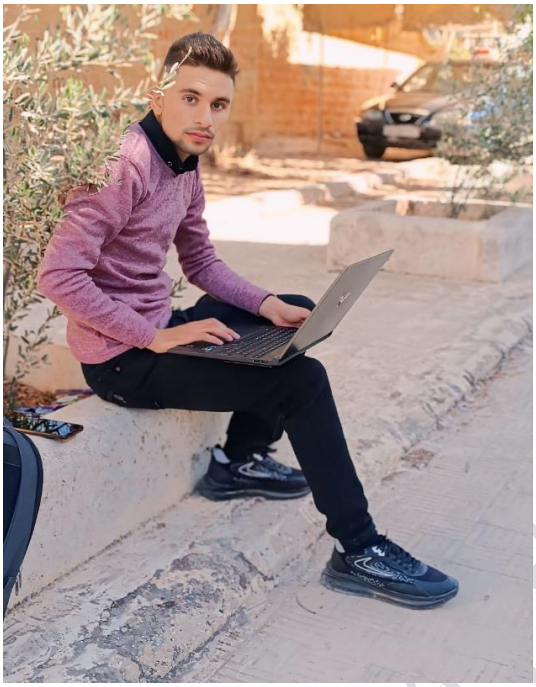
أساسيات بايثون

python

م.حسين محمود الجلعود



حسين محمود الجلعود

بايثون

حسين محمود الجلعود

PYTHON

كتاب المبتدئين

Hussein Mahmoud Al Jaloud

ماذا اتعلم من هذا الكتاب ؟

اساسيات بشكل كامل بالإضافة إلى البرمجة الكائنية أيضاً هناك الكثير من الأمثلة العملية و الشرح الواضح ومراعاة جميع الطلاب المبتدأ و المتوسط .

هذا الكتاب للمبتدئين في لغة بايثون أما الطالب الذي لديه معرفة سابقا عن هذه اللغة يمكن ان يفيدك إذا أردت أن تراجع بعض المعلومات ولكن الطالب الذي يكون لديه معرفة حول هذه اللغة بشكل كامل هذا الكتاب ليس لك.

ملاحظة . إذا درست خطوة بخطوة من أول الكتاب مع تطبيق كل الأمثلة العملية فأنت لست بحاجة إلى تسجيل كورس لتعلم أساسيات البايثون هذا الكتاب كافي و وافي جدا.

إهداء إلى روح أمي

بسم الله الرحمن الرحيم

إعداد : حسين محمود الجلعود

PYTHON

بايثون هي لغة عالية المستوى تستخدم على نطاق واسع في تطوير

البرمجيات وتطبيقات الويب وتحليل البيانات والتعلم الالي تم تصميم بايثون لتكون سهلة القراءة والكتابة مما يجعلها مثالية للمبتدئين والمحترفين على حد سواء تتميز لغة بايثون عن اللغات الأخرى بوجود مكتبات وأطر عمل قوية مثل Flaskو Django المستخدمة للتطوير الويب و Pandasو Numpy لتحليل البيانات و TensorFlow للتعلم الالي .

تدعم لغة بايثون البرمجة الكائنية والبرمجة الوظيفية وتتيح للمستخدمين كتابة كود نظيف ومنظم كما تتمتع بجتمع كبير وداعم .

مقدمة :

في عالم البرمجة المتسارع والمتغير باستمرار تبرز لغة بايثون كواحدة من أكثر اللغات شعبية وطلباً في سوق العمل في وقتنا الحالي تم تصميم لغة بايثون لتكون سهلة الاستخدام مما يجعلها الخيار المثالي لكل مبتدأ تأسست بايثون في أواخر الثمانينات على يد غيدوفان روسوم .ومنذ ذلك اليوم شهدت نمو هائلاً في شعبيتها تستخدم بايثون في مجموعة واسعة من التطبيقات بدأ من تطوير الويب والتطبيقات وعلوم البيانات والذكاء الاصطناعي والتعلم الآلي .

هذا الكتاب يهدف إلى تقديم دليل شامل للمبتدئين لفهم لغة بايثون وتطبيقاتها العملية سنبدأ في أساسيات اللغة ثم ننتقل لاستكشاف المفاهيم الأكثر تقدماً معا تقديم أمثلة عملية ومشاريع حقيقية. نأمل أن يكون هذا الكتاب مصدرإلهام لك لتطوير مهاراتك في البرمجة واستكشاف إمكانيات بايثون الواسعة.

فلنبدأ رحلتنا في عالم بايثون!

Variable : هو متغير يتم أعطاه قيمة ويمكنك تغير تلك القيمة في الذاكرة ويتم استدعائها من خلال المتغير .

تسمية المتغير :

هناك شروط يجب تحققها لتمسية المتغيرات :

-من الممكن أن يبدأ اسم المتغير بحرف أو شرطة سفلية "_" لكن بشرط أن يكون أول حرف هو حرف عادي وليس رقم

-يجب الأ يكون اسم المتغير محجوز في لغة البايثون مثل الكلمات المحجوزة وهي for و If و true و false وغيرها من الكلمات المحجوزة

- استخدام أسماء واضحة وصريحة للمتغيرات لتسهيل قراءة الشفرة

- قم بتجنب استخدام الاختصارات غير واضحة أو أسماء غير معبرة

- لايسمح بأن يكون اسم المتغير يبدأ بالأحرف اللاتينية وبعض الرموز مثل # و*و@و؟

بعض الأمثلة:

```
x=10
y=15
name = "hussein"
n = 1.0
```

عملية الطباعة

```
print ("hussein")
print (111)
```

التنفيذ :

hussein
111

التعليقات : عن طريق # إفادة التعليق هي تكتب فيه أن الكود عن ماذا يتحدث
أي تم تسليمه لشخص ثاني .
مثال :

- الربط بين متغيرين :

مثال :

```
name1="hussein"
name2="ali"
print (name1+name2)
```

التنفيذ:

hussein ali

أنواع البيانات :

String: يخزن قيمة نصية (كلمات أو نصوص)

مثال :

```
name="hussein"
name1="ali"
```

- لطباعة الحرف الأول فقط من الكلمة .

مثال :

```
 name = "hussein"
print (name[0])
```

التنفيذ:

نرى أثناء تنفيذ البرنامج أنه تطبع h فقط لأن في البرمجة نبدأ بعد الاحرف من الصفر

- أما إذا أردنا طباعة الحرف الأخير على الشكل التالي :

```
print (x[-1])
```

التنفيذ :

n

نلاحظ ناتج البرنامج طبع الحرف الأخير فقط

- أما إذا اردنا طباعة مجموعة من الأحرف .

مثال:

```
name = "ABCDEFG"
print (name [0:5])
```

التنفيذ:

ABCDE

- طباعة الأحرف بشكل كبير .

مثال :

```
name = "ali"
print (name.upper())
```

التنفيذ:

ALI

نلاحظ أن النص كان مكتوب بأحرف صغيرة وعند استدعاء upper() تمت كتابته بشكل أحرف كبيرة .

- طباعة الأحرف بشكل صغير .

مثال :

```
n = "ABCD"
print (n.lower())
```

التنفيذ:

abcde

نلاحظ أن النص كان مكتوب على شكل أحرف كبيرة والآن تمت كتابته بشكل صغير عند استعمال lower()

- طباعة أول حرف فقط بشكل كبير .

مثال :

```
n = "abcd"
print (n.capitalize())
```

التنفيذ:

Abcd

نلاحظ أن تم طباعة الحرف ألأول فقط بشكل كبير.

- طباعة أول حرف من كل كلمة بشكل كبير .

مثال:

```
n = "hussein amll"
print (n.title())
```

التنفيذ :

Hussein Amll

نلاحظ أن تم طباعة الحرف بشكل كبير من كلمة عند استخدام title()

- لطباعة قيمة ترتيب الحرف نستخدم index()

مثال:

```
n = "abcdef"
print (n.index('c'))
```

التنفيذ:

2

تمت طباعة قيمة ترتيب الحرف c .

- لمعرفة عدد أحرف أي نص نستخدم len() .

مثال :

```
x = "abcdefgh"
print(len(x))
```

التنفيذ:

8

تم حساب عدد الحروف في النص وتمت طباعة العدد.

- إذا أردنا تبديل بين حرفين نستخدم replace() .

مثال :

```
x = "BCDE"
print (x.replace("D","K"))
```

التنفيذ:

BCKE

نلاحظ أن تم حذف حرف D وضع مكانه حرف ال K .

- لمعرفة نوع البايانات إذا كانت نصية أو عددية.

مثال :

```
print (type (50))
```

التنفيذ:

<class 'int'>

نلاحظ عند استخدام type () حصلنا على نوع البيانات المستخدم .

مثال:

```
print (type (5.5))
```

التنفيذ:

<class 'float'>

مثال:

```
print (type (5+2j))
```

التنفيذ:

<class 'complex'>

- التحويل من int إلى float

مثال:

```
print (float(7))
```

التنفيذ:

7.0

نلاحظ أن تم تحويل نوع البيانات من الشكل الصحيح إلى الفواصل .
والعكس صحيح يمكن أيضاً أن نقوم بتحويل البيانات من الشكل العشري إلى الفواصل.

مثال:

```
print (int(8.5))
```

التنفيذ:

8

لاحظنا ايضاً أن تم تحويل البيانات من الشكل العشري إلى الصحيح.

- التقريب الأعداد .

مثال:

```
print (round(7.4))
```

التنفيذ:

7

عند تنفيذ البرنامج شاهدنا أن تم التقريب نحو الأسفل .

- ا لعملية بتم تقريب بشكل رياضي

مثال:

```
print (round(8.9))
```

التنفيذ:

9

أيضاً عند تنفيذ البرنامج شاهدنا أن تم التقريب ولكن نحو الأعلى .

الأرقام والعمليات الحسابية .

التعامل معا الأرقام والعمليات الحسابية هي نفسها يلي موجودة في الآلة الحاسبة .

مثال:

```
print(2+2)
print(4-2)
print (18/3)
print(18/7)
print(18.5/6.7)
print(9%4)
print(8%4)
print(8.75%5)
print(6*6)
print(7*7*7)
print(8**3)
print(-10**3)
```

التنفيذ :

4

2

6.0

2.5714285714285716

2.761194029850746

1

0

3.75

36

343

512

1000-

عملية

الأدخال:

مثال :

```
name=input("enter your name:")
age =input ("enter your age:")
print ("your is age" +age)
print ("your is name "  + name)
```

التنفيذ :

enter your name:hussein
enter your age:19
your name is hussein
your age is 19

مثال: اكتب برنامج يقوم بجمع عددين

```
N1=input ("enter number1:")
N2=input ("enter number2:")
Sum =int (N1)+int (N2)
print(Sum)
```

قبل تنفيذ البرنامج يجب أن نشرح بعض الأمور الغامضة.

عندما نقوم بعملية الادخال فأن أي شيء ندخله سواء كان نص أو عدد إذا كان عدد واستخدمنا عملية حسابية يجب بعدما نقوم بأدخال العدد وأردنا جمع العدد معا عدد ثاني لازم نقوم بتحويله إلى عدد صحيح لأن أي شيء يقوم المستخدم بأدخاله فأنه يدخل على شكل نص وبالتالي لايمكن أن نقوم بجمع نصيين معا بعض وبالتالي كتبنا الامر التالي حتى تجرى عملية التحويل من النص إلى عدد .

Sum =int (N1)+int (N2)

التنفيذ:

enter number1:5
enter number2:4
9

مثال: اكتب برنامج يقوم بجمع ثلاث أعداد .

```
N1=input ("enter number1:")
N2=input ("enter number2:")
N3=input ("enter number3:")
sum =(int (N1)+ int (N2)+ int(N3))
print ("sum =" + str(sum))
```

التنفيذ:

enter number1:4
enter number2:6
enter number3:5
sum=15

نشرح البرنامج .

بالبداية برنامجنا يقوم بجمع ثلاث اعداد ادخلنا هذه الاعداد ولكن لا تنفذ عملية الجمع ألا نقوم بتحويل من نص إلى اعداد صحيحة لأن سابقاً ذكرنا أي عدد يتم إدخاله عن طريق المستخدم يدخل على شكل نص ولايمكن أن تتم عملية جمع نصين معا بعض لذلك كتبنا هذا الأمر

```
sum =(int (N1)+ int (N2)+ int(N3))
```

بعدما تمت عملية الجمع أردنا إضهار النتيجة على الشكل التالي .
sum = ولكن ايضاً لا يمكن أن تتم عملية ربط بين نص وعدد عن طريق إشارة + الأ بشرط تكون أما الأثنين عدديين أو نصين لذلك أجرينا تحويل ال sum من أعداد صحيحة على نص لكي تتم عملية الربط بشكل صحيح على الشكل التالي .

```
print ("sum =" + str(sum))
```

مثال: اكتب برنامج يقوم بإدخال نصوص وطباعتها.

```
X=input("enter your lan1:")
Y=input ("enter your lan2:")
print("I love"+X)
print("I love"+Y)
```

التنفيذ:

enter your lan1: java
enter your lan: python
I love java
I love python

أنواع البيانات في لغة بايثون:

لغة البايثون تحتوي على5أنواع قياسية للبيانات وهم:

1-الأرقام
2-النصوص

3-القائمة |List

4-Tuple

5-القواميس |Dictionary

1-القائمة |List:

القائمة تحتوي على عدة عناصر يفصل بينها بفاصلةومغلقة بالرمز[]

مثال على انشاء قائمتين بالبايثون حيث نلاحظ انه من الممكن ان تحتوي على عدة

أنواع من البيانات.

