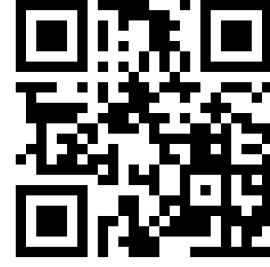


شكراً لتحميلك هذا الملف من موقع



مذكرة البرمجة بلغة البايثون

موقع المناهج ⇨ الصف الأول الثانوي ⇨ علوم وتقانة ⇨ الفصل الثاني ⇨ الملف

تاريخ نشر الملف على موقع المناهج: 2023-10-29 07:46:49

التواصل الاجتماعي بحسب الصف الأول الثانوي



روابط مواد الصف الأول الثانوي على تلغرام

[الرياضيات](#)

[اللغة الانجليزية](#)

[اللغة العربية](#)

[التربية الاسلامية](#)

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

[شرح و ملخص درس تصميم صفحات الويب الديناميكية مقرر تقن 105 درس لغة pHP](#)

1

[شرح و ملخص درس تصميم صفحات الويب الديناميكية مقرر تقن 105 درس النماذج forms](#)

2

[شرح و ملخص درس تصميم صفحات الويب الديناميكية مقرر تقن 105 درس لغة pHP](#)

3

[الإجابة النموذجية لامتحان المنتصف inventions](#)

4

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

[نموذج نهاية امتحان الفصل الثاني مقرر تصميم صفحات الويب](#)
[الدناميكية، رمز المقرر تقن 105](#)

5

مملكة البحرين
وزارة التربية والتعليم
مدرسة جدحفص الثانوية للبنات
قسم الحاسوب

رؤية المدرسة
وطن بجيل جدحفص رائد في مواجهة تحديات المستقبل

البرمجة بلغة البايثون

ملف الطالبة



الشعبة

اسم الطالبة

رقم الجهاز

اسم المعلمة

مديرة المدرسة
أ.خلود يعقوب

اعداد
أ.أمينة محمد الصايغ

الفهرس

3	معلومات عن المقرر
3	التواصل مع المعلمة
4	اختباراتي
4	مكافآتي
5	الدرس 1: الأنظمة العددية
7	الدرس 2: الذكاء الاصطناعي
9	الدرس 3: الخوارزميات ودورة تطوير البرمجيات
10	الدرس 4: الخرائط التندقية
12	الدرس 5: مقدمة في البايثون
16	الدرس 6: المتغيرات و جملة الادخال
19	الدرس 7: التعامل مع الأرقام
24	الدرس 8: عوامل المقارنة و العوامل المنطقية
28	الدرس 9: الجملة الشرطية
32	الدرس 10: التعامل مع القوائم
37	الدرس 11: الدوال الاحصائية
38	الدرس 12: دوال الاختيار العشوائي
41	الدرس 13: الجملة التكرارية for
45	الدرس 14: الجملة التكرارية while
47	الدرس 15: انشاء دوال
50	المرفقات : جدول المحتوى

الاتفاقية

في المختبر

- أحافظ على سلامة الاجهزة.
- التزم بالنظام و الترتيب و النظافة.
- اتجنب احضار الاطعمة و المشروبات (ماعدا الماء) و الاجهزة الالكترونية للمختبر.

قبل الذهاب للمختبر

- احضار الكراسة في كل حصة.
- أجهز نفسي للذهاب الى المختبر بداية كل حصة.
- ألتزم بالمشي مع زميلاتي بهدوء حتى الوصول الى المختبر.

قبل مغادرتي المختبر

- اتأكد من اغلاق ملفاتي و ارجاع الكرسي الى مكانه.
- اخزن ملفاتي في مجلدي الخاص.

أثناء الدرس

- أحترم معلمتي و زميلاتي.
- أتعاون مع زميلاتي .
- اسجل ملاحظاتي و أهتم بملفي.

مهم جدا

- ألتزم بتثبيت البرمجيات المطلوبة في حاسوبي بالمنزل خلال الاسبوعيين الأولين من بدء الفصل الدراسي.

توقيع الطالبة :

معلومات عن المقرر

بيانات المقرر:

اسم المقرر	البرمجة بغلة البايثون	رمز المقرر	تقن 106	عدد الساعات	4
------------	-----------------------	------------	---------	-------------	---

الدرجات :

الملاحظة المنظمة			ملف الطالب				المهمة		
سلوك الطالب	الأداء الأكاديمي	المجموع	الإجراءات التنظيمية	تحسين التعلم	المبادرات	المجموع	المشروع	العرض	المجموع
4	4	8	4	4	4	12	15	5	20

درجة التقويم التكويني	المجموع الكلي	الاختبارات				
		المجموع	4	3	2	1
40.0	100.0	60.00	15	15	15	15

المتطلبات :

تثبيت البرامج الخاصة بالمقرر خلال اول اسبوعين من الفصل الدراسي وهذا تحت مسؤولية الطالبة.

التواصل مع المعلمة

يمكن للطالبة أو ولي الأمر التواصل مع المعلمة عن طريق :

عن طريق البوابة.

