Functional Programming:

- It purly mathematical function style.

lambda:

- By using lambda keyword we can develop the function.
- It is anynomous function in single line.
- syntax:
  variable_name = lambda argumnets:output_expression
  - how to call the lambda
    variable_name(arguments)

In [1]:
```
# To print addition of two numbers using lambda?
add = lambda x,y:x+y
add(100,345)
```

Out[1]: 445

In [2]:
```
def addition(a,b):
    return a+b
addition(100,45)
```

Out[2]: 145

In [6]:
```
# To print the odd numbers using lambda?
odd = lambda n:n%2!=0
L2 = [i for i in range(1,15) if odd(i)]
L2
```

Out[6]: [1, 3, 5, 7, 9, 11, 13]

iterator:

- To fetch the one value at a time
- iter, next
- syntax:
  variable_name = iter(iterable)
  next(variable_name)

In [7]:
```
L1 = [10,20,30,40]
a = iter(L1)
next(a)
```

Out[7]: 10

In [8]:
```
next(a)
```

Out[8]: 20

In [9]:
```
next(a)
```

Out[9]: 30

In [10]:
```
next(a)
```

Out[10]: 40

In [11]:
```
next(a)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-11-15841f3f11d4> in <module>
----> 1 next(a)

StopIteration:
```

generator:

- To generate the sequence of values from the function using 'yield' keyword in formatt of list.

In [13]:
```
# To generate the 1 to 10 natural numbers using generator?
def natural(n):
    i = 1
    while(i<=n):
        yield i
        i = i+1
list(natural(10))
```

Out[13]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

In [15]:
```
def natural(n):
    i = 1
    while(i<=n):
        return i
        i = i+1
natural(10)
```

Out[15]: 1

filter:

- syntax: filter(function_name,sequence)

In [17]:
```
L5 = [10,20,-3,-56,12,-7]
# To filter only the negative numbers?
def negative(n):
    return n<0
f1 = list(filter(negative,L5))
f1
```

Out[17]: [-3, -56, -7]

In [18]:
```
f2 = list(filter(lambda x:x<0,L5))
f2
```

Out[18]: [-3, -56, -7]

map:

- syntax:
  list(map(function_name,sequnce)

In [20]:
```
f2
```

Out[20]: [-3, -56, -7]

In [21]:
```
m2 = list(map(lambda n:n+100,f2))
m2
```

Out[21]: [97, 44, 93]

In [22]:
```
'''int a,b;
scanf("%d%d",&a,&b);'''
a = int(input("Enter a value"))
b = int(input("Enter b value"))
print(a,b)
```

```
Enter a value10
Enter b value30
10 30
```

In [23]:
```
a,b,c=map(int,input().split())
print(a,b,c)
```

```
10 20 30
10 20 30
```

reduce:

- It return the single value from the function.
- syntax:
  reduce(function_name,sequnce)
- we can import the functools
- from functools import reduce

In [24]:
```
L = [1,2,3,4]
# TO print the sum..
from functools import reduce
r1 = reduce(lambda n1,n2:n1+n2,L)# n1=1,n2=2:1+2=>3
                                 # n1=3,n2=3:3+3=>6
                                 #n1=6,n2=4:6+4=>10
r1
```

Out[24]:
```
10
```

In [25]:
```
L1 = [1,2,3,4,5]
# To print the list product..
r2 = reduce(lambda a,b:a*b,L1)
r2
```

Out[25]:
```
120
```

In [ ]: