تم تحميل هذا الملف من موقع المناهج البحرينية

مذكرة تقن 106

موقع المناهج ⇦ المناهج البحرينية ⇦ الصف الأول الثانوي ⇦ علوم وتقانة ⇦ الفصل الثاني ⇦ الملف

تاريخ إضافة الملف على موقع المناهج: 2024-05-23 20:24:22

التواصل الاجتماعي بحسب الصف الأول الثانوي




اضغط هنا للحصول على جميع روابط "الصف الأول الثانوي"

روابط مواد الصف الأول الثانوي على تلغرام

التربية الاسلامية	اللغة العربية	اللغة الانجليزية	الرياضيات

المزيد من الملفات بحسب الصف الأول الثانوي والمادة علوم وتقانة في الفصل الثاني

1	ملخص تقن 106
2	المواضيع المطلوبة لمقرر تقن 106
3	الإجابة النموذجية لبنك أسئلة الامتحان النهائي مقرر تقن 106
4	شرح نظام الأنظمة العددية
5	الدروس المطلوبة من مقرر البرمجة بلغة البايثون مقرر تقن 106

موضوع الدرس :	الدرس 1 : الأنظمة العدديّة
التاريخ :	

التقييم

مهارات الدرس :

م	المهارة	
1	مفهوم الأنظمة العددية	
2	تعريف نظامي العد : العشري والثنائي	

م	المهارة	
3	تحويل عدد من النظام العشري الى الثنائي	
4	تحويل عدد من النظام الثنائي الى العشري	

ما هي الأنظمة العدديّة Numbering system

هي أنظمة عّد أو ترقيم تعتمد أّولا على مجموعة محّددة من الرموز لتمثيل الأعداد وثانيا على منهجّية معّينة لكتابتها وعرضها. يحتل كل رمز من رموز النظام العددي موضعا يمّثل قيمته.مثال: الرمز 1ضمن العدد 10أقل قيمة منه ضمن العدد 100

النظام العشري Decimal numbering system :

من أهّم الأنظمة العددّية الذي يعتمد على الرموز من 0إلى 9لتمثيل الأعداد العشرّية نسبة إلى العدد 10الذي يمّثل بدوره عدد الرموز من 0إلى 9. وهو النظام العددي الأكثر شيوعا والذي نعتمده في حساباتنا اليومّية وفي أنظمتنا الاقتصادية وإدا ارتنا التعليمّية ومختلف المجالات الأخرى.

النظام الثنائي Binary numbering system :

هو نظام عّد يعتمد على الرموز من 0إلى 1لتمثيل الأعداد الثنائّية نسبة إلى العدد 2الذي يمّثل بدوره عدد الرموز من 0إلى 1

تحويل عدد من النظام العشري الى الثنائي:

للتحويل من النظام العشري إلى الثنائي يتم قسمة العدد على الرقم 2 وكتابة الباقي من القسمة في تسلسل من اليسار إلى اليمين أو استخدام جدول الأعداد العشرية قوة2 .

2^7	$^6 2$	$^5 2$	$^4 2$	$^3 2$	$^2 2$	$^1 2$	$^0 2$
128	64	32	16	8	4	2	1

طريقة استخدام جدول الأعداد العشرية قوة2:

2^7	$^6 2$	$^5 2$	$^4 2$	$^3 2$	$^2 2$	$^1 2$	$^0 2$
128	64	32	16	8	4	2	1
		1	0	1	1	1	1

لايجاد نتيجة تحويل الرقم (47)$_{10}$ الى عدد ثنائي اتبع الخطوات التالية:

ابدأ بوضع الرقم 1في الخانة أسفل أقرب عدد عشري للعدد المطلوب تحويله. في حالتنا هذه أقرب عدد عشري للعدد 47 ضمن الجدول أعلاه هو 32 ، بعدها اقوم بالتالي:

- انقاص العدد 32 من الرقم 47 سيكون الناتج هو 15 ، ابحث عن اقرب وأصغر رقم من 15 سيكون 8 لذا سأضع 1 تحت الرقم 8 .
- انقاص الرقم من 8 من 15 سيكون الناتج 7 ابحث عن اقرب وأصغر رقم من 7 سيكون 4 لذا سأضع 1 تحت الرقم 4.
- انقاص الرقم من 4 من 8 سيكون الناتج 4 ابحث عن اقرب وأصغر رقم من 4 سيكون 3 لذا سأضع 1 تحت الرقم 3.
- اكمل العملية حتى يكون الناتج هو صفر.

تحويل عدد من النظام الثنائي الى العشري:

للتحويل من النظام الثنائي إلى العشري نستخدم أيضا جدول الأعداد العشرية قوة 2 ، نقوم بكتابة الرقم الثنائي في خانات الجدول، مثلا لو طلب مني تحويل العدد 100010 الى عدد عشري ، سأقوم بكتابة الارقام في الجدول ، بعدها اقوم بجمع الأعداد التي تحتها الرقم 1 فقط ، وهي 2 + 32 سيكون الناتج هو 34 .

2^7	^{6}2	^{5}2	^{4}2	^{3}2	^{2}2	^{1}2	^{0}2
128	64	32	16	8	4	2	1
		1	0	0	0	1	0

الملاحظات :

موضوع الدرس :	الدرس 2: الذكاء الاصطناعي
التاريخ :	

التقييم

مهارات الدرس :

م	المهارة	
1	مفهوم الذكاء الاصطناعي وأهميته	
2	مفهوم التعلم الآلي	

م	المهارة	
4	استخدامات التعلم الآلي	
5	لوغاريتمات التعلم الآلي	

تعريف الذكاء الاصطناعي:

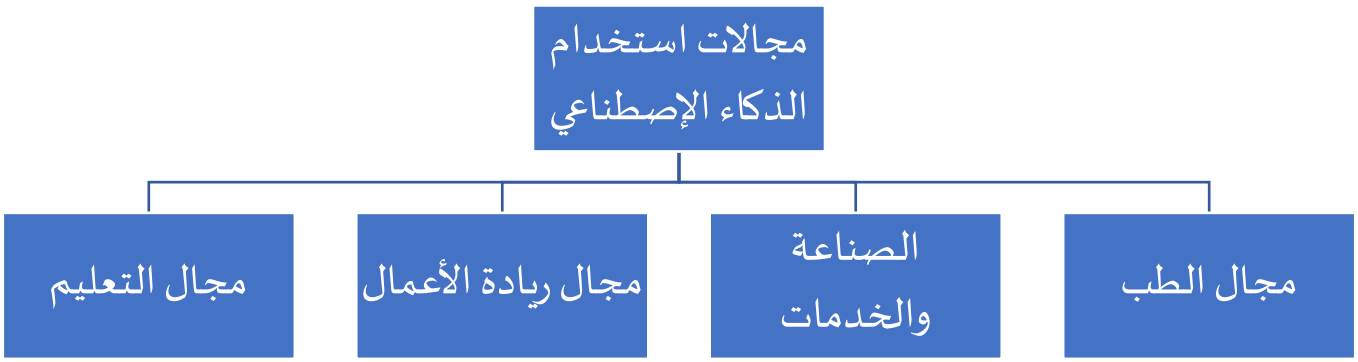
يعرف الذكاء الاصطناعي بأنه الذكاء الذي تبديه الآلات والبرامج بما يحاكي القدرات الذهنية البشرية وأنماط عملها، مثل القدرة على التعلم والاستنتاج ورد الفعل على أوضاع لم تبرمج في الآلة.

هدف الذكاء الاصطناعي:

إنّ الهدف الرئيسي من الذكاء الاصطناعي يكمن في اتخاذ القرار الصحيح في أقل وقت وبأقل التكاليف ممّا سيساعد المؤسسات على:

- الزيادة في الإنتاج.
- التنبؤ للتهديدات الأمنيّة المستقبلية.
- تقليل التكلفة وربح الوقت.
- سرعة اتخاذ القرار.
- كما يمكن أيضا الاستفادة من الذكاء الاصطناعي في مجال التعليم من خلال تحليل البيانات وتحديد الموضوعات والدروس التي يجب إعادة تقييمها ووضع أفضل برنامج تعليمي للطالب.

مجالات استخدام الذكاء الاصطناعي:



مفهوم التعلم الآلي

التعلم الآلي (Machine Learning) هو فرع من فروع الذكاء الاصطناعي يركز على تطوير النظم التي تمكن الأجهزة الحاسوبية من التعلم والتكيف تلقائيًا بناءً على البيانات والتجارب السابقة، دون الحاجة إلى برمجة صريحة.

أهمية التعلم الآلي وعلاقته بالذكاء الاصطناعي:

1. **يساعد الأعمال عن طريق:**
    1. **دفع عجلة النمو.**
    2. **فتح سبل إيرادات جديدة.**
    3. **حل المشكلات الصعبة.**

طريقة عمل التعلم الآلي:

- **يخمن من خلال** تحليل كمّ كافٍ من البيانات علاقة رياضية بين المخرجات والمدخلات
- استخدام اللوغاري**تمات المحددة بناءً على المدخلات و المخرجات**

أنواع لوغاريتمات التعلم الآلي:

1. التعلم الآلي تحت الإشراف (Supervised Learning)
2. التعلم الآلي بدون اشراف (Unsupervised Learning)
3. التعلم الآلي تحت الإشراف الجزئي(Semi-Supervised Learning)
4. التعلم الآلي المعزز (Reinforcement Learning)

الملاحظات :

موضوع الدرس :	الدرس 3: الخوارزميات ودورة تطوير البرمجيات
التاريخ :	

التقييم

مهارات الدرس :

م	المهارة	
1	مفهوم الخوارزميات	
2	كتابة خوارزمية لحل مشكلة ما	

م	المهارة	
3	مراحل تطوير البرمجيات	
4		

مفهوم الخوارزميات:

الخوارزمية تُمثل تسلسلا منطقيا قصد الوصول إلى الهدف أو النتيجة المطلوبة.