مثال:

القائمة الأولى:

```
list = [ 'abcd', 745 , 2.23,
'Moustafa', 70.2 ]
smalllist = [123, 'Ali']
print (list)  طباعة بشكل كامل
print (list[0])  طباعة فقط الجزء الذي يكون ترتيبه صفر

print (list[1:3] 3  طباعة من ذو الترتيب 1حتى ال
print (list[2:])    طباعة من ذو الترتيب ال 2حتى اللانهاية
print (smalllist * 2)  طباعتهما مرتين
print (list + smalllist)  طباعتهما بشكل كامل كل من المتغيرين
```

التنفيذ:

['abcd', 745, 2.23, 'Moustafa', 70.2]

abcd

[745, 2.23]

[2.23, 'Moustafa', 70.2]

[123, 'Ali', 123, 'Ali']

['abcd', 745, 2.23, 'Moustafa', 70.2, 123, 'Ali']

Tuples 2-

الـTuplesمشابهة للقائمة ولكن الفرق الوحيد هو انها للقراءة فقط اي لايمكن اضافة عناصر جديدة بعد انشائها.

مثال:

```
tuple = ( 'abcd', 786 , 2.23,
'Moustafa', 70.2 )
tinytuple = (123, 'Ali')
print (tuple)          طباعة جميع القيم
print (tuple[0])       طباعة العنصر الأول
print (tuple[1:3]) طباعة العنصر من 1حتى 3
print (tuple[2:]) ابدأ طباعة من 2    حتى لانهاي
print (tinytuple * 2)                  طباعتها مرتان
print (tuple + tinytuple) طباعتهما معا بعض
```

التنفيذ:

```
   ('abcd', 786, 2.23, 'Moustafa',
70.20000000000003)
abcd
(786, 2.23)
(2.23, 'Moustafa', 70.20000000000003)
(123, 'Ali', 123, 'Ali')
('abcd', 786, 2.23, 'Moustafa',
70.20000000000003, 123, 'Ali')
```

-القواميس |Dictionary:

القواميس مشابهة للقائمة ولكن الفرق أنها تحتوي مفتاح-قيمة.

مثال:

```
dict = {'ali': 'john','code':6734,
'dept': 'sales'}
print (dict['ali']          طباعة القيمة المحددة(
print (dict['code'])       طباعة القيمة المحددة
print (dict)              طباعتها بشكل كامل
print (dict.keys())        طباعة جميع المفاتيح
print (dict.values())  طباعة جميع القيم
```

التنفيذ:

john

6734

{'dept': 'sales', 'code': 6734, 'ali':

'john'}

['dept', 'code', 'ali']

['sales', 6734, 'john']

Methods

داخل ال

Set

إذا أردنا دمج هاتين المتغيرين معا بعض نستخدم

Union(): تستخدم للدمج

مثال:

```
X={1,2,3}
Y={4,5,6}
print (X.union(Y))
```

التنفيذ:

{1,2,3,4,5,6}

لدينا أيضاً طريقة ثانية للدمج دون استخدام

Methods

مثال:

```
X={1,2,3}
Y={4,5,6}
print (X|Y)
```

التنفيذ:

{1,2,3,4,5,6}

add() : تستخدم لأضافة عناصر داخل متغير من النوع set

مثال:

```
X={1,2,3}
X.add(4)
print(X)
```

التنفيذ:

{1,2,3,4}

remove(): تستخدم لحذف عنصر من داخل متغير من النوع set

مثال:

```
X={1,2,3}
X.remove(1)
print(X)
```

التنفيذ:

{2,3}

discard() : تستخدم لحذف عنصر من داخل متغير من النوع set.

ولكن عندما نضع بداخلها عنصر حتى تقوم بحذفه ولم يكن هذا العنصر مخزن داخل المتغير set البرنامج ينفذ ولم يتم حذف أي عنصر ولكن نلاحظ أنها نفس عمل remove() ولكن الفرق بينهما

أن remvoe() أثناء وضع داخلها عنصر لم يكن مخزن داخل المتغير البرنامج أثناء التنفيذ يقوم بطباعة error.

مثال:

```
X={1,2,3}
X.discard(2)
print(X)
```

التنفيذ:

{1,3}

Clear(): تستخدم لحذف جميع العناصر المخزنة داخل المتغير يلي من النوع set.

مثال:

```
X={1,2,3}
X.clear()
print(X)
```

التنفيذ:

set()

Function

نتعلم كيف صناعة تابع وكيف نقوم بإستدعائه طبعاً سابقاً الكثير منا درس بعض لغات البرمجة أنا بظن معضمنا يعرف ماهو التابع وكيف استخدامه.

ولكن أنا سأشرح في هذا المحتوى كيف نتعلم صناعة تابع واستدعائه

يتم التصريح عن ال Function بالعبارة

Def متبوعة باسمها .

مثال:

```
def myname(str):
    print(str)
#Call function
myname('moustafa')
```

التنفيذ:

moustafa

نشرح الكود السابق .

Def هي العبارة المسبوقة باسم ال function (myname).

وتحوي على بارامتر من النوع str وتقوم بطباعة هذا البارامتر عند استدعائه .myname('moustafa')

مثال : اكتب برنامج يقوم بجمع عددين مستخدم ال function

```
def number(x,y):
        print(x+y)
number(3,4)
```

التنفيذ:

7

مثال: اكتب برنامج يقوم بحساب ضرب عددين .

```
def num (h,k):
       print(h*k)
num(4,5)
```

التنفيذ:

20

هناك بعض العمليات الرياضية والمنطقية يجب أن نأخذ فكرة عن هذه العمليات ولكن هذه العمليات موجودة في كل اللغات :

العملية الرياضية	عملها في البرمجة
++	للزيادة
--	للنقصان
/	القسمة
*	للضر ب
%	باقي القسمة
=	التساوي

نتعرف ايضاً على العمليات المنطقية .

العملية المنقطية	عملها في البرمجة
>><<	لأزاحة العنصر إلى اليمين أو اليسار
&	للربط بين شيئين and
^	رفع القوة
|	الخيار بين شيئين or

المعاملات :

المعامل	العملية المكافئة
Spam+=2	Spam=spam+2
Spam-=2	Spam=spam-2
Spam*=2	Spam=spam*2
Spam/=2	Spam=spam/2

من أجل اسناد قيمة جديدة لمتغير باالعتماد على قيمة المتغير الحالية نقوم باستخدام معامالت

الزيادة كما في المثال التالي:

```
spam=25
spam=spam+5
print(spam)
```

التنفيذ:

30

يمكن القيام بهذه العملية من خلال استخدام المعامل+=

مثال:

```
spam =25
spam+= 5
print(spam)
```

التنفيذ:

30

لدينا مثال لكن المدخلات عبارة عن نصوص وليس اعداد .

```
spam="hello"
spam+="world!"
print (spam)
```

التنفيذ:

helloworld!