ارسال بريد على :

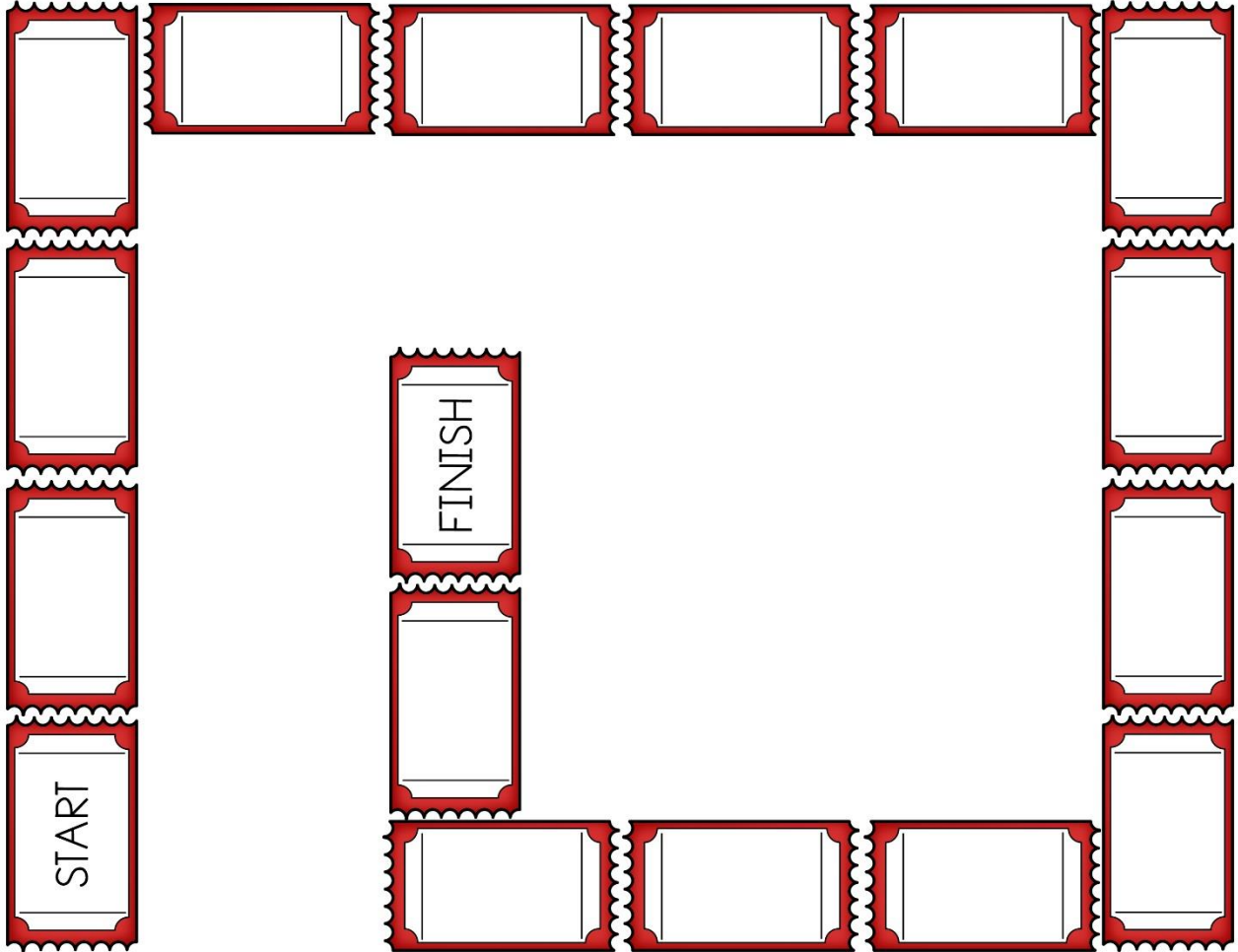
الساعات المكتبية وبحسب الجدول التالي :

اليوم			
الحصة			

اختباراتي

الاختبار	التاريخ	درجة الطالبة	توقيع ولي الأمر	ملاحظات
الأول				
الثاني				
الثالث				
الرابع				
المشروع				

مكافاتي



التقييم

موضوع الدرس : الدرس 1 : الأنظمة العددية
التاريخ :

مهارات الدرس :

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

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

ما هي الأنظمة العددية Numbering system

هي أنظمة عد أو ترقيم تعتمد أولاً على مجموعة محددة من الرموز لتمثيل الأعداد وثانياً على منهجية معينة لكتابتها وعرضها. يحتل كل رمز من رموز النظام العددي موضعاً يمثل قيمته. مثال: الرمز 1 ضمن العدد 10 أقل قيمة منه ضمن العدد 100

النظام العشري Decimal numbering system :

من أهم الأنظمة العددية الذي يعتمد على الرموز من 0 إلى 9 لتمثيل الأعداد العشرية نسبة إلى العدد 10 الذي يمثل بدوره عدد الرموز من 0 إلى 9. وهو النظام العددي الأكثر شيوعاً والذي نستخدمه في حساباتنا اليومية وفي أنظمتنا الاقتصادية وإدارتنا التعليمية ومختلف المجالات الأخرى.