كتابة خوارزمية لحل مشكلة :

لكتابة أي خوارزمية نحتاج الى كتابتها على 3 مراحل :

1- **المدخلات:** نوضح خلالها جميع المدخلات (المتغيرات و الثوابت) التي سنتعامل معها خلال البرنامج.

2- **المعالجة الحسابية والمنطقية:** يتم تحديد العمليات الحسابية ان وجدت أو المقارنة .

3- **المخرجات:** تحديد ماذا سيتم عرضه .

مثال: اكتب خوارزمية للبرنامج التالي:

برنامج يستقبل عددان ويقوم بتخزينهما في متغيرين، يتم جمع العددين و عرض الناتج .

المدخلات:	ادخال العدد الأول وحفظه في المتغير x ادخال العدد الثاني وحفظه في المتغير y
المعالجة:	جمع العددين x+y وحفظ الناتج في المتغير result
المخرجات:	طباعة فيمة المتغير result

مراحل تطوير البرمجيات:

1	التحليل Analysis	في هذه المرحلة يعرّف المبرمج المشكلة المراد حلّها ويحدّد الاحتياجات والمعطيات المطلوبة من البرنامج وطريقة معالجتها والمخرجات الناتجة.
2	التصميم Design	بعد تحديد المدخلات وطريقة معالجتها والمخرجات المنتظرة من البرمجيّة ، تكون هذه المرحلة لتحديد طريقة إنشاء البرمجية من خلال الخوارزميات Algorithms
3	البرمجة Programming	تحويل الخوارزميات إلى لغة برمجة.
4	التجربة Testing	تكون بصورة مكثّفة من خلال تجربة المدخلات بكلّ أنواعها لتحديد المشاكل الناتجة عنها وتتبّعها وتصحيحها بحيث تكون البرمجيّة خالية من أيّة شو ائب وقادرة على التعامل مع المدخلات في جميع الحالات والحصول على المخرجات المطلوبة بدقّة.

موضوع الدرس :	الدرس 4: الخرائط التدفقية
التاريخ :	

التقييم

مهارات الدرس :

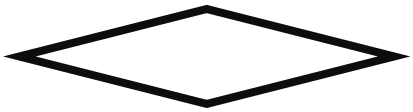
م	المهارة	
1	مفهوم الخرائة التدفقية	
2	التعرف على أشكال الخرائط التدفقية	

م	المهارة	
3	انشاء خريطة تدفقة لحل مشكلة برمجية	
4		

الخرائط التدفقية :

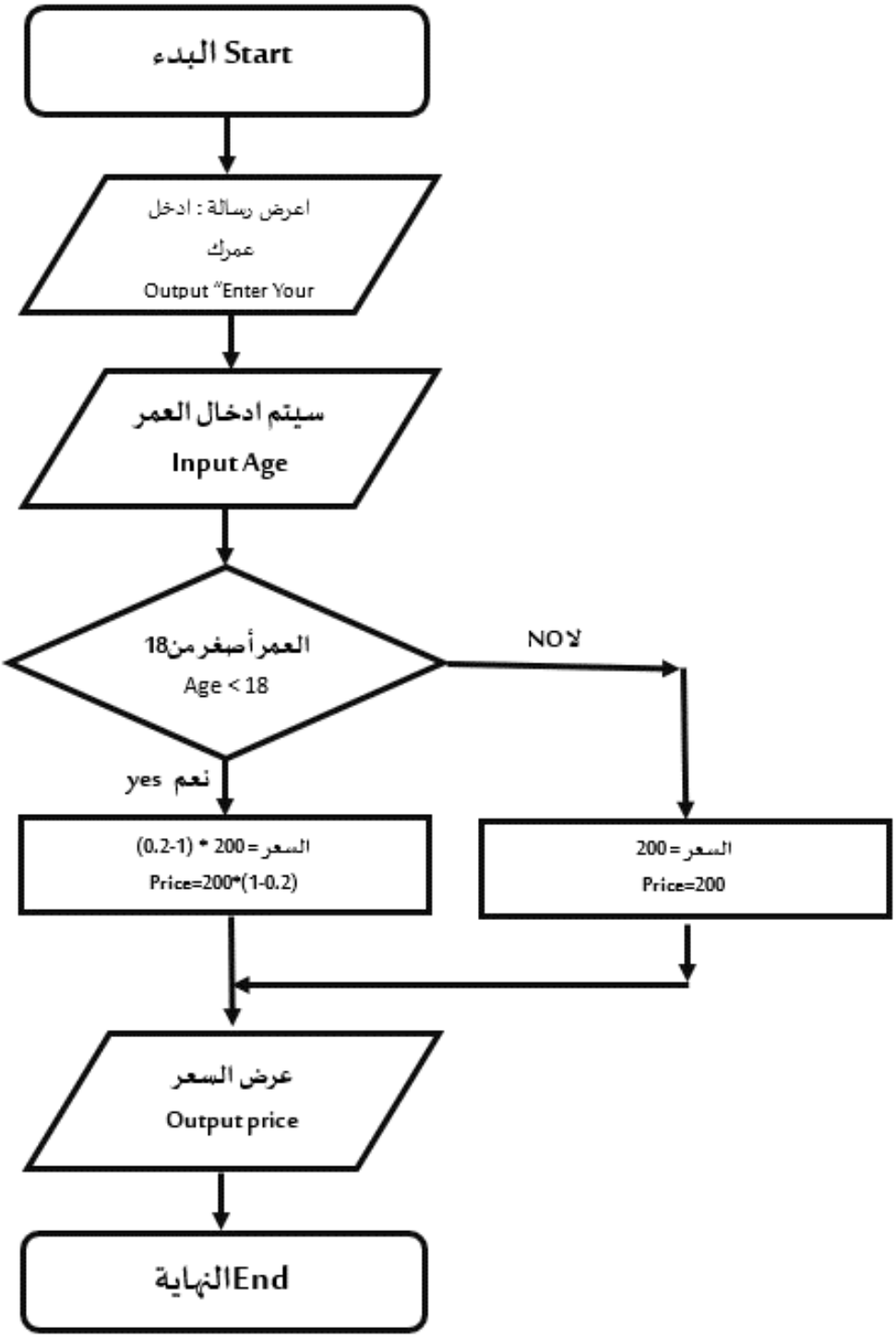
حل رسوميّ للمشكلة، حيث ترتبط مجموعة من الأشكال الهندسية بعضها ببعض في ترتيب منطقي لتسلسل الأحداث والإجراءات البرمجية

أشكال الخرائط التدفقية:

الشكل	الوظيفة	الوصف
	البداية/النهاية	يستخدم في بداية ونهاية الخريطة التدفّقيّة
	مدخلات/مخرجات	يستخدم عند إدخال المعطيات و/أو عرض المخرجات
	معالجة	عمليات حسابية | منطقيّة، تعليمة برمجيّة
	اتخاذ القرار	عندما يكون هناك اجراء سيتّخذبناء على شرط نتيجته (نعم / لا)

مثال:

يبلغ سعر تذكرة إلى دولة ما 200دب ،فإذا كان سن المسافر أقل من 18 سنة فإنه سيتمتع بـ20% خصم.



الملاحظات :

موضوع الدرس :	الدرس 5: مقدمة في البايثون
التاريخ :	

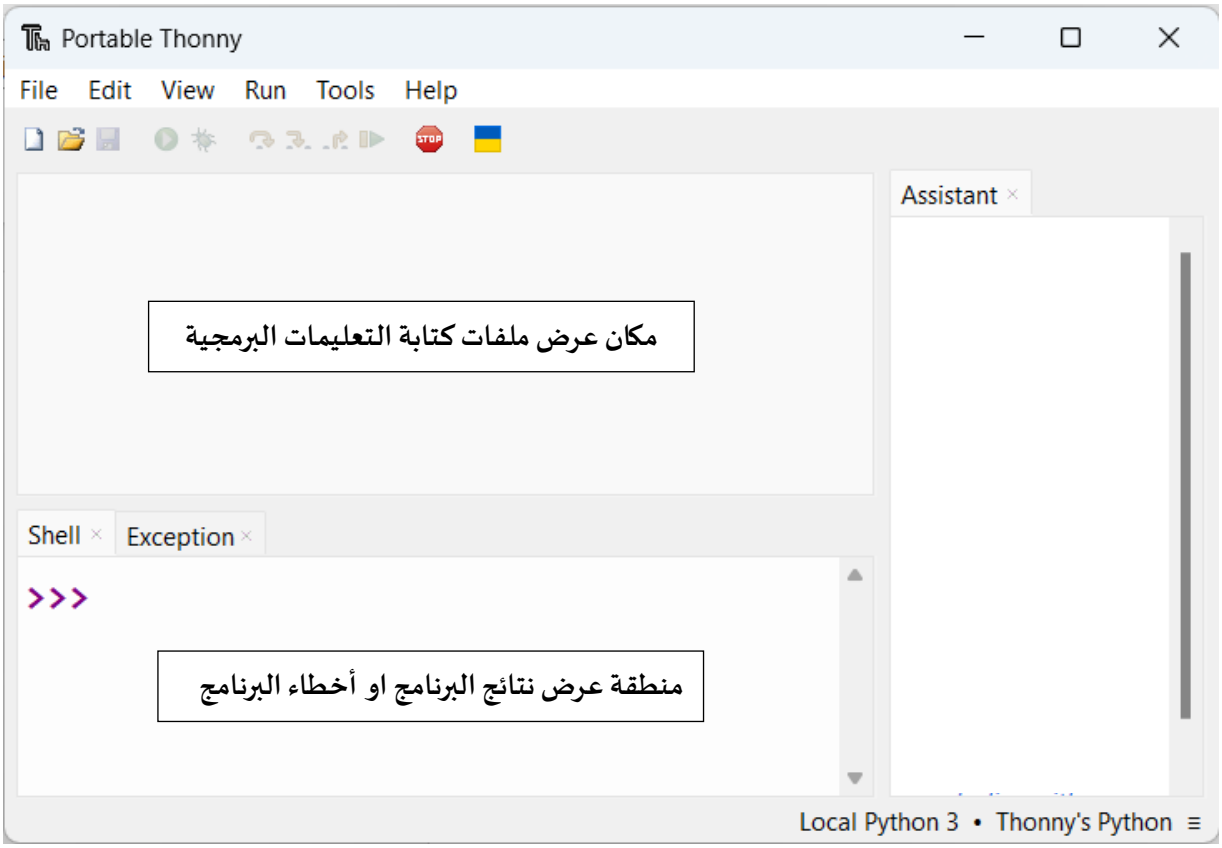
التقييم

مهارات الدرس :

