In [ ]:
```
Two Important functions related to File handling..
1.tell()
2.seek()
```

In [ ]:
```
Explaination of tell Function()
tell functions --> we can use tell() to return the current position of the cursor(file pointer)
from the begining of the file.
The Starting pointer will always 0
```

In [3]:
```
#Example to Demostrate Tell Function
f=open("abc.txt","r")
print(f.tell())  #
print(f.read(4))
print(f.tell())
print(f.read(8))
print(f.tell())
f.close()
```

```
0
Prat
4
yush
Moh
13
```

In [ ]:
```
Explaination of seek Function()
seek function is used to move our cursor(file pointer)
```

In [13]:
```
#Example to Demonstrate seek Functions()
f=open("abc.txt","r+")
text = f.read()
#print(text)
print("Current pointer is",f.tell())
f.seek(97)
print("Current pointer is",f.tell())
f.read()
```

```
Current pointer is 123
Current pointer is 97
```

Out[13]:
```
'File handling is very easy'
```

In [15]:
```
#Nested Functions --> Functions inside functions
def add(x,y):
    def sub(x,y):
        return x-y
    return x+y
print(sub(10,20))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [15], in <cell line: 6>()
      4         return x-y
      5     return x+y
----> 6 print(sub(10,20))

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

In [30]:
```
def outer():
    print("Outer function executed")   #1

    def inner():
        print("Inner Function executed")    #3

    print("Outer function calling inner function")   #2
    return inner

f1=outer()
```

```
Outer function executed
Outer function calling inner function
```

In [29]:
```
def outer():
    print("Outer function executed")   #1

    def inner():
        print("Inner Function executed")    #3

    print("Outer function calling inner function")   #2
    inner()

f1=outer()
```

```
Outer function executed
Outer function calling inner function
Inner Function executed
```

In [31]:
```
def add(a,b):
    def sub(c,d):
        return c-d
    return sub(a,b)
    return a
res=add(5,10)
print(res)
```

```
-5
```

In [34]:
```
def  div(a,b,c):
    def mul(c,d,e):
        return c*d*e
    return mul(a,b,c)
    return a
res=div(5,10,9)
print(res)
```

```
450
```

In [37]:
```
#Variable length Arguments
def add(*args):
    print(type(args))
    print(args)
print(add(10,20,30,40))
#If we have single star then we can only pass positional argument
```

```
<class 'tuple'>
(10, 20, 30, 40)
None
```

In [42]:
```
#Variable length Arguments
def add(**args):
    print(type(args))
    print(args)
    for k,v in args.items():
        print(k,v)
print(add(n1=10,n2=20,n3=30,n4=40))
#If we want to pass key value pair(keyword argument) then we need to give **
```

```
<class 'dict'>
{'n1': 10, 'n2': 20, 'n3': 30, 'n4': 40}
n1 10
n2 20
n3 30
n4 40
None
```

In [50]:
```
#sort vs sorted
x=[10,200,125,40,50,60,70]
y=x.sort(reverse=True)
print(y)
print(id(y))
print(id(x))
print(x)
#Note --> if you are using sort function then the sorting will be done on the original list
```

```
None
140713129385176
1887457523072
[200, 125, 70, 60, 50, 40, 10]
```

In [51]:
```
x=[10,200,125,40,50,60,70]
print(id(x))     #12345
y=sorted(x,reverse=True)       #[10,40,50,60,70,125.200]
print(y)            #[10,40,50,60,70,125.200]
print(id(x))             #different
print(x)                #unsorted
print(id(y))
#Note --> if you are using sort function then the sorting will be done on the original list
```

```
1887457459712
[200, 125, 70, 60, 50, 40, 10]
1887457459712
[10, 200, 125, 40, 50, 60, 70]
1887457273984
```

In [52]:
```
#global vs local
x=200
def add():
    print(x)
    x=300
add()
#Note: You cannot access the value before assignment if you will try then you will get an error
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Input In [52], in <cell line: 6>()
      4     print(x)
      5     x=300
----> 6 add()

Input In [52], in add()
      3 def add():
----> 4     print(x)
      5     x=300

UnboundLocalError: local variable 'x' referenced before assignment
```

In [80]:
```
x=10
def add():
    x=x+10#Error
    print(x)
add()
print(x)
#Note --> you cannot modify the global variable inside the function.
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Input In [80], in <cell line: 5>()
      3     x=x+10#Error
      4     print(x)
----> 5 add()
      6 print(x)

Input In [80], in add()
      2 def add():
----> 3     x=x+10#Error
      4     print(x)

UnboundLocalError: local variable 'x' referenced before assignment
```

In [ ]:
```
```