نشرح ماكتبناه في الكود

بالتفصيل.

التعليمةspam += 2تعطي نفس نتيجة التعليمة spam = spam +2 ولكن التعليمة الأولى هي مختصرة.

المعامل=+يتم استخدامه لإضافة أعداد صحيحة مع بعضها أو لإضافة سلاسل نصية مع بعضها أو حتى لإضافة مواد أو قيم للقوائم.

Function – return

التعليمة return تتبع باسم المتغير المراد إعادة قيمته أو يمكن أن

تتبع بعبارة جبرية بدال ً من اسم المتغير وفي هذه الحالة يتم إعادة القيمة بعد حساب العبارة الجبرية.

مثال: اكتب برنامج جمع عددين ولكن يقوم بإعادة النتيجة.

```
def sum(a,b):
      return a+b
spam = sum(4,5)
print (spam)
```

التنفيذ:

9

التابع sum(4,9) سوف يقوم بحساب القيمة 42 والتعليمة return سوف تقوم بإعادة هذه القيمة.

ملاحظة : سطر ال return يجب أن يكون اخر سطر في ال function لأن أي سطر يتم كتابته بعدها لا ينفذ .

مثال:

```
def sum(a,b):
       return a+b
       print ("hello world")
spam = sum(4,5)
print (spam)
```

عند تنفيذ هذا الكود :

9

لكنه لم ينفذ عملية الطباعة يلي كتبت

.

مثال

:

```
def sum(a,b):
       print("hello world")
       return a+b
spam = sum(4,5)
print (spam)
```

التنفيذ:

hello world

9

نلاحظ أن تم تنفيذ الكود كامل وبشكل صحيح أيضاَ .

مثال : اكتب برنامج يقوم بحساب عمرك بلإيام .

```
def clacdays(age):
        return "your age in days is "+str(age*365)+"days"
print (clacdays(19))
```

عند تنفيذ البرنامج .

your age in days : 6935 days

مثال: اكتب برنامج يقوم بحساب العمر بلإيام ولكن يلي بقوم بإدخال العمر المستخدم . (اكيد فهمت ماذا بقصد).

```
def clacdays(age):
        return "your age in days is "+str(age*365)+"days"
print (clacdays(int(input("enter your age:"))))
```

تنفيذ البرنامج.

enter your age :25

your age in days:14600 days

نذكر بعض الرموز:

الإشارة	وظيفتها	مثال
>	اكبر	a>b
<	اصغر	a<b
>=	اكبر او يساوي	a>=b
<=	اصغر او يساوي	a<=b
==	يساوي	a==b
=!	لاتساوي	a!=b

المعاملات المنطقية or,and : المعاملات المنطقية and , or
تساعدنا في الشروط المعقدة الخاصة بتعليمات if and while

المعامل and يوضع بين عبارتين جبرييتين ويعيد القيمة Trueإذا كانت كلتا العبارتين الجبريتين محققتين.

المعامل or يوضع بين عبارتين جبريتين ويعيد القيمة Trueذا كانت إحدى العبارتين محققة.

جدول الحقيقة للمعامل and :

A	and	B	النتيجة
True	and	True	True
True	and	False	False
False	and	True	False
False	and	False	False

جدول الحقيقة للمعامل or :

A	or	B	النتيجة
True	or	True	True
Treu	or	False	True
False	or	True	True
False	or	False	False

جدول الحقيقة للمعامل not:

not A	النتيجة
not True	False
not False	True

المعامل and يستخدم من أجل اختصار تعليمات if

مثال:

```
if 25 > 5:
      if 5 < 9:
            print ("test")
```

التنفيذ:

test

نلاحظ أن تم تنفيذ البرنامج بعد تحقق الشرطين .

مثال : انتبه حتى تعرف أكثر ماهي وظيفة المعامل وماهو الفرق بين الكود السابق والكود التالي.

```
if 25 > 5 and 5 < 9:
        print ("test")
```

التنفيذ : نلاحظ الفرق بين الكود السابق والكود التالي ومن هذا الكود يمكن أن نقول المعامل and من أجل اختصار تعليمات if .

المعامل or يستخدم بدلاً من تعليمة elif .

مثال:

```
if 5  !=  5 :
       print("test")
elif 15 > 10:
      print ("test")
```

التنفيذ:

test

مثال : في حال استخدمنا المعامل or .

```
if 5 != 5 or 15 > 10:
         print ("test")
```

التنفيذ :

test

بالمقارنة بين الكودين السابقين نلاحظ الفرق بينهما وحصلنا أيضاً على وظيفة المعامل .

and

test

العبارة الشرطية if :

وهي لأختبار حالة معينة إّذا كانت صحيحة فسوف ينفذ مجموعة من الأوامر .

مثال:

```
x=4
if x==4:
    print(x)
```

التنفيذ:

4

العبارة الشرطية else – elif:

تستخدم else عندما نريد تنفيذ مجموعة أوامر عنا لا يتحقق الشرط أي يكون False

تستخدم elif عندما نريد اختبار اكثر من حالة للشرط .

مثال :

```
x=3
o=6
y=5
if x==4:
    print(x)
elif o==6:
     print(o)
else:
    print(y)
```

يكون الخرج6 لان الشرط الثاني محقق وتمت طباعة المتحولO
أما لو لم يتحقق الشرطان لتمت طباعة المتحول y .

التنفيذ :

```
6
```

مثال 2: في حال لم يتم تحقق الشرطين .

```
x=3
o=6
y=5
if x==4:
    print(x)
elif o==7:
     print(o)
else:
    print(y)
```

التنفيذ :

5

لأن الشرط الأول والثاني لم يتحقق وبالتالي تمت طباعة قيمة ال y

لدينا مثال التالي :

```
email =" x@gmail.com "
password = 1234
if email == "x@gmail.com " and password ==1234:
      print("welcome")
elif email == "x@gmail.com " and password ==1634:
      print("invalid password")
elif email == "y@gmail.com " and password ==1234:
      print("invalid email")
else :
       print("invalid password and email")
```

التنفيذ :

invalid password and email

مثال: اكتب برنامج يقوم بحساب نتيجتك في مادة الرياضيات ولكن استخدم الرمز x يدل على علامتك في المادة . يعني العلامة تخزن ضمن الرمز x

```
x= input ("enter x")
if (x) >= 90:
     print ("excellent")
 elif (x) >= 80:
      print ("very good")
 elif (x) >= 70:
      print ("good")
 elif (x) >= 50:
      print ("successful")
else:
      print("not successful")
```

التنفيذ : في حال كانت علامة الطالب تساوي 91.

excellent

التنفيذ: في حال كانت علامة الطالب تساوي 70.

good

التنفيذ : في حال كانت علامة الطالب تساوي 40.