م	المهارة	
1	انشاء – حفظ – فتح ملف بايثون	
2	جملة عرض/طباعة المعلومات print	
3	طباعة تعليق	

م	المهارة	
4	طباعة : سطر جديد – tab – " - "	
5	كتابة الاجراء على أكثر من سطر	

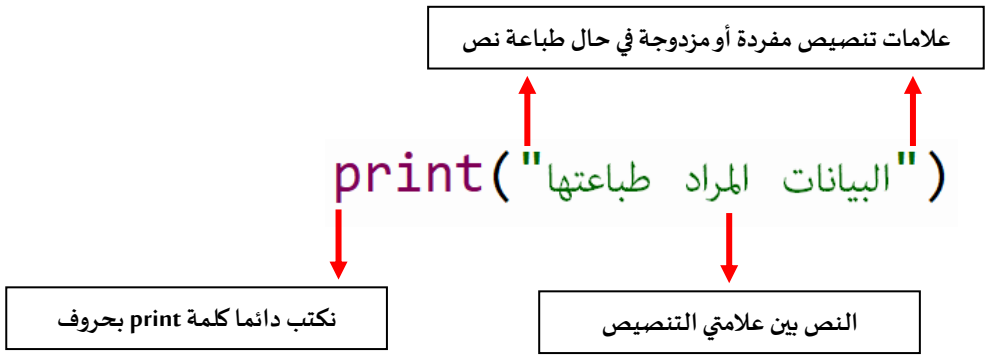
التعامل مع واجهة برنامج thony :



انشاء ملف		File → New
فتح ملف		File → Open
حفظ ملف		File → Save As - File →Save
تشغيل البرنامج		Run → Run Current Script

جملة عرض / طباعة المعلومات :

نستخدم هذه الجملة لعرض البيانات قبل/ بعد معالجتها او عرض نتيجة أي معادلة أو متغيرات ،وتكون صيغتها كالتالي:



على سبيل المثال، أود أن اطبع الرسالة الترحيبية : "Welcome" ، سنكتب:

```
1 print("Welcome")
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
 Welcome
```

ايضا يمكننا استخدام الجملة لعرض نتيجة معادلة حسابية، ويكون كالتالي:

```
1 print(5*6)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
 30
```

يمكننا طباعة نص متبوعا بمعادلة أو دالة أو حتى متغير، ويكون كالتالي:

```
1 print("The result= ",5*6)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
 The result=  30
```

ملاحظات مهمة:

- عند استخدام الجملة لعرض قيمة نصية يجب أن تكون القيمة النصية بين علامتي تنصيص مزدوجة " " أو علامتي تنصيص مفردة ' '.
- عند استخدام الجملة لعرض معادلة / رقم / متغير **لا نستخدم** علامتي التنصيص.
- عند طباعة أكثر من عنصر باستخدام جملة الطباعة يجب أن نفصل بين كل عنصر و الثاني بالفاصلة , .

طريقة ادراج تعليق :

ليتم ادراج تعليق في البرنامج ليتمكن المبرمج من كتابة ملاحظاته حتى تكون الاوامر البرمجية واضحة له ويتذكرها، علما بأن البرنامج **لا ينفذ التعليق.** يمكننا استخدام علامة # لادراج تعليق من سطر واحد ، ولو أردنا كتابة تعليق من أكثر من سطر نكتب " " " في بداية التعليق و نهايته.

```
1  #هذا تعليق من سطر واحد
2  """ هذا تعليق
3  من عدة أسطر """
```

طريقة ادراج سطر:

نستخدم print لطباعة النصوص، و سنلاحظ انه لا يمكننا ادراج سطر بين النصوص عند الضغط على زر الادخال enter ، لذا نستخدم الرمز n\ من اجل ادراج السطر في المكان الذي نريده.

فعلى سبيل المثال لو قمنا بكتابة العلامة البرمجية التالية وتشغيل البرنامج:

print("Hello Ameena")

ستكون النتيجة :

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
 Hello Ameena
>>>
```

فلو أردنا طباعة الاسم في سطر منفصل فسنستخدم الرمز n\ ، كالتالي:

```
1
2  print("Hello \n Ameena")
3
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
 Hello
 Ameena
```

طريقة ادراج علامة تنصيص مزدوجة أو مفردة:

نستخدم علامة التنصيص المزدوجة او المفردة في جملة print لطباعة النصوص، لكن بعض الأوقات نحتاج أن نقوم بطباعة علامة تنصيص لتظهر مع النص على الشاشة ، مثل النتيجة التالية:

```
>>> %Run -c $EDITOR_CONTENT
 "Hello"
>>>
```

لنحصل على النتيجة السابقة يجب أن نكتب الرمز \" للحصول على علامة تنصيص مزدوجة و \' للحصول على علامة تنصيص مفردة:

```
print(" \"علامة تنصيص مزدوجة\" ")
print(" \'علامة تنصيص مفردة\' ")
```

```
Shell  Exception

>>> %Run -c $EDITOR_CONTENT
  "علامة تنصيص مزدوجة"
  'علامة تنصيص مفردة'

>>>
```

طريقة ادراج مسافة tab:

نستخدم علامة tab لادراج مسافة محددة بين النصوص، و يتم ادراجها عن طريق استخدام الرمز \t ، مثال :

```
print("Hello My name is \t  Ameena")
```

```
Shell  Exception

>>> %Run -c $EDITOR_CONTENT
 Hello My name is          Ameena

>>>
```

كتابة الاجراء على أكثر من سطر:

نستخدم الرمز \ لتقسيم الامر على أكثر من سطر ، علما بأن الأوامر التي تحتوي على الأقواس () أو {} أو [] لا تحتاج لاضافة \

```
x=num1+\
    num2+ \
    num3
```

الملاحظات :

موضوع الدرس :	الدرس6: المتغيرات و جملة الادخال
التاريخ :	

التقييم

مهارات الدرس :

م	المهارة	
1	المتغيرات	

م	المهارة	
2	جملة الادخال	

المتغيرات:

عبارة عن مواقع في الذاكرة تخزن البيانات بشكل مؤقت ويمكن تغيير قيمتها أثناء تشغيل البرنامج، يمكننا تسمية المتغير بأسماء مختلفة لكن مع مراعاة بعض الأمور مثل:

- يتكون من حروف وأرقام، ويمكن استخدام _ اذا كان اسم المتغير من كلمتين .
- لا يجب أن يبدأ برقم.
- لا يحتوي على الرمز @ ، #
- يمكن أن يحتوي الاسم على حروف كبيرة وصغيرة، مع الانتباه الى أن أسماء المتغير حساسة لنوع الحرف، مثلا: المتغيران التاليان مختلفان TotalPay و totalpay .
- لا نستخدم الكلمات المفتاحية للغة البرمجة، مثلا : لا يمكنني تسمية متغير print لأن كلمة print تُعتبر كلمة خاصة بلغة البايثون. وهذه الكلمات المفتاحية للبايثون:

if	print	for	break	in	finally	none	not	return
elif	import	nonlocal	raise	class	or	true	try	while
else	continue	except	from	is	with	yield	input	and
assert	del	false	pass	lambda	global	exec	def	

أنواع المتغيرات:

نوع المتغير يكون بحسب نوع البيانات المسندة اليه، فلو كانت قيمة نصية فسيكون المتغير نصي str (string) و لو كان عدد صحيح فسيكون نوعه int (integer) والعدد العشري يكون float.

اسناد قيم للمتغيرات:

عملية اسناد اي قيمة للمتغير تكفي باستخدام علامة = ، فلو كان لدي متغير اسمه stName و اردنا اسناد قيمة له سيكون كالتالي:

```
stName="ameena"
```

لماذا كتبنا القيمة بين علامتي تنصيص " " ؟ لأنها قيمة نصية string

حين نرغب باسناد قيمة رقمية للمتغير نكتفي بكتابة القيمة دون علامتي تنصيص ، و لو تم كتابة قيمة رقمية بين علامتي تنصيص فسيتم التعامل معها على أنها نص و ليس رقم.

المتغير stMark1 من نوع int (رقمي) .	**stMark1=15**
المتغير stMark2 من نوع نصي str لأن القيمة المسندة اليه موجودة بين علامتي تنصيص " "	**stMark2="15"**

عند اسناد قيمة للمتغير بالطريقة السابقة، يعتبر ثابت

عرض قيمة المتغير:

يمكننا عرض قيمة المتغير عن طريقة استخدام جملة print() ، كالتالي:

```
1 st_mark1=15
2 print(st_mark1)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
 15
```

كما يمكن عرض قيمة نصية يليها المتغير، كالتالي:

```
1 st_mark1=15
2 print("your grade is",st_mark1)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
 your grade is 15
```

تطبيق:

1- افتح برنامج thonny ، وانشيء ملفاً احفظه باسم welcome

2- صمم برنامجاً كالاتي:

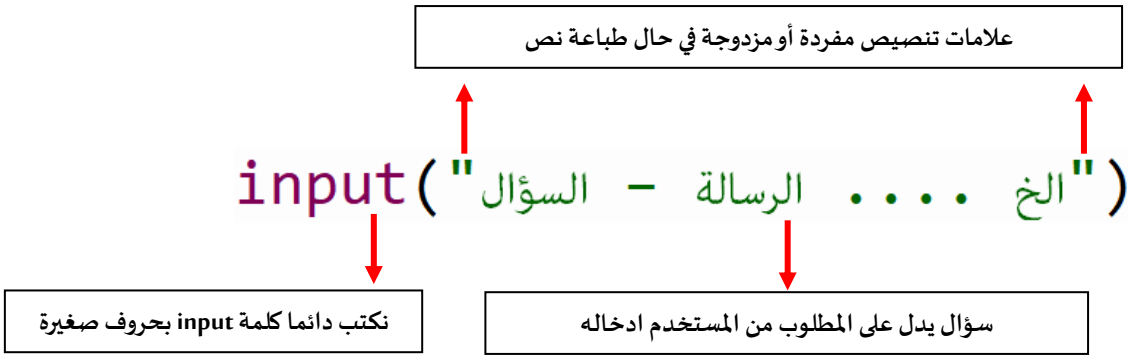
- ادخال اسم المستخدم وحفظ القيمة في المتغير mname.
- عرض الرسالة "Welcome " مع اسم المستخدم.

