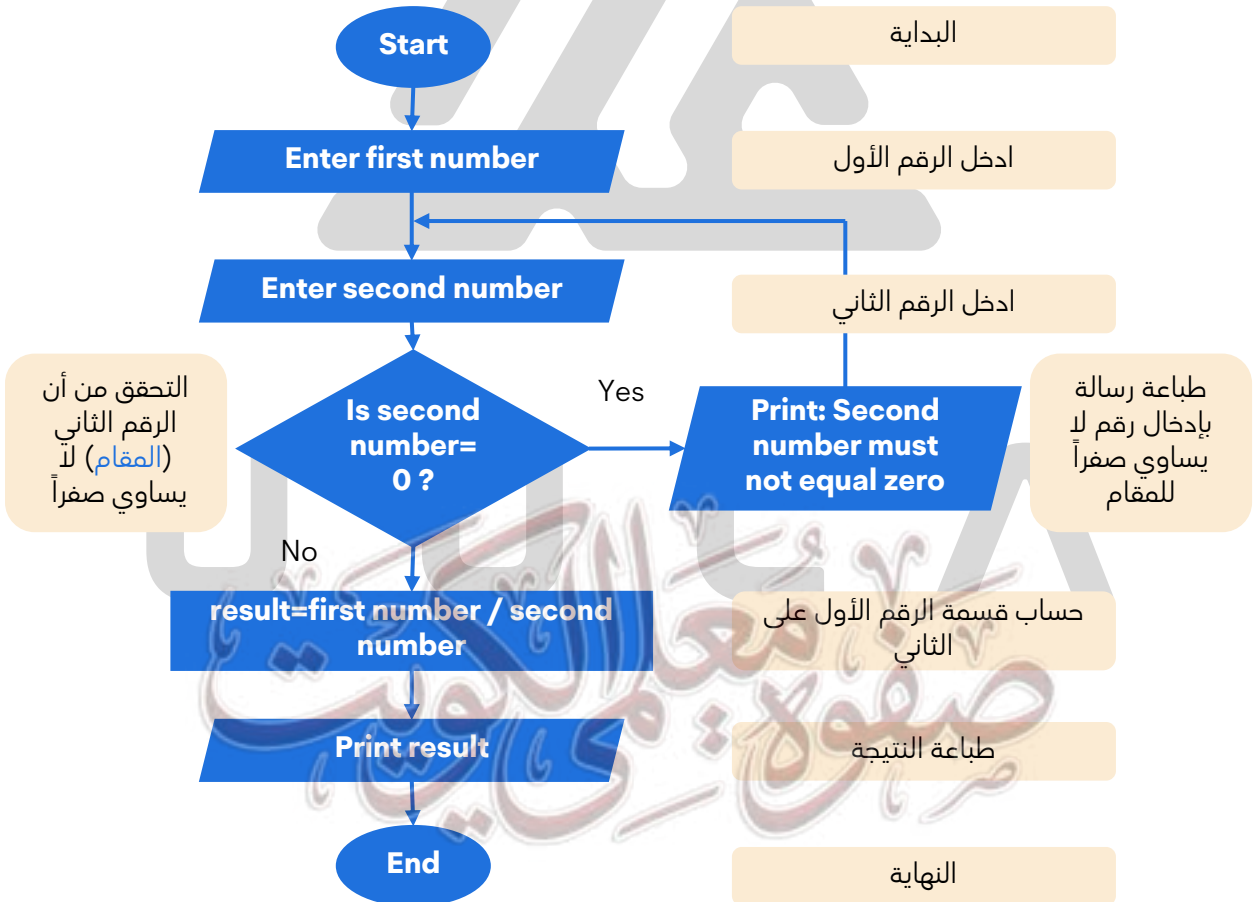
الأشكال الأساسية في خرائط التدفق Flowchart

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

مثال : ارسم خريطة التدفق لحساب مساحة مستطيل.