not successful

مثال: اكتب برنامج يقوم بمقارنة بين ثلاث اعداد واكبر عدد يقوم بطباعته.

```
def prr (x,y,z):
    if x > y and x > z:
         return x
    elif y > z and y > x:
           return y
    else:
           return z
print(prr (50,40,100))
```

التنفيذ:

100

مثال : اكتب برنامج آلة حاسبة.

```
num1 = float(input("enter number:"))
operator = input("enter: / +,-,*:")
num2 = float(input ("enter number:"))
if operator=="+":
      print(num1+num2)
elif operator =="-":
       print (num1-num2)

elif operator =="*":
       print (num1*num2)
elif operator =="/":
       print (num1/num2)
else:
        print("error")
```

التنفيذ:

```
enter number :5
enter : /,+,-,*: +
enter number:10
15
enter number :5
enter : /,+,-,*: *
enter number:10
50
enter number :50
enter : /,+,-,*: -
enter number:10
40
enter number :50
enter : /,+,-,*: /
enter number:10
5
```

حلقة while :

طالما أن الشرط محقق سوف تكرر تنفيذ التعليمات التي تليها.

```
i = 4
while i < 9:

    print(i)
i = i + 2
```

سوف يتم طباعة المتحول x ثالثة مرات الى ان يصبح المتحول I أكبر من الـ9

ويتم الخروج من الحلقة.

التنفيذ:

4

6

8

ملاحظة:

يجب الانتباه الى الحلقات اللا نهائية حيث أنه طالما الشرط صحيح سيتم التنفيذ بعدد لا نهائي.

break :

تستخدم للمقاطعة في اللحلقات التكرارية ال for ,while

طبعاً حلقة ال for نشرحها بعد إنهاء حلقة while والتدريب على أكثر من مشروع حتى نثبت المعلومة بشكل صحيح .

مثال :

```
x =1
while x<=10:
    if x==8:
        break
    print (x)
    x=x+1
```

التنفيذ :

1 2 3 4 5 6 7

نلاحظ أنها تمت عملية المقاطعة عند العدد 8 .

continue : تستخدم لإعادتنا إلى بداية الحلقة وتجاهل باقي التعليمات كما أنها يمكن ان تستخدم في for , while.

```
for letter in 'Python':
    if letter == 'h':
          continue
    print ('Current Letter :', letter)
```

عند تنفيذ البرنامج :

Current Letter : P

Current Letter : y

Current Letter : t

Current Letter : o

Current Letter : n

نشرح كل خطوة بخطوة .

في هذا المثال سيتم طباعة كل حرف في كلمةPythonوعند الوصول الى المحرفH سيتم تجاهل باقي التعليمات والعودة لبداية الحلقة .

مثال شامل : سانقوم بكتابة برنامج يطرح لنا سؤال إذا كان الجواب صح يطبع لنا (فزت) وإذا كانت النتيجة خطأ يطبع لنا (خسرت).

```
X=5
Y
I=0
Z=3
Lsoe =False
while X=Y and not Lose:
        if I<Z:
          Y=int (input("what is 2+3?")
               I=I+1
        else :
            Lose=True
    if Lose:
      print ("you Lose")
    else :
        print ("you win")
```

نشرح الكود خطوة بخطوة .

بدايةً برنامجنا هو يطرح لك سؤال وأنت تقوم بالإجابة عن السؤال السؤال الذي سيطرح على المستخدم هو what is 3+2?

الرمز X هو الجواب الصحيح

الرمز Y هو الجواب الذي يكتبه المستخدم

الرمز I هو عبارة عن عداد يبدأ من الصفر وكلما السؤال يطرح على المستخدم تزداد قيمة I بمقدار واحد

الرمز Z هو يدل على عدد المحاولات التي يطرح بها السؤال على المستخدم

التنفيذ : في حال كانت جميع الإجابات التي أدخلها المستخدم تكون خطأ.

what is 2+3? 7

what is 2+3? 2

what is 2+3? 6

you Lose

التنفيذ: في حال تكون الإجابة التي أدخلها المستخدم تكون صحيحة.

what is 2+3? 5

you win

بتمنى أنك فهمت الكود بشكل صحيح .

حلقة For :

تعتبر نوع اخر من الحلقات التكرارية في بايثون .

1.لكلمة for

2.اسم المتغير

3.الكلمة in

4.القيمة النصية أو المتغير الذي يحويعلى قيمة نصية

5.النقطتان :

6.كتلة من التعليمات

مثال :

```
for x in "hussein":
        print(x)
```

التنفيذ :

h

u

s

s

e

i

n

في هذا المثال نلاحظ أن تمت طباعة كل حرف بحرفه لأن ال

x بالبداية لايملك أي قيمة وبعد وضع كلمة hussein يبدأ ال x يأخذ قيمة ويطبعها (حرف واحد ويطبعها وتدور الحلقة حتى يرجع يأخذ قيمة ثانية ويطبعها هكذا بتم الأمور .)

range: يأخذ قيمة عدد صحيح ويعيد قيم لمجال هذه القيمة وقيم هذا المجال يمكن أن تستخدم في حلقة for من أجل تكرار الحلقة عدد معين من المرات.

نأخذ مثال بسيط ثم نقوم بشرحه بالتفصيل.

```
for x in range(5):
    print(x)
```

التنفيذ :

0

1

2

3

4

بالبداية المتغير x تكون قيمته صفر لأن ما وضعنا بدايته من أي رقم لكن فقط وضعنا نهايته تكون عند الرقم 5 .

الرقم 5 يدل على عدد الأرقام التي يأخذها المتغير x لأن عملية الطباعة توقفت عند الرقم 4 وليس 5 لأن الرقم 5 يدل فقط على عدد الأرقام التي يأخدها المتغير .

دائماً المتغير معا حلقة for تكون بدايته من الصفر عندما لانضع له بداية فقط .

مثال:

```
for x in range (4,10):
      print(x)
```

التنفيذ:

4

5

6

7

8

9

هنا المتغير x أخذ بداية من الرقم 4 حتى الرقم 10 وبالتالي الأرقام الموجودة ضمن المجال من 4 حتى 10 هي 6 ارقام وبالتالي المتغير توقف عند العدد 9 وليس العدد 10 .

لأن لو المتغير توقف عند الرقم 10 كان تمت طباعة 7 ارقام وليس 6

مثال:

```
lan =["java","c++","html"]
for x in range(len(lan)):
   if len(lan)=="c++":
        print(len(lan)+"I love c++")
    else:
        print(len(lan))
```

التنفيذ :

java

c++ I love c++

html

بالبداية المتغير x بما أن لانحدد له بداية فتكون قيمته صفر ولكن نهايته تكون عند len(lan) . وسابقاً شرحنا التابع len(lan) ماهي وظيفته سنعود بشرح مرة ثانية حتى نفهم بشكل صحيح كيف الأمور بتم .

len(lan): يقوم بحساب عدد الاحرف في نص من النوع string

ويقوم بحساب عدد الكلمات في نص من النوع list.

