Day Objectives

- Maps
- Lambda
- Filter
- Use cases - File/Data Encryption

List Comprehesion is used for both Map and Filter

Map

- Map generate a mathematical outcome from our program
- Map Doesn't use for conditional computation
- Map fails because it doesn't apply for conditional checking
- Mapping - Mapping between Entity with Function
- f : x^2 + 3x + 9
- [x belongs to 1,10]
- f(x)
- f(1) --> 13
- f(2) --> 19 .... ..

y = f(x)

x --- y 1 ----13 2 ----- 3 4 5 6 7 8 9 10

In [ ]:
```

y = f(x)

y = x^2

x         y
1         1
2         4
3         9
4         16


map( function, Iterable)

map will not print anything it will just return
```

In [4]:
```
def powerN(a,n):
    #return a**n
    r = 1
    for i in range(0,n):
        r *= a
    return r
powerN(2,10)

def recursivePowerN(a,n):
    if n == 0:
        return 1
    else:
        return a * recursivePowerN(a,n-1)
recursivePowerN(2,10)
```

Out[4]:
```
1024
```

In [7]:
```
def cube(n):
    return n ** 3
li = [1,2,3,4,5]

# print(map(cube,li))   #-->  print won't work here  --> we got output as --
set(map(cube,li))
```

Out[7]:
```
{1, 8, 27, 64, 125}
```

In [8]:
```
def cube(n):
    return n ** 3
li = [1,2,3,4,5]

set(map(cube,123)) #--> we got error here because we haven't take a list or

```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-bb654475ff89> in <module>
      3 li = [1,2,3,4,5]
      4
----> 5 set(map(cube,123))

TypeError: 'int' object is not iterable
```

In [9]:
```
def cube(n):
    return n ** 3
li = [1,2,3,4,5]

set(map(cube,[1,3]))
```

Out[9]:
```
{1, 27}
```

In [10]:
```
def cube(n):
    return n ** 3
li = [1,2,3,4,5]

list(map(cube,[1,3]))
```
Out[10]:
```
[1, 27]
```

In [50]:
```
def cube(n):
    return n ** 3
li = ['1','2','3','4','5']
li2 = list(map(int,li))
li2
```
Out[50]:
```
[1, 2, 3, 4, 5]
```

In [51]:
```
li3 = list(map(str,li))
li3
map(float,li2)
li2
tuple(map(float, li2))



```
Out[51]:
```
(1.0, 2.0, 3.0, 4.0, 5.0)
```

In [52]:
```
numbers = [int(i) for i in li]

[cube(i) for i in numbers]
```
Out[52]:
```
[1, 8, 27, 64, 125]
```

Filter

- Used to check boolean values

In [ ]:
```
### Filter
- Used to check boolean values
    - f : x  -> {T,F}

- y is a subset of x

#### Identify the prime number
 x          y
 1
 2          2
 3          3
 4
 5          5


```

In [31]:
```
li =[1, 2,'a','b','c',3]

def isDigit(c):
    c = str(c)
    if c.isdigit():
        return 100
    return 0
#isDigit('a')    # o/p --> False
list(filter(isDigit,li))
```

Out[31]:
```
[1, 2, 3]
```

In [32]:
```
##
li =[1, 2,'a','b','c',3]

def isDigit(c):
    c = str(c)
    if c.isdigit():
        return 0
    return 100
#isDigit('a')    # o/p --> False
list(filter(isDigit,li))
```

Out[32]:
```
['a', 'b', 'c']
```

In [33]:
```
li =[1, 2,'a','b','c',3]

def isDigit(c):
    c = str(c)
    if c.isdigit():
        return 100
    return -1
#isDigit('a')    # o/p --> False
list(filter(isDigit,li))
```

Out[33]:
```
[1, 2, 'a', 'b', 'c', 3]
```

In [36]:
```
### OTHERTHAN 0 IT WILL TAKE ALL NUMBERS AS TRUE
li =[1, 2,'a','b','c',3]

def isDigit(c):
    c = str(c)
    if c.isdigit():
        return 100
    return 10292
#isDigit('a')    # o/p --> False
list(filter(isDigit,li))
```

Out[36]:
```
[1, 2, 'a', 'b', 'c', 3]
```

In [47]:
```
# Identity all Primes in a range
def checkPrime(n):
    if n < 2:
        return False
    for i in range(2,n//2+1):
        if n%i == 0:
            return False
    return True