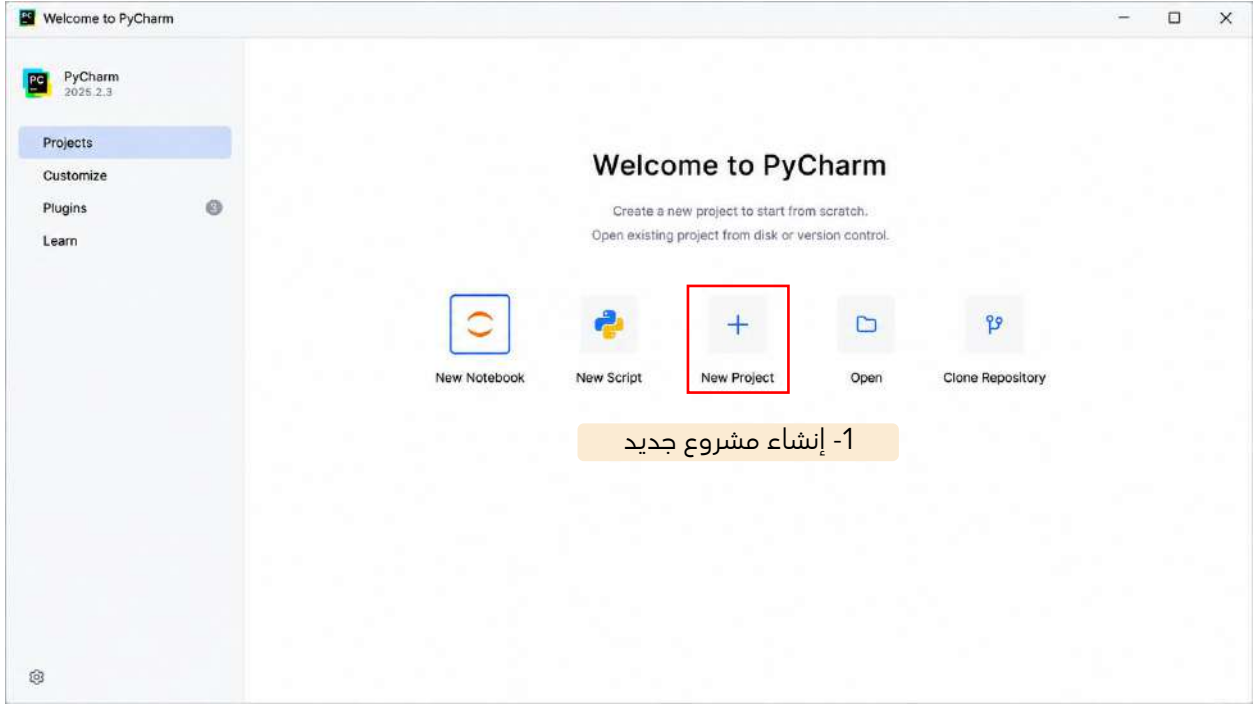
مثال : ارسم خريطة التدفق بحيث يتم قسمة رقمين مع مراعاة أن لا يكون المقام **صفر** .





1- إنشاء مشروع جديد

عند تشغيل **PyCharm** يظهر البرنامج بهذه الواجهة يتم إنشاء مشروع جديد من خلال الخطوات التالية :



1- إنشاء مشروع جديد

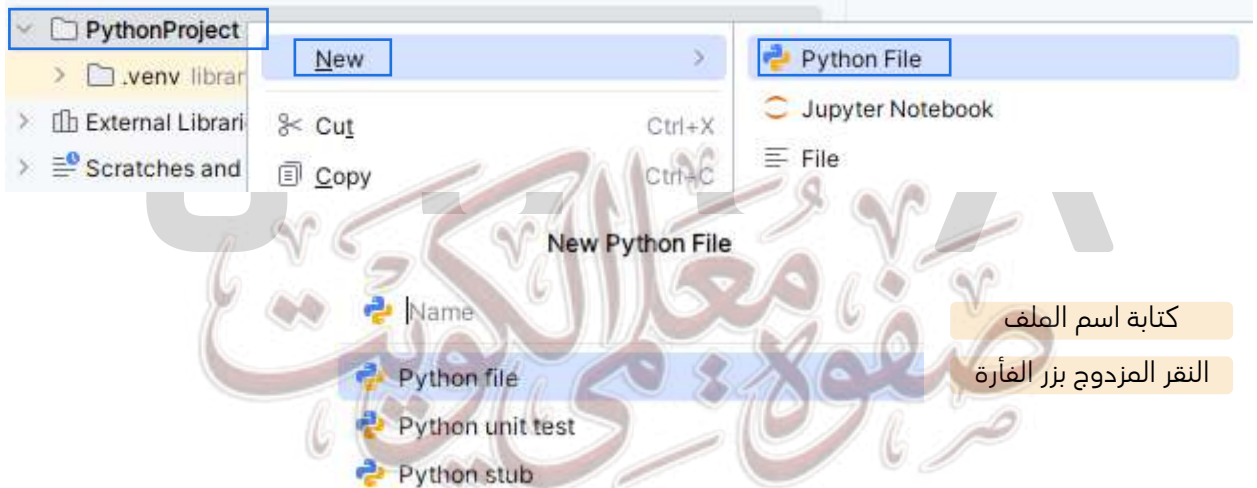
Location: C:\Users\ACER\Desktop\PythonProject

3- كتابة اسم المشروع

2- تحديد مكان المشروع

2- إنشاء ملف Python :

الضغط بالزر الأيمن على مجلد المشروع ومن القائمة المختصرة نختار أمر **New** ثم **Python File** وكتابة اسم الملف .



