

PYTHON COURSE

Python course , from scratch
to Professionalism ...

Made by : Taha Khaldoun Amro 

Built in functions

تكلّمنا في الدرس السابق عن الـ `functions` , وذكرنا وجود نوعين من الـ `functions` و هما الـ `functions` التي يقوم المستخدم بكتابتها و تحديد الأوامر التي تقوم بها , و `functions` مدمجة في اللغة موجودة أصلاً لتسهل علينا بعض الأمور و بعض العمليات المهمة التي قد نحتاجها في عملنا بشكل عام .

سنتناول في هذا الدرس كل الـ `functions` المدمجة في لغة بايثون .
في الحقيقة , هذا ليس أول تعامل لنا مع الاقترانات المدمجة , فقد تعاملنا من قبل مع الاقترانات الخاصة بتحويل البيانات (Type conversion functions) التي تحول البيانات من نصوص إلى `list` أو إلى أرقام و هكذا.

1.all()

يستعمل الأمر `all` بالعادة في شروط الجمل الشرطية , حيث يعمل على أخذ قيمة عنصر `iterable` مثل الـ `lists` على سبيل المثال و يقوم بفحص ما إذا كانت كل عناصر الـ `list` تعطي القيمة `True` :

```
print(all([False , True , True]))
print(all([True , True , True]))
print(all([0 , 1 , 1]))
```

Built in functions

2. any()

يعمل هذا الأمر عكس عمل all لأنه يعيد قيمة true إذا كان أي عنصر داخل iterable يحمل القيمة True , عكس الall التي تعيد true فقط إذا كانت كل عناصر الlist تعيد القيمة True:

```
print(any([False , True , True]))  
print(any([True , True , True]))  
print(any([0 , 1 , 1]))
```

3. bin()

يعمل هذا الأمر على أخذ قيمة integer و يعيده بلغة الحاسوب بكل بساطة .

```
print(bin(12))
```

Built in functions

4. id()

يعمل هذا الأمر على قبول أي متغير , حيث يعيد لك كود موقعه التخزيني في حاسوبك .

```
x = 1323  
  
print(id(x))
```

5. sum()

يعمل هذا الأمر على أخذ قيمة iterable يحتوي على أرقام فقط , و يعيد لنا مجموع هذه الأرقام .

```
print(sum([1 , 12 , 3453 ,13, 24 ,63]))
```

Built in functions

6. round()

يقوم هذا الأمر بأخذ قيمتين , الأولى رقم عشري , و الثانية رقم يعبر عن عدد المنازل العشرية التي ستبقى بعد الفاصلة بعد القيام بعملية التقريب , حيث يعمل الاقتران بشكل أساسي على تقريب الرقم الى أقرب 10 , و ذلك فقط في المنازل العشرية :

```
print(round(1232.13 , 1))  
print(round(1232.56 , 1))  
print(round(1232.563 , 2))
```

Built in functions

7. range()

كنا قد تعاملنا سابقاً مع هذا الاقتران في ال for loops حيث يعمل على إنشاء list من أرقام تبدأ من موقع معين وتنتهي عند موقع آخر .

يطلب ال range منك ثلاث قيم (start , end , step) حيث بإمكانك إدخال قيمة واحدة فقط و هي ال end و عندها سيبدأ العد من الصفر , و عند إدخالك لقيمتين , ستكون القيمة الأولى هي البداية و القيمة الثانية هي النهاية , أما بالنسبة لل steps فهي تعبر عن عدد الخطوات التي يخطوها العداد , فهي في الأساس 1 , عندما نضع 2 يتم العد عبر القفز عن عدد على النمط التالي : 2 4 6 8 ...

```
print(list(range(0)))  
print(list(range(10)))  
print(list(range(5 , 20 , 2)))
```

Built in functions

8. print()

يمكن أن يكون هذا الأمر من أكثر الأمور التي مرت علينا في مسيرتنا هذه , لكن هناك ما يجب التكم عنه في خبايا هذا الأمر فمثلا يوجد مجموعة من ال arguments التي يقبلها هذا الاقتران , أهمها :

1.sep=

يعبر هذا ال argument عن الفاصل بين ما يتم طباعته في عملية ال string concatenation , حيث تكون القيمة التلقائية لهذا ال argument هي المسافة .

```
print('Hi' , 'Wassap !?' , sep=' | ')
```

2. end=

لنقل أنك تريد طباعة list من خلال for loop في سطر واحد , بحيث تفصل بين كل عنصر و الآخر بفاصلة أو مسافة بدل من سطر من كامل .

```
for number in range(10):  
    print(number , end="|")
```

Built in functions

9. abs()

يعمل هذا الأمر على إعادة القيمة المطلقة للعدد (absolute value) , جرب بنفسك .