```
1 mname=input("Enter your name")
2 print("Welcome ",mname)
3
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
 Enter your nameAmeena
 Welcome  Ameena
>>>
```

جملة ادخال البيانات:

لا يوجد برنامج يعمل بشكل كامل دون أن يستقبل مدخلات من المستخدم حتى تتم معالجتها وعرضها. وتنفيذ هذه الجملة يجعل الحاسوب ينتظر المستخدم لادخال قيمة معينة، وتكون صيغتها:



فلو أردنا أن نطلب من المستخدم ادخال اسمه ستكون الطريقة المستخدمة:

```
input("Enter your name:")
```

عند تشغيل البرنامج سنلاحظ ظهور السؤال و عند الضغط على زر الادخال Enter سينتهي تشغيل البرنامج:

```
1 input("Enter Your Name ")
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
 Enter Your Name Ameena Mohamed
```

عند قيام المستخدم بادخال قيم عن طريق البرنامج، يجب أن يتم تخزين هذه القيم في مكان ما حتى نستخدمها لاحقا في البرمجية، لذا سنحتاج الى استخدام المتغيرات.فلوافترضنا أننا نود أن يقوم المستخدم بادخال اسمه و بالتالي حفظ الاسم في متغير ليتم استخدامه لاحقا، فستكون الطريقة :

```
stName=input("Enter your name")
```

اسم المتغير و بعده علامة =

ولعرض القيمة التي تم استخدامها سيكون عن طريق استخدام الجملة print .

مثال: صمم برنامج يقوم المستخدم بادخال عمره وتخزين القيمة في المتغير age ، عرض عمر المستخدم يسبقه العبارة "Your age is"

```
1 age=input("Enter your age ")
2 print("Your age is ",age)
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
 Enter your age 15
 Your age is  15
```

الملاحظات :

موضوع الدرس :	الدرس 7: التعامل مع الأرقام	التقييم
التاريخ :		

مهارات الدرس :

م	المهارة	
1	دالة type( )	
2	دالة int( )	
3	دالة float ( )	
4	دالة str( )	

م	المهارة	
5	دالة round()	
6	دالة abs()	
7	العمليات الحسابية	

سنقوم بتصميم برنامج لحساب ناتج جمع عددين وعرض الناتج على الشاشة، لذلك سنستخدم المتغيرn1 لتخزين العدد الأول، المتغير n2 لتخزين المتغير الثاني ، المتغير result لناتج جمع العددين. علما بأن الأعداد سيتم ادخالها من قبل المُستخدم، سيكون البرنامج كالتالي:

```
n1=input("Enter number1: ")
n2=input("Enter number2: ")
result=n1+n2
print("the result = ",result)
```

عند تشغيل البرنامج سيكون الناتج :

نلاحظ أن المستخدم قام بادخال الرقم 5 و بعدها الرقم 4 و ناتج الجمع يجب أن يكون 9 و ليس 54 كما يظهر لنا في الصورة.

```
>>> %Run -c $EDITOR_CONTENT
 Enter number1: 5
 Enter number2: 4
 the result =  54
>>>
```

السبب في ذلك أن جملة الادخال input تعتبر أي قيمة يتم ادخالها قيمة نصية، لذا عند اتمام عملية الجمع لأي قيمتين نصيتين سيتم دمج الاثنين معا.

لحل هذه المشكلة يجب أن نستخدم دوال خاصة لتحويل اي قيمة نصية الى قيمة عددية. و المقصود ب الدالة هي كتلة من الجمل البرمجيّة الجاهزة مسبقا، يتمّ استدعاؤها عند اللزوم، عادة ما ترافقها بيانات خاصّة بها تسمى " parameters " لإستخدامها للوصول إلى النتيجة المطلوبة.

دالة type( ) :

يتم استخدام هذه الدالة للتعرف على نوع البيانات، و يتم استخدامها بالطريقة التالية:

type(البيانات – اسم المتغير)

```
type("Bahrain")
```

ويجب أن نتأكد اذا كانت البيانات نصية يجب أن تكتب بين علامتي التنصيص.

الجملة البرمجية	النتيجة	نوع البيانات
st_mark="15" print(type(st_mark))	>>> %Run -c $EDITOR_CONTENT <class 'str'> >>>	نصية str
print(type("Ameena"))	>>> %Run -c $EDITOR_CONTENT <class 'str'> >>>	نصية Str
st_mark=15 print(type(st_mark))	>>> %Run -c $EDITOR_CONTENT <class 'int'> >>>	رقم صحيح int
print(type(20))	>>> %Run -c $EDITOR_CONTENT <class 'int'> >>>	رقم صحيح int
st_mark=15.0 print(type(st_mark))	>>> %Run -c $EDITOR_CONTENT <class 'float'> >>>	رقم عشري float

نلاحظ من المثال الأول في الجدول أن نوع البيانات نصية على الرغم من أن قيمة المتغير هي 15 ، و السبب في أن الرقم تم كتابته بين علامتي التنصيص " " .

الدوال الخاصة لتحويل القيم النصية الى قيم رقمية:

يمكننا ان نحول القيم النصية الى عدد صحيح او عدد عشري عن طريق الدوال التالية:

الدالة	وصف الدالة	صيغة الدالة	مثال
int( )	التحويل الى عدد صحيح int	int(نكتب الرقم – اسم المتغير - جملة الادخال)	int(x) int(15.5)
float( )	التحويل الى عدد عشري float	float(نكتب الرقم – اسم المتغير - جملة الادخال)	float(x) float(15)

طريقة تحويل أي قيمة الى عدد صحيح او عشري:

يمكننا استخدام الدالة بطريقتين ، الاولى تحويل القيمة المُدخلة مباشرة الى عدد ، كالتالي:

```
n1=int(input("Enter Number1 = "))
```

الطريقة الثانية ، عن طريق تحويل القيمة الى عدد أثناء العملية الحسابية المطلوبة، كالتالي:

```
result=int(n1) + int(n2)
```

مثال:

سنقوم بتصميم برنامج لحساب ناتج جمع عددين وعرض الناتج على الشاشة، لذلك سنستخدم المتغيرn1 لتخزين العدد الأول ونوعه عدد صحيح int، المتغير n2 لتخزين المتغير الثاني ونوعه عدد عشري float، المتغير result لناتج جمع العددين. علما بأن الأعداد سيتم ادخالها من قبل المُستخدم.

الطريقة الاولى :

تحويل المتغير n1 الى عدد صحيح و المتغير n2 الى عدد عشري مباشرة عند ادخال القيم باستخدام جملة ( ) input

```
1 n1=int(input("Enter Number1 = "))
2 n2=float(input("Enter Number2 = "))
3
4 result=n1+n2
5 print("the result =",result)
6
```

Shell × Exception ×

```
>>> %Run -c $EDITOR_CONTENT
 Enter Number1 = 5
 Enter Number2 = 6
 the result = 11.0
```

الطريقة الثانية :

تحويل المتغير n1 الى عدد صحيح و المتغير n2 الى عدد عشري ، خلال العملية الحسابية المطلوبة.

ملاحظة: استخدام هذه الطريقة تعني أن المتغير n1 و n2 سيكون نوعها رقم فقط في هذا السطر البرمجي، وفي االماكن الاخرى يعتبر نوعهما نصي.