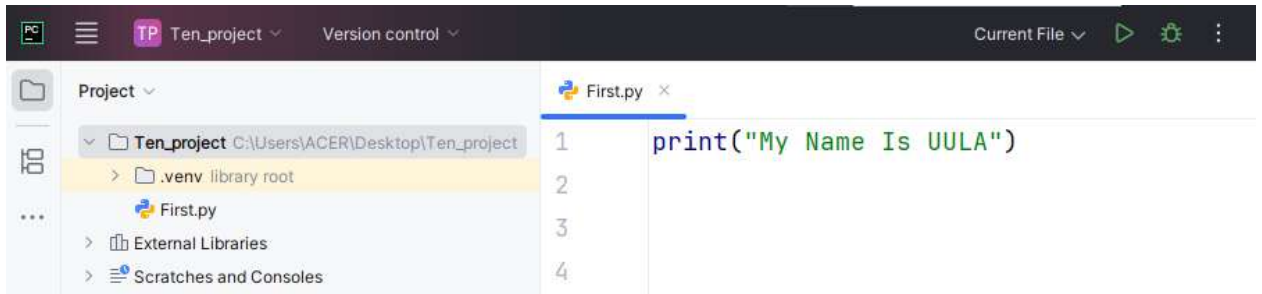
كتابة اسم الملف

النقر المزدوج بزر الفأرة

البرنامج : هو مجموعة من التعليمات .

تعليلة print : تطبع المحتويات بين القوسين في نافذة التشغيل **Run** .

مثال : اكتب التعليلة لطباعة اسمك .



```
1 print("My Name Is UULA")
2
3
4
```

تشغيل البرنامج

يمكن تشغيل البرنامج بإحدى الطرق التالية : -

- 1- القائمة المختصرة لمنطقة كتابة التعليمات البرمجية واختيار الأمر **Run** .
- 2- استخدام مفاتيح الاختصار **Ctrl + Shift + F10** .
- 3- منطقة شريط الأدوات **Toolbar** واختيار أداة **Run** .



ملاحظة :

لغة **Python** : تتعرف على نوع البيانات تلقائياً .



ورقة عمل 4 و 5



صفحة: ٥ من ٥



المتغيرات Variables

المتغير

هو مكان يتم حجزه في الذاكرة لتخزين البيانات بصورة مؤقتة أثناء تنفيذ البرنامج .

أنواع المتغيرات

تتميز لغة Python بالتعرّف إلى نوع البيانات.

نوع المتغير	نوع البيانات	استخداماته	مثال
str	نص	لتخزين الأسماء والحروف وتكتب بين " " أو ' '	my_name= " UULA "
int	عدد صحيح	تخزين قيم صحيحة	days= 7
float	عدد بفاصلة عشرية	تخزين أعداد عشرية	Salary = 1200.450
bool	قيمة منطقية	تخزين قيمة صح True أو خطأ False	x = 10 x > 5 #True

💡 **تذكر :** الرمز # يستخدم لإضافة التعليقات ولا ينفذها البرنامج كتعليمة برمجية .

دالة type : تستخدم لمعرفة نوع المتغير .

مثال : اكتب تعليمات لطباعة نوع المتغيرات my_name, my_class

```
1 my_name = "My Name Is UULA"
2 my_class = 10
3 print(type(my_name))
4 print(type(my_class))
```

النتيجة عند تشغيل البرنامج

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

دالة الإخراج print :

تستخدم لعرض البيانات والمخرجات على الشاشة ويمكن استخدام الفاصلة التي تضيف مسافة بين القيم .

مثال : ربط بين قيم المتغيرات

```
1 w= "Free"
2 z = "Kuwait"
3 print(w,z)
```

النتيجة عند تشغيل البرنامج

```
Free Kuwait
```

مثال : ربط بين النصوص وقيم المتغيرات

```
1 age = 16
2 print("My Age Is :",age)
```

النتيجة عند تشغيل البرنامج

```
My Age Is : 16
```

العمليات الحسابية :

تُمثّل العمليات الحسابية بالمعاملات التالية :

الرمز	العمليات الحسابية
+	الجمع
-	الطرح
*	الضرب
/	القسمة
//	القسمة الصحيحة
%	باقي القسمة
**	رفع عدد إلى أس محدد

اكتب التعليمات لتخصيص قيمة 25 للمتغير var1 وقيمة 15 للمتغير var2 .

```
var1 = 25
var2=15
```

اكتب تعليمة طباعة نتيجة جمع المتغيرين var1 و var2 .

```
print(var1+var2)
```

النتيجة عند تشغيل البرنامج : 40

اكتب تعليمة طباعة نتيجة طرح المتغير var2 من المتغير var1 .

```
print(var1-var2)
```

