(6.2) Special Function

```
Anonymous Function/Lambda Function/Inline Function
filter()
map()
reduce()
```

1. Anonymous / Lambda / Inline

```
Sometimes we can declare a function without any name,such type of nameless
functions are called anonymous functions or lambda functions.

The main purpose of anonymous function is just for instant use(i.e for one time
usage)

we can define bu using lambda keyword
```

ref = lambda arg1, arg2, ... : return_statement

Note:Lambda Function internally returns expression value and we are not required to write return statement explicitly.

In [5]:
```
hi = lambda : print('Hello world!')

a = hi()
print(a)
```

```
Hello world!
None
```

In [6]:
```
print(hi)
```

```
<function <lambda> at 0x000001B4A0CE90D0>
```

A program to create a lambda function to find sum of square of two number

In [7]:
```
sq = lambda a, b: a**2 + b**2

r = sq(4, 5)
print(r)

print(sq(4, 5))
```

```
41
41
```

program to find greatest of three number using lambda function

In [8]:
```
g = lambda a,b,c: a if a>b and a>c else b if b>c else c

g(1, 20 , 5)
```

Out[8]:
```
20
```

lambda with default parameter

```
In [7]: Discount_Price = lambda p, d=0.2 : p - (p*d) #bydefault 20% discount

        r = Discount_Price(100,0.5) # 50 % discount
        print(r)
```

```
50.0
```

lambda with recursion

```
In [9]: p = lambda n,c=2:True if c > n//2 else False \
        if n % c == 0 else p(n,c+1)
```

```
In [12]: print(p(15))
         print(p(127))
```

```
False
True
```

Note: Sometimes we can pass function as argument to another function. In such cases lambda functions are best choice.

We can use lambda functions very commonly with filter(),map() and reduce() functions,b'z these functions expect function as argument.

2. filter()

```
We can use filter() function to filter values from the given sequence based on some
condition.
```

filter(function,sequence)

```
where function argument is responsible to perform conditional check
sequence can be list or tuple or string.
```

Q. Program to filter only even numbers from the list by using filter() function?

```
In [14]: # without lambda

         def isEven(x):
             if x%2==0:
                 return True
             else:
                 return False

         l = [0,5,10,12,20,25,30]
         l1=list(filter(isEven,l))
         print(l1)
```

```
[0, 10, 12, 20, 30]
```

```
In [17]: # with lambda

         l = [0,5,10,15,20,25,30]
         print(list(filter(lambda x:x%2==0, l)))
```

```
[0, 10, 20, 30]
```

3. map()

```
For every element present in the given sequence,apply some functionality and
generate new element with the required modification. For this requirement we should
go for map() function.

used to map a function on a sequence
```

map(function, iterable)

program to make square of each element of list

In [19]:
```
lst = [ 1, 2, 3, 4]

ans = list(map(lambda x:x**2, lst))
print(ans)
```

```
[1, 4, 9, 16]
```

We can apply map() function on multiple lists also.But make sure all list should have same length.

In [20]:
```
l1 = [1,2,3,4]
l2 = [2,3,4,5]
l3 = list(map(lambda x,y:x*y, l1,l2))
print(l3)
```

```
[2, 6, 12, 20]
```

4. reduce()

```
reduce() function reduces sequence of elements into a single element by applying
the specified function.
```

reduce(function,sequence)

```
reduce() function present in functools module and hence we should write import
statement
```

In [22]:
```
from functools import*
l = [10,20,30,40,50]
result=reduce(lambda x,y:x+y, l)
print(result)
```

```
150
```

In [23]:
```
result=reduce(lambda x,y:x*y,l)
print(result) #12000000
```

```
12000000
```

sum of 1 to 100

In [24]:
```
from functools import *
result=reduce(lambda x,y:x+y,range(1,101))
print(result) #5050
```

```
5050
```

```
In [ ]:
```