وبالتالي حصلنا على نهاية المتغير x وهي 3 هذا يؤدي أن المتغير xبدايته من الصفر ونهايته عند الرقم 3 وبالتالي الأرقام يلي موجودة ضمن المجال هي 0 و1 و2 وهنا نرى أن تمت طباعة ثالاث جمل

أما الشرط if عند وصول الرقم 1 مقابل c++ يكون الشرط محقق وبالتالي نرى أنه تمت عملية طباعة I love c++ إذا كانت عندك مشكلة فيك تزور بعض الفيديوهات الموجودة في اليوتيوب أو استمعال الذكاء الأصطناعي في مساعدتك .

مثال:

```
x={ "name": "hussein",
      "age": "25"}
for y in x :
       print (x[y])
```

التنفيذ:

hussein

25

بالبداية قيمة y تكون صفر ولكن حتى نعرف نهايته استخدمنا المتغير x الذي يخزن قيم نصية من النوع dictionary نستخدم المتغير لتحديد مجال ال y عندما يكون لدينا نص من النوع dictionary.

try و except : تستخدم للتعامل معا الأخطاء التي قد تحدث أثناء تنفيذ البرنامج .

إليك الشرح المفصل .

try :

يكتب الكود الذي قد يسبب الاستثناء

except SomeException as error:

هنا يكتب الكود الذي قد يسبب الاستثناء

try : في داخلها تضع الكود الذي قد يسبب الخطأ إذا لم يتنفذ هذا الكود دون أي مشاكل يتم تجاهل except .

except : إذا حدث استثناء في try يتم الانتقال مباشرة إليها .

ويمكنا أن نحدد نوع الاستثناء الذي ترغب في التعامل معه مثل .

. (ValueError)

مثال:

```
try :
   n=int (input("enter number"))
   print(n)
   x=1/0
except ZeroDivisionError as error:
      print(error)
except ValueError as error:
      print(error)
print("hello")
```

التنفيذ: نشرح ثم نرى تنفيذ البرنامج بالبداية نطلب من المستخدم يقوم بإدخال عدد .

إذا المستخدم أدخل قيمة غير رقمية سيتم رفع ال ValueError

أما في حال دخل المستخدم قيمة رقمية سيتم رفع ال ZeroDivisionError .لأن في خطأ في الكود وهو 1/0 .

ومنه نستنتج أن في حال المستخدم دخل قيمة غير رقمية فيكون هذا الخطأ العام في البرنامج أما في حال أدخل المستخدم قيمة رقمية فيكون الخطأ جزء من البرنامج وهو 1/0 زيجب أن نأخذ بعين العتبار أن البقية من الكود يتم تنفيذه يعني يقوم بطباعة hello حتى إذا كان لدينا خطأ في البرنامج وفينا نقول أن هذه في وظيفة try except, تجعل البرنامج ينفذ و تنبه أن الأخطاء .

نرى التنفيذ معاً .

enter number: test

invalid litera for int () with base 10 : ‘jhfauylsf’

hello

المستخدم أدخل test بدلاً من قيمة رقمية فظهر الخطا العام في البرنامج

ولكن لم يتوقف عند اللخطأ فقام بطباعة hello .

ولكن لو أدخل قيمة رقمية يظهر لنا الخطأ الثاني في البرنامج وهو 1/0

الخلاصة : فينا نقول أن اخر except في البرنامج هي الخطأ العام و أي شيء قبلها من ال except فينا نقول عنه خطأ جزءي من البرنامج.

القراءة من الملفات:

حتى الان كل دخل نريد أن نعطيه للبرنامج هو دخل يقوم المستخدم بكتابته

برامج بايثون يمكنها فتح وقراءة الملفات بشكل مباشر من القرص الصلب

يوجد ثلاث خطوات من أجل قراءة محتوى أي ملف:

1.فتح الملف

2.قراءة المحتوى عند طريق اسناده إلى متغير

3.إغالق الملف

Open() : يأخذ قيمتين اول قيمة هي مسار الملف الذي تريد قراءته أو التعديل عليه ثانية قيمة هي التي تحدد لك على ان تريد الملف للقراءة أو الكتابة وهذه القيم تكون (r,r+,w,w+,a,a+) .

نشرح وظيفة هذه القيم بالتفصيل.

r : تسمح لك بقراءة الملف فقط.

r+ :تسمح لك بقراءة الملف وايضاً الكتابة فيه (تعديل عليه).

w : تسمح لك بالكتابة ولكن بشرط تمسح كل المحتوى الذي كان في الملف .

w : تسمح بالكتابة والقراءة ايضاً تمسح المحتوى القديم .

a : تسمح لك بالكتابة دون مسح المحتوى القديم .

a : تسمح لك بالكتابة والقراءة دون مسح المحتوى القديم.

لكن في ملاحظة يجب أن نأخذها بعين الاعتبار وهي .

في حال كانت القيمة الثانية (w,w+,a,a+) وكان مسار الملف الذي وضعه المستخدم أي القيمة الأولى غير موجودة (غير موجود ملف) تسمح لك هذه القيم الثانية بخلق ملف جديد وتبدأ بالكتابة داخله ولكن معا تلك القيم فقط.

مثال:

```
filse = open("test.txt","r"):
      print(filse.readable())
flise .close
```

نشرح البرنامج بالتفصيل .

test.txt : مسار ملف معين ولكن بشرط أن يكون هذا الملف جنب ملف البايثون .

r هي القيمة الثانية شرحناها سابقاً .

filse : اسم المتغير .

open بداية فتح الملف .

readable : تساعدنا لنعرف اذا كان الملف قابل للقراءة أو لا إذا كان قابل للقراءة يكون خرج البرنامج true في حال كان غير قابل يكون الخرج false .

filse.close : تقوم بتقفيل الملف.

التنفيذ:

True

بما أن الناتج true فهذا يؤدي أن الملف قابل للقراءة.

مثال:

```
filse = open("test.txt","r")
      print(filse.readline())
flise .close
```

readline () : تقوم بطباعة الملف سطر سطر .

لو أفرتضنا أن داخل الملف يوجد هذه الأسماء hussein,ali

حتى البرنامج يطبع الاسمين يجب ان نكرر هذا السطر مرتين

print(filse.readline())

وبالتالي يكون شكل البرنامج بشكل .

```
filse = open("test.txt","r")
    print(filse.readline())
    print(filse.readline())
flise .close
```

التنفيذ:

hussein

ali

مثال:

```
filse = open("test.txt","r")
    print(filse.readlines())
flise .close
```

readlines : طباعة الملف بشكل كامل ولانحتاج إلى تكرار هذا السطر

print(filse.readline())

التنفيذ:

hussein

ali

الربط بين نصوص والأرقام معا بعض .

سابقاً استخدمنا اثناء الربط بين نص ورقم عملية تحويل الرقم الى نص لكي تنجح عملية الربط .