النتيجة عند تشغيل البرنامج : 10

اكتب تعليمة طباعة نتيجة ضرب المتغيرين var1 و var2 .

```
print(var1*var2)
```

النتيجة عند تشغيل البرنامج : 375

اكتب تعليمة طباعة نتيجة قسمة المتغير var1 على var2 .

```
print(var1/var2)
```

النتيجة عند تشغيل البرنامج : 1.6666666666666667

اكتب تعليمة طباعة نتيجة القسمة الصحيحة للمتغير var1 على var2 .

```
print(var1//var2)
```

النتيجة عند تشغيل البرنامج : 1

اكتب تعليمة طباعة نتيجة باقي القسمة للمتغير var1 على var2 .

```
print(var1%var2)
```

النتيجة عند تشغيل البرنامج : 10

اكتب تعليمة طباعة نتيجة رفع المتغير var1 للأس 2 .

```
print(var1**2)
```

النتيجة عند تشغيل البرنامج : 625



دالة input

- تستخدم لإدخال الأرقام والنصوص عند تشغيل البرنامج .
- تُعدّ أي قيمة تُدخل من لوحة المفاتيح نصًا من النوع **string** .
- يتم تحويل القيمة النصية إلى عدد صحيح باستخدام دالة **int** .
- يتم تحويل القيمة النصية إلى عدد عشري باستخدام دالة **float** .

اكتب تعليمات لإدخال قيم عددية باستخدام **input** للمتغيرين وإجراء عملية الجمع لهما.

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

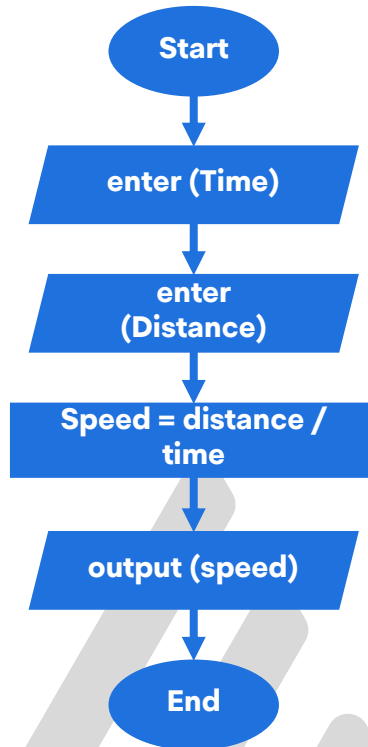
إذا تم إدخال الرقم 20 للمتغير الأول و 5 للمتغير الثاني فإن النتيجة عند تشغيل البرنامج : 205
السبب : البرنامج يعتبر أي قيمة مدخلة عن طريق **input** قيمة نصية وليس عددية .

اكتب تعليمات لإدخال قيم عددية باستخدام **input** للمتغيرين وتحويلهما إلى عدد صحيح ثم إجراء عملية الجمع لهما.

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

إذا تم إدخال الرقم 20 للمتغير الأول و 5 للمتغير الثاني فإن النتيجة عند تشغيل البرنامج : 25
لأنه تم تحويل القيمة المدخلة إلى عدد صحيح .

مثال تطبيقي : أكتب برنامج بلغة بايثون لحساب السرعة بالكيلومتر مستعيناً بخريطة التدفق التالية :



الحل :

```

1 d=float(input("Enter Distance :"))
2 t=float(input("Enter Time :"))
3 speed=d/t
4 print(speed)
  
```

رقم السطر	وظيفة التعليمة
1	إدخال قيمة من لوحة المفاتيح باستخدام input وتحويلها لعدد عشري ثم تخزينها بالمتغير d
2	إدخال قيمة من لوحة المفاتيح باستخدام input وتحويلها لعدد عشري ثم تخزينها بالمتغير t
3	حساب السرعة بالمتغير speed ويساوي قسمة المسافة على الوقت
4	طباعة النتيجة وهي السرعة



ورقة عمل 6



ورقة عمل 7





السلاسل النصية Strings

مجموعة من الأحرف والأرقام والرموز تكتب داخل الرمزين " " أو ' ' .

أولاً : عمليات السلاسل النصية :

1. تعريف وتخصيص وطباعة السلاسل النصية : يتم بعلامات اقتباس **فردية** أو **زوجية** .

