Lambda

❖ تعریف لامبدا (Lambda)

روشی ساده برای تعریف تابع (Function) در پایتون می باشد . این توابع معمولا به نام **توابع لامبدا** یا **عملگرهای لامبدا** نامیده می شوند .

❖ طریقه نوشتن تابع با استفاده از کلمه کلیدی def

پیش از بررسی توابع لامبدا (Lambda) یک تابع ساده با استفاده از کلمه کلیدی def می نویسیم .

```
def Func(Num1, Num2):
    return Num1*Num2

print(Func(5, 10))      # output: 50
```

اگر بخواهیم همین تابع را با عبارت لامبدا (Lambda) بنویسیم ، اینگونه است :

```
Func = lambda Num1, Num2: Num1*Num2

print(Func(5, 10))      # output: 50
```

نکته : از lambda به جای def استفاده شده است .

نکته : کلماتی که بعد از کلید واژهٔ lambda می آیند ، پارامترها هستند .

نکته : از علامت ":" برای جدا کردن عبارت و پارامترها استفاده می شود .

نکته : نیازی به کلید واژهٔ return نیست ، چرا که lambda خود این عملکرد را به صورت خودکار انجام می دهد .

❖ نگاشت (Map)

تابع نگاشت (Map) دو آرگومان ورودی می‌گیرد که یکی تابع (Function) و دیگری لیست (List) است . این تابع ، از تابع ورودی استفاده کرده و آن را روی لیست اجرا می‌کند و لیست اصلاح شده را به صورت یک شیء نگاشت (Map) باز می‌گرداند .

استفاده از نگاشت (Map) بدون لامبدا (Lambda) :

```
List = [2, 4, 6, 8]

print(List)          # output: [2, 4, 6, 8]

def Func(Num):
    return Num+10

New_List = list(map(Func, List))

print(New_List)      # output: [12, 14, 16, 18]
```

استفاده از نگاشت (Map) همراه لامبدا (Lambda) :

```
List = [2, 4, 6, 8]

print(List)          # output: [2, 4, 6, 8]

New_List = list(map(lambda Num: Num+10, List))

print(New_List)      # output: [12, 14, 16, 18]
```

❖ فیلتر (Filter)

Filter یک تابع را می‌گیرد و آن را بر روی همه عناصر یک فهرست اعمال می‌کند و لیست جدیدی ایجاد می‌کند .

استفاده از فیلتر (Filter) بدون لامبدا (Lambda) :

```
List = [2, 3, 4, 5, 6, 7, 8, 9]

print(List)          # output: [2, 3, 4, 5, 6, 7, 8, 9]

def Func(Num):
    if Num%2==0:
        return True
    else:
        return False

New_List = list(filter(Func, List))

print(New_List)      # output: [2, 4, 6, 8]
```

استفاده از فیلتر (Filter) همراه لامبدا (Lambda) :

```
List = [2, 3, 4, 5, 6, 7, 8, 9]

print(List)          # output: [2, 3, 4, 5, 6, 7, 8, 9]


New_List = list(filter(lambda Num: Num%2==0, List))

print(New_List)      # output: [2, 4, 6, 8]
```

❖ کاهش (Reduce)

Reduce یک محاسبه چرخشی بر روی همه عناصر یک لیست انجام می‌دهد . از این تابع برای محاسبه جمع یا ضرب کردن همه اعداد لیست با هم می‌توان استفاده کرد .

استفاده از کاهش (Reduce) بدون لامبدا (Lambda) :

```
from functools import reduce

List = [2, 3, 4, 5, 6, 7, 8, 9]

print(List)     # output: [2, 3, 4, 5, 6, 7, 8, 9]

def Func(x, y):
    return x+y

result = reduce(Func, List)

print(result)       # output: 44
```

استفاده از کاهش (Reduce) همراه لامبدا (Lambda) :

```
from functools import reduce

List = [2, 3, 4, 5, 6, 7, 8, 9]

print(List)     # output: [2, 3, 4, 5, 6, 7, 8, 9]


result = reduce(lambda x, y: x+y, List)

print(result)       # output: 44
```