Python_BootCamp_3

February 10, 2019

1 Python Session 3 :

Shashank Shawak

MAP
FILTER
REDUCE
LAMBDA
generators
iterators
decorators

lambda argumnets:expression

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

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

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

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

```
In [ ]:
```

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

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

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

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

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

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

2 Filter

```
In [ ]: import statistics

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

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

In [ ]: avg

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

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

In [ ]: filter(None,data)
```

3 Reduce

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

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

In [ ]: data

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

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

In [ ]: product

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

In [ ]: product
```

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 [ ]: 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 [ ]: def func():
            return 1

In [ ]: func()

In [ ]: s = 'Global Variable'

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

In [ ]: check_for_locals()

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

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

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

In [ ]: greeting=hello
        greeting()

In [ ]: del hello
```

```
In [ ]: hello()

In [ ]: greeting()

In [ ]: 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 [ ]: x = hello()

In [ ]: x

In [ ]: print(x())

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

In [ ]: print x()
```

6 Functions as Arguments

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

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



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

In [ ]: other(hello)
```

6.0.2 creating Decorator

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

            def wrap_func(*args):

                print "I have been executed inside the decorator before function_to_be_run exe
```

```
                function_to_be_run(*args)


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

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

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

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

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

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