النظام الثنائي Binary numbering system :

هو نظام عد يعتمد على الرموز من 0 إلى 1 لتمثيل الأعداد الثنائية نسبة إلى العدد 2 الذي يمثل بدوره عدد الرموز من 0 إلى 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 قوة 2.

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

طريقة استخدام جدول الأعداد العشرية قوة 2:

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

الملاحظات :

التقييم	موضوع الدرس : الدرس 2: الذكاء الاصطناعي
	التاريخ :

مهارات الدرس :

م	المهارة	م	المهارة
4	استخدامات التعلم الآلي	1	مفهوم الذكاء الاصطناعي وأهميته
5	لوغاريتمات التعلم الآلي	2	مفهوم التعلم الآلي

تعريف الذكاء الاصطناعي:

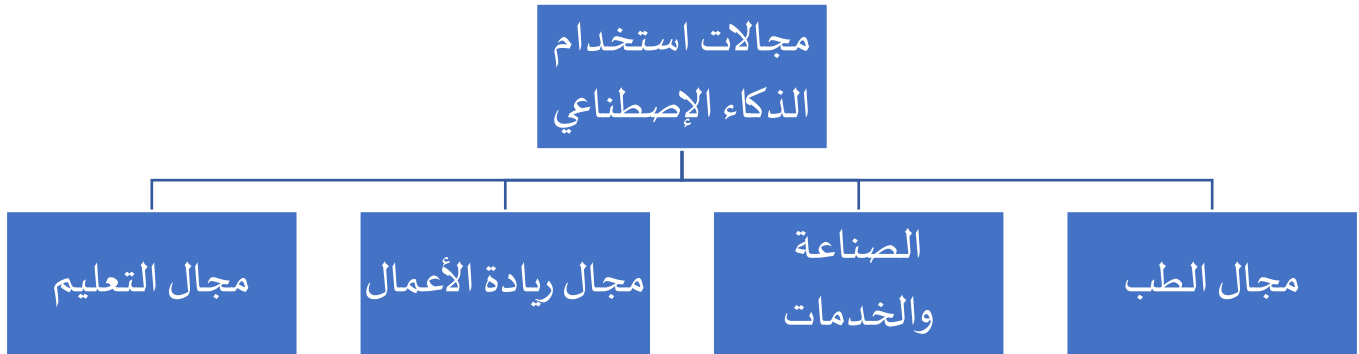
يعرف الذكاء الاصطناعي بأنه الذكاء الذي تبديه الآلات والبرامج بما يحاكي القدرات الذهنية البشرية وأنماط عملها، مثل القدرة على التعلم والاستنتاج ورد الفعل على أوضاع لم تبرمج في الآلة.

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 الأمنية المستقبلية.
- تقليل التكلفة وبيع 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:



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

مهارات الدرس :

م	المهارة	م	المهارة
3	مراحل تطوير البرمجيات	1	مفهوم الخوارزميات
4		2	كتابة خوارزمية لحل مشكلة ما

مفهوم الخوارزميات:

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

كتابة خوارزمية لحل مشكلة :

لكتابة أي خوارزمية نحتاج إلى 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 عدداً 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 قيمة 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 البرمجية ، 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 وتتبعها وتصحيحها بحيث تكون البرمجية خالية من أية شوائب وقادرة على التعامل مع المدخلات في جميع الحالات والحصول على المخرجات المطلوبة بدقة.

التقييم	الدرس 4: الخرائط التدفقية	موضوع الدرس :
		التاريخ :




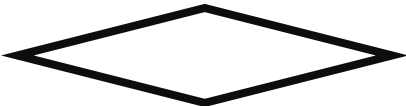
مهارات الدرس :

م	المهارة	م	المهارة
3	انشاء خريطة تدفقة لحل مشكلة برمجية	1	مفهوم الخرائط التدفقية
4		2	التعرف على أشكال الخرائط التدفقية