10. pow()

يقبل هذا الأمر عددين , حيث يقوم برفع الأول لقوة الثاني :

```
print(pow(4,2)) # 4**2
```

11. min()

يعمل هذا الأمر على قبول iterable object يحتوي على أرقام و يعيد لك أصغر قيمة فيه

12. max()

يعمل هذا الامر بنفس مبدأ الذي قبله , ولكنه يرجع لنا أكبر قيمة :

```
mynums = [1 , 2 ,3 ,4,-12 ,4,-10 ,6 , 10 ,34 , -2]
```

```
print(min(mynums))  
print(max(mynums))
```


Built in functions

13. map()

يعتبر هذا الأمر واحداً من أهم الأوامر في لغة بايثون , و هذا لأنه يعمل على التالي :

- يأخذ function و يقوم بتنفيذه على عناصر iterable

يبدو الأمر معقداً في البداية , و لكن مع التدريب التالي سنعمل على فهمه بشكل مفصل .

سنقوم في البداية بتعريف اقتران يقبل منا قيمة نصية , و يعيدها لنا بعد إجراء مجموعة من التنسيقات عليها .

```
def stringformat(text) :
```

```
    return f"$ {text.upper()} $"
```

قم بإنشاء list تحتوي على مجموعة من الأسماء .

سنعمل الان على تنفيذ الfunction الذي كتبناه على كل عناصر الlist من خلال الأمر map .

Built in functions

كما ذكرنا سابقاً , فإن الfunction يأخذ اقتراناً و iterable , حيث يطبق الاقتران على عناصر الiterable :

```
names = ['moammed' , 'rama' , 'badr' , 'rami']
```

```
names = map(stringformat , names)
```

```
print(names)
```

لو جربت الكود السابق , ستتم طباعة رسالة تخبرك بالشكل التخزيني لهذا الأمر الذي تم على عناصر الlist , لكن لماذا ؟ ببساطة لأن النظام يتعامل مع هذا الfunction على أنه function و من أجل طباعة ما بداخله , يجب علينا أن نعمل for loop أو أن نحول نوعه من map object إلى list , سنقوم بالخيار الأول الان :

```
for name in names :
```

```
    print(name)
```

Built in functions

14. filter()

هذا الاقتران شبيه إلى حد ما بالاقتران السابق , حيث أنهما يكتبان بنفس الطريقة , و يقبلان نفس القيم , حيث يأخذ هذا الأمر function و iterable لإجراء الfunction على جميع عناصره , و لكن ما يقوم به الفلتر هنا مختلف . يقوم الفلتر ببساطة بإرجاع القيم التي يتحقق عليها شرط يوضع داخل الfunction , حيث إذا تم إعادة القيمة True , تتم إعادة القيمة التي تحقق عليها الشرط , و من خلال المثال التالي , سنوضح بعض الأمور المهمة :

سنعمل الان على نظام بسيط , يفحص في list عن الأعداد التي تكون أكبر من 10 و يعيدها لنا بكل بساطة .
في البداية سنعرف الfunction :

```
def checknum(number):  
    if number > 10 :  
  
        return True # the filter needs to have a boolean value .
```

Built in functions

لا بد من أنك لاحظت الفكرة التي تكلمنا عنها سابقاً , ألا و هي أن ال filter يعيد لنا القيم التي تعيد القيمة True عند تنفيذ الشرط عليها , و لهذا السبب لا نقوم بإرجاع البيانات نفسها , بل نعيد قيمة منطقية .
الان , قم بإنشاء list فيها مجموعة أرقام عشوائية , و قم بتنفيذ الأمر filter عليها :

```
nums = [1,5,2,33,5,2,7,444,13,64,2,0]

myfilterednums = filter(checknum , nums)

for num in myfilterednums :
    print(num)
```

تجربة للتنفيذ :
جرب كتابة كود يعمل على إرجاع الأسماء التي تحتوي على الحرف (t) ضمن لائحة أسماء عشوائية .