# n = int(input())
# checkPrime(n)
lb,ub = 500,601
primeList=list(filter(checkPrime,range(lb,ub)))
primeList

        ;..
```

Out[47]:
```
[503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599]
```

In [48]:
```
primelist2 = [i for i in range(lb,ub+1) if checkPrime(i) ]
primelist2
```

Out[48]:
```
[503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601]
```

Type *Markdown* and LaTeX: α^2

Lambda

- Lambda is a key word in python
- Anonymous Functions ---> function which don't have a name
- Can be embedded into List Comprehensions, Maps, Filters

Keywords in Python programming language

- False class finally is return
- None continue for lambda try
- True def from nonlocal while
- and del global not with
- as elif if or yield
- assert else import pass
- break except in raise

In [53]:
```
a = [lambda x: x%2 == 0 for x in range(1,11)]
a
```

Out[53]:
```
[<function __main__.<listcomp>.<lambda>(x)>,
 <function __main__.<listcomp>.<lambda>(x)>,
 <function __main__.<listcomp>.<lambda>(x)>,
 <function __main__.<listcomp>.<lambda>(x)>,
 <function __main__.<listcomp>.<lambda>(x)>,
 <function __main__.<listcomp>.<lambda>(x)>,
 <function __main__.<listcomp>.<lambda>(x)>,
 <function __main__.<listcomp>.<lambda>(x)>,
 <function __main__.<listcomp>.<lambda>(x)>,
 <function __main__.<listcomp>.<lambda>(x)>]
```

In [55]:
```
a = lambda x: x**3
print(a(3))

list(map(lambda x:x**3, [1,2,3,4,5,6]))
```

```
27
```

Out[55]:
```
[1, 8, 27, 64, 125, 216]
```

In [58]:
```
# here we have used filter because it boolean expression because saying even
list(filter(lambda x:(x%2 == 0),[1,2,3,4,5,6]))
```

Out[58]:
```
[2, 4, 6]
```

In [59]:
```
# here we have used filter because it boolean expression because saying even
list(filter(lambda x:(x%2 != 0),[1,2,3,4,5,6]))
```

Out[59]:
```
[1, 3, 5]
```

In [64]:
```
from random import randint

internal1 = [randint(0,25) for i in range(10)]
internal2 = [randint(0,25) for i in range(10)]

averageInternal = list(map(lambda x,y: (x+y)/2, internal1,internal2))
averageInternal
```

Out[64]:
```
[24.0, 12.0, 10.0, 21.5, 16.5, 20.5, 7.5, 10.0, 5.0, 9.5]
```

In [65]:
```
from random import randint

internal1 = [randint(0,25) for i in range(10)]
internal2 = [randint(0,25) for i in range(10)]
internal3 = [randint(0,25) for i in range(10)]

averageInternal = list(map(lambda x,y,z: (x+y+z)//3, internal1,internal2,int
averageInternal
```

Out[65]:
```
[15, 18, 6, 13, 10, 14, 2, 15, 14, 17]
```

In [67]:
```python
from random import randint

internal1 = [randint(0,25) for i in range(10)]
internal2 = [randint(0,25) for i in range(10)]
internal3 = [randint(0,25) for i in range(10)]

averageInternal = list(map(lambda x,y,z: (x+y+z)//3, internal1,internal2,int

failedmarks = list(filter(lambda x: x<15,averageInternal))
failedmarks
```

Out[67]:
```
[14, 11, 14, 13, 14, 13, 8, 6, 7]
```

In [1]:
```python
# Generate Marks data
from random import randint

def generateMarks(n,lb,ub):
    filename = 'DataFiles/marks.txt'
    with open(filename,'w') as f:
        for i in range(n):
            marks = randint(lb,ub)
            f.write(str(marks)+'\n')
        return