```
1 n1=input("Enter Number1 = ")
2 n2=input("Enter Number2 = ")
3
4 result=int(n1)+float(n2)
5 print("the result =",result)
6
```

Shell × Exception ×

```
>>> %Run -c $EDITOR_CONTENT
 Enter Number1 = 5
 Enter Number2 = 6
 the result = 11.0
```

تحويل القيمة الرقمية الى قيمة نصية:

نحتاج احيانا للتعامل مع الارقام كقيمة نصية،لذا يمكننا استخدام الدالة الخاصة بالتحويل الى قيمة نصية وهي str()، وطريقة استخدامها:

str(جملة الادخال - اسم المتغير – نكتب الرقم)

مثال:

```
num=int(input("Enter Number "))
print(str(num))
```

دالة تقريب الاعداد:

تُستخدم دالة round( ) للتقريب ، وطريقة استخدامها:

round(نكتب الرقم – اسم المتغير)

```
1 n1=15.37
2 print(round(n1))
3
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
15
```

دالة القيمة المطلقة :

دالة abs( ) للحصول على القيمة المطلقة لأي عدد و المقصود بها القيمة الموجبة ،وطريقة الاستخدام:

abs(- نكتب الرقم – اسم المتغير)

```
1 a=-12
2 print(abs(a))
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_C
12
```

العمليات الحسابية في لغة البايثون :

العامل الحسابي	الوصف
+	للجمع في حال البيانات الرقميّة
+	للربط في حال البيانات النصيّة
-	للطرح
*	للضرب
/	للقسمة
//	للحصول على الناتج الصحيح من عمليّة القسمة
%	للحصول على باقي عمليّة القسمة
**	للحصول على قوّة العدد

للبدء في كتابة المعادلات الحسابية، يجب الانتباه الى ترتيب العمليات و هي كالتالي:

الترتيب	1	2	3	4
العلامة	( )	**	* و /	+ و -
الوصف	الأقواس	قوة العدد (الأُس)	الضرب و القسمة	الجمع والطرح

مثال:

العامل الحسابي	مثال	النتيجة
+ جمع الاعداد	x=3 y=6 r=x + 6 print(r)	9
- طرح الاعداد	x=13 y=6 r=x - y print(r)	7
* ضرب الاعداد	x=8 y=5 r=x * y print(r)	40
/ قسمة الاعداد	x=63 y=5 r=x / y print(r)	12.6

//
ناتج القسمة بحيث يكون
عدد صحيح من غير اعداد عشرية
	x=63
	y=5
	r=x // y
	print(r)
		12

% ايجاد باقي القسمة	x=63 y=5 r=x % y print(r)	3
** قوة العدد (الأس)	x=5 y=2 r=x ** y print(r)	25

الملاحظات :

موضوع الدرس :	الدرس 8: عوامل المقارنة و العوامل المنطقية
التاريخ :	

التقييم

مهارات الدرس :

م	المهارة	
1	عوامل المقارنة	

م	المهارة	
3	العوامل المنطقية	

عوامل المقارنة:

نحتاج الى استخدام عوامل المقارنة لنقارن بين قيمتين، و تكون كالتالي:

رمز عامل المقارنة	الوصف	مثال	ملاحظات
==	يساوي	X == 5 cnum == "Bahrain" Y == X	1- يمكننا المقارنة بين : • متغير و قيمة • متغير ومتغير 2-يمكننا استخدام معاملات المقارنة للقيم النصية ايضا . 3- قراءة المعاملات من اليسار الى اليمين 4-يجب أن تكون المقارنة بين عدد و عدد أو نص ونص ، لا يمكنك أن تقارن بين عدد ونص.
!=	لا يساوي	X != 5 Y != X	
<	اصغر من	X < 5 Y <= X	
<=	أصغر من أو يساوي	X <= 5 Y <= X	
>	أكبر من	X > 5 Y > X	
>=	أكبر من أو يساوي	X >= 5 Y>=X	

يمكننا طباعة نتيجة المقارنة مباشرة باستخدام جملة print ، كالتالي:

```
1 print(2 > 5)
```

```
Shell × Exception ×
>>> %Run -c $EDITO
False
```

```
1 print(2 < 5)
```

```
Shell × Exception ×
>>> %Run -c $EDITO
True
```

نلاحظ أن نتيجة المقارنة كانت True او False ، وتعتبر من أنواع البيانات و تسمى Boolean او bool كاختصار لها.

يمكننا تخزين نتيجة المقارنة في متغير وسيكون نوع المتغير bool ، نلاحظ المثال التالي:

```
1 x= 2 > 5
2 print(x)
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
False
```

المقارنة قد تكون بين متغير و قيمة:

في المثال التالي تم ادخال رقم من قبل المستخدم وطباعة نتيجة مقارنته اذا كان أكبر من 5 .

```
1 x=int(input("Number= "))
2 print(x > 5)
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
Number= 2
False
```

و قد تكون المقارنة بين متغير و متغير :

في المثال التالي تم ادخال رقميين من قبل المستخدم و طباعة نتيجة المقارنة اذا العدد الاول أكبر من العدد الثاني:

```
1 num1=int(input("Enter number 1 : "))
2 num2=int(input("Enter number 2 : "))
3 print(num1 > num2)
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
Enter number 1 : 5
Enter number 2 : 2
True
```

ألاحظ في المثال السابق ، عند ادخال العددين و المقارنة بينهما ، اظهر الاجابة True ، لأن العدد الاول أكبر من العدد الثاني.

العوامل المنطقية:

تُستخدم العوامل المنطقية للمقارنة بين شرطين في نفس الوقت، وتكون نتيجة استخدامها اما تحقق **True** أو لم يتحقق **False**.

الرمز	المعنى	الوصف	مثال	النتيجة
and	و	عند تحقق الشرطين معاً في نفس الوقت ستكون النتيجة true، عدم تحقق شرط واحد يعطي نتيجة false.	5 > 2 and 5 < 10	True
			6 > 10 and 5 < 10	False
			5 > 8 and 10 < 1	False
or	أو	عند تحقق شرط واحد ستكون النتيجة true، عدم تحقق أي شرط ستكون النتيجة false.	5 > 2 or 5 < 10	True
			6 > 10 or 5 < 10	True
			5 > 8 or 10 < 1	False
not	لا	تستخدم لنفي شرطٍ ما	not(3 > 1)	False
			not(3 < 1)	True

ملاحظة: المقارنة قد تكون بين قيمة وقيمة أخرى، قيمة و متغير ، متغير و متغير

في المثال التالي سيتم المقارنة بين شرطين وبينهما العامل and ، اذا تحقق الشرطان ستكون النتيجة True ، يشترط تحقق الشرطين معا:

```
1 print( 5 >2 and 5 <10)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
True
```

في المثال التالي سيتم المقارنة بين شرطين وبينهما العامل or ، اذا تحقق أحد الشرطان ستكون النتيجة True :

```
1 print( 5 == 2 or 5 <10)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
True
```

في المثال التالي سيتم عكس شرط محدد فرقم 5 لا يساوي 2 ، لكن مع ذلك النتيجة اصبحت True و السبب انه تم استخدام العامل not الذي يعكس الشرط

```
1 print( not(5 == 2 ))
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
True
```

الملاحظات :

موضوع الدرس :	الدرس 9: الجملة الشرطية
التاريخ :	

التقييم

مهارات الدرس :

م	المهارة	
1	الجملة الشرطية البسيطة	
2	الجملة الشرطية الكاملة	

م	المهارة	
3	الجملة الشرطية المتداخلة	

الجملة الشرطية البسيطة:

نستخدم الجملة الشرطية البسيطة لتنفيذ اجراءات معينة بناءً على تحقق شرط محدد وتكون صيغة الجملة بالشكل التالي:

نهاية الشرط يجب أن يكون :

if الشرط المطلوب :

↔ جملة برمجية 1

↔ .....

يجب ترك مسافة بعد كل جملة برمجية تابعة للجملة الشرطية.

الرجوع إلى بداية السطر يعني نهاية الجملة الشرطيّة

مثال:اكتب برنامجاً بلغة البايثون كالتالي:

a. يطلب من المستخدم ادخال درجته.
b. عرض رسالة "ناجح" اذا كانت الدرجة أكبر من أو يساوي 60 .

```
1  mark=int(input("enter your mark:"))
2
3  if mark >=60:
4      print("ناجح")
```

Shell × Exception ×

```
>>> %Run -c $EDITOR_CONTENT
enter your mark:60
ناجح
```

في المثال السابق استخدمنا علامة المقارنة أكبر من أو يساوي ، في الشرط الخاص بالجملة الشرطية

ملاحظة: لا تنس استخدام دالة ( )int لتحويل البيانات المدخلة الى رقم حتى يمكنك المقارنة بشكل صحيح

الجملة الشرطية الكاملة

نستخدم الجملة الشرطية لتنفيذ اجراءات معينة بناءً على تحقق شرط محدد من عدمه، وتكون الصيغة لها:

تفيد جملة else لاضافة الاجراءات التي نريدها اذا لم يتحقق الشرط الموجود في جملة if .

مثال:اكتب برنامجاً بلغة البايثون كالتالي:

a. يطلب من المستخدم ادخال درجته.
b. عرض رسالة "ناجح" اذا كانت الدرجة أكبر من أو يساوي 60 .
c. عرض رسالة "راسب" اذا كانت الدرجة اقل من 60

```
1  mark=int(input("enter your mark:"))
2
3  if mark >=60:
4      print("ناجح")
5  else:
6      print("راسب")
```

```
Shell  Exception
>>> %Run -c $EDITOR_CONTENT
 enter your mark:50
 ر ا سب
```

بعد اضافة else سيتم عرض رسالة "راسب" اذا تم ادخال درجة أقل من 60 .

ملاحظة: لا تنس استخدام دالة ( )int لتحويل البيانات المدخلة الى رقم حتى يمكنك المقارنة بشكل صحيح

الجملة الشرطية المتداخلة

نستخدم الجملة الشرطية في حال وجود عدد من الشروط الواجب مراعاتها، وتكون صيغتها بالشكل التالي:

ملاحظة: يتم استخدام الجملة elif بحسب عدد الشروط التي لدينا.

مثال:اكتب برنامجاً بلغة البايثون كالتالي:

a. يطلب من المستخدم ادخال رمز لون اشارة المرور وتخزين القيمة في متغير.
b. تظهر رسالة للمستخدم بحسب التالي:
   i. اذا كان اللون الأحمر red : يجب عليك التوقف.
   ii. اذا كان اللون الأخضر green : يمكنك المرور.
   iii. اذا كان اللون الأصفر yellow : عليك الاستعداد.