Built in functions

15. reduce()

على سبيل التغيير , لن نقوم بشرح هذا الاقتران لسببين , أولاً , قد يكون شرحه صعباً و معقداً بعض الشيء في حال تم شرحه بالكلام فقط , لذلك أنت بحاجة إلى شيء نظري و عملي أكثر , لذلك سنترك لك الرابط التالي :

https://www.youtube.com/watch?v=bgV0RHfRhB4&list=PLDoPjvoNmBAyE_gei5d18qkfle-Z8mocs&index=74&pp=iAQB

ثانياً , نريد رؤية النتائج التي ستقع على المبرمجين في حال غيروا طريقة التعلم , في الأساس لا يجب أن يتأثرو على الإطلاق , ولكن قد تحدث بعض المشاكل , و عند حدوثها يمكننا أن نعلم أن هناك شيئاً ما خاطئاً.

و لأهمية الأوامر الثلاثة الأخيرة، سيتم إرفاق رابط لشرح الموضوع بشكل مبسط و عملي:

<https://youtu.be/Az2ggGkFwEM?si=v9Ip2SMUV1mHCxYV>

Built in functions

16. enumerate()

أحد أطرف الاقترانات التي يمكن التعامل معها , حيث يعمل على ترقيم عناصر iterable يتم إعطاؤه من قبل المستخدم , و لكن الخدعة تكمن في أن نوع البيانات التي يعيدها لنا هذا الاقتران هي (enumerate) و تكون على شكل العنصر الذي تم ترقيمه و رقمه داخل اقواس <(1 , element) و هو ما قد يكون غير عملي , مما يرغبنا عند رغبتنا في طباعة عناصره على إنشاء for loop عادية , و لكن و بدلاً من المتغير الواحد , فإننا نضع متغيرين اثنين , يعبر الأول عن الرقم و الثاني عن العنصر , على النحو التالي :

```
mylist = ['one' , 'Two' , 'Three' , 'Four' , 'Five' , 'Six']

mylistnumbered = enumerate(mylist) # enumerate(iterable , start=0)

for number , element in mylistnumbered :
    print(f'{number} -> {element}')
```

الان جرب أن تقوم بوضع موضع بداية كما هو موضح في التعليق في الكود .

Built in functions

17. help()

يمكننا هذا الاقتران من التعرف على أي موضوع داخل اللغة , كل ما عليك هو إعطاء هذا الاقتران كلمة مفتاحية عن موضوع معين، و سيقوم هو بإعادة شرح كامل عنه.

18. reversed()

يقوم هذا الاقتران بعملية عكس لأي عنصر iterable , حيث يقوم بترتيبه من الآخر إلي الأول , و يحتاج لعملية for loop للوصول إلى البيانات المعكوسة , أو يمكننا ببساطة تحويل نوع البيانات إلى list مجدداً.

```
mylist = [199,323,34,678,3423,522,45,15,45]
```

```
mylist = reversed(mylist)
```

```
for num in mylist :  
    print(num)
```

Built in functions

مرفق لكم روابط شرح لما جاء في الدرس :

https://youtu.be/-PfCcZ2Q_MI?si=GD2Hp3zJ-w1ghpY4

https://youtu.be/2ed3aomFliA?si=85hS_nQRCZEZnUen

<https://youtu.be/XRw7mArOyok?si=ttO3tAq4jbp3Awb8>

<https://youtu.be/JvbLI0z8t8c?si=g9u-qDD1mG6ZjTWg>

https://youtu.be/0Zmdu7OgVl0?si=zrL_sz4ex5cN2TP4

https://youtu.be/bgV0RHfRhB4?si=OoA_UOI2Iy-DlUt4

<https://youtu.be/nS-uled9bil?si=IVv0VRL2WUjZLfGg>

PYTHON COURSE

Python course , from scratch
to professionalism ...

Made by : Taha Khaldoun Amro 