Practice Question

In [ ]:
```
How many Times Hello will be Printed ?

for i in range(1,5):
    for j in range(1,5):
        if j==i:
            break
    print("Hello")

a. 0 Time
b. 4 Times  #Correct
c. 16 Times
d. 1 Time
```

Normal Functions/User Defined Functions

In [ ]:
```
Normal Functions/User Defined Functions
--> Prepared by Developers
--> Based on Business Logic
```

Example of User Defined Functions

In [1]:
```
def square(n):
    return n**2
print(square(5))
```

```
25
```

In [2]:
```
def add(x,y):
    return x+y
print(add(10,5))
```

```
15
```

In [3]:
```
def maximum(a,b):
    if a>b:
        return "a is greater"
    else:
        return "b is greater"
print(maximum(10,20))
```

```
b is greater
```

Anonymous Function

In [ ]:
```
Anonymous Function --> An anonymous function is a function that is defined without a name.
                       While normal functions are defined using the def keyword in Python,
                       anonymous functions are defined using the lambda keyword. Hence,
                       anonymous functions are also called lambda functions.

                   --> The main purpose of anonymous functions is just to instant use(one time usage).

                   --> In Python we have one anonymous function only and that function is known as
                       Lambda.
```

Lambda Function

In [ ]:
```
We can define lambda function in python with the help of lambda Keyword

Syntax of lambda Function:

    lambda parameter : expression

Basically lambda function is used to make your code concise and it incrases readibility.
```

Example of lambda Function

Python Program to find the square of a Number with and Without Lambda Function

In [4]:
```
#Normal Function
def square(n):
    return n**2
print(square(5))
```

```
25
```

In [5]:
```
#Lambda Function
x = lambda n : n**2
x=int(input("Enter a number"))
print("Square of  a number",x(n))
```

```
Enter a number5
Square of  a number 25
```

Note --> You cannot directly used lambda function with loops but indirectly you can use loops with lambda functions.

In [6]:
```
#Lambda function with for loop
list = [4,2,3,4,5,6]
l1=[]
for i in list:
    temp=lambda i : i**2
    l1.append(temp(i))
print(l1)
```

```
[16, 4, 9, 16, 25, 25]
```

Python Program to find the addition of two Numbers with and Without Lambda Function

In [7]:
```
#Normal Function
def add(x,y):
    return x+y
print(add(10,5))
```

```
15
```

In [8]:
```
#Lambda Function
y = lambda x,y:x+y
print(y(10,20))
```

```
30
```

Python Program to find the maximum of two Numbers with and Without Lambda Function

In [9]:
```
#Normal Function
def maximum(a,b):
    if a>b:
        return "a is greater"
    else:
        return "b is greater"
print(maximum(10,20))
```

```
b is greater
```

In [ ]:
```
#Lambda Function
s=lambda a,b :"a is greater" if a>b  else "b is greater "
print(s(29,10))
```

Python Program to Check weather a Given Number is Even or Odd with lambda Function

In [11]:
```
s=lambda a :  "a is even" if a%2==0 else "a is odd "
print(s(9))
```

```
a is even
```

Python Program to find the maximum of three Numbers with and Without Lambda Function

In [12]:
```
s=lambda a,b ,c:  a if a>b and a>c else b if b>c else c
print(s(20,15,60))
```

```
60
```

High order Functions

In [ ]:
```
High order Functions --> are those functions that will take a another function as an input
                         and a sequence(list,set,tuple,range)
                     --> In case of high order functions we need to pass two things first one
                         is a function and second is a sequence
                     --> if you want to apply a certain logic or functionality to each and every
                         element of a sequence then we should go for high order functions.
```

Filter Function

In [ ]:
```
Filter --> The filter() method filters the given sequence with the help of a function that tests
           each element in the sequence to be true or not. Wherever filter function encountered
           True that will be taken under filter function rest all will be ignored.


       --> In case of filter function Inside the function whatever conditon is given
           it will check that condition for every element and based on that condtion whatever
           elements are true that will be return.
```

Filter Function without Lambda Function

In [ ]:
```
#without lambda Functions
def is_even(x):
    if x%2==0:
        return True

l1=[10,20,14,20,25,19,27]
l1=list(filter(is_even,l))
print(l1)
```

