Data Structures

Lists القوائم

القائمة عبارة عن مجموعة أو سلسلة من العناصر المحفوظة في مكان واحد .
القوائم في البايثون لها المزايا التالية :

- يمكن حفظ اي نوع من البيانات .
- قابلة للتعديل ، يمكنك اضافة ، حذف ، تعديل على عناصر المجموعة .
- تحفظ العناصر بالترتيب .
- يمكنك حفظ أكثر من عنصر بنفس القيمة .

إنشاء قائمة جديدة

لإنشاء قائمة جديدة خالية :

```
my_empty_list = list()
```

لإنشاء قائمة جديدة مع بعض العناصر:

```
my_list = ["apple", "kiwi", "orange"]
```

اضافة عناصر للقائمة

لإضافة عنصر جديد لآخر القائمة:

```
my_list.append("cherry")
```

لإضافة عنصر جديد في مكان محدد في القائمة:

```
my_list.insert(1, "banana")
```

استرجاع قيمة من القائمة

لإسترجاع (الدخول على) قيمة معينة داخل القائمة ، نستخدم موقع العنصر داخل القائمة ، و التي تبدأ من الموقع 0 .
للدخول على العنصر الثالث داخل المجموع برقم الموقع :

```
print(my_list[2])
#outpu : kiwi
```

تتيح لنا البايثون الدخول على العناصر بإستخدام المواقع السلبية ، حيث عندما نبدأ بالسالب ، يتم استرجاع العنصر من نهاية القائمة .

للدخول على العنصر الأخير من القائمة :

```
print(my_list[-1])
#output: Cherry
```

اضافة قائمة لـ قائمة

لإضافة قائمة من العناصر إلى قائمة أخرى نستخدم الدالة extend

```
my_list2 = ["grapes", "plums"]
my_list.extend(my_list2)
```

للجمع بين قائمتين ، نستخدم الرمز +

```
new_list = my_list + my_list2
```

حذف عنصر من القائمة

لحذف عنصر من القائمة بإستخدام موقع العنصر

```
popped_item = my_list.pop(0)
```

لحذف عنصر من القائمة بإستخدام del

```
del my_list[1]
```

تنفيذ عملية (تكرار) على كل عنصر في القائمة

```
#looping through a list
for element in my_list:
    print(element)
```

لمعرفة حجم القائمة (عدد العناصر)

```
#finding the length of a list
list_length = len(my_list)
print(list_length)
```

لإختبار وجود عنصر في قائمة نستخدم الكلمة المفتاحية in ، بهذا سنسترجع قيمة منطقية إما بالإيجاب True او النفي False

```
#finding if a member of a list exists
if "grapes" in my_list:
    print("Grapes is in the list !")
else:
    print("Grapes is not in the list")
```

تجزيء القائمة slicing للحصول على قائمة جديدة مكونة من المدى المختار

```
new_list = my_list[1:3]
print(my_list[1:3])
print(my_list[:3])
print(my_list[1:])
```

Tuples

التوبلز مشابهة في مفهومها للقوائم ، بحيث أنها مكان لحفظ مجموعة من العناصر ، لكن الفرق الجوهري هو في أنها غير قابلة للتعديل . فبعد انشائها لن تتمكن من التعديل على العناصر داخل التوبل ، الاضافة لها أو حذفها احدها .
مزايا التوبل :

- غير قابلة للتعديل Immutable .
- يمكن حفظ اي نوع من البيانات داخلها.
- تحفظ العناصر بالترتيب.
- يمكنك حفظ عناصر بنفس القيمة.

إنشاء Tuple توبل جديدة

لإنشاء tuple جديد خالية :

```
new_tuple = ()
```

لإنشاء tuple جديدة مع بعض العناصر:

```
my_tuple = ("one_item", "second_item", 125)
```

```
#another way
my_tuple2 = "first item", 14, "third item"
```

استرجاع قيمة من التوبل Tuple

لإسترجاع (الدخول على) قيمة معينة داخل التوبل Tuple ، نستخدم موقع العنصر داخل التوبل ، و التي تبدأ من الموقع 0 .
للدخول على العنصر الأول داخل المجموعة برقم الموقع :

```
print(my_tuple[0])
#outpu : one_item
```

تتيح لنا البايثون الدخول على العناصر بإستخدام المواقع السلبية ، حيث عندما نبدأ بالسالب ، يتم استرجاع العنصر من نهاية القائمة .

للدخول على العنصر الأخير من التوبل :

```
print(my_tuple[-1])
#output: 125
```

جمع tuple لـ tuple

للجمع بين مجموعتي tuple ، نستخدم الرمز +

```
new_tuple = my_tuple + my_tuple2
```

تنفيذ عملية (تكرار) على كل عنصر في القائمة

```
#looping through a tuple
```

```
for element in my_tuple:
    print(element)
```

لمعرفة حجم القائمة (عدد العناصر)

```
#finding the length of a tuple
tuple_length = len(my_tuple)
print(tuple_length)
```

لإختبار وجود عنصر في قائمة نستخدم الكلمة المفتاحية in ، بهذا سنسترجع قيمة منطقية إما بالإيجاب True او النفي False

```
#finding if a member of a tuple exists
if 125 in my_tuple:
    print("125 is in the tuple !")
else:
    print("125 is not in the tuple")
```

تجزيء التوبل slicing للحصول على توبل جديد مكون من المدى المختار

```
new_tuple = my_tuple[1:3]
print(my_tuple[1:3])
print(my_tuple[:3])
print(my_tuple[1:])
```

تفريغ محتوى التوبل tuple unpacking

بالإمكان تفريغ محتوى (عناصر) مجموعة التوبل في متغيرات على شرط أن يكون عدد المتغيرات نفس عدد مجموعة التوبل

```
#tuple unpacking
```

```
var1, var2, var3 = my_tuple
```