

PYTHON COURSE

Python course , from scratch
to Professionalism ...

Made by : Taha Khaldoun Amro 

Sets

سنتعرف الان إلى النوع الثالث من أنواع البيانات التي يمكن تخزين البيانات داخلها.

ال set , هي شكل من أشكال البيانات التي يمكن من خلالها تخزين البيانات , و لكنها تختلف عن باقي أخواتها تخزين البيانات من حيث:

1. عناصر ال set توضع داخل {}
 2. عناصر ال set غير مرتبة و لا يوجد لها أرقام مواقع
 3. لا يمكن الوصول إلى العناصر التي بداخلها من خلال ال indexing أو ال slicing.
 4. لا يمكن أن تحتوي ال set على عناصر يكون نوع بياناتها : lists or dictionaries
 5. العناصر داخل ال set لا يمكن أن تتكرر (مميزة و فريدة)
- و بالنسبة للنقطة الأخيرة , فهي التي تعتبر الفارق الأكبر بينها و بين أشكال البيانات الأخرى.

من أجل التعرف بشكل أكبر على نوع البيانات هذا , نفذ الكود التالي :

Sets

إنشاء ال set و طباعتها :

```
myset = {1 , 2, 3 , 'Hi' , False}

print(myset)

print(myset[0])

mysettwo = {5, 6 , 'Taha', [22 , True]}

print(mysettwo)
```

هل لاحظت شيئاً غريباً ؟ نعم , عند تنفيذنا للكود السابق يظهر لنا error , ما يحاول ال error إخبارنا به هو أن أمر الطباعة الثاني لا يمكن تنفيذه , لماذا ؟ لقد أجبنا على هذا السؤال في السابق , فقد قلنا من قبل أن ال set لا تخضع لل indexing حيث لا يمكن الوصول إلى العناصر داخلها , و لذلك امسح أمر الطباعة الثاني و نفذ الكود .

Sets

ظهر لك error اخر أليس كذلك ؟
ال error الان يحاول إخبارك بأن ال set الثانية تحتوي على عنصر يمكن تغييره (list) , و قد أشرنا سابقاً الى أن ال set لا يمكن أن تحتوي على عناصر قابلة للتغير كال list أو ال dictionary .
احذف ال list من ال set و نفذ الكود مجدداً .
الان , و كما تعلمنا سابقاً فإن ال set لا يمكن أن تحتوي على عناصر متكررة , و لكن , ماذا يحدث إذا أدخلنا مجموعة من العناصر المتكررة ؟ هل سنحصل على error ؟ نفذ الكود الاتي لتعرف الإجابة :

```
myset = {1, 2, 3, 'Hi', 'python', 3, 'Hi', 'hi'}
```

```
print(myset)
```

يمكننا الان أن نستنتج من الكود السابق أن ال set لا يمكن أن تحتل نفس القيمة أكثر من مرة , و هذا ما يميزها عن ال lists و ال tuples , و في حال إدخال قيمة مكررة , فإنها تقوم بحذف التكرار و الإبقاء على واحدة من القيم المكررة , و هو ما سنستخدمه كثيراً في مجالنا بإذن الله في المستقبل القريب.

Sets

سنتعرف الان على مجموعة من التقنيات التي تختص بال set .

1. clear()

تعمل هذه التقنية على إفراغ ال set من كل العناصر التي بداخلها .

2. union()

تتم عملية الدمج بين اثنتين من ال sets عن طريق وضع إشارة (|) بينهما .

```
myset1 = {1 , 2, 3 }  
myset2 = { 'Hi' , 'two'}  
print(myset1 | myset2) # try using the union method
```

Sets

3. add()

عندما نقول أن العناصر التي تقع داخل الـ set غير قابلة للتغيير ، فإن هذا لا يعني عدم قدرتنا على إضافة عناصر جديدة ، وهذا بالتحديد ما تقوم به هذه التقنية.

```
myset = {1, 2, 3}
myset.add('hi') # add something that is already in the set
print(myset)
```

4. copy()

لقد تكلمنا عن هذه التقنية في درس الـ Lists ، حيث أنها مشتركة بين الـ Lists و الـ Sets ، و تعمل التقنية على نسخ البيانات نسخاً سطحياً ، و في حال تم تغيير الـ Set الرئيسة ، فإن النسخة لن تتغير.