```
1 color=input("Enter the color : red - green - yellow\n")
2 if color=="red":
3     print("التوقف عليك يجب")
4 elif color=="green":
5     print("المرور يمكنك")
6 elif color=="yellow":
7     print("الاستعداد عليك")
8 else:
9     print("الصحيح اللون ادخل")
```

Shell × Exception ×

```
>>> %Run -c $EDITOR_CONTENT
Enter the color : red - green - yellow
red
يجب عليك التوقف
```

يمكننا استخدام العوامل المنطقية في حال وجود أكثر من شرط لنفس المعيار،

مثال:اكتب برنامجاً بلغة البايثون كالتالي:

a. يطلب من المستخدم ادخال درجة المشروع وتخزينه في متغير m1 و ادخال درجة الامتحان وتخزينها في المتغير m2.

b. عرض رسالة "ناجح" اذا كانت درجة المشروع أكبر من 10 و درجة الامتحان أكبر من 50.

c. عرض رسالة "راسب" اذا لم يتحقق الشرط.

```
1 m1=int(input("ادخل درجة المشروع"))
2 m2=int(input("ادخل درجة الامتحان"))
3 if m1>15 and m2 > 50:
4     print("ناجح")
5 else:
6     print("راسب")
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
ادخل درجة المشروع12
ادخل درجة الامتحان23
راسب
```

الملاحظات :

موضوع الدرس :	الدرس 10: التعامل مع القوائم
التاريخ :	

التقييم

مهارات الدرس :

م	المهارة	
1	دالة split()	
2	دالة sorted( )	
3	دالة remove( )	
4	دالة append( )	

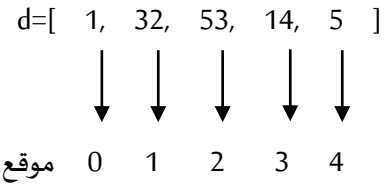
م	المهارة	
5	دالة index ( )	
6	دالة pop( )	
7	دالة del( )	
8	طباعة عناصر القائمة من الاتجاه العكسي	

أحد أنواع البيانات هي القائمة list والتي تسمح لنا بتخزين أكثر من قيمة في متغير واحد، ويمكن أيضا لهذه القيم أن تتكرر بداخل القائمة، بالامكان تغيير / حذف / اضافة أي قيمة أثناء عمل البرنامج.

مثال:لدينا متغير من نوع قائمة اسمه d لاسناد قيم رقمية بداخل المتغير يكون بالشكل التالي:

```
d=[1,32,53,14,5]
```

نلاحظ أن المتغير يحتوي على 5 قيم رقمية، و لكل قيمة موقع نسميه index ، دائما يبدأ الموقع من 0 ، فالمثال السابق لو أردنا طباعة القيمة الثانية و هي 32 ، يكون موقعها 1 و ليس 2 .



يمكننا أيضا اسناد قيم نصية للمتغير من نوع قائمة، كالتالي:

```
num=["Muharraq","Jidhafs","Manama"]
```

وقد تكون نوع البيانات متنوعة (رقمية – نصية ... الخ)، كالتالي:

```
v=["Muharraq",5.5,"Manama",20]
```

ملاحظة مهمة: القيم النصية يجب أن تكون بين علامتي تنصيص مزدوجة " " او تنصيص مفردة ' '

طباعة قيم متغير من نوع قائمة:

لدينا المتغير التالي :

```
num=["Muharraq","Jidhafs","Manama"]
```

تعرفنا سابقا أن موقع القيم يبدأ من 0 ، فلو أردنا طباعة القيم ستكون بالشكل التالي:

	القيمة الاولى	القيمة الثانية	القيمة الثالثة
الطريقة	num[0]	num[1]	num[2]

يمكننا بطباعة القيم الموجودة في المتغير num بهذه الطريقة :

```
num=["Muharraq","Jidhafs","Manama"]
print (num[0])
print (num[1])
print (num[2])
```

```
Shell  Exception
>>> %Run -c $EDITOR_CONTENT
Muharraq
Jidhafs
Manama
```

أو هذه الطريقة :

```
num=["Muharraq","Jidhafs","Manama"]
print (num[0],num[1],num[2])
```

```
Shell  Exception
>>> %Run -c $EDITOR_CONTENT
Muharraq Jidhafs Manama
```

تقسيم النص الى كلمات وتخزينها في متغير من نوع قائمة :

تستخدم دالة split( ) لتقسيم النص الى كلمات وتخزين هذه الكلمات على هيئة قائمة list . طريقة استخدامها كالتالي:

split( ).اسم المتغير الذي يحتوي على نص

مثال:

الجملة البرمجية	النتيجة
theText="Hello my name is Ameena" **theWords=theText.split()** **print(theWords)**	>>> %Run -c $EDITOR_CONTENT ['Hello', 'my', 'name', 'is', 'Ameena'] >>>
theText="Hello my name is Ameena" **print(theText.split())**	>>> %Run -c $EDITOR_CONTENT ['Hello', 'my', 'name', 'is', 'Ameena'] >>>

نلاحظ من الأمثلة السابقة أنه تم اعتماد المسافة كفاصل لتقسيم النص الى كلمات، و من شكل الناتج الذي تمت طباعته يتضح لنا أنه تم تخزين البيانات على شكل قائمة.

التعامل مع القوائم:

هناك مجموعة من الدوال التي نستخدمها للتعامل مع متغير القائمة، سيتم استخدام القوائم التالية أثناء توضيح طريقة استخدام الدوال:

```
cname=["Muharraq","Jidhafs","Manama"]
stid=[3,1,18,4,10]
```

التعرف على موقع عنصر معين في القائمة :

نستخدم دالة index( ) للحصول على رقم موقع عنصر معين، سنبحث عن موقع العنصر Manama من القائمة cname:

```
1 cname=["Muharraq","Jidhafs","Manama"]
2 indx=cname.index("Manama")
3 print(indx)
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
 2
```

ملاحظة: يمكننا مباشرة طباعة النتيجة دون تخزينها في متغير :

```
print(cname.index("Manama"))
```

ادراج عنصر جديد في نهاية القائمة :

نستخدم دالة append( ) لادراج عنصر جديد و سيكون Sanabis للقائمة cname :

```
1 cname=["Muharraq","Jidhafs","Manama"]
2 cname.append("Sanabis")
3 print(cname)
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
 ['Muharraq', 'Jidhafs', 'Manama', 'Sanabis']
```

بعد طباعة القائمة نلاحظ أنه تم ادراج العنصر الجديد في نهايتها.

حذف عنصر من القائمة بحسب موقعه:

نستخدم دالة pop( ) لحذف عنصر معين بحسب موقعه، فمثلا أود ان احذف العنصر الثاني من القائمة cname :

ملاحظة: العنصر الثاني من القائمة يكون رقمه 1

```
1 cname=["Muharraq","Jidhafs","Manama"]
2 cname.pop(1)
3 print(cname)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
['Muharraq', 'Manama']
```

بعد طباعة القائمة نلاحظ أن العنصر الثاني من القائمة تم حذفه.

حذف عنصر من القائمة بحسب قيمته:

نستخدم دالة ()remove لحذف عنصر معين بحسب قيمته، فمثلا أود ان احذف العنصر Jidhafs من القائمة cname :

```
1 cname=["Muharraq","Jidhafs","Manama"]
2 cname.remove("Jidhafs")
3 print(cname)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
['Muharraq', 'Manama']
```

بعد طباعة القائمة نلاحظ أن العنصر Jidhafs من القائمة تم حذفه

ترتيب عناصر القائمة:

نستخدم دالة ()sorted لترتيب عناصر القائمة تصاعديا، فمثلا لترتيب القائمة stid تصاعديا يكون بالطريقة التالية:

```
1 stid=[3,1,18,4,10]
2 newid=sorted(stid)
3 print(newid)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
[1, 3, 4, 10, 18]
```

بعد طباعة القائمة نلاحظ أن عناصر القائمة تم ترتيبها من الأصغر الى الأكبر.

يمكننا استخدامها لترتيب العناصر تنازليا مع اضافة تغيير بسيط الى دالة sorted :

```
stid=[3,1,18,4,10]
newid=sorted(stid,reverse=True)
print(newid)
```

```
Shell  Exception
>>> %Run -c $EDITOR_CONTENT
 [18, 10, 4, 3, 1]
```

بعد طباعة القائمة نلاحظ أن عناصر القائمة تم ترتيبها من الأكبر الى الأصغر.

ملاحظة مهمة: في العبارة reverse=True يجب أن تبدء كلمة True بحرف كبير، كتابة true تعتبر خطأ برمجي

حذف القائمة:

نستخدم دالة ()del لحذف القائمة بأكملها، فمثلاً لو أردنا حذف قائمة stid يكون بالطريقة التالية:

```
stid=[3,1,18,4,10]
del(stid)
print(stid)
```

```
Shell  Exception
>>> %Run -c $EDITOR_CONTENT
 Traceback (most recent call last):
   File "<string>", line 3, in <module>
 NameError: name 'stid' is not defined
```

لاحظ بعد تشغيل البرنامج تظهر لنا رسالة خطأ بأن المتغير stid غير موجود.

طباعة عناصر القائمة من الاتجاه العكسي:

يمكنني طباعة قيم أي دالة من اليمين الى اليسار و ذلك عن طريق استخدام الارقام السالبة، فميلا لو أردت طباعة الرقم الأخير (الأول من اليمين) اذ سيكون رقم العنصر الأخير -1

