PYTHON COURSE

Python course , from scratch to profissionalism ...

Made by : Taha Khaldoun Amro

دورة تعليم لغة بايثون

Sets

سنتعرف الان الى النوع الثالث من أنواع البيانات التي يمكن تخزين البيانات الأخرى داخلها .
الset , هي شكل من أشكال البيانات التي يمكن من خلالها تخزين البيانات , و لكنها تختلف عن باقي كائنات تخزين البيانات من حيث :

1. عناصر الset توضع داخل {}
2. عناصر الset غير مرتبة و لا يوجد لها أرقام مواقع
3. لا يمكن الوصول الى العناصر التي بداخلها من خلال الindexing او الslicing
4. لا يمكن أن تحتوي الset على عناصر يكون نوع بياناتها : lists or dictionaries
5. العناصر داخل الset لا يمكن أن تتكرر (مميزة و فريدة)

و بالنسبة للنقطة الأخيرة , فهي التي تعتبر الفارق الأكبر بينها و بين اشكال البيانات الأخرى .

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

دورة تعليم لغة بايثون

Sets

انشاء الset 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 الerror اخبارنا به هو أن أمر الطباعة الثاني لا يمكن تنفيذه , لماذا ؟ لقد أجبنا على هذا السؤال في السابق , فقد قلنا من قبل أن الset لا تخضع للindexing حيث لا يمكن الوصول الى العناصر داخلها , امسح أمر الطباعة الثاني و نفذ الكود .

Sets

ظهر لك error اخر أليس كذلك ؟
الerror الان يحاول اخبارك بأن الset الثانية تحتوي على عنصر يمكن تغييره (list) , و قد أشرنا سابقاً الى أن الset لا يمكن أن تحتوي على عناصر قابلة للتغير كالlist او الdictionary .
احذف الlist من الset و نفذ الكود مجدداً .
الان , و كما تعلمنا سابقاً فان الset لا يمكن أن تحتوي على عناصر متكررة , و لكن , ماذا يحدث اذا أدخلنا مجموعة من العناصر المتكررة ؟ هل سنحصل على error ؟ نفذ الكود الاتي لتعرف الاجابة :

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

print(myset)
```

يمكننا الان أن نستنتج من الكود السابق أن الset لا يمكن أن تحتمل نفس القيمة أكثر من مرة , و هذا ما يميزها عن الlists و الtuples , و في حال ادخال قيمة مكررة , فانها تقوم بحذف التكرار و الابقاء على واحدة من القيم المكررة , و هو ما سنستخدمه كثيراً في مجالنا باذن الله .

دورة تعليم لغة بايثون

Sets

سنتعرف الان على مجموعة من التقنيات التي تختص بالset .

1. clear()

تعمل هذه التقنية على افراغ الset من كل العناصر التي بداخلها .

2. union()

تتم عملية الدمج بين اثنتين من الsets عن طريق وضع اشارة (|) 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 الset غير قابلة للتغيير , فان هذا لا يعني عدم قدرتنا على اضافة 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 الLists , حيث أنها مشتركة بين الLists و الSets , و تعمل التقنية على نسخ البيانات نسخاً سطحياً , و في حال تم تغيير الSet الرئيسة , فان النسخة لن تتغير .

Sets

5. remove()

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

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

6. discard()

تمعل هذه التقنية على ازالة العناصر من الset , تماما مثل التقنية السابقة , و لكنها عند عدم ايجاد العنصر المراد ازالته , لا تقوم بطباعة error , بل تكتفي بالمضي قدما في الكود , على عكس تقنية remove التي اذا 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 الset .

Sets

7. pop()

كما تقوم هذه التقنية في الLists باعادة العنصر الذي يتم اعطاء قيمته , تقوم بنفس العمل هنا , الا انها و بما ان عناصر الSets غير مرقمة , تعيد عنصرا عشوائيا .

```
myset = {'Python' , 'css' , 'js' , 'Html'}

print(myset.pop())
```

8. update()

تعمل هذه التقنية على تحديث الset و ذلك باضافة قيم جديدة من set اخرى او حتى list اخرى .

```
myset = {'Python' , 'css' , 'js' , 'Html'}
mylist = [1 , 2 , 3 ,4]

myset.update(mylist)
print(myset)
```

Sets

9. difference()

من الأشياء التي تميز الset هي قدرتنا على الحصول على العناصر الموجودة فيها , و ليست موجودة في set اخرى , 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 الset الى ناتج الفرق الذي حصلنا عليه .

جرب الكود السابق و لكن استبدل difference ب difference_update

Sets

11. intersection()

كما تمكننا في السابق من الوصول الى الفرق بين الsets , فاننا بامكاننا الوصول الى 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()

تماما مثل التقنيات السابقة , فنحن بامكاننا ايضا تحديث قيمة الset الى العناصر المشتركة بينها و بين الset الاخرى .

استخدم الكود السابق , ولا تنسى استبدال intersection ب intersection_update .

Sets

13. issuperset()

من التقنيات الجميلة التي يمكننا من خلالها تحديد ما اذا كانت set محتواة داخل set اخرى , 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 اذا كانت الsetمشتقة من set اخرى اكبر

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

Sets

14. isdisjoint()

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

استخدم الأكواد السابقة و استبدل التقنيات , وحاول تغيير الsets .

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