Python BootCamp 3

February 10, 2019

1 Python Session 3 :

```
            Shashank Shawak

MAP
FILTER
REDUCE
LAMBDA
generators
iterators
decorators

lambda argumnets:expression
```

```
In [13]: x=1
         square=lambda x:x*x
```

```
In [14]: square(4)
```

```
Out[14]: 16
```

```
In [15]: x=[1,2,3,4,5]
```

```
In [20]: def even(x):
             for values in x:
                 if values%2!=0:
                     x.remove(values)
             print x
```

```
In [21]:
```

```
[2, 4]
```

```
In [22]: import numpy as np
```

```
In [70]: x=list(range(11, 17))
         even(x)
```

```
[12, 14, 16]
```

```
In [75]: y=np.array(list(range(11, 17)));
         y[y%2==0]

Out[75]: array([12, 14, 16])

In [107]: x=list(range(11, 17))

          print list(map(lambda y:y%2==0,y))

[False, True, False, True, False, True]


In [108]: map(lambda y:y*y,y)

Out[108]: [121, 144, 169, 196, 225, 256]
```

2 Filter

```
In [112]: import statistics

In [113]: data=[1.3,1.9,1.5,1.8,3.6,3.8,2.4,2.5,3.1,1.9]

In [114]: avg =statistics.mean(data)

In [115]: avg

Out[115]: 2.38

In [116]: filter(lambda x :x>avg,data)

Out[116]: [3.6, 3.8, 2.4, 2.5, 3.1]

In [129]: data=["",1,2,3,4,5]

In [130]: filter(None,data)

Out[130]: [1, 2, 3, 4, 5]
```

3 Reduce

```
In [150]: def f(x):
              return x*x
          out=f(f(f(f(f(2)))))
          out

Out[150]: 4294967296
```

```
In [151]: data=list(range(11,20))

In [152]: data

Out[152]: [11, 12, 13, 14, 15, 16, 17, 18, 19]

In [153]: mulitplier=lambda x,y:x*y

In [154]: product=reduce(mulitplier,data)

In [155]: product

Out[155]: 33522128640

In [157]: product=1
          for values in data:
              product=product*values

In [158]: product

Out[158]: 33522128640
```

4 Generator

```
In [ ]: def fib(mymax):
            a,b=0,1
            while True:
                c=a+b
                if c<mymax:
                    yield c
                    a=b
                    b=c
                else:
                    break

In [ ]: val=fib(15)

In [ ]: next(val)

In [ ]: mylist=[1,2,3,4,5,6,7,8,9]

In [ ]: val=iter(mylist)

In [ ]: next(val)

In [159]: mylist=list(range(11))

In [ ]: def list_reader(mylist):
            i=0
            if i in range(len(mylist)):
                yield(mylist[i])
                i+=1

In [ ]: gen=list_reader(mylist)

In [ ]: next(gen)
```

5 Decorators

```
In [160]: def func():
              return 1

In [161]: func()

Out[161]: 1

In [165]: s = 'Global Variable'

          def check_for_locals():
              n=5
              print(locals())

In [166]: check_for_locals()

{'n': 5}


In [168]: globals()['s']

Out[168]: 'Global Variable'

In [169]: def hello(name='shashank'):
              return 'Hello '+name

In [181]: greeting=hello(name=raw_input('enter your name please   :   '))
          greeting

Out[181]: <function __main__.hello>

In [183]: greeting=hello
          greeting()

Out[183]: 'Hello shashank'

In [184]: del hello

In [185]: hello()
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-185-a75d7781aaeb> in <module>()
    ----> 1 hello()


        NameError: name 'hello' is not defined
```

```
In [187]: greeting()

Out[187]: 'Hello shashank'

In [188]: def hello(name='anything'):

              def greet():
                  return '\t This is inside the greet() function'

              def welcome():
                  return "\t This is inside the welcome() function"

              if name == 'anything':
                  return greet
              else:
                  return welcome

In [189]: x = hello()

In [190]: x

Out[190]: <function __main__.greet>

In [191]: print(x())

         This is inside the greet() function


In [196]: x=hello(name='sam')

In [199]: print x()

         This is inside the welcome() function
```

6 Functions as Arguments

6.0.1 Now let's see how we can pass functions as arguments into other functions:

```
In [205]: def hello():
              return 'Hi Jose!'



          def other(func):
              print('Other code would go here')
              print(func())

In [206]: other(hello)

Other code would go here
Hi Jose!
```

6.0.2 creating Decorator

```
In [224]: def new_decorator(function_to_be_run):

              def wrap_func(*args):

                  print "I have been executed inside the decorator before function_to_be_run ex

                  function_to_be_run(*args)


                  print "I have been executed inside the decorator after function_to_be_run exe
              return wrap_func

          def func_needs_decorator(*args):
              print("This function is in need of a Decorator")
              print ("done")
```

```
In [225]: func_needs_decorator = new_decorator(func_needs_decorator)
```

```
In [226]: func_needs_decorator(5)
```

```
I have been executed inside the decorator before function_to_be_run execution
This function is in need of a Decorator
done
I have been executed inside the decorator after function_to_be_run execution
```

```
In [227]: @new_decorator
          def func_needs_decorator(x):
              print("This function is in need of a Decorator")
              print x*x
```

```
In [228]: func_needs_decorator(5)
```

```
I have been executed inside the decorator before function_to_be_run execution
This function is in need of a Decorator
25
I have been executed inside the decorator after function_to_be_run execution
```