```
sid=[123,147,159,145]
print(sid[-1])
```

```
Shell  Exception
>>> %Run -c $EDITOR_CONTENT
 145
```

موضوع الدرس :	الدرس 11: الدوال الاحصائية
التاريخ :	

التقييم

مهارات الدرس :

م	المهارة	
1	دالة min( )	
2	دالة max( )	

م	المهارة	
3	دالة sum( )	

تستخدم الدوال الاحصائية لايجاد معلومات معينة لقيم محددة، مثلا أكبر قيمة، أصغر قيمة و مجموع القيم، الدوال التالية تتعامل فقط مع بيانات من نوع قائمة.

الدالة	الوصف	مثال	ملاحظات
min( )	ايجاد أصغر قيمة في القائمة	x=[1,34,22,34,3] mi=min(x) print(mi)	1- يتم استخدام هذه الدوال مع القوائم فقط. 2- يجب التأكد من أن القائمة تحتوي فقط على أرقام.
max( )	ايجاد أكبر قيمة في القائمة	x=[1,34,22,34,3] mx=max(x) print(mx)	
sum( )	جمع الأعداد الموجودة في القائمة	x=[1,34,22,34,3] s=sum(x) print(s)	

مثال :

صمم برنامجاً يقوم بعرض أكبر قيمة ، أصغر قيمة و مجموع الاعداد الموجودة في القائمة التالية:

a=[4,5,2,44,14,9]

```
1 a=[4,5,2,44,14,9]
2 mi=min(a)
3 mx=max(a)
4 s=sum(a)
5 print("the minimum number= ",mi)
6 print("the maximam number= ",mx)
7 print("the sum of the numbers= ",s)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
the minimum number=  2
the maximam number=  44
the sum of the numbers=  78
```

نلاحظ أنه تم اسناد قيم الدوال الى متغيرات وبعدها تم طباعة قيم هذه المتغيرات.

موضوع الدرس :	الدرس 12: دوال الاختيار العشوائي	التقييم
التاريخ :		

مهارات الدرس :

م	المهارة	
1	دالة ( )random	
2	دالة ( )randint	

م	المهارة	
3	دالة ( )randrange	

توفر لغة البايثون مكتبة من الدوال التي يمكن استخدامها لاستخراج قيمة بشكل عشوائي، و قبل البدء باستخدام هذه الدوال يجب أن يتم استدعاء المكتبة الخاصة بها و اسمها random ، لذا يتم كتابة السطر التالي في ملف البايثون:

```
1 import random
```

عند استخدام اي دالة تابعة للمكتبة random يجب أن نذكر اسم المكتبة قبلها، ويكون بالشكل التالي:

دالة ( )random :

تستخدم الدالة للحصول على عدد عشوائي عشري بين 0.01 و 1.0 مثال: عند كتابة البرنامج التالي وتشغيله، نلاحظ أن الرقم يختلف في كل مرة .

```
1 import random
2 a=random.random()
3 print(a)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
 0.05801497180534176 4
```

دالة ( )randint :

تستخدم لاختيار عدد عشوائي صحيح مع تحديدمجال الاختيار، مثلا يكون العدد بين الرقمين 10 و 20.مثال:

```
import random
y=random.randint(10,20)
print(y)
```

```
Shell  Exception
>>> %Run -c $EDITOR_CONTENT
12
```

دالة ( ) randrange :

تستخدم أيضا لاختيار عدد عشوائي صحيح مع تحديد المجال، يمكن استخدامها بعدة طرق.

1- اختيارعدد عشوائي صحيح ضمن مجال محدد يبدء من الصفر، في المثال التالي تم تحديد المجال 15 ، لذا سيكون الرقم الذي سيتم اختياره بين 0 و 14 (الرقم 15 ليس ضمن المجال).

```
import random
rand=random.randrange(15)
print(rand)
```

```
Shell  Exception
>>> %Run -c $EDITOR_CONTENT
11
```

2- اختيارعدد عشوائي صحيح ضمن مجال محدد يتم تحديد بدايته ونهايته، في المثال التالي تم تحديد المجال بين 5 و 40 ، لذا سيكون الرقم الذي سيتم اختياره بين 5 و 39 (الرقم 40 ليس ضمن المجال).

```
import random
num=random.randrange(5,40)
print(num)
```

```
Shell  Exception
>>> %Run -c $EDITOR_CONTENT
18
```

3- اختيار عدد عشوائي صحيح ضمن مجال محدد يتم تحديد بدايته ونهايته مع تحديد خطواته، في المثال التالي تم تحديد المجال بين 5 و16 والخطوات 2، لذا سيكون الرقم الذي سيتم اختياره من الأرقام التالية : 5 – 7 – 9 – 11 – 13 – 1

```
1 import random
2 num=random.randrange(5,16,2)
3 print(num)
```

```
Shell × Exception ×
>>> %Run -c $EDITOR_CONTENT
 7
```

الملاحظات :

موضوع الدرس :	الدرس 13: الجملة التكرارية for
التاريخ :	

التقييم

مهارات الدرس :

م	المهارة	
1	for x in range( )	
2	for x in list	

م	المهارة	
3	for x in "text"	
4	break - continue	

استخدم الجملة التكرارية for لتنفيذ عدد من الاجراءات المعينة لعدد محدد من التكرار، وتكون الصيغة العامة لها:

متغير يعتبر العداد counter
يمكن تسمية المتغير بأي اسم آخر

يتم تحديد المجال بثلاثة طرق:
الدالة ( )range - قيمة نصية - قائمة

```
for x in المجال :
    الاجراءات
    .....
```

:
في نهاية الجملة التكرارية

يجب ترك مسافة قبل كل اجراء تابع للجملة التكرارية.

الرجوع إلى بداية السطر يعني نهاية الجملة التكرارية

الجملة التكرارية مع ( )range :

لدالة ( )range ثلاث معاملات يكن استخدام جميع المعاملات او اثنان او واحد، و تكون كالتالي:

range(بداية , نهاية , خطوات)

الطريقة	الوصف	ملاحظة
range(5)	يبدأ المجال من الرقم 0 الى 4	5 ليست ضمن المجال
range(6,20)	يبدأ المجال من الرقم 5 الى 19	20 ليست ضمن المجال
range(6,20,2)	يبدأ المجال من الرقم 6 الى 18 و ينتقل خطوتين، سيكون المجال: 18-16-14-12-10-8-6	20 ليست ضمن المجال

سأصمم برنامج يقوم بعرض الأرقام من 5 الى 10 لذا سأستخدم الدالة ( )range و التي ستمكنني من تحديد بداية المجال ونهايته وستكون بالشكل التالي range(5,11) تم كتابة الرقم 11 عوضا عن 10 لأن نهاية المجال لا تكون ضمن الارقام.

```
1 for y in range(5,11):
2     print(y)
```

Shell × Exception ×

```
>>> %Run -c $EDITOR_CONTENT
5
6
7
8
9
10
```

ادخال قيم للقائمة:

يمكنني ايضا استخدام الجملة التكرارية لادخال عناصر في القائمة، فمثلا في المثال التالي سأطلب من المستخدم ادخال اربعة اسماء، و ادخال الاسم مباشرة في القائمة، ألاحظ انه تم استخدام متغيرو اسناد قيمة له قبل البدء بالجملة التكرارية، وذلك حتى يتعرف البرنامج على المتغيرويتمكن من استخدامه:

```
1 thelist=[]
2 for i in range(4):
3     x=input("enter the name ")
4     thelist.append(x)
```

Shell × Exception ×

```
>>> %Run -c $EDITOR_CONTENT
enter the name amina
enter the name moona
enter the name noor
enter the name fatema
```

قمت بادراج قيم للقائمة ، كما يمكنني ايضا حذف عنصر معين بحسب موقعه أو قيمته.

الجملة التكرارية مع القائمة :

يمكن أن يكون متغير من نوع قائمة هو المجال في الجملة التكرارية، ونستخدم هذه الطريقة لقراءة محتوى القائمة أو لطباعتها أو لاجراء معالجة بحسب المطلوب. في المثال التالي سأقوم بطباعة محتوى القائمة stName :

```
1 stName=["Ameena","Noor","Mariam"]
2 for i in stName:
3     print(i)
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
Ameena
Noor
Mariam
```

الجملة التكرارية مع القيم النصية :

يمكن استخدام نص ليكون هو المجال الخاص بالجملة التكرارية، فمثلا لو كانت الكلمة Amal هي المجال، واستخدمتها مع الجملة التكرارية فسأتمكن من طباعة كل حرف من الكلمة، أو حتى التعامل معه ضمن اجراءات معينة.

```
1 for lett in "Amal":
2     print(lett)
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTE
A
m
a
l
```

ايقاف حلقة التكرار break :

يمكنني ايقاف التكرار في الجملة التكرارية بناءً على شرط محدد، و سوف نحتاج الى استخدام الجملة الشرطية if ، في المثال التالي سنقوم بعرض عناصر القائمة nbr و اذا كانت قيمة العنصر 4 سيتم ايقاف التكرار.

```
1 nbr=[2,8,4,9,5,6]
2 for i in nbr:
3     if i==4:
4         break
5     print(i)
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
2
8
```

مواصلة حلقة التكرار continue:

يمكنني هذا الامر من تجاوز عنصر محدد ضمن الجملة التكرارية بناءً على شرط محدد ، سنحتاج الى استخدام الجملة الشرطية if ، في المثال التالي سنقوم بعرض عناصر القائمة nbr و اذا كانت قيمة العنصر 4 لن يعرضها وسيواصل عرض باقي القيم.

```
1  nbr=[2,8,4,9,5,6]
2  for i in nbr:
3      if i==4:
4          continue
5      print(i)
```

```
Shell  Exception
>>> %Run -c $EDITOR_CONTENT
2
8
9
5
6
```

الملاحظات :

موضوع الدرس :	الدرس 14: الجملة التكرارية while	التقييم
التاريخ :		selectQuery.php

مهارات الدرس :

م	المهارة	
1	الجملة التكرارية :While.....	

م	المهارة	
3	While ..... else	

تستخدم الجملة التكرارية while في تنفيذ الاجراءات الموجودة ضمنها طيلة تحقق الشرط ، وتكون صيغتها كالتالي:

while الشرط المطلوب :

الاجراءات المطلوبة

.....

في نهاية الجملة التكرارية :

يجب ترك مسافة قبل كل اجراء تابع للجملة التكرارية.

الرجوع إلى بداية السطر يعني نهاية الجملة التكرارية

سأصمم برنامج يقوم بطلب ادخال رقم ويتوقف عند ادخال الرقم 0 .

