

LẬP TRÌNH PYTHON

HÀM LAMBDA

NGUYỄN HẢI TRIỀU¹

¹Bộ môn Kỹ thuật phần mềm,
Khoa Công nghệ thông tin, Trường ĐH Nha Trang

NhaTrang, February 2022

Cấu trúc điều khiển

- 1 Ôn tập hàm
- 2 Lambda function

1 Ôn tập hàm

2 Lambda function

Đối số bất kì, *args

Nhắc lại *args

Nếu không biết có bao nhiêu đối số sẽ được truyền vào hàm thì thêm * vào trước tên tham số trong định nghĩa hàm. Hàm sẽ nhận được một tuple các đối số truyền vào.

Ví dụ 1.1

```
1 def my_function(*khoa):  
2     print("Lop chung ta đang hoc khoa: " + khoa[0])  
3 my_function("CNTT", "XHNV", "KT")  
4 #->Lop chung ta đang hoc khoa: CNTT
```

Keyword Arguments

Ngoài ra chúng ta có thể gửi các đối số cho hàm dạng keyword arguments: **key=value**. Lưu ý, thứ tự của các đối số không quan trọng.

Ví dụ 1.2

```
1 def my_function(x,y,z):  
2     print("Lop chung ta đang hoc khoa: " + x)  
3 my_function(x="CNTT", y="XHNV", z="KT")  
4 #->Lop chung ta đang hoc khoa: CNTT
```

Vấn đề gì sẽ xảy ra khi **key** trong đối số truyền vào hàm không trùng với tham số trong định nghĩa hàm?

Các đối số dạng Keyword bất kì, ** kwargs

Trong trường hợp không biết có bao nhiêu đối số dạng keyword arguments được truyền vào hàm, ta thêm 2 dấu ** vào trước tên tham số bên trong định nghĩa hàm.

Ví dụ 1.3

```
1 def my_function(**fname):
2     print("ten va ho cua toi la: "+fname['ten']+' '+fname['ho'])
3 my_function(ten = "Trieu", ho = "Nguyen")
4 #->ten va ho cua toi la: Trieu Nguyen
```

Sử dụng tham số mặc định

Python cho phép chúng ta sử dụng giá trị mặc định cho tham số. Nếu gọi hàm mà không có đối số truyền vào, thì nó sẽ sử dụng giá trị mặc định.

```
1 def my_function(country = "Vietnam"):
2     print("I am from " + country)
3 my_function()
4 my_function("Khanh Hoa")
5 #I am from Vietnam
6 #I am from Khanh Hoa
```

1 Ôn tập hàm

2 Lambda function

Hàm ẩn danh Lambda

Trong Python, hàm ẩn danh Lambda là một hàm nhỏ được định nghĩa mà không có tên. Một hàm lambda có thể nhận bất kỳ số lượng đối số nào, nhưng chỉ có một biểu thức biểu diễn.

Cú pháp 2.1

***lambda** arguments : expression*

- hàm ẩn danh được định nghĩa bằng cách sử dụng từ khóa **lambda**
- biểu thức sẽ được thực thi và trả về
- hàm lambda có thể được sử dụng ở bất cứ nơi nào đối tượng hàm được yêu cầu

Ví dụ 2.1

Cộng 1 vào đối số a và trả về kết quả

```
1 x = lambda a : a + 1
2 print(x(5))#->6
```

Nhân đối số a với đối số b và trả về kết quả:

```
1 x = lambda a, b : a * b
2 print(x(2, 10))#2->0
```

*Các hàm lambda ở trên trả về **một đối tượng hàm** được gán định danh là x (tương tự như `def x(a):`)*

Ưu điểm của hàm lambda là sử dụng nó như một hàm ẩn danh bên trong một hàm bình thường.

Ví dụ 2.2

Trường hợp hàm bình thường chỉ nhận một đối số và đối số đó sẽ nhân với một số chưa biết:

```
1 def myfunc(n):  
2     return lambda a : a * n  
3 mydoubler = myfunc(2)  
4 print(mydoubler(11))#->22
```

*Dựa vào hàm **myfunc**, ta sẽ nhận được giá trị gấp đôi đối số truyền vào thông qua đối tượng hàm **mydoubler**.*

Các hàm *lambda* được sử dụng khi cần một hàm ẩn danh trong khoảng thời gian ngắn (thường dùng làm đối số cho hàm khác). Hàm Lambda được sử dụng phổ biến cùng với các hàm Python tích hợp sẵn như *filter()*, *map()*,...

Ví dụ 2.3

Hàm *map()* nhận các tham số là một **hàm** và một **list**.

```
1 ds = [1, 2, 3, 4, 5]
2 dsmoi = list(map(lambda a: a*2, ds))
3 print(dsmoi)
```

Ví dụ 2.4

Sử dụng hàm **lambda** để chuyển tất cả các phần tử có kiểu **string** sang **int** cho một list các số nguyên?

Các hàm *lambda* được sử dụng khi cần một hàm ẩn danh trong khoảng thời gian ngắn (thường dùng làm đối số cho hàm khác). Hàm Lambda được sử dụng phổ biến cùng với các hàm Python tích hợp sẵn như *filter()*, *map()*,...

Ví dụ 2.3

Hàm *map()* nhận các tham số là một **hàm** và một **list**.

```
1 ds = [1, 2, 3, 4, 5]
2 dsmoi = list(map(lambda a: a*2, ds))
3 print(dsmoi)
```

Ví dụ 2.4

Sử dụng hàm *lambda* để chuyển tất cả các phần tử có kiểu **string** sang **int** cho một list các số nguyên?

```
1 ds=['1','2','3']
2 ds=list(map(lambda i:int(i),ds))
3 print(ds)
```

Ví dụ 2.5

Hàm ***filter()*** nhận các tham số là một ***hàm*** và một ***list***. Hàm được gọi với tất cả các phần tử trong *list* (cho kết quả ***True*** or ***False***) và *list* mới sẽ được trả về.

```
1 ds=[1,2,3]
2 nl=list(filter(lambda i: i%2==0, ds))
3 print(nl)
```

Bài tập

Ví dụ 2.6

Sử dụng hàm lambda, viết hàm cho ra kết quả gấp 15 lần đối số truyền vào, viết hàm nhân 2 đối số x , y và cho ra kết quả.

Ví dụ 2.7

Sử dụng hàm lambda để sắp xếp một danh sách các tuple cho trước.

Bài tập

Ví dụ 2.6





Sử dụng hàm *lambda*, viết hàm cho ra kết quả gấp 15 lần đối số truyền vào, viết hàm nhân 2 đối số x , y và cho ra kết quả.

Ví dụ 2.7

Sử dụng hàm *lambda* để sắp xếp một danh sách các tuple cho trước.

```
1 #sort
2 subject_marks = [('English', 88), ('Science', 90), ('Maths', 97),
3                  ('Social sciences', 82)]
4 print("Original list of tuples:")
5 print(subject_marks)
6 subject_marks.sort(key = lambda x: x[1])
7 print("\nSorting the List of Tuples:")
8 print(subject_marks)
```


Tài liệu tham khảo

-  V.H. Quân, C.X. Nam, H.T. Hiếu, N.H. Triều, V.C. Tài
Tự học lập trình Python căn bản. *NXB DHQG Tp.HCM, 2019.*
-  Mark Lutz
Learning Python (5th Edition). *O'Reilly Media, Inc, 2013.*
-  Luciano Ramalho
Fluent Python (2nd Edition). *O'Reilly Media, Inc, 2021.*
-  Python Software Foundation
<https://docs.python.org/3/tutorial/>