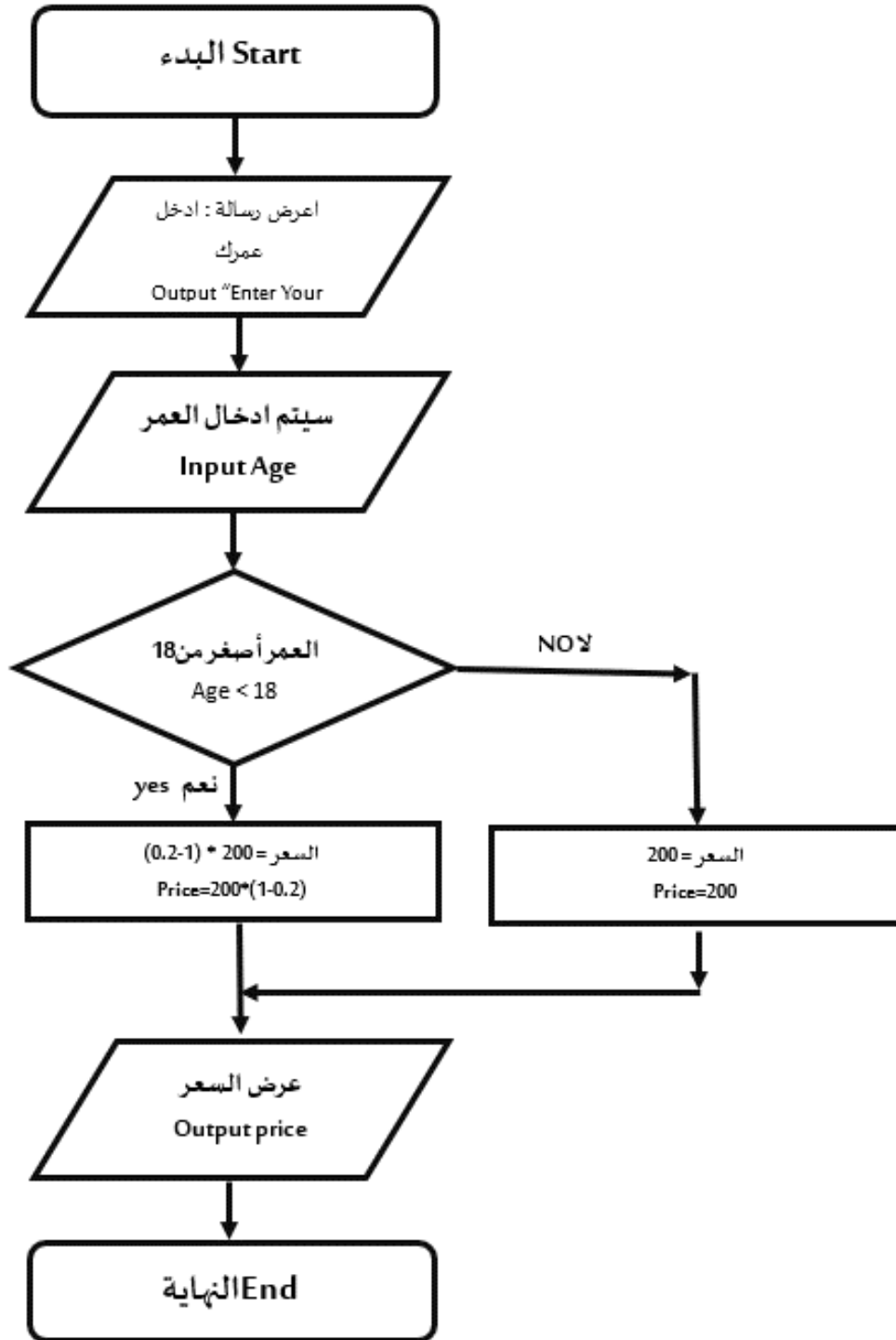
الخرائط التدفقية :

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

أشكال الخرائط التدفقية:

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

يبلغ سعر تذكرة إلى دولة ما 200 د.ب ، فإذا كان سن المسافر أقل من 18 سنة فإنه سيتمتع بـ 20% خصم.



الملاحظات :

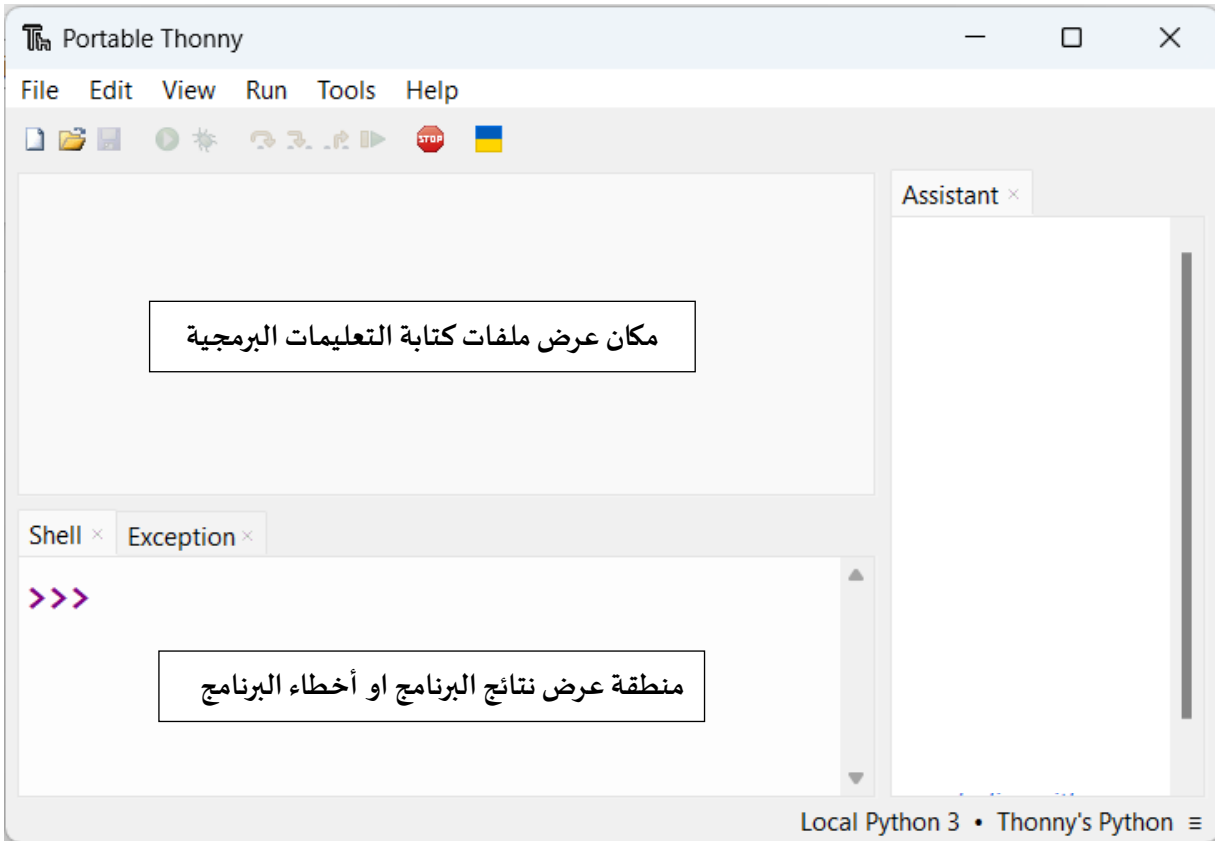
التقييم	الدرس 5: مقدمة في البايثون	موضوع الدرس :
		التاريخ :