```
1 n=1
2 while n != 0 :
3     n=int(input("Enter a number: "))
4     print("The number = ",n)
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
Enter a number: 5
The number =  5
Enter a number: 4
The number =  4
Enter a number: 0
The number =  0
```

نلاحظ في البرنامج السابق :

- تم اسناد قيمة الى المتغير n قبل بدء الجملة التكرارية، حتى نتمكن من استخدامه خلال الجملة ويتعرف عليه البرنامج.
- استخدام الدالة ()int مع جملة input و ذلك لأنني سأقوم بادخال قيم رقمية .
- أثناء تشغيل البرنامج يظهر السؤال لادخال رقم و طباعة هذا الرقم، ويتوقف عند ادخال الرقم 0 .

استخدام else مع الجملة التكرارية:

استخدم else مع الجملة التكرارية، لتنفيذ اجراءات معينة عند توقف التكرار، فمثلا في المثال التالي ، سيقوم المستخدم بادخال اسماء الطلبة وعند ادخال النص end سيتوقف البرنامج ويعرض رسالة "شكرا لكم"

```
1 sname=""
2 while sname != "end" :
3     sname=input("ادخل اسم الطالب: ")
4 else:
5     print("شكرا لكم")
```

Shell Exception

```
>>> %Run -c $EDITOR_CONTENT
ادخل اسم الطالب: ameena
ادخل اسم الطالب: amal
ادخل اسم الطالب: end
شكرا لكم
```

ملاحظة:

- يجب تعريف المتغير واسناد قيمة له قبل الجملة التكرارية.
- تم كتابة end بين علامتي تنصيص لأنه نص.
- تم كتابة :else في بداية السطر .

الملاحظات :

موضوع الدرس :	الدرس 15: انشاء دوال	التقييم
التاريخ :		

مهارات الدرس :

م	المهارة	
1	انشاء دالة بدون ارجاع قيمة	

م	المهارة	
2	انشاء دالة مع ارجاع قيمة	

الدالة هي كتلة من الجمل البرمجية الجاهزة مسبقا، يتّم استدعاؤها عند اللزوم، عادة ما ترافقها عوامل خاصة بها تسمى " parameters" لاستخدامها للوصول إلى النتيجة المطلوبة. عادة ما تقوم الدالة بإرجاع النتيجة إلى البرنامج الرئيسي الذي تم استدعاؤها منه. كما أنها تساهم في حل المسائل المعقّدة بحيث يتم تقسيم المشكل الرئيسي إلى مشكلات بسيطة يتم حلّها بطريقة منفصلة ثم تجميعها للحصول على الحل النهائي. ويوجد لدينا نوعان من الدوال، دالة بدون ارجاع قيمة و دالة مع ارجاع قيمة.

القالب العام للدالة بدون ارجاع قيمة:

كلمة def يجب كتابتها قبل اسم الدالة

```
def اسم الدالة(العوامل):
    الاجراءات المطلوبة
    .....
```

يجب ترك مسافة قبل كل اجراء تابع للدالة.

الرجوع إلى بداية السطر يعني نهاية الدالة

يتم انشاء الدوال في بداية الملف او قبل استدعاؤها ، يتم استدعاء الدوال عن طريق كتابة اسم الدالة مع الأقواس فلو كان اسم الدالة printHello فطريقة استدعاؤها كتابة printHello() في المكان المطلوب.

سأقوم بانشاء دالة اسمها printHello بدون عوامل (متغيرات) تقوم بعرض رسالة ترحيبية:

```
1 def printHello():
2     print("Hello world")
3
4 printHello()
```

Shell × Exception ×

```
>>> %Run -c $EDITOR_CONTENT
 Hello world
```

الأحظ أنه تم انشاء الدالة في بداية الملف و تم استدعاؤها في السطر رقم 4.

الان سأقوم بانشاء دالة اسمها sumTwo مع عاملين و تقوم بجمعهما وعرض النتيجة، الاستدعاء يكون مع ارسال قيم للدالة مثلا sumTwo(5,6) او sumTwo(x,y) اذا كان لدي قيم للمتغير x و y

```
1 def sumTwo(a,b):
2     x=a+b
3     print("The result= ", x)
4
5 sumTwo(5,6)
```

```
Shell  Exception
>>> %Run -c $EDITOR_CONTENT
 The result=  11
```

الاحظ في السطر رقم 5 تم استدعاء الدالة مع اضافة قيم ، لأن الدالة sumTwo تستقبل متغيرات ، من الممكن كتابة الارقام مباشرة او اسماء متغيرات بحسب الحاجة.

القالب العام للدالة مع ارجاع قيمة:

```
def اسم الدالة(المتغيرات):
    الاجراءات المطلوبة
    .....
    return اسم المتغير الذي سيتم ارجاعه
```

يجب ترك مسافة قبل كل اجراء تابع للدالة.

الرجوع إلى بداية السطر يعني نهاية الدالة

نستخدم return لارجاع قيمة المتغير

عند استدعاء الدالة في البرنامج الاساسي يكون بالشكل التالي:

```
z=sumTwo(5,6)
```

مثال: الان سأقوم بانشاء دالة اسمها sumTwo مع عاملين و تقوم بجمعهما وارجاع الناتج الى البرنامج الرئيسي، الاستدعاء يكون مع ارسال قيم للدالة مثلا sumTwo(5,6) او sumTwo(x,y) اذا كان لدي قيم للمتغير x و y

```
1 def sumTwo(a,b):
2     x=a+b
3     return x
4
5 z=sumTwo(5,6)
6 print("The result= ", z)
```

Shell × Exception ×

```
>>> %Run -c $EDITOR_CONTENT
 The result=  11
```

ألاحظ أن طباعة النتيجة كانت ضمن البرنامج الرئيسي و ليس داخل الدالة، والسبب أن الدالة اعادت النتيجة الى البرنامج الرئيسي.

الملاحظات :

المرفقات : جدول المحتوى

المهارة	الوصف	مثال
المهارات العامة		
print( )	جملة الطباعة / العرض	print("The result",m)
input( )	جملة الادخال	X=input("Enter your name")

if ....: else:	الجملة الشرطية الكاملة	if ***condition1***:
		do this
		else:
		do this

if ...: elif...:else:	الجملة الشرطية المتداخلة	if ***condition1***:
		do this
		elif ***condition2:***
		do this
		else:
		do this

for.....:	الجملة التكرارية	for counter in range: ***do this***
while .....:	الجملة التكرارية	while condition: ***do this***

while.....: else:	الجملة التكرارية	while condition:
		do this
		else:
		do this

Def functionName()	انشاء دالة	def functionName(): instruction
أنواع البيانات		
int	رقم صحيح	X=15
float	رقم عشري	X=15.5
str	نص	X="hello"
list	قائمة	X=[1,2,"hello"]

bool	منطقي	X=True
الدوال الخاصة بأنواع البيانات		
type( )	عرض نوع المتغير	type(x)
int()	تحويل النص المدخل الى عدد صحيح	int(x)
float()	تحويل النص المدخل الى عدد عشري	float(x)
str()	تحويل النص المدخل الى نص	str(x)
الدوال الخاصة بالنصوص والعمليات الحسابية		
round()	دالة التقريب لأقرب عدد صحيح	round(x)
abs()	دالة القيمة المطلقة	abs(x)
len()	دالة حساب عدد أحرف النص	len(x)
الدوال الخاصة بالقوائم		
sorted()	ترتيب عناصر القائمة تصاعديا أو تنازليا	sorted(thelist)
index()	البحث عن Index عنصر معين	thelist.index("Manama")
remove()	حذف عنصر بحسب قيمته	thelist.remove("name")
split()	تقسيم النص الى عناصر في قائمة	text.split( )
pop()	حذف عنصر بحسب موقعه	thelist.pop(1)
del()	حذف القائمة	del(theList)
append	اصافة عنصر للقائمة	thelist.append("text")
الدوال الخاصة بالعمليات الاحصائية وتستخدم فقط مع القوائم		
min()	الحصول على أصغر قيمة في القائمة	min(listName)
max()	الحصول على أكبر قيمة في القائمة	max(listName)
sum()	جمع عناصر القائمة	sum(listName)
دوال خاصة بالجملة التكرارية for		
range()	تحديد المجال للجملة التكرارية	range(start,stop,step)
الدوال الخاصة بالاختيار العشوائي		
random()	اختيار عدد عشوائي عشري بين 0.01 و1	random.random()
randint()	اختيار عدد صحيح عشوائي بين مجالين	random.randint(start, stop)
randrange ( )	اختيار عدد عشوائي بين مجالين	randrange (start,stop,step)
العمليات الحسابية		
+ للنص	ربط النصوص	"تقن" + "106" = تقن106
+ للأرقام	جمع الأعداد	8 + 1 = 9

-	طرح الأعداد	8 – 1 = 7
*	ضرب الأعداد	3 * 5 = 15
/	قسمة الأعداد	25 / 12 = 2.0833
//	ناتج القسمة فقط عدد صحيح	25 // 12 = 2
%	باقي القسمة	25 % 12 = 1
**	قوة العدد	2 ** 2 = 4
العوامل المنطقية		
and	يجب أن يتحقق شرطان	2 > 5 and 12 < 80 = True
or	يتحقق احد الشرطان	2 > 5 or 4 <5 = True
not	نفي شرط محدد	not(1==0) = False
عوامل المقارنة		
للأمثلة التالية اعتبر أن : a= 5 - b= 8 - c =2		
==	يساوي	a == c = False
=!	لا يساوي	a != 0 = True
<	أصغر من	a < 5 = False
<=	أصغر من أو يساوي	a <= 8 = True
>	اكبر من	a > b = False
>=	اكبر من أو يساوي	a >= 8 = False