FUNCTIONS

- Anonymous Function(Lambda)
- map()
- filter()
- reduce()

In [3]:

```
x=20
y=10
print("Addition--->",x+y)
print("Substraction--->",x-y)
print("Multiplication--->",x*y)
print("***********************************")
x=200
y=100
print("Addition--->",x+y)
print("Substraction--->",x-y)
print("Multiplication--->",x*y)
print("***********************************")
x=2000
y=1000
print("Addition--->",x+y)
print("Substraction--->",x-y)
print("Multiplication--->",x*y)
print("===============================")
```

```
Addition---> 30
Substraction---> 10
Multiplication---> 200
***********************************
Addition---> 300
Substraction---> 100
Multiplication---> 20000
***********************************
Addition---> 3000
Substraction---> 1000
Multiplication---> 2000000
===============================
```

In [38]:

```
def arthimetic(x,y):
    print("Addition--->",x+y)
    print("Substraction--->",x-y)
    print("Multiplication--->",x*y)
arthimetic(20,10)
print("--------------------")
arthimetic(200,100)
print("--------------------")
arthimetic(2000,1000)
print("--------------------")
```

```
Addition---> 30
Substraction---> 10
Multiplication---> 200
--------------------
Addition---> 300
Substraction---> 100
Multiplication---> 20000
--------------------
Addition---> 3000
Substraction---> 1000
Multiplication---> 2000000
--------------------
```

In [39]:

```
z=200 # global value
def f1():
    z=20 # local variable
    print("z=",z)
def f2():
    print(z)
f1()
f2()
```

```
z= 20
200
```

In [43]:

```
a=100
def f1():
    global a
    a=500
    print(a)
def f2():
    print(a)
f1()
f2()
```

```
500
500
```

In [41]:

```
def f1():
    p=500
    print(p)
def f2():
    print(p)
f1()
f2()
```

```
500

---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-41-cd456228f401> in <module>
      5     print(p)
      6 f1()
----> 7 f2()

<ipython-input-41-cd456228f401> in f2()
      3     print(p)
      4 def f2():
----> 5     print(p)
      6 f1()
      7 f2()

NameError: name 'p' is not defined
```

Anonymous Functions(lambda)

- No Identity
- Nameless function
- Instant use(one time usage)

In [44]:

```
def squreit(n):
    return n*n
squreit(5)
```

Out[44]:

```
25
```

In [36]:

```
# syntax
# lambda inputarguments:expression
```

In [45]:

```
ob=lambda n:n*n
print(ob(6))
```

```
36
```

In [47]:

```
s=lambda x,y:x*y
print(s(10,20))
```

```
30
```

In [48]:

```
m=lambda x,y,z:x*y*z
print(m(10,20,2))
```

```
400
```

In [49]:

```
# program to print range of values
s=lambda n:n*n
for i in range(1,11):
    print("The Squre of {} is {}".format(i,s(i)))
```

```
The Squre of 1 is 1
The Squre of 2 is 4
The Squre of 3 is 9
The Squre of 4 is 16
The Squre of 5 is 25
The Squre of 6 is 36
The Squre of 7 is 49
The Squre of 8 is 64
The Squre of 9 is 81
The Squre of 10 is 100
```

In [51]:

```
b=lambda x,y:x if x>y else y
print(b(20000,100))
```

```
20000
```

map() function

- for every element present in the given sequence apply some function and generate new element with the required modification

In [37]:

```
# Syntax
# map(function,sequence) sequences------>list,tuple,dict,set
# lst=[1,2,3,4,5]------->[1,2,9,16,25]
```

In [54]:

```
lst=[1,2,3,4,5,6,7,8,9,10]
def squre(n):
    return n*n
lst1=list(map(squre,lst))
print(lst1)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [57]:

```
# map performs operations on all elements
lst=[1,2,3,4,5,6]
lst1=tuple(map(lambda n:n**n,lst))
print(lst1)
```

```
(1, 4, 27, 256, 3125, 46656)
```

In [62]:

```
# program to multiply two list of elements
l1=[1,2,3,4,5,6,7,8,9,10]
l2=[6,7,8,9,10,11,12]
l3=list(map(lambda x,y:x*y,l1,l2))
print(l3)
```

```
[6, 14, 24, 36, 50, 66, 84]
```

In [63]:

```
# program to add text to existing list of strings
l1=['gitam','cse','JOhn','it','Ravi']
print(tuple(map(lambda x:x+'Hyderabad',l1)))
```

```
('gitamHyderabad', 'cseHyderabad', 'JOhnHyderabad', 'itHyderabad', 'RaviHy
derabad')
```

In [ ]:

```
# program to find the length of each word in a given line of text
```

In [68]:

```
s='welcome to python programming coding culture in gitam'
#[7,2,6,11,6,7,2,5]======>output
words=s.split()
print(words)
print(list(map(lambda w:len(w),words)))
```

```
['welcome', 'to', 'python', 'programming', 'coding', 'culture', 'in', 'git
am']
[7, 2, 6, 11, 6, 7, 2, 5]
```

In [69]:

```
# program to add of three lists elements
l1=[1,2,3,4,5]
l2=[10,20,30,40,50]
l3=[100,200,300,400,500]
l4=list(map(lambda x,y,z:x+y+z,l1,l2,l3))
print(l4)
```

```
[111, 222, 333, 444, 555]
```

filter()

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

Syntax

filter(function,sequence)

In [9]:

```
l1=[10,3,5,6,4,2,12,18,20,1]
def even(x):
    if x%2==0:
        return True
    return False
print(list(filter(even,l1)))
```

```
[10, 6, 4, 2, 12, 18, 20]
```

In [11]:

```
lst=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
print(list(filter(lambda x:x%2==0,lst)))
```

```
[2, 4, 6, 8, 10, 12, 14]
```

In [ ]:

```
# program to print duplicate string in a given list
# input===>['cse','gitam','coding','cse','culture','cse','cse']
```

In [13]:

```
l=['cse','gitam','coding','cse','culture','cse','cse']
print(tuple(filter(lambda w:w=='cse',l)))
```

```
('cse', 'cse', 'cse', 'cse')
```

reduce()

functools module

In [14]:

```
l1=[1,2,3,4,5,6,7,8,9,10]
prod=1
for i in l1:
    prod=prod*i
print(prod)
```

```
3628800
```

In [16]:

```
from functools import reduce
print(reduce(lambda a,b:a*b,l1))
```

```
3628800
```

In [17]:

```
# write a python program to perform addition of 1-100 elements
sum=0
for i in range(1,101):
    sum+=i
print(sum)
```

```
5050
```

In [19]:

```
from functools import reduce
# for i in range(1,101):
print(reduce(lambda a,b:a+b,range(1,101)))
```

```
5050
```

In [20]:

```
# How reverese a string 'GITAM======MATIG'
s="GITAM"
res=s[::-1]
print(res)
```

```
MATIG
```

In [21]:

```
rev=''
ln=len(s)-1
while ln>=0:
    rev=rev+s[ln]
    ln-=1
print(rev)
```

```
MATIG
```

In [24]:

```
rev=reversed(s)
print(''.join(rev))
```

```
MATIG
```

In [28]:

```
# write a program to reverse the given line of text
s1="python programming using data structures"
l=s1.split()
print(l)
l1=l[::-1]
print(" ".join(l1))
```

```
['python', 'programming', 'using', 'data', 'structures']
structures data using programming python
```

In [ ]:

```

```