In [2]:
```
#User Defined Function / Normal Functions
def square(n):
    return n**2
print(square(5))
def add(a,b):
    return a+b
def maximum(a,b):
    if a>b:
        print("a is greater")
    else:
        print("b is greater")
#In normal functions we are passing a variable like integer , float , string...etc
```

```
25
```

In [ ]:
```
Anonymous functions
Some times we can declare a function without any name such type of functions are callled
Anonymous Function
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.
```

In [ ]:
```
Lambda function--> we can define lambda function with the help of lambda keyword.
```

In [ ]:
```
Syntax of lambda functions:
lambda n : n**2
lambda parameter = expression
Lambdafunctions  --> code is concise readility will be increases
```

In [ ]:
```
examples of lambda function
```

In [3]:
```
s=lambda n : n**2
print("Sqaure of 5 is ",s(5))
```

```
Sqaure of 5 is  25
```

In [4]:
```
s=lambda a,b:a+b
print(s(10,20))
```

```
30
```

In [6]:
```
s=lambda a,b: "a IS GREATER" if a>b else "B is greater"
print(s(10,20))
```

```
B is greater
```

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
```

In [ ]:
```
Filter function --> to filter values based on some condition
[10,20,30,40,50]
function  --> if x[i]<30 return x[i]
#Note: 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 ar e
true that will be return.
```

In [8]:
```
#Example of filter functions
#Without lambda function
def iseven(x):
    if x%2!=0:
        return True
    else:
        return False
l=[5,10,15,20,30,50]
l1=list(filter(iseven ,l))
print(l1)
```

```
[5, 15]
```

In [9]:
```
#With lambda expression
l=[5,10,15,20,30,50]
l1=list(filter(lambda x:x%2==0 ,l))
print(l1)
```

```
[10, 20, 30, 50]
```

In [11]:
```
#With lambda expression  (whose cost is less than 500)
cart=[500,700,800,200,100,200,40,600,900]
l1=list(filter(lambda x:x<500 ,cart))
print(l1)
```

```
[200, 100, 200, 40]
```

In [12]:
```
#range(1,500) --> that are divisilbe by 7
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, 2
31, 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, 43
4, 441, 448, 455, 462, 469, 476, 483, 490, 497]
```

In [ ]:
```
map function-->Functions and Sequence
if we are using filter function --> not same/same
if we are using map function  --> length always be same to orginial list
Note: In map whatever function you are passing that function functionality will be applied
    for each and every element of the given sequence
Synatx of map function  -->map (function_name , sequence)
```

In [39]:
```
#Example of map function
l=[5,10,15,20,30,50]
l1=list(map(lambda x:x%2==0 ,l))
print(l1)
#False
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [39], in <cell line: 3>()
      1 #Example of map function
      2 l=[5,10,15,20,30,50]
----> 3 l1=list(filter(map(lambda x:x%2==0 ,l),l))
      4 print(l1)

TypeError: 'map' object is not callable
```

In [14]:
```
#Without lambda
def double(x):
    return 2*x
l1=[10,20,30,40,50]
l=list(map(double,l1))
print(l)
```

```
[20, 40, 60, 80, 100]
```

In [15]:
```
#With lambda
l1=[10,20,30,40,50]
l=list(map(lambda x:x*2,l1))
print(l)
```

```
[20, 40, 60, 80, 100]
```

In [25]:
```
x=[10,20,30,40,50,60,70]
l=list(map(lambda x:x**2,x))
sum(l)
```

Out[25]:
```
14000
```

In [19]:
```
x=[1,2,3,4]
y=[2,3,4,5]
l=list(map(lambda x,y:x*y,x,y))
l
```

Out[19]:
```
[2, 6, 12, 20]
```

In [27]:
```
n=int(input("Enter number of elements "))
a=list(map(int ,input().split()))
print(a)
```

```
Enter number of elements 5
10  20 30 40 50
[10, 20, 30, 40, 50]
```

In [30]:
```
def f1():
    print("hello")
print(f1)
print(type(f1))
```

```
<function f1 at 0x0000028896C87D30>
<class 'function'>
```

In [ ]:
```
reduce --> reduces the elements into single element on the basis of specified functionality
reduce function is present in functools module
```

In [32]:
```
from functools import reduce
l=[10,20,30,40,50,60]
result=reduce(lambda x,y:x+y,l)
print(result)
```

```
210
```

In [37]:
```
#Function Aliasing
def wish(name):
    print("GOOd Evening ",name)

greeting=wish
print(id(wish))
print(id(greeting))
greeting("Priyanka")
wish("Ashish")
del wish
greeting("Ashu")
```

```
2785669000496
2785669000496
GOOd Evening  Priyanka
GOOd Evening  Ashish
GOOd Evening  Ashu
```

In [ ]:
```
```