Sets

5. remove()

في الواقع , يمكننا ملاحظة أن الكثير من تقنيات ال Lists يتم استخدامها هنا , حيث استخدمنا هذه التقنية من قبل , و ذلك بهدف إزالة عنصر من القائمة , و كذلك الأمر هنا :

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

6. discard()

تعمل هذه التقنية على إزالة العناصر من ال set , تماماً مثل التقنية السابقة , و لكنها عند عدم إيجاد العنصر المراد إزالته , لا تقوم بطباعة error , بل تكتفي بالمضي قدماً في الكود , على عكس تقنية remove التي إذا لم تجد العنصر , تقوم بطباعة خطأ يفيد بأن العنصر غير موجود من الأساس .

استخدم الكود السابق , استبدل remove ب discard و جرب وضع قيمة غير موجودة في ال set .

Sets

7. pop()

كما تقوم هذه التقنية في ال Lists بإعادة العنصر الذي يتم إعطاء قيمته , تقوم بنفس العمل هنا , إلا انها و بما ان عناصر ال Sets غير مرقمة , تعيد عنصرا عشوائيا .

```
myset = {'Python' , 'css' , 'js' , 'Html'}  
  
print(myset.pop())
```

8. update()

تعمل هذه التقنية على تحديث ال set و ذلك بإضافة قيم جديدة من set أخرى أو حتى list أخرى .

```
myset = {'Python' , 'css' , 'js' , 'Html'}  
mylist = [1 , 2 , 3 ,4]  
  
myset.update(mylist)  
print(myset)
```


Sets

9. difference()

من الأشياء التي تميز ال set هي قدرتنا على الحصول على العناصر الموجودة فيها , و ليست موجودة في set أخرى , أو بالأحرى الحصول على العناصر المختلفة بينهما.

```
myset1 = {1, 2, 3, 4}
myset2 = {1, 2, 3.3, 4.4}
print(myset1.difference(myset2)) # things in set 1 that are not in set 2
print(myset1 - myset2) # things in set 1 that are not in set 2
```

10. difference_update()

الفرق بين هذه التقنية و التي تسبقها هي أن هذه التقنية تغير من قيمة ال set الى ناتج الفرق الذي حصلنا عليه .

جرب الكود السابق و لكن استبدل difference ب difference_update

Sets

11. intersection()

كما تمكنا في السابق من الوصول إلى الفرق بين الـ sets , فإننا بإمكاننا الوصول إلى نقاط التقاطع , أو العناصر المتشابهة بينهما.

```
myset1 = {1 , 2 , 3 , 4}
myset2 = {1,2 , 3.3 , 4.4}
print(myset1.intersection(myset2)) # the common thing between both sets
print(myset1 & myset2) # the common thing between both sets
```

12. intersection_update()

تماماً مثل التقنيات السابقة , فنحن بإمكاننا أيضاً تحديث قيمة الـ set إلى العناصر المشتركة بينها و بين الـ set الأخرى .

استخدم الكود السابق , ولا تنسى استبدال intersection بـ intersection_update .

Sets

13. issuperset()

من التقنيات الجميلة التي يمكننا من خلالها تحديد ما إذا كانت set محتواة داخل set أخرى , تعيد لنا قيمة منطقية .

```
a = {1 , 2, 3 , 4}
b = {1,2,3}
print(a.issuperset(b))
```

14. issubset()

تعمل هذه التقنية عكس السابقة تماماً , حيث تحدد ما إذا كانت set مشتقة من set أخرى أكبر.

```
a = {1 , 2, 3 , 4}
b = {1,2,3}
print(a.issubset(b))
```

Sets

14. isdisjoint()

تمكننا هذه الخاصية من معرفة ما إذا كانت اثنتين من الـ sets لا تحتويان على أي عناصر متشابهة , أي أن كل العناصر داخلهما مختلفة تماماً .

استخدم الأكواد السابقة و استبدل التقنيات , وحاول تغيير الـ sets .

مرفق لكم فيديوهات شرح لما جاء في الدرس :

<https://youtu.be/PSc6QX4Py7k?si=tu9e6x8w2KZQ4Mng>

https://youtu.be/N06_D5wWobg?si=txw7mG0u6W8Ar_--

<https://youtu.be/o8pr--y5vuU?si=HLftcxsryPxFcWKM>

https://youtu.be/rs9eebZpcaE?si=gghk_0lSzf1gahZW

PYTHON COURSE

Python course , from scratch
to professionalism ...

Made by : Taha Khaldoun Amro 