النتيجة عند التشغيل

Free Kuwait

التعليمات

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

2. الفهرسة **Indexing** : الوصول إلى موقع الحرف في سلسلة نصية [x] حيث تبدأ عملية الفهرسة من **صفر**

F
r
e

```
1 v= "Free Kuwait"
2 print(v[0])
3 print(v[1])
4 print(v[2])
```

3. **القطع Slicing** : يمكن الحصول على جزء من السلسلة النصية بأرقام الفهارس [x:y] وتكتب بالصيغ التالية :

الوصف	الصيغة	مثال
تعيد النص من x حتى ما قبل y	str[x:y]	v[5:11]
تعيد النص من البداية حتى ما قبل y	str[:y]	v[:5]
تعيد النص من x حتى النهاية	str[x:]	v[5:]

Free
Kuwait
ee Ku

```
1 v= "Free Kuwait"
2 print(v[:5])
3 print(v[5:])
4 print(v[2:7])
```

4. **جمع السلاسل النصية Concatenation** : يستخدم معامل عملية الجمع + لجمع السلاسل النصية .

Free Kuwait

```
1 v= "Free"
2 v2="Kuwait"
3 print(v + ' ' +v2 )
```

5. **تكرار السلاسل النصية** : تكرار سلسلة نصية باستخدام معامل الضرب * .

Kuwait Kuwait Kuwait

```
1 print('Kuwait ' * 3 )
```

6. **مقارنة السلاسل النصية** : تتم عملية المقارنة من خلال الرمز == .

False

```
1 name1="UULA"
2 name2="uula"
3 print(name1==name2)
```



ثانياً : دوال السلاسل النصية :

1- **len()** : للحصول على طول السلسلة النصية .

التعليمات

النتيجة عند التشغيل

11

```
1 v="Free Kuwait"
2 print(len(v))
```

2- **capitalize()** : تحول الحرف الأول فقط إلى حرف كبير.

Hi man

```
v="hi man"
print(v.capitalize())
```

3- **upper()** : تحول جميع الأحرف في السلسلة النصية إلى أحرف كبيرة.

HI MAN

```
1 v="hi man"
2 print(v.upper())
```

4- **lower()** : تحول جميع الأحرف في السلسلة النصية إلى أحرف صغيرة.

welcome

```
1 v1="WELCOME"
2 print(v1.lower())
```

5- **find()** : تعطي رقم الحرف او بداية كلمة محددة في السلسلة النصية .

7

```
1 v="Try To Pray Every Day "
2 print(v.find('Pray'))
```

6- **replace()** : استبدال نص بنص آخر في السلسلة النصية.

hi uula

```
1 v="hi sa3d "
2 print(v.replace( __old: 'sa3d', __new: 'uula'))
```

7- **f-string()** : إنشاء سلسلة نصية تتضمن قيماً من أنواع بيانات مختلفة .

Name :Ahmad, Age:24

```
1 v1='Ahmad'
2 age=24
3 m=f'Name : {v1}, Age:{age}'
4 print(m)
```

8- **int()** : تحويل السلسلة النصية إلى عدد صحيح

First Number : 15
Second Number : 23
38

```
1 v1=input('First Number :')
2 v2=input('Second Number : ')
3 print(int(v1)+int(v2))
```

9- float() : تحويل السلسلة النصية إلى عدد عشري.

التعليمات

```
1 v1=input('First Number :')
2 v2=input('Second Number : ')
3 print(float(v1)+float(v2))
```

First Number : 15
Second Number : 23
38.0



10- count() : إيجاد تكرار سلسلة نصية (عدد تكرار كلمة أو حرف بالجملة) .

```
1 d="Try To Be Or Not To Be "
2 print(d.count("Be"))
```

2



11- center() : طباعة السلسلة وسط رمز مكرر بعدد معين .

```
1 v1='Kuwait'
2 print(v1.center( __width: 20, __fillchar: '*' ))
```

*******Kuwait*******



ورقة عمل 8



ورقة عمل 9



ورقة عمل 10



صفحة معلمي الكويت



الشروط Conditions

يُستخدم الشرط لاتخاذ قرار وتنفيذ تعليمات بناءً على تحقق شرط محدد.

1- عمليات المقارنة

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

تذكر:

- معامل = : إعطاء (تحديد) قيمة للمتغير .
- معامل == : مقارنة بين عنصرين **Comparison** .

مثال : اكتب التعليمات لطباعة نتيجة مقارنة المتغير **v_number** بالرقم **100** وكذلك **80**

النتيجة عند التشغيل

التعليمات

False

True

```

1 v_number = 80
2 print (v_number == 100 )
3 print(v_number == 80 )

```

مثال : إذا قيمة المتغير **x** هي **40** وقيمة المتغير **y** هي **20** فإن نتيجة عمليات المقارنة بينهما حسب الجدول التالي :

النتيجة	العملية
False	$x == y$
True	$x != y$
False	$x < y$
False	$x <= y$
True	$x > y$
True	$x >= y$

2- العمليات المنطقية

- نتيجة المعامل **and** تكون **True** إذا تحقق الشرطان .
- نتيجة المعامل **or** تكون **True** إذا تحقق أحد الشرطين على الأقل .
- المعامل **not** النفي أو عكس القيمة .