Filter Function with Lambda Function

In [8]:
```
#with the help of lambda Function
l1=list(filter(lambda x:x%2==0 ,l))
l1
```

Out[8]:
```
[10, 20, 14, 20]
```

Filter the Prices of a product from a CART using filter function

In [11]:
```
cart = [100,200,500,700,800,900,1000,1200,1400,1600]
l1=list(filter(lambda x:x>500,cart))
l1
```

Out[11]:
```
[100, 200]
```

Filter the Number from 1 to 500 which are divisible by 7

In [12]:
```
l1=list(filter(lambda x:x%7==0  ,range(1,500)))
print(l1)
```

```
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, 105, 112, 119, 126, 133, 140, 147, 154, 161, 168, 175, 182, 189, 196, 203, 210, 217, 224, 231, 238, 245, 252, 259, 266, 273, 280, 287, 294, 301, 308, 315, 322, 329, 336, 343, 350, 357, 364, 371, 378, 385, 392, 399, 406, 413, 420, 427, 434, 441, 448, 455, 462, 469, 476, 483, 490, 497]
```

map Function

In [ ]:
```
map --> Map in Python is a function that works as an iterator to
        return a result after applying a function to every item of an iterable (tuple, lists, etc.).
        It is used when you want to apply a single transformation function to all the iterable
        elements. The iterable and function are passed as arguments to the map in Python.

    --> if you want to perform any functionality or logic for each and every element
        of a list then we should use map function.

    --> if we are using filter function --> not same length/same length

    --> if we are using map function  --> length always be same to original list

    --> In map whatever function you are passing that function functionality will be applied
        for each and every element of the given sequence

Syntax of map function:

    map (function_name , sequence)
```

Map Function Without Lambda Function

In [13]:
```
#without lambda
def double(n):
    return n*2
x = [10,20,30,40,50]
l1=list(map(double,x))
l1
```

Out[13]:
```
[20, 40, 60, 80, 100]
```

Map Function With Lambda Function

In [14]:
```
#with lambda
l1=list(map(lambda n:n*2,x))
l1
```

Out[14]:
```
[20, 40, 60, 80, 100]
```

Example of Map Function

In [15]:
```
l=[10,20,14,20,25,19,27]
l1=list(map(lambda x:x%2==0 ,l))
l1
```

Out[15]:
```
[True, True, True, True, False, False, False]
```

In [16]:
```
x=[1,2,3]
y=[3,4,5]

l=list(map(lambda a,b:a+b,x,y))
l


#Note --> We can also use map function for Two lists or sequence
```

Out[16]:
```
[4, 6, 8]
```

How to take input of a List Using Map Function

In [18]:
```
#This function is also used to take input as a list
x=int(input("Enter the number"))
a=list(map(int, input().split()))
print(a)
```

```
Enter the number1
2
[2]
```

Reduce Function

In [ ]:
```
reduce --> reduce all the elements of a list into a single unit.based on certain condtion
       --> reduce function is presenet in functools module
       --> A single answer or element you will get in this case.
```

Example of Reduce Function

In [19]:
```
#With Reduce Function
from functools import reduce
l=[10,20,30,40,50,60]
result=reduce(lambda x,y:x*y,l)
print(result)
```

```
720000000
```

In [20]:
```
#Without Reduce Function
prod=1
for i in l:
    prod=prod*i
print(prod)
```

```
720000000
```

Functions are also an object in Python

In [ ]:
```
In Python Function everything is treated as an object
Even functions are also treated as an object in python internally
```

In [10]:
```
def f1():
    print("Hello")
print(f1)
print(id(f1))
```

```
<function f1 at 0x000002DE1DA090D0>
2878125743408
```

Function Aliasing

In [ ]:
```
Function aliasing --> means to give another name to the existing function.
                  --> After giving another name if we are deleting one name still we can
                      access that function with another name.
```

In [45]:
```
def wish(name):
    print("Good Evening :",name)
greetingwish
print(wish)

wish("Pratyush")
del wish
greeting("Ashu")

if you are deleting one name (reference variable) then still we can
access that function with another name.
```

```
<function wish at 0x000002088A9D0160>
Good Evening : Pratyush
Good Evening : Ashu
```

In [ ]:
```

```