Python講座

第三回高階関数とラムダ式

takumi kaimai

2017 年 7 月 6 日

目次

1　高階関数

　高階関数とは，他の関数を引数として受け取る関数のことである．
pythonには予め用意された高階関数として，map，filter，reduceなどがある．
これらの関数はループ構文の省略として使用することができる．
出力結果が変わらないfor文を使用した場合のソースと，高階関数を使用した場合のソースを比較していく．

2　map

```
1    map(function,iterable)
2    #引数functionには関数
3    #引数iterableにはリスト
4    #返り値はiterableの要素を一つ一つ引数にとったfunctionの返り値のリスト
```

以下は，引数で取得した値に 1 を加算する関数plusOneをリストinput_listの値全てに使用して，
その結果をリストoutput_listに代入するスクリプトである．

2.1　for文

```
1  # coding: utf-8
2  def plusOne(x):
3      return x+1
4
5  input_list = [1,2,3]
6  output_list = []
7
8  for i in input_list:
9      output_list.append(plusOne(i))
10
11 for i in output_list:
12     print(i)
```

Listing 1: lec3-1.py

output

```
2
3
4
```

2.2 map

```
1  # coding: utf-8
2
3  def plusOne(x):
4      return x+1
5
6  input_list = [1,2,3]
7  output_list = map(plusOne,input_list)
8
9  for i in output_list:
10     print(i)
```

Listing 2: lec3-2.py

このようにfor文を書かずにリストに対しての処理を行うことができる．
変数の宣言も減らすことができ，バグを少なくすることができる．
for文のみでプログラムを記述すると可読性も下がるのでなるべく高階関数を利用することをオススメする．
さらにラムダ式を使用することで，高階関数用の関数を定義する必要がなくすことができる．

3　ラムダ式

ラムダ式を使用することで，名前の無い関数(匿名関数)を定義することができる．
基本的には，高階関数の引数に使用される．

3.1　ラムダ式による関数の定義

```
lambda 引数: 処理
```

```
#関数定義
def plusOne(x):
  return x+1

#ラムダ式
plusOne = lambda x:x+1
```

Listing 3: 関数定義との比較

3.2　ラムダ式の具体例

以下のようにラムダ式を使用することで，関数定義ができる．

```
# coding: utf-8

plusOne = lambda x:x+1
result = plusOne(2)
print(result)
```

Listing 4: lec3-3.py

output

```
3
```

多変数関数も定義できる．

```python
# coding: utf-8

plus = lambda x,y:x+y
result = plus(1,2)
print(result)
```

Listing 5: lec3-4.py

output

```
3
```

次に，高階関数のソースをラムダ式に変更してみる．

```python
# coding: utf-8
input_list = [1,2,3]
output_list = map(lambda x:x+1,input_list)

for i in output_list:
    print(i)
```

Listing 6: lec3-5.py

output

```
2
3
4
```

高階関数の引数にラムダ式を使用することで，高階関数に関する処理を一つの行にまとめることができる．

4 filter

高階関数filterは，入力したリストの中から関数で指定した条件を満たす値だけ抽出することができる．

```python
filter(function,iterable)
#引数functionには関数（返り値はTrue or Falseの真偽値型のみ）
#引数iterableにはリスト
#返り値はiterableの要素の中でfunctionの返り値がTrueになったもののリスト
```

4.1 for文

以下のスクリプトは入力したリストから偶数のみを抽出するものになる.

```
# coding: utf-8

input_list = [1,2,3,4,5,6]
output_list = []

for var in input_list:
    if var % 2 == 0:
        output_list.append(var)
    else:
        continue

for i in output_list:
    print(i)
```

Listing 7: lec3-6.py

output

```
2
4
6
```

4.2 filter

```
# coding: utf-8

input_list = [1,2,3,4,5,6]
output_list = filter(lambda x:x % 2 == 0,input_list)

for i in output_list:
    print(i)
```

Listing 8: lec3-7.py

output

```
2
4
6
```

filterはmap以上に記述を省略することが可能である.
また，条件式をラムダ式で表すのが困難な場合もあり，その時は関数で定義した方がいい.

5 reduce

具体例として，リストの値を順番に掛け合わせていくスクリプトで解説していく.

5.1 for文

```
1  # coding: utf-8
2  input_list = [1,2,3,4,5,6]
3  result = 1
4
5  for var in input_list:
6      result = result * var
7
8  print(result)
```

Listing 9: lec3-8.py

output

```
720
```

5.2 reduce

```
1  # coding: utf-8
2
3  #reduceを使用するためのおまじない
4  from functools import reduce
5
6  input_list = [1,2,3,4,5,6]
7  result = reduce(lambda x,y:x*y,input_list)
8  print(result)
```

Listing 10: lec3-9.py

output

```
720
```

このようにリストの要素に対して順番に指定の演算を行うことができる.

6 高階関数使用方法

6.1 Stringをintに変換したい時

```
1  # coding: utf-8
2
3  input_data = "1,3,4,5,6,7,8"
4  #空白で区切り, Stringのリストを作成
5  tmp_list = input_data.split(",")
6
7  #int型に変換する関数intをtmp_listに適用する
8  result = map(int,tmp_list)
9  for var in result:
10     print("value:"+str(var))
11     print(type(var))
```

Listing 11: lec3-10.py

output

```
value:1
<type 'int'>
value:3
<type 'int'>
value:4
<type 'int'>
value:5
<type 'int'>
value:6
<type 'int'>
value:7
<type 'int'>
value:8
<type 'int'>
```

6.2 複数のintをStringに変換する

```
1  # coding: utf-8
2  from functools import reduce
3
4  input_data = [1,3,4,5,6,7,8]
5
6  tmp_list = map(str,input_data)
7
8  result = reduce(lambda x,y:x+","+y,tmp_list)
9  print(result)
10 print(type(result))
```

Listing 12: lec3-11.py

output

```
1,3,4,5,6,7,8
<type 'str'>
```

7　おまけ

7.1　zip

リストとリストの結合を行うことができる

```
1  # coding: utf-8
2  list1 = [1,2,3]
3  list2 = [4,5,6]
4  list = zip(list1,list2)
5  for var in list:
6      print(var)
```

Listing 13: lec3-12.py

output

```
(1, 4)
(2, 5)
(3, 6)
```

7.2　enumerate

リストに数字を振ってくれる関数

```
1  # coding: utf-8
2  list = ["bigroom","dubstep","dnb"]
3
4  for i,var in enumerate(list):
5      output = str(i)+":"+var
6      print(output)
```

Listing 14: lec3-13.py

output

```
0:bigroom
1:dubstep
2:dnb
```