مثال : اكتب التعليمات لتخصيص قيمة **True** للمتغير **x** و **y** وطباعة نتيجة المعامل **and**

النتيجة عند التشغيل

التعليمات

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

True ←

مثال : اكتب التعليمات لتخصيص قيمة **True** للمتغير **x** وقيمة **False** للمتغير **y** وطباعة نتيجة المعامل **and**

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

False ←

مثال : اكتب التعليمات لتخصيص قيمة **True** للمتغير **x** و **y** وطباعة نتيجة المعامل **or**

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

True ←

مثال : اكتب التعليمات لتخصيص قيمة **True** للمتغير **x** وقيمة **False** للمتغير **y** وطباعة نتيجة المعامل **or**

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

True ←

مثال : اكتب التعليمات لتخصيص قيمة **True** للمتغير **x** وقيمة **False** للمتغير **y** وطباعة نتيجة المعامل **not** لكل منهما

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

False ←
True ←

مثال : اكتب التعليمة لطباعة نتيجة نفي المتغير **z**

```
1 z=False
2 print(not(z))
```

True ←

3- العمليات الشرطية

- **if** : لاختبار شرط منطقي وتنفيذ تعليمات برمجية **إذا تحقق الشرط** .
- **else** : تستخدم لتحديد ما يجب تنفيذه **إذا لم يتحقق الشرط** .
- **elif** : تستخدم لإضافة شروط متعددة ويتم تنفيذ كتلة التعليمات إذا تحقق أحد الشروط .

مثال : اكتب التعليمات بحيث يطبع كلمة **Excellent** إذا الدرجة **st_grade** المدخلة أكبر من أو تساوي **90**

```
1 st_grade =int(input("Enter Student Grade : "))
2 if st_grade >=90 :
3     print("Excellent")
```

مثال : اكتب التعليمات بحيث يطبع كلمة **Excellent** إذا الدرجة **st_grade** المدخلة أكبر من أو تساوي **90** وإذا لم تكن كذلك اطبع **Good Luck**

```
1 st_grade =int(input("Enter Student Grade : "))
2 if st_grade >=90 :
3     print("Excellent")
4 else :
5     print("Good Luck !!! ")
```

مثال : اكتب التعليمات بحيث يطبع تقدير الطالب حسب درجته **st_grade** بحيث إذا الدرجة أكبر من أو تساوي **90** يطبع **Excellent** وإذا أكبر من أو تساوي **70** يطبع **Good** وإذا أكبر من أو تساوي **50** يطبع **Acceptable** غير ذلك يطبع **Failed** .

```
1 st_grade =int(input("Enter Student Grade : "))
2 if st_grade >=90 :
3     print("Excellent")
4 elif st_grade >=70:
5     print("Good")
6 elif st_grade >=50 :
7     print("Acceptable")
8 else :
9     print("Failed")
```

💡 **تذكر :**

- العملية الشرطية **elif** : تقابل **else if** وتعني عدم تحقق الشرط السابق والتحقق من شرط إضافي .
- عند استخدام العملية الشرطية **else** في نهاية العملية الشرطية **elif** : تعني عدم تحقق جميع الشروط السابقة .



مثال تطبيقي : اكتب التعليمات لإظهار العدد الأقصى للإلكترونات حسب رقم مستوى الطاقة

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

```

1 element_symbol =input("Enter Element Symbol :")
2 if element_symbol == "K":
3     print(2)
4 elif element_symbol == "L":
5     print(8)
6 elif element_symbol == "M" :
7     print(18)
8 elif element_symbol == "N" :
9     print(32)
10 else :
11     print("please enter symbol : K - L- M - N ")

```



ورقة عمل 11



ورقة عمل 12



ورقة عمل 13



صفحة معلمي الكويت



التكرار Loops

تنفيذ مجموعة من التعليمات بشكل متكرر .

1. الحلقة التكرارية **while** .
2. الحلقة التكرارية **for** .

1- الحلقة التكرارية **while** :

القاعدة العامة

while condition : ما دام الشرط صحيح نفذ

code block to be executed until the condition becomes False

التعليمات التي يعاد تنفيذها حتى يصبح الشرط خطأ

مثال : أكتب التعليمات لطباعة الأعداد من 1 إلى 4 باستخدام دالة **while** .

النتيجة عند التشغيل

1
2
3
4

التعليمات

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

تذكر :

- الشرط في الحلقة التكرارية **while** عادة يتضمن متغير يحدد له قيمة ابتدائية .
- الخروج من الحلقة التكرارية **while** يكون بطريقتين : -
- عند تحول الشرط إلى False .
- استخدام التعليمة البرمجية **break**

مثال : في الحلقة التكرارية **while** التالية المطلوب كتابة التعليمات للخروج من الحلقة عندما تصبح **c** تساوي 5 .