مهارات الدرس :

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

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

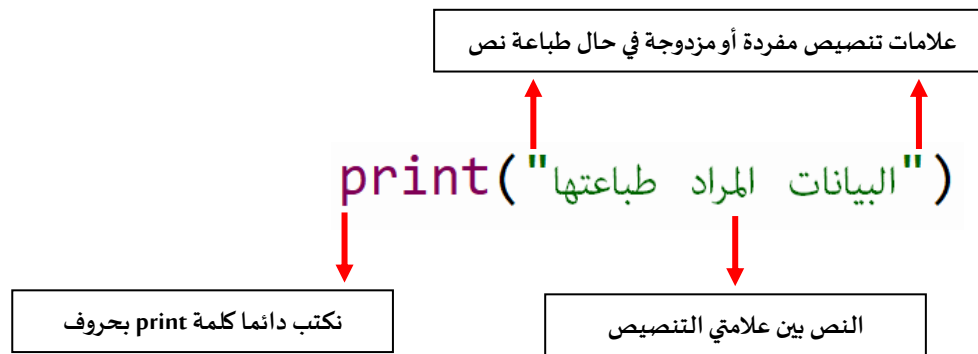
التعامل مع واجهة برنامج thony :



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

📄 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 تنصيب مزدوجة " " أو علامتي تنصيب مفردة ' ' .
- عند استخدام الجملة لعرض معادلة / رقم / متغير لا نستخدم علامتي التنصيب.
- 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 ونهايته.

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

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 x Exception x
>>> %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 x Exception x
>>> %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:

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

```
Shell x Exception x
>>> %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 :

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

```
Shell x Exception x
>>> %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 `\`

```
1 x=num1+\
2   num2+ \
3   num3
```

الملاحظات :

التقييم	موضوع الدرس : الدرس 6: المتغيرات و جملة الادخال
	التاريخ :

مهارات الدرس :

م	المهارة	م	المهارة
1	المتغيرات	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 _ إذا 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 totalpay و TotalPay مختلفان .
- 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 (string) و لو كان عدد صحيح فسيكون نوعه (integer) int 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

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

Shell × Exception ×

```
>>> %Run -c $EDITOR_CONTENT
Enter your nameAmeena
Welcome Ameena
>>> |
```

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.

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:

علامات تنصيص مفردة أو مزدوجة في حال طباعة نص

`input("الخ ... الرسالة - السؤال")`

نكتب دائما كلمة `input` بحروف صغيرة

سؤال يدل على المطلوب من المستخدم ادخاله

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 ")
```

```
>>> %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 وبعده 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)
```

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

الملاحظات:

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

مهارات الدرس :

م	المهارة	م	المهارة
5	دالة round()	1	دالة type()
6	دالة abs()	2	دالة int()
7	العمليات الحسابية	3	دالة float()
		4	دالة str()

سنقوم بتصميم برنامج لحساب ناتج جمع عددين وعرض الناتج على الشاشة، لذلك سنستخدم المتغير n1 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 :

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

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.

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 البرمجية الجاهزة مسبقا، يتم استدعاؤها عند اللزوم، عادة ما ترافقها بيانات خاصة بها تسمى " parameters " لإستخدامها للوصول إلى النتيجة المطلوبة.

دالة type() :

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

(اسم المتغير – البيانات) type()

type("Bahrain")

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

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

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	(نكتب الرقم – اسم المتغير - جملة الادخال) int	int(x) int(15.5)
float()	التحويل الى عدد عشري float	(نكتب الرقم – اسم المتغير - جملة الادخال) float	float(x) float(15)

طريقة تحويل أي قيمة الى عدد صحيح او عشري:

يمكننا استخدام الدالة بطريقتين ، الاولى تحويل القيمة المدخلة 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 المتغير n1 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 المستخدم.

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

الطريقة الاولى:

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

```
Shell x Exception x
```

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

```
Enter Number1 = 5
Enter Number2 = 6
the result = 11.0
```

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

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 الأماكن الأخرى يعتبر نوعهما نصي.