generateMarks(100,0,100)

```

In [8]:
```python
# Reading MarksList
def readMarksList(filepath):
    with open(filepath,'r') as f:
        filedata = f.read().split()
    return list(map(int,filedata))
readMarksList('DataFiles/marks.txt')
```

Out[8]:
```
[9,
 5,
 91,
 62,
 100,
 60,
 85,
 46,
 29,
 94,
 42,
 19,
 17,
 29,
 29,
 12,
 38,
 1,
 71,
 58,
 43,
 84,
 41,
 44,
 74,
 62,
 42,
 68,
 23,
 28,
 88,
 34,
 78,
 95,
 60,
 21,
 23,
 50,
 10,
 21,
 35,
 12,
 67,
 71,
 58,
 65,
 36,
 54,
 43,
 41,
                78,
                97,
                68,
                85,
                83,
                88,
                77,
                40,
                73,
                34,
                67,
                98,
                90,
                67,
                80,
                48,
                75,
                43,
                66,
                11,
                55,
                14,
                23,
                58,
                100,
                15,
                57,
                80,
                69,
                89,
                34,
                21,
                79,
                75,
                23,
                76,
                22,
                36,
                16,
                55,
                94,
                93,
                45,
                28,
                73,
                22,
                60,
                31,
                29,
                18]
```

In [9]:
```
# Marks Analysis
# Class Average
# % of Passed, Failed and Distinction
# Frequency of Highest and Lowest Marks.

import re
def classAverage(filepath):

    with open(filepath,'r') as f:
        filedata = f.read()
        markslist = re.split('\n',filedata)
        markslist = list(map(int,markslist))
        return sum(markslist)//len(markslist)

filepath = 'DataFiles/marks.txt'
classAverage(filepath)



# # % of Failed
# def percentageFailed(filepath):
#     markslist = readMarksList(filepath)
#     failedcount = len(list(filter(lambda x:x<30,markslist)))
#     return failedcount

# falepath='DataFiles/marks.txt'
# percentageFailed(filepath)
```

Out[9]:
```
52
```

In [10]:
```
# %  of Failed
def percentageFailed(filepath):
    markslist = readMarksList(filepath)
    failedcount = len(list(filter(lambda x:x<30,markslist)))
    return failedcount

filepath='DataFiles/marks.txt'
percentageFailed(filepath)
```

Out[10]:
```
28
```

In [15]:
```
# % of Pass
def percentagePassed(filepath):
    markslist = readMarksList(filepath)
    passedcount = len(list(filter(lambda x:x>=30,markslist)))
    return passedcount
filepath = 'DataFiles/marks.txt'
percentagePassed(filepath)
```

Out[15]:
```
72
```

In [16]:
```python
# % of disction
def percentagedistinction(filepath):
    markslist = readMarksList(filepath)
    distinctioncount = len(list(filter(lambda x:x>=75,markslist)))
    return distinctioncount
filepath = 'DataFiles/marks.txt'
percentagedistinction(filepath)
```

Out[16]:
```
26
```

In [14]:
```python
# Highest Mark Frequency
def highestMarkFrequency(filepath):
    markslist=readMarksList(filepath)
    return [markslist.count(max(markslist)),max(markslist)]

highestMarkFrequency(filepath)
```

Out[14]:
```
[2, 100]
```

In [17]:
```python
# Lowest Mark Frequency
def lowestMarkFrequency(filepath):
    markslist=readMarksList(filepath)
    return [markslist.count(min(markslist)),min(markslist)]
lowestMarkFrequency(filepath)
```

Out[17]:
```
[1, 1]
```

Data Encryption

- **Key -** Mapping of characters with replaced characters
- assigning a key to each number here i have used 4 for each number
- 0 --> 4
- 1 --> 5
- 2 --> 6
- 3 --> 7
- 4 --> 8
- 5 --> 9
- 6 --> 0
- 7 --> 1
- 8 --> 2
- 9 --> 3

0 4

1 5

2 6 ... ...

In [19]:
```
# Function to generate key for encryption
keypath = 'DataFiles/key.txt'
def generateKey(keypath):
    with open(keypath,'w') as f:
        for i in range(10):
            if i < 6:
                f.write(str(i)+ ' ' + str(i+4)+'\n')
            else:
                f.write(str(i) + ' ' + str(i-6)+'\n')
    return
generateKey(keypath)
```

In [23]:
```
# Function to encrypt a data file

keyfile ='DataFiles/key.txt'
def dictionaryKeyFile(keyfile):
    keyDic = {}
    with open(keyfile,'r') as f:
        line = f.readline().split()    # For first line reading
        keyDic[line[0]] = line[1]
    return keyDic
dictionaryKeyFile(keyfile)


