19. Lambda, filter, reduce and map

Lambda

Toán tử lambda hoặc hàm lambda là một cách để tạo các hàm ẩn danh nhỏ, nghĩa là các hàm không có tên. Chúng chỉ cần thiết ở nơi chúng được tạo ra. Lambda chủ yếu được sử dụng kết hợp với hàm filter(), map() và reduce(). Tính năng lambda đã được thêm vào Python do yêu cầu từ các Lisp programers. Cú pháp chung của hàm lambda khá đơn giản: “Lambda argument_list: biểu thức”

Danh sách đối số bao gồm một danh sách đối số tách biệt bằng dấu phẩy và biểu thức là một biểu thức số học sử dụng các đối số này. Bạn có thể gán hàm cho một biến để đặt tên cho nó. Trong phần bài học này tôi sẽ dùng CMD để thực hiện các đoạn mã lệnh thay vì dùng IDE.

Ví dụ 1:

```
>>> f = lambda x,y: x*y

>>> f(1,2)

2
```

Tôi sẽ tiếp tục sử dụng tính năng này trong các phần tiếp theo.

Map

Ưu điểm của toán tử lambda có thể được nhìn thấy khi nó được sử dụng kết hợp với hàm map().
map() là một hàm với hai đối số, func và seq: r = map(func, seq)

Đối số đầu tiên, func, là tên của hàm,trong khi đối số còn lại là dữ liệu kiểu tuần tự, e.g., list. Mỗi phần tử trong seq sẽ là đầu vào của hàm func. Hàm map() sẽ trả về dữ liệu kiểu list với các phần tử bị thay đổi bởi hàm func.

Ví dụ 2 ta không sử dụng “lambda”

```
>>> def convertUSDtoVND(m):

...     return m*22000

...

>>> def convertVNDtoUSD(m):

...     return float(m)/22000

...

>>> vnd = [20e6,50e6,70e7]

>>> usd = map(convertVNDtoUSD, vnd)

>>> usd

[909.0909090909091, 2272.7272727272725, 31818.18181818182]

>>> map(convertUSDtoVND, usd)

[20000000.0, 49999999.99999999, 700000000.0]
```

```
>>>
```

Ví dụ 3: Khi ta sử dụng lambda ta không cần định nghĩa tường minh hai hàm convertUSDtoVND() và convertVNDtoUSD(). Bạn có thể quan sát ví dụ dưới đây.

```
>>> vnd = [20e6,50e6,70e7]

>>> usd = map(lambda x: float(x)/22000, vnd)

>>> usd

[909.0909090909091, 2272.7272727272725, 31818.18181818182]

>>> map(lambda x: x * 22000, usd)

[20000000.0, 49999999.99999999, 700000000.0]

>>>
```

Filtering

Hàm filter(func,seq) cung cấp một giải pháp tuyệt vời để lọc các phần tử có trong tập dữ liệu kiểu liệt kê.

Hàm filter(func,seq) có hai đối số. Đối số đầu tiên là một hàm trả về một kiểu Boolean, trong khi đối số thứ hai là dữ liệu kiểu liệt kê. Dữ liệu trả về của filter sẽ là dữ liệu trong seq nhưng phần tử đó phải thỏa mãn điều kiện trong hàm func.

Ví dụ 4:

```
>>> salaries = (123000,143000,125000,231000,100000)

>>> filter(lambda x: True if x > 120000 else False, salaries)

(123000, 143000, 125000, 231000)

>>>
```

Reduce

Hàm reduce(func, seq) có cơ chế hoạt động như cách giải thích sau:

Nếu seq = [s1,s2,s3,…,sn]

Đầu tiên hai phần tử đầu tiên của seq là (s1,s2) sẽ là đầu vào của hàm func, e.g., func(s1,s2). List trong reduce() sẽ hoạt động như sau [func(s1,s2),s3, …, sn].

Bước tiếp theo,dữ liệu (func(s1,s2),s3) là đầu vào của func vậy list mới trong reduce sẽ có dạng [func(func(s1,s2),s3),s4 … sn]

Quá trình này sẽ tiếp tục diễn ra như vậy cho đến khi còn lại một phần tử trong list.

Ví dụ 5: Tính tích các phần tử trong một danh sách

```
>>> nums = (1,2,3,4,5,6,7,8,9,10)

>>> reduce(lambda x,y: x*y, nums)
```

```
3628800
```

Ví dụ 6: Tìm min của dãy số

```
>>> nums = (1,2,3,4,5,67,8,9,10)

>>> reduce(lambda x,y: x if x < y else y, nums)

1
```

Kết luận

Như vậy chúng tôi vừa giới thiệu đến các bạn một số hàm hữu ích được định nghĩa sẵn trong Python là lambda, map(), filter() và reduce(). Bằng cách sử dụng lambda hoặc kết hợp việc sử dụng lambda với các hàm map, filter và reduce giúp cho việc gọi hoặc triển khai hàm khá ngắn gọn, súc tích và tiết kiệm được nhiều thời gian trong việc lập trình. Trong bài tiếp theo chúng tôi sẽ đề cập đến một phương thức khác (List comprehension) có thể thay thế cho lambda, map(), filter() và reduce().