In [ ]:
```
#Agenda of Today :
                    1. Continue to File handling
                    2. List Comprehensions
                    3. Funtional Programming (map(),filter(),reduce(),lambda)
                DAY-16  - Oops
                DAY-17  - Data Science Packages
                DAY-18  - Data Science Packages and Test on Python
```

In [ ]:
```
#File Operations(open,read,write,append,close)
```

In [12]:
```
#Write and append
f=open("newfile.txt","w")
f.write("Today is Wednesday")

f=open("newfile.txt","a")
f.write("Python is Awesome")
f=open("newfile.txt","w")
f.write("This is a new line")
f = open("newfile.txt","r")
f.read()
```

Out[12]:
```
'This is a new line'
```

In [ ]:
```
#close operation (we have different way of closing the file)
```

In [19]:
```
#1-way
f = open("newfile.txt","r")
f.read()
f.close()
```

In [21]:
```
#How to know the file object attributes (fileobject.attributename)
print(f.name)
print(f.mode)
print(f.closed)
print(f.encoding)
```

```
newfile.txt
r
True
cp1252
```

In [23]:
```
#2-way to close the file
try:
    f=open("newfile.txt","a")
    f.read()
finally:
    f.close()
```

```
---------------------------------------------------------------------------
UnsupportedOperation                      Traceback (most recent call last)
<ipython-input-23-aa7c89d20feb> in <module>
      2 try:
      3     f=open("newfile.txt","a")
----> 4     f.read()
      5 finally:
      6     f.close()

UnsupportedOperation: not readable
```

In [24]:
```
f.closed
```

Out[24]:
```
True
```

In [29]:
```
#3-way to close the file(simply open the file with "with" keyword)
with open("newfile.txt","r") as f:
    print(f.read())
```

```
This is a new line
```

In [30]:
```
f.closed
```

Out[30]:
```
True
```

In [32]:
```
#Files using with functions:
#Write data into file:
def writetofile(filename):
    with open(filename,"w") as f:
        data=f.write("Hello are you there?")
writetofile(input("enter your filename:"))
```

```
enter your filename:hello.txt
```

In [33]:
```
#read from file:
def readfile(filename):
    with open(filename,"r") as f:
        print(f.read())
readfile(input("enter your filename"))
```

```
enter your filenamehello.txt
Hello are you there?
```

In [34]:
```
def writetofile(filename):
    with open(filename,"w") as f:
        data=f.write("Hello are you there?")
    with open(filename,"r") as f:
        print(f.read())
writetofile(input("enter your filename:"))
```

```
enter your filename:hello.txt
Hello are you there?
```

In [ ]:
```
#tell() and seek() methods:
#tell()- used to know the where the cursor is placed at now
#seek() - used to change the position of cursor to origin,previous,end
    #seek(0,0)-origin
    #seek(0,1)-prevoius
    #seek(0,2)-end point
```

In [113…
```
#ex:
with open("newfile.txt","r") as f:
    #print(len(f.read()))
    data = f.read(6)
    position = f.tell()
    print(position)
    position1 = f.seek(0,0)  #change to origin position
    print(position1)
    position2 = f.seek(0,2)  #change to end position
    print(position2)
    position3 = f.seek(0,0)  #change to previous position
    print(position3)
```

```
    position4 = f.seek(0,1)
    print(position4)
```

```
6
0
18
0
0
```

In [ ]:
```
#List Compreshensions in Python:
 List Compreshensions are used to creating new lists by using existing lists
    or from other iterables (lists,sets,tuples,dictionaries)
#syntax:
 output=[result for val in input_sequence if val satisfies the condition]
```

In [46]:
```
#Ex: (using normal code)
li = [1,2,3,4,5,6,7,8,9]
out = []
for i in li:
    out.append(pow(i,2))
print(out)
```