```
Shell x Exception x
```

```
>>> %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:

(نكتب الرقم - اسم المتغير - جملة الادخال)str

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:

الوصف	العامل الحسابي
لجمع في حال البيانات الرقمية	+
لربط في حال البيانات النصية	+
للطرح	-
للضرب	*
للقسمة	/
للحصول على الناتج الصحيح من عملية القسمة	//
للحصول على باقي عملية القسمة	%
للحصول على قوة العدد	**

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

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

النتيجة	مثال	العامل الحسابي
9	x=3 y=6 r=x + 6 print(r)	+ جمع الاعداد
7	x=13 y=6 r=x - y print(r)	- طرح الاعداد
40	x=8 y=5 r=x * y print(r)	* ضرب الاعداد
12.6	x=63 y=5 r=x / y print(r)	/ قسمة الاعداد
12	x=63 y=5 r=x // y print(r)	// ناتج القسمة بحيث يكون عدد صحيح من غير اعداد عشرية
3	x=63 y=5 r=x % y print(r)	% ايجاد باقي القسمة
25	x=5 y=2 r=x ** y print(r)	** قوة العدد (الأس)

الملاحظات :

التقييم	موضوع الدرس : الدرس 8: عوامل المقارنة والعوامل المنطقية
	التاريخ :

مهارات الدرس :

م	المهارة	م	المهارة
1	عوامل المقارنة	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 وقيمة • متغير ومتغير
!=	لا يساوي	X != 5 Y != X	2- يمكننا استخدام معاملات المقارنة للقيم النصية ايضا .
<	اصغر من	X < 5 Y <= X	3- قراءة المعاملات من اليسار الى اليمين
<=	أصغر من أو يساوي	X <= 5 Y <= X	4- يجب أن تكون المقارنة بين عدد و عدد أو نص ونص ، لا يمكنك أن تقارن بين عدد ونص.
>	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 $EDITC
False
```

```
1 print(2 < 5)
Shell × Exception ×
>>> %Run -c $EDITC
True
```

نلاحظ أن نتيجة المقارنة كانت True او False ، وتعتبر من أنواع البيانات وتسمى 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.

تُستخدم العوامل المنطقية للمقارنة بين شرطين في نفس الوقت، وتكون نتيجة استخدامها إما 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	$\text{not}(3 > 1)$	False
			$\text{not}(3 < 1)$	True

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

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

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

Shell × Exception ×

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

```
True
```

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

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

Shell × Exception ×

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

```
True
```

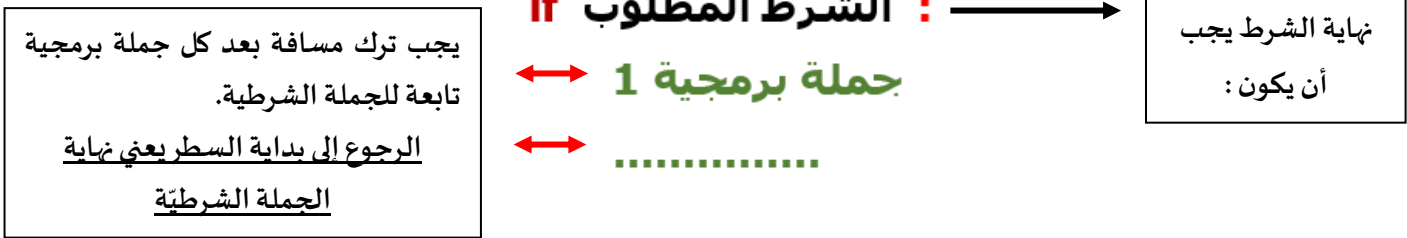

التقييم	الدرس 9: الجملة الشرطية	موضوع الدرس :
		التاريخ :

مهارات الدرس :

م	المهارة	م	المهارة
3	الجملة الشرطية المتداخلة	1	الجملة الشرطية البسيطة
		2	الجملة الشرطية الكاملة

الجملة الشرطية البسيطة:

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



مثال: اكتب برنامجاً بلغة البايثون كالتالي:

- يطلب من المستخدم ادخال درجته.
- 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")

