Day objectives

- Packages and Modules
- File Handling
- Comprehensions
- Functional programming

```
- Packages and Modules

    - Package
        - Collection of Modules
    - Module
        - Collection functions,methods,classes etc...
        - Python code
        - We have two types of modules
            - 1. Built-in or Pre defined modules
                - keyword,math,random, os etc...
            - 2. Used defined modules

```

In [10]:

```
## Pre defined modules

import math
```

In [11]:

```
print(dir(math),end=' ')
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'ac
osh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos',
'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial',
'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose',
'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'l
og2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sq
rt', 'tan', 'tanh', 'tau', 'trunc']
```

In [12]:

```
import math as m
```

In [13]:

```
m.sqrt(3)
```

Out[13]:

```
1.7320508075688772
```

In [17]:

```
import math as m

m.sqrt(3)
m.floor(56.89)
m.ceil(56.89)
m.pow(2,4)
```

Out[17]:

```
16.0
```

In [1]:

```
from Package import module
```

In [20]:

```
module.is_even(98)
```

Out[20]:

```
True
```

In [21]:

```
module.is_even(99)
```

Out[21]:

```
False
```

In [2]:

```
module.is_pal('madam')
```

```
Palindrom
```

In [3]:

```
module.is_pal('house')
```

```
Not palindrom
```

In [4]:

```
print(dir(module),end=' ')
```

```
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name_
_', '__package__', '__spec__', 'is_even', 'is_pal']
```

In [2]:

```
module.is_perfect(6)
```

```
Perfect
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

```
### File Hanling

- What is file?
- What is file handling
- File methods
- Different modes of file
- Use of with statement
```

In [ ]:

```
- File
    - Which is a named location on disk or memory containing information/data
    - ex: .ipynb,.pdf,.txt,.py,.mp3,.mp4,.jpg,.png etc

- File Handling
    - Python has several function to create file,read file,update file
    and delete file

- File methods

    - Create
    - Open
    - WORK
    - Close
```

```
- open('filename','filemode')

- file_variable.close()

- r --> default mode(purpose of reading data)
```

In [5]:

```
f = open('file.txt','r')
if f:
    print('Opened Successfully...!')
f.close()
```

```
Opened Successfully...!
```

```
- file_varaible.read(size) --> it will read data uptp size what we mentiond
- file_variable.read() --> it will read entire data of file
```

In [6]:

```
## Reading data and printing

f