مثال :

```
name = "hussein"
 age = 25
 print("my name is "+name+"my age is "+age)
```

في هذا المثال لم نستخدم عملية تحويل الرقم الى نص وبالتالي الكود لم ينفذ بشكل صحيح .

تنفيذ البرنامج :

Traceback (most recent call last):

File "<pyshell#3>", line 1, in <module>

print("my name is "+name+"my age is "+age)

TypeError: can only concatenate str (not "int") to str

نلاحظ رسالة الخطأ بأنها تخبرنا لايمكن الربط بين النص والرقم ويجب علينا عملة التحويل .

مثال:

```
name = "hussein"
age = 19
print("my name is "+name+"\nmy age is "+str(age))
```

التنفيذ :

my name is hussein

my age is 19

بهذا المثال أثناء عملية الربط بين الرقم والنص استخدمنا قاعدة تحويل الأرقام الى نصوص وبالتالي تنفذ الكود بشكل صحيح.

لدينا طريقة ثانية لربط بين النصوص والأرقام دون حدوث خطأ في البرنامج وهي مختصرة بشكل افضل وهي طريقة ال formating .

مثال :

```
name = "hussein"
age = 19
print("my name is %s my age is %s" % (name , age ))
```

التنفيذ:

my name is hussein my age is 19

%s : تستخدم معا النصوص .

%f : تستخدم معا الأرقام العشرية .

%d : تستخدم معا الأرقام الصحيحة .

مثال :

```
    name = "hussein"

age = 19

print("my name is %s my age is %f" % (name , age ))
```

التنفيذ:

my name is hussein my age is 19.000000

نلاحظ عندما استخدمنا %f ظهر لنا الرقم 19 بشكل 19.000000

ولكن يمكن أن نحدد عدد الاصفار بعد الرقم 19

مثال:

```
name = "hussein"

age = 19

print("my name is %s my age is %0.3f" % (name , age ))
```

التنفيذ:

my name is hussein my age is 19.000

لدينا ايضاً مثال ثاني ولكن بطرق مختلفة .

```
name = "hussein"

age = 19
print("my name is {:s} my age is {:f}".format(name,age))
```

التنفيذ:

my name is hussein my age is 19.000000

ترتيب العناصر باستخدام ال formating .

مثال:

```
a = 4
b = 5
c = 8
        c  b   a
        ↓  ↓   ↓
print("{2} {1} {0}".format(a,b,c))
```

التنفيذ:

8 5 4

ويمكن ايضاً أن نحدد نوع البيانات (الاعداد) الذي نريد ترييبها .

مثال:

```
a = 1
b = 2
c = 3
print("{2:d} {0:f} {1:d}".format(a,b,c))
```

التنفيذ:

3 1.000000 2

هناك أيضاً طريقة ثانية يمكن أن نستخدمها في ترتيب العناصر
ويمكن القول إنها طريقة مختصرة وتوفر الوقت للمبرمج

مثال:

```
a = 1
b = 2
c = 3
print(f"{a} {b} {c}")
```

التنفيذ:

1 2 3

Modules :

في لغة البرمجة بايثون يوجد الكثير ضمنها من ال modules وهذا مما جعلها مميزة وسهلة عن بقيات اللغات مثل .

Math : مسؤولة عن التعامل معا الأرقام والعمليات الرياضية

Random : مسؤولة عن التعامل معا ظهور الحالات العشوائية.

Modules : هو ملف بايثون يحتوي على مجموعة من المتغيرات والكلاسات والدوال ويسمح لنا بإعادة استخدام الكود ويساعد في تنظيم المشاريع .

مثال: ليكن لدينا ملف python اسمه x.py ويحوي هذا الملف على function اسمها callName المطلوب استدعدي هذا الملف إلى ملف بايثون اخر اسمه y.py .

الملف x.py

```
def callName (name):
    print("my name is "+name)
```

الملف y.py

```
import x
x . callName("hussein")
```

Import تستخدم لستدعاء الملفات (المكتبات)

X هو اسم الملف المراد استدعاءه إلى الملف p.py

callName اسم ال function .

تنفيذ البرنامج :

إذا كان لدينا ال modules : يحتوي على أكثر من function وبالتالي يمكنا أن نستدعي ال function بشكل التالي .

```
form x import callName
callName (name)
```

ولكن بعض المكتبات تكون جاهزة فقط علينا أن نقوم بستدعاء المكتبة في الأمثلة السابقة قمنا في نفسنا بصناعة المكتبات ومن ثم استدعاءها

إليك مثال نقوم بستدعاء المكتبة .

```
import math
print(math.cos(0))
```

التنفيذ :

1

مثال 2:

```
import random
print(random.randint(0,15))
```

التنفيذ:

```
2
print(random.randint(0,15))
14
print(random.randint(0,15))
3
print(random.randint(0,15))
14
print(random.randint(0,15))
5
print(random.randint(0,15))
10
```

Random : هي عبارة عن مكتبة (ملف)

Randint : تأخذ مجال من الاعداد وتبدأ بظهور الاعداد بشكل عشوائي نلاحظ أن كلما ضغطنا على زر التنفيذ يظهر لنا عدد مختلف عن العدد السابق .

لمعرفة جميع ال function الموجودة داخل ال modules .

إليك الحل :

```
import random
print(dir(random))
```

التنفيذ:

['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST', 'SystemRandom', 'TWOPI', '_ONE', '_Sequence', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '_accumulate', '_acos', '_bisect', '_ceil', '_cos', '_e', '_exp', '_fabs', '_floor', '_index', '_inst', '_isfinite', '_lgamma', '_log', '_log2', '_os', '_pi', '_random', '_repeat', '_sha512', '_sin', '_sqrt', '_test', '_test_generator', '_urandom', '_warn', 'betavariate', 'binomialvariate', 'choice', 'choices', 'expovariate', 'gammavariate', 'gauss', 'getrandbits', 'getstate', 'lognormvariate', 'normalvariate', 'paretovariate', 'randbytes', 'randint', 'random', 'randrange', 'sample', 'seed', 'setstate', 'shuffle', 'triangular', 'uniform', 'vonmisesvariate', 'weibullvariate']

مثال2:

```
import math
print(dir(math))
```

التنفيذ:

['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'cbrt', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'exp2', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'sumprod', 'tan', 'tanh', 'tau', 'trunc', 'ulp']

نلاحظ أن تمت عملية ظهور جميع ال function الموجودة داخل ال modules.

يمكنا ايضاً استدعاء المكتبات بطريقة ثانية .

إليك مثال يوضح لك .

```
from random import*
print(randint(0,5))
```

التنفيذ:

4

print(randint(0,5))

2

print(randint(0,5))

5

print(randint(0,5))

3

print(randint(0,5))

3

OOP

Object Orlented Porgramming

البرمجة الكائنية التوجه: هي نمط برمجي يركز على استخدام الكائنات التي تحتوي على بيانات (object) في (methods) وسلوكيات (Attributes) ال oop في بايثون تدعم المطورين بإنشاء كائنات تعكس المفاهيم من العالم الحقيقي .

