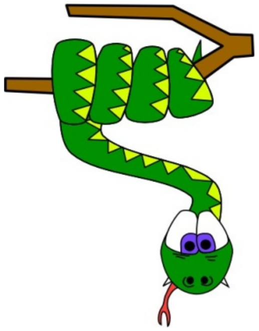

Higher Order Functions in Python
Part-I
KNC-402

What are HOFs

- *A function is called **Higher Order Function** if it contains other functions as a parameter or returns a function as an output i.e, the functions that operate with another function are known as **Higher order Functions**.*

- *Functions and methods are **first-class objects** in Python, so if we want to pass a function as an argument to another function, we can just treat it as any other object.*

Properties of HOFs

Properties of higher-order functions:

- A function is an instance of the Object type.
- You can store the function in a variable.
- You can pass the function as a parameter to another function.
- You can return the function from a function.
- You can store them in data structures such as hash tables, lists,

HOFs: Function as an object

In Python, a function can be assigned to a variable. This assignment does not call the function, instead a reference to that function is created.

```
# Example1
 def square(x):
     return x*x
 print(square)
 print(square(3))
 f=square
 print(f)
 print(f(3))
```

Output:

```
<function square at 0x08818468>
9
<function square at 0x08818468>
9
```

HOFs: Passing Function as an argument to other function

Functions are like objects in Python, therefore, they can be passed as argument to other functions.

```
#Example 2 sum of square till n natural numbers
def sum(n, square):
    total=0
    for num in range(1,n+1):
        total = total + square(num)
    return total

def square(x):
    return x*x
print(sum(3,square))
```

Output:14

Note: Here square function is passed as a parameter to the sum function, hence sum function is a HOF.

HOFs: Passing Function as an argument to other function

```
# Example 3
def sum(n,square):
   total=0
   for num in range(1,n+1):
      total = total + square(num)
   return total

def square(num):
   return num*num

def cube(num):
   return num*num*num

print("Sum of Squares:",sum(3,square),"Sum of Cubes:",sum(3,cube))
```

Output:Sum of Squares: 14 Sum of Cubes: 36

HOFs: Returning function

As functions are objects, we can also return a function from another function.

```
#Example 4
def make_adder(x):
    def adder(y):
        return x + y
    return adder

addition = make_adder(15)
print(addition(10))
```

Output:25

HOFs: Returning function

```
# Example 5
from random import choice
Def make_greeting_func (person):
    def get_greet():
        ch=choice(("Good Morning ","Good Evening ", "Good    Night "))
        return ch+person
    return get_greet
greet=make_greeting_func("Anshuman")
print(greet())
print(greet())
print(greet())
```

OutPut:

```
Good Morning Anshuman
Good    Night Anshuman
Good    Night Anshuman
```

Lambda Expression

- ❖ A lambda function is a small anonymous function.
- ❖ A lambda function can take any number of arguments, but can only have one expression.

Syntax

```
lambda arguments : expression
```

Example

```
square = lambda x : x * x
print(square(5))
```

```
Output:25
```

New More Example

Example 2:

```
g =lambda x: x*x*x
print(g(7))
```

Output:343

Example 3:

```
x = lambda a, b : a * b
print(x(5, 6))
```

Output:30

Why use Lambda Functions?

- ❖ The power of lambda is better shown when you use them as an anonymous function inside another function.

```
def myfun1(n):
    return lambda a:a*n
def myfun2(n):
    return lambda a:a*n
mydoubler=myfun1(2)
mytripler=myfun2(3)
print(mydoubler(11))
print(mytripler(11))
```

Output:

```
22
33
```