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

بعد التعديل

التعليمات تطبع من 1 حتى 4 ويتم الخروج من الحلقة التكرارية عند رقم 5

```
1 c=1
2 while c<=10 :
3     print(c)
4     c=c+1
```

قبل التعديل

التعليمات تطبع من 1 حتى 10

الحلقات التكرارية while اللانهائية :

تكرر تنفيذ التعليمات البرمجية إلى ما لا نهاية .

مثال : اكتب التعليمات لحلقة تكرارية لا نهائية لطباعة كلمة **uula** بحيث تتكرر ما لا نهاية .

التعليمات

```
1 while True :
2     print("uula")
```

النتيجة عند التشغيل

سيتم طباعة كلمة **uula**
بشكل لا نهائي

مثال تطبيقي : اكتب برنامج يظهر رسالة خطأ عند محاولة المستخدم إدخال كلمة المرور **ثلاث** مرات بصورة خاطئة .

```
1 n=1
2 correct_password='12345'
3 while n<4 :
4     password=input("Enter Password :")
5     if password ==correct_password :
6         print("You Entered Correct Password")
7         break
8     n=n+1
9 else :
10    print("You Entered Password Three Times Wrong !!! ")
```

تفسير التعليمات :

رقم السطر	وظيفة التعليمة
1	تحديد قيمة ابتدائية للمتغير n بالرقم 1
2	تحديد قيمة كلمة السر الصحيحة 12345
3	حلقة تكرارية بشرط قيمة المتغير n أقل من 4
4	استقبال كلمة السر من المستخدم وتخزينها بالمتغير password
5	مقارنة كلمة السر المدخلة مع كلمة السر الصحيحة
6	طباعة جملة بأن الكلمة المدخلة صحيحة إذا تحقق الشرط في سطر 5
7	الخروج من الحلقة التكرارية
8	زيادة قيمة المتغير n بواحد
9	إذا لم يتم إدخال كلمة السر بشكل صحيح ثلاث مرات .
10	طباعة جملة بأنه تم الإدخال ثلاث مرات خطأ .



2- الحلقة التكرارية For :

تستخدم لتكرار تنفيذ تعليمات بعدد محدد .

الحلقة التكرارية for :

القاعدة العامة

for variable in iterable : يستخدم متغير يعتبر عداد يحدد له بداية ونهاية

code block to be executed for each element

التعليمات التي تنفذ لكل قيمة من عند البداية حتى النهاية

مثال : اكتب التعليمات لطباعة الأرقام من 0 إلى 3 مستخدماً الحلقة التكرارية **for** .

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

مثال : اكتب التعليمات لطباعة الأرقام من 6 إلى 10 مستخدماً الحلقة التكرارية **for** .

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

مثال : اكتب التعليمات لطباعة الأرقام التي أكبر من 5 مستخدماً الحلقة التكرارية **for** حيث البداية 1 والنهاية 11

```
1 for x in range(1,11) :  
2     if x > 5 :  
3         print(x)
```

تذكر :

صيغة دالة **range** : تستخدم بثلاثة أشكال :

range(start,stop,step)

- **range(11)** : الرقم بين الأقواس يعتبر نهاية التكرار .
- **range(1,11)** : الرقم الأول البداية والرقم الثاني حد التوقف ولا يدخل ضمن التكرار .
- **range(1,11,3)** : الرقم الأول البداية والرقم الثاني حد التوقف للتكرار والثالث الزيادة في كل مرة .

مثال : اكتب التعليمات لإدخال الاسم ثم طباعة كل حرف في سطر منفصل .

```
1 v_name = input("Enter Your Name : ")  
2 for ch in v_name :  
3     print(ch)
```

ملاحظة :

عند التعامل مع النصوص (السلاسل النصية) لا نحتاج إلى كتابة كلمة **range** في التكرار .

تتجاوز ما تبقى من تعليمات الدورة الحالية وتنتقل إلى الدورة التالية .

مثال : اكتب التعليمات لطباعة الأرقام الزوجية مستخدماً الحلقة التكرارية **for** و **while** .
تعليمة **continue** مع الحلقة التكرارية **while**

```

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

```

تعليمة **continue** مع الحلقة التكرارية **for**

```

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

```

تذكر : 

تعليمة **continue** تنتقل للقيمة التالية في التكرار عند تحقق شرط معين .

ورقة عمل 14



ورقة عمل 15



صفحة من مجلة



تصحيح الأخطاء والاستثناءات Debugging and Exceptions

أولاً : الأخطاء

1- الأخطاء اللغوية Syntax Errors

- من الأمثلة على ذلك :
- مسافة إضافية أو ناقصة بين الكلمات والرموز .
- استخدام الأقواس أو علامات الاقتباس بشكل غير صحيح .