# def encryptMarksData(datafile,keyfile):
#     # consturct a dictionary for key data

```

Out[23]:
```
{'0': '4'}
```

In [27]:
```
keyfile ='DataFiles/key.txt'
def dictionaryKeyFile1(keyfile):
    keyDic = {}
    with open(keyfile,'r') as f:
        for line in f:
            line = line.split()
            keyDic[line[0]] = line[1]
    return keyDic
#dictionaryKeyFile1(keyfile)

```

Out[27]:
```
{'0': '4',
 '1': '5',
 '2': '6',
 '3': '7',
 '4': '8',
 '5': '9',
 '6': '0',
 '7': '1',
 '8': '2',
 '9': '3'}
```

In [5]:

```python
keyfile ='DataFiles/key.txt'
def dictionaryKeyFile1(keyfile):
    key = {}
    with open(keyfile,'r') as f:
        for line in f:
            line = line.split()
            key[line[0]] = line[1]
    return key
def encryptMarksData(datafile,keyfile):
    # Consturct a dictionary for key data
    key = dictionaryKeyFile1(keyfile)
    with open(datafile,'r') as f:
        filedata = f.read().split('\n')
    with open('DataFiles/encryptedMarks.txt','w') as f:
        for mark in filedata:
            line = ''
            for n in mark:
                line += key[n]
            f.write(line+'\n')
        return
filedata='DataFiles/marks.txt'
encryptMarksData(filedata,keyfile)


```

In [22]:

```python
# Function to decrypt an encrypted file
def decryptionMarkData(encryptedfile,keyfile):
    key = dictionaryKeyFile1(keyfile)
    newkey = {}
    for key,value in key.items():
        newkey[value]=key
    with open(encryptedfile,'r') as f:

        encrypteddata = f.read().split('\n')
    with open('DataFiles/decryptedMarks.txt','w') as f:
        for encryptedmark in encrypteddata:
            line =''
            for n in encryptedmark:
                line += newkey[n]
            f.write(line+'\n')
    return
encryptedfile ='DataFiles/encryptedMarks.txt'
import timeit     # -->To check the time this program is taken
st = timeit.default_timer()
#keyfile='DataFiles/key.txt'
decryptionMarkData(encryptedfile,keyfile)
print(timeit.default_timer()-st)
```

```
0.018331129002035595
```

In [15]:
```
# Comprehendsions
keyfile = 'DataFiles/key.txt'
key = dictionaryKeyFile1(keyfile)
evenkeys = {item for item in key.items() if int(item[0])%2 ==0}
evenkeys
```

Out[15]:
```
{('0', '4'), ('2', '6'), ('4', '8'), ('6', '0'), ('8', '2')}
```

In [16]:
```
keyfile = 'DataFiles/key.txt'
key = dictionaryKeyFile1(keyfile)
evenkeys = {item for item in key}
evenkeys
```

Out[16]:
```
{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'}
```

In [17]:
```
keyfile = 'DataFiles/key.txt'
key = dictionaryKeyFile1(keyfile)
evenkeys = {item for item in key}
evenkeys[0]   # ---> it shows error because set is not supports index operat
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-17-9361112141df> in <module>
      2 key = dictionaryKeyFile1(keyfile)
      3 evenkeys = {item for item in key}
----> 4 evenkeys[0]

TypeError: 'set' object is not subscriptable
```

https://scipy.org/ (https://scipy.org/) search in google

https://www.numpy.org (https://www.numpy.org)

https://www.numpy.org/devdocs/user/quickstart.ht (https://www.numpy.org/devdocs/user/quickstart.h

In [ ]:
```

```