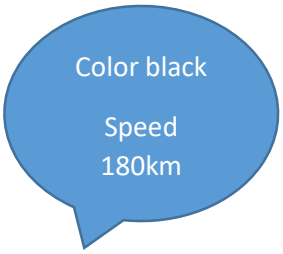
الكائنات (object) : هي مثيلات من فئات ال (Classes) وتحتوي على البيانات والسلوكيات.(أي شيء نراه يعتبر كائن مثل السيارة الطائر .....)

الفئات (Classes) : هي قوالب أو نماذج تستخدم لإنشاء كائنات تحدد الفئة الخصائص والدوال التي ستشاركها الكائنات .

مثل السيارة هي كائن لها صفات وسلوكيات إليك الجدول التالي

Actions	Attributes
Stop	Color black
Move	Speed 180km

Function



Variables

العملي للمثال السابق .

```
class car:
    def __init__ (self,color,speed):
        self.color1=color
        self.speed1=speed


BMW= car("black",180)
print(BMW.color1)
print(BMW.speed1)
```

التنفيذ:

black

180

نشرح الكود السابق بالتفصيل.

في هذا الكود تم إنشاء class بكتابة كلمة class فقط واسم هذا ال class هو car

color, speed : هي عبارة عن متغيرات

BMW : هو ال object الذي تم إنشاءه عن طريق اسم الكلاس وتمرير له قيم .

مثال:

```
class car :

    def prr (self):
        print("hussein")

    def __init__(self):
        print("hello")


car()
```

نلاحظ في هذا المثال يوجد كلاس اسمه car ويوجد ضمنه اثنين من function ولكن اثناء استدعاء الكلاس تمت عملية طباعة hello فقط لأن اثناء استدعاء الكلاس نفذ ال function الرئيسية فقط وهي

```
def __init__(self):
        print("hello")
```

التنفيذ:

hello

ولكن إذا أردنا أن تنفذ عملية طباعة اسم hussein علينا أن نستدعي ال function وليس اسم الكلاس فقط.

```
class car :

    def prr (self):
        print("hussein")

    def __init__(self):
        print("hello")


car().prr()
```

نلاحظ هنا أن تم استدعاء اسم ال function (prr) وبالتالي ستتم عملية طباعة اسم hussein .

التنفيذ:

hussein

ملاحظة : بالنسبة لإسماء ال function او الكلاسات أنت مخير فيها ولكن انا استخدمت أسماء سهلة واضحة مراعياً الطالب الضعيف في اللغة الإنكليزية .

بالتوفيق لكم .

مثال:

سنستخدم الـ Function التي استخدمناها في مثالنا السابق والتي تقوم بجمع عددين وسنقوم بانشاءClass ينفذ العمليات الحسابية الأساسية

(الجمع،الطرح،الضرب،القسمة)

```
class num:
         def add(self,x,y):
               print(x+y)
         def sub(self,x,y):
               print(x-y)
         def div(self,x,y):
               print(x/y)
        def mul(self,x,y):
                print(x*y)
result=num()
result.add (0,2)
result.div (5,10)
result.mul (2,8)
result.sub (3,9)
```

التنفيذ :

2

5

4

27

شرح الكود السابق.

Class اسمهnum ويحتوي على4 Functions تمثل العمليات الاساسية.

الوسيط هنا اسمه Result وتم استدعاء الـ Functions واعطاء قيم

مثال :

```
class Car :
        def __init__(self,price,model):
                self.myPrice = price
                self.myModel = model

        def myCar (self):
                if self.myModel >= 2015:
                self.myPrice += 75
                print(f"new model {self.myPrice}")
                else:
                      self.myPrice -= 75
                       print(f"new model {self.myPrice}")

BMW = Car(150,2020)
ta = Car (100,2013)
BMW.myCar()
ta.myCar()
```

بالبداية في عندنا كلاس اسمه car ويحتوي عل تابعين تابع رئيسي وتابع اسمه myCar التابع الرئيسي نمرر له قيمتين هما سعر السيارة و موديل السيارة اما التابع الثاني يحوي على شرط إذا كان موديل السيارة فوق 2015 يزيد فوق سعر السيارة الأساسي 75 ألف

أما إذا كان موديل السيارة تحت 2015 يقوم بخصم 75 ألف عن سعرها الأساسي أيضاً لدينا object (BMW)and object(ta)

اعطينا لكل object قيمتين هما سعر السيارة الأساسي و موديل السيارة .

التنفيذ :

new model 225

new model 25

نلاحظ عندما كان سعر السيارة 150 وكان موديلها 2018 اصبح سعرها الجديد 225.

وعندما كان سعر السيارة 100 وكان موديلها 2013 اصبح سعرها الجديد 25 .

الوراثة Inheritance :

الوراثة في لغة بايثون كما في لغات البرمجة الأخرى هي أن يرث class بعض خصائص class اخر.

```
class Parent:
       def myMethod2(self):
               print (' parent ')
class Child(Parent):
        def myMethod(self):
              print (' child ')
c = Child()
c.myMethod2()
c.myMethod()
```

التنفيذ :

parent

child

نشرح الكود بالتفصيل .

في هذا المثال لدينا class اسمه Parent و class اخر اسمه child ال class Child يرث من ال class Parent

الوسيط هنا هو ال c وهو لل class Child

نلاحظ أن عن طريق ال class Child تم استدعاء ال function mymethod2 من ال class parent .

في ختام هذا الكتاب الذي تناول أساسيات لغة البرمجة بايثون، نأمل أن يكون قد قدّم لك الأسس الصلبة لتطوير مهاراتك في هذه اللغة الرائدة. بايثون ليست مجرد لغة برمجة، بل هي أداة قوية متعددة الاستخدامات تمكّنك من التعامل مع العديد من المجالات مثل تطوير الويب، تحليل البيانات، الذكاء الاصطناعي، والتعلم الآلي.

لقد مررنا معًا على مفاهيم البرمجة الأساسية، مثل المتغيرات، الحلقات، الشروط، والدوال، وصولًا إلى المكتبات المتخصصة التي تجعل بايثون مرنة وقادرة على التعامل مع تحديات معقدة. الآن، الكرة في ملعبك للاستمرار في استكشاف هذه اللغة وتطبيق ما تعلمته.

تذكّر أن البرمجة تتطلب الصبر والممارسة المستمرة. ومع كل مشروع جديد أو تحدٍ تحلّه، ستزداد قدرتك على استخدام بايثون بفاعلية وكفاءة.

نتمنى لك رحلة موفقة وممتعة في عالم البرمجة مع بايثون، ونتطلع لرؤية الإبداعات التي ستبنيها مستقبلاً.

بالتوفيق لكم