2- أخطاء وقت التشغيل Runtime Errors

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

3- الأخطاء المنطقية Logic Errors : هي الأخطاء التي لا يتعرف عليها البرنامج وتؤدي إلى نتائج غير متوقعة

- ومنها على سبيل المثال :
- استخدام شرط غير صحيح .
- استخدام العمليات الحسابية بشكل غير صحيح .

تذكر

- الخطأ اللغوي : يظهر أسفل منه خط بلون أحمر وعند التشغيل يتم تحديد السطر ومكان الخطأ من قبل برنامج بايثون.
- السؤال في الاختبار عادة يكون تصحيح الخطأ اللغوي .

مثال : صحح الأخطاء في التعليمة بالسطر الثاني

عدد الأخطاء تقديري وغير دقيق

ظهور خط أحمر عند الخطأ

عند تشغيل البرنامج تظهر الشاشة التالية :

رقم السطر الذي يوجد فيه خطأ

مؤشر على مكان الخطأ

وجود أخطاء

Process finished with exit code 1

الحل : حذف القوس الزائد من التعليمة .

مثال : صح الأخطاء في التعليمة بالسطر الثاني للتحقق من أن قيمة x تساوي 10 :

<pre>1 x=10 2 if x==10 : 3 print("Welcome")</pre>		<pre>1 x=10 2 if x=10 : 3 print("Welcome")</pre>
---	--	--

الحل : الخطأ اللغوي عند التحقق من قيمة معينة في تعليمة if يجب أن يكتب == عند المقارنة .

أمثلة على الأخطاء اللغوية Syntax

الخطأ	التعليمة
عدم كتابة نقطتين في نهاية سطر تعليمة if	<pre>1 if x < 0 2 print (x)</pre>
عدم إغلاق القوس	<pre>1 print(" Hello, UULA "</pre>
عدم إزاحة تعليمة print جهة اليمين حتى يفهم البرنامج بأنها تابعة لتعليمة for	<pre>1 for c in range(101) : 2 print(c)</pre>

تذكر

- كتابة نقطتين : في نهاية سطر تعليمة كل من (if - else -elif - while - for) .

ثانياً : الاستثناءات

يمكن استخدام try/except لمعالجة خطأ متوقع كي يستمر البرنامج في التنفيذ بدلاً من التوقف .

أنواع الاستثناءات : TypeError - IndexError - NameError- ValueError - ZeroDivisionError

تنقسم try/except إلى :

- try :** يشمل التعليمات البرمجية التي قد تحدث فيها أخطاء .
- except :** معالجة الأخطاء التي قد تحدث في كتلة try .
- else :** إذا لم يحدث خطأ في كتلة try .
- finally :** لتنفيذ التعليمات سواء حدث أو لم يحدث خطأ في كتلة try .

الاستثناءات المحددة :

- ValueError :** خطأ عند محاولة تحويل نص غير صالح إلى نوع رقمي مثال int('abcd') .
- ZeroDivisionError :** خطأ يعبر عن قسمة عدد على صفر .

مثال : التعليمات التالية تحسب قيمة مشاركة كل طالب في تكلفة الرحلة المدرسية

```

1  try :
2      total_cost = float(input("Enter Total Price"))
3      students = int(input("Enter Number participating students"))
4      share= total_cost / students
5  except ValueError :
6      print("Please Enter numbers !!! " )
7  except ZeroDivisionError :
8      print("Cannot divide by zero " )
9  else :
10     print(share)
11 finally:
12     print("program ended")

```

وظيفة التعليمات	الكتلة
إدخال التكلفة الإجمالية وتحويلها إلى عدد عشري	try
إدخال عدد الطلاب المشاركين وتحويله إلى عدد صحيح	
معالجة خطأ الإدخال عند إدخال نص بدل رقم بطباعة جملة تفيد ذلك	except ValueError
معالجة خطأ القسمة على صفر بطباعة جملة تفيد ذلك	except ZeroDivisionError
عند عدم حدوث خطأ يتم طباعة قيمة مشاركة كل طالب share	else
بجميع الأحوال سواء حدث خطأ أم لا ، تطبع جملة program ended	finally



ورقة عمل 16



ورقة عمل 17



مُعَلِّمٌ كَوَيْتٌ
صَفْوَةٌ مِائِي كَوَيْتٌ

إعداد المشروع ونماذج الاختبارات

إعداد المشروع



المشروع الجزء الأول



المشروع الجزء الثاني



نماذج الاختبارات



نموذج فاينل 1



نموذج فاينل 2



نموذج فاينل 3



مُعَلِّمُ الْكُويْتِ
صَفْوَةُ مِيسْ