Recursion , Lambda

تكرار الدالة Function Recursion

تكرار الدالة هي أن تستدعي الدالة نفسها تكرارا الى ان يتم تحقيق شرط أساسي معين . يستفاد منها في تقليل كمية الكود المكتوب ، و اضافة وضوح لعملية الدالة . و احيانا تضيف بعض السرعة .
لاحظ في حال عدم وجود شرط اساسي Base Condition ستستمر الدالة في استدعاء نفسها مرارا و تكرارا الى أن يتم ايقافها من قبل مترجم البايثون ، في حالة البايثون لك حد الى 1000 استدعاء ، بعدها يتم رفع خطأ من نوع RecursionError .

في المثال التالي ، المطلوب من الدالة هي جمع الأرقام المتتالية للرقم المطلوب

```
#a recursive function to sum the consecutive numbers
def sumNumbers(number):
    if number == 0:
        return number
    return number + sumNumbers(number-1)
print(sumNumbers(100))
#output 5050
```

الدوال المجهولة (Lambda)

الدوال المجهولة في البايثون هي عبارة عن عملية (وظيفة) من غير اسم . بينما نعرف الدالة المعتادة بإستخدام def ، الدوال المجهولة نعرفها بإستخدام الكلمة المفتاحية lambda .

الدوال المجهولة في البايثون يمكن تعيينها بأي عدد من المعطيات ، و لكن فقط مع عملية واحدة (ترجع لنا ناتج بالعادة). فيتم تنفيذ العملية و ارجاع النتيجة . و يمكن حفظها داخل متغير ، أو تمريرها كـ مُعطى لدالة أخرى و هذا بالعادة هو الاستخدام الأكثر لهذا النوع من الدوال .

كمثال سننشيء دالة مجهولة تستقبل مُعطيين من نوع الأرقام ، و ستقوم بعدها بضربهما في بعض .

```
#define a lambda function
multiply_numbers = lambda x , y : x*y
print(multiply_numbers(5, 3))
```

مثال آخر لتمرير الدوالة المجهولة كمعطى للبايثون ، سنستخدم هنا دالة الـ filter و التي تقوم بعملية لترشيح القائمة بناء على الشرط الذي نوفره بإستخدام دالة مجهولة لامبدا .

```
#using a lambda function inside filter function
my_list = [1, 4, 5, 6, 7]
filtered_list = list(filter(lambda element: element%2 == 0, my
_list ))
print("the filtered list is :")
print(filtered_list)
```