```

```

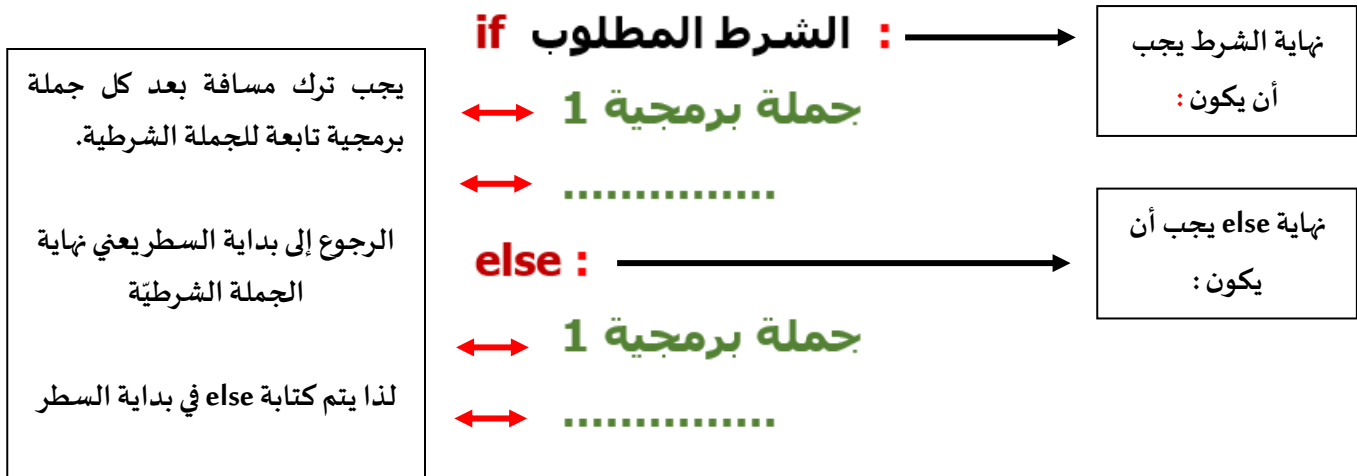
Shell x Exception x
>>> %Run -c $EDITOR_CONTENT
enter your mark:60
ناجح

```

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

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
راسب

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

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 x Exception x

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

Shell × Exception ×

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

```
12 ادخل درجة المشروع
23 ادخل درجة الامتحان
راسب
```

الملاحظات :

التقييم	الدرس 10: التعامل مع القوائم	موضوع الدرس:
		التاريخ:

مهارات الدرس :

م	المهارة	م	المهارة
5	دالة () index	1	دالة () split
6	دالة () pop	2	دالة () sorted
7	دالة () del	3	دالة () remove
8	طباعة عناصر القائمة من الاتجاه العكسي	4	دالة () append

أحد أنواع البيانات هي القائمة list والتي تسمح لنا بتخزين أكثر من قيمة في متغير واحد، ويمكن أيضا لهذه القيم أن تتكرر بداخل القائمة، بالإمكان تغيير / حذف / إضافة 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 نسيمه index ، دائما يبدأ الموقع من 0 ، فالمثال السابق لو أردنا طباعة القيمة الثانية و هي 32 ، يكون موقعها 1 و ليس 2 .

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

↓ ↓ ↓ ↓ ↓

موقع القيمة 0 1 2 3 4

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 :

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

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

أو هذه الطريقة :

```
1 num=["Muharraq","Jidhafs","Manama"]
2 print (num[0],num[1],num[2])
3 |
```

```
Shell x Exception x
>>> %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:
اسم المتغير الذي يحتوي على نص split()

مثال:

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

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 x Exception x
>>> %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 x Exception x
>>> %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، فمثلاً 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، فمثلاً 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 تصاعدياً، فمثلاً لترتيب القائمة stid تصاعدياً 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 تنازلياً مع اضافة تغيير بسيط الى دالة sorted :

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

```
Shell x Exception x
>>> %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 stdid يكون بالطريقة التالية:

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

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

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

طباعة عناصر القائمة من الاتجاه العكسي 

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

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

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

التقييم	الدرس 11: الدوال الاحصائية	موضوع الدرس :
		التاريخ :

مهارات الدرس :

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

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

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

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.

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

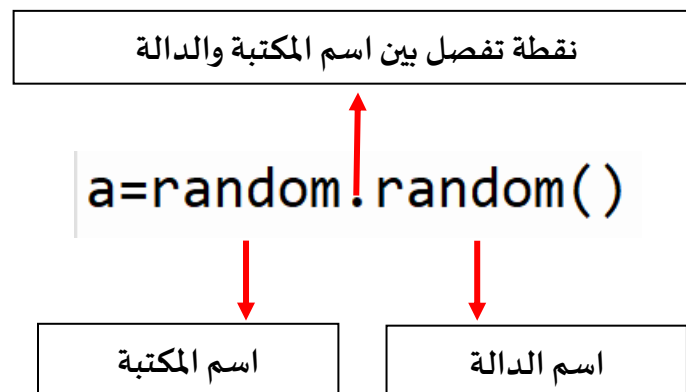
مهارات الدرس :

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

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
0.058014971805341764
```

دالة randint() :

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

```
1 import random
2 y=random.randint(10,20)
3 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- اختيار عدد عشوائي صحيح ضمن مجال محدد يبدأ من الصفر، في المثال التالي تم تحديد المجال 15 ، لذا سيكون الرقم الذي سيتم اختياره بين 0 و 14 (الرقم 15 ليس ضمن المجال).

```
1 import random
2 rand=random.randrange(15)
3 print(rand)
```