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

In [48]:
```
 #using comprehension:
[pow(i,2) for i in li]
```

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

In [49]:
```
#Ex2:
li = []
for i in range(1,100):
    if i%5==0 and i%6!=0:
        li.append(i)
print(li)
```

```
[5, 10, 15, 20, 25, 35, 40, 45, 50, 55, 65, 70, 75, 80, 85, 95]
```

In [52]:
```
[i for i in range(1,100) if (i%5==0 and i%6!=0)]
```

Out[52]:
```
[5, 10, 15, 20, 25, 35, 40, 45, 50, 55, 65, 70, 75, 80, 85, 95]
```

In [ ]:
```
#Dictionary Comprehension:
#Syntax:
{key:value for (key,value) in iterable if (key,value) are satisfies}
```

In [53]:
```
#Ex:
li = [1,2,3,4,5,6,7,8,9]
odict= {}
for i in li:
    if i%2==0:
        odict[i]=pow(i,3)
print(odict)
```

```
{2: 8, 4: 64, 6: 216, 8: 512}
```

In [54]:
```
{i: pow(i,3) for i in range(1,10) if i%2==0}
```

Out[54]:
```
{2: 8, 4: 64, 6: 216, 8: 512}
```

In [59]:
```
#K:pow(V,3) for (k,v) in d}
```

In [ ]:
```
#Functional Programming in Python
  - what is functional Programming?
     map()- mapping (each element to element)
     filter() - filtering
     reduce() - serial mathimatical operations
     lambda - its keyword act as function without a name
Note: these functions are taken other funtion names as its parameters
```

In [ ]:
```
#map() syntax:
 map(funtionname,sequence)
```

In [65]:
```
#exmaple: (map() functions always return map objects)
def addsum(x):
    return x*x+50
li = [1,2,3,4,5,6,7,8,9]
data=set(map(addsum,li))
print(data)
```

```
{66, 99, 131, 75, 114, 51, 54, 86, 59}
```

In [ ]:
```
#filter(): to filter out the required data from given sequence
#synatx:
filter(functioname,sequence)
```

In [67]:
```
#to filter out who are eligible to vote in given list of ages
def numFilter(x):
    if x>=18:
        return x
data=filter(numFilter,(30,15,18,12,17,65,89,100,3))
print(list(data))
```

```
[30, 18, 65, 89, 100]
```

In [75]:
```
def stringFilter(s):
    if s==s[::-1]:
        return s
data=filter(stringFilter,input("enter strings").split())
print(list(data))
```

```
enter stringsmam surya 121 535 lol python
['mam', '121', '535', 'lol']
```

In [93]:
```
#reduce (its function of predefined math modules)
#syntax: reduce(functioname, sequence)
from functools import reduce
def fact(x,y):
    return x*y
fact1=reduce(fact,range(1,7))
print(fact1)
```

```
720
```

In [96]:
```
def add(x,y):
    return x+y
data=reduce(add,[10,20,30,40,50,60])
print(data)
```

```
210
```

In [ ]:
```
#Lambda:
```

```
#syntax:
      lambda arguments : expression
#Note: we can use any no.of arguments but its evalutes only one expression
```

In [97]:
```
#ex:
def cube(x):
    return x*x*x
result =lambda y:y*y*y
print(cube(5))
print(result(3))
```

```
125
27
```

In [ ]:
```
#map() with lambda filter and reduce with lambda
```

In [99]:
```
list(filter((lambda x:x%2==0),[23,4,5,6,7,8,90,85]))
```

Out[99]:
```
[4, 6, 8, 90]
```

In [104…
```
#reduce with lambda:
reduce((lambda x,y:x+y),[10,20,30,40,50,60,70,80,90,100])
```

Out[104…
```
550
```

In [102…
```
#map() filter() reduce() with lambda:
li = [23,4,5,67,88,9,94,7458,394304]
reduce(lambda x,y:x+y,filter(lambda x:x>20,map(lambda x:x+x,li)))
```

Out[102…
```
804068
```

In [ ]:
```

```