Shell × Exception ×

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

2- اختيار عدد 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).

```
1 import random
2 num=random.randrange(5,40)
3 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

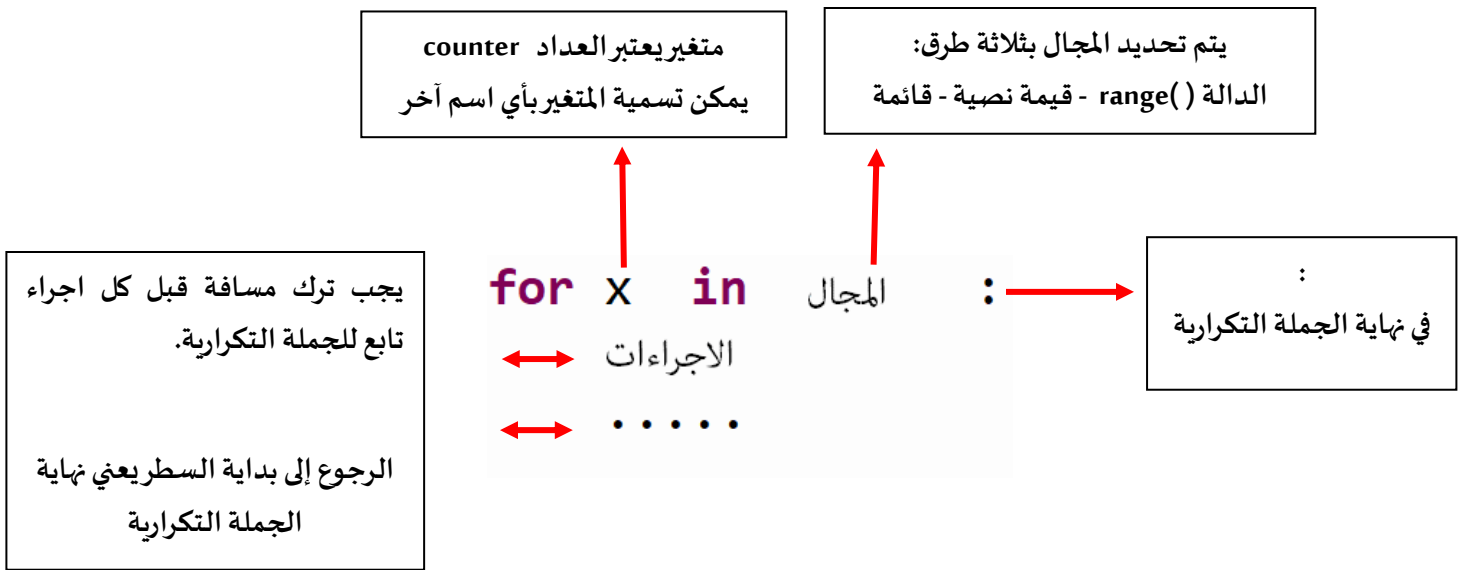
الملاحظات :

التقييم	موضوع الدرس : الدرس 13: الجملة التكرارية for
	التاريخ :

مهارات الدرس :

م	المهارة	م	المهارة
3	for x in "text"	1	for x in range()
4	break - continue	2	for x in list

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



📌 الجملة التكرارية مع range() :

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

(خطوات , نهاية , بداية) range()

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

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، وادخال الاسم مباشرة في القائمة، ألاحظ انه تم استخدام متغير واسناد قيمة له قبل البدء بالجملة التكرارية، وذلك حتى يتعرف البرنامج على المتغير ويتمكن 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 x Exception x

```
>>> %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 x Exception x

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

```
A
m
a
l
```

📖 إيقاف حلقة التكرار break :

يمكنني إيقاف 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 إيقاف التكرار.

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

Shell x Exception x

```
>>> %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 x Exception x

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

```
2
8
9
5
6
```

الملاحظات :

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

مهارات الدرس :

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

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



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 .

التقييم	الدرس 15: انشاء دوال	موضوع الدرس :
		التاريخ :

مهارات الدرس :

م	المهارة	م	المهارة
1	انشاء دالة بدون ارجاع قيمة	2	انشاء دالة مع ارجاع قيمة

الدالة هي كتلة من الجمل البرمجية الجاهزة مسبقا، يتم 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 المعقدة بحيث يتم تقسيم المشكل الرئيسي إلى مشكلات بسيطة يتم حلها 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 (العوامل) اسم الدالة :
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.

الآن سأقوم بإنشاء دالة اسمها sumTwo مع عاملين و تقوم بجمعها 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 x Exception x

```
>>> %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:



عند استدعاء الدالة في البرنامج الاساسي يكون بالشكل التالي:

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

مثال: الآن سأقوم بإنشاء دالة اسمها sumTwo مع عاملين و تقوم بجمعها 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

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 <u>condition1</u> : <u>do this</u> else: <u>do this</u>
if: elif....:else:	الجملة الشرطية المتداخلة	if <u>condition1</u> : <u>do this</u> elif <u>condition2</u> : <u>do this</u> else: <u>do this</u>
for.....:	الجملة التكرارية	for counter in range: <u>do this</u>
while:	الجملة التكرارية	while condition: <u>do this</u>
while.....: else:	الجملة التكرارية	while condition: <u>do this</u> else: <u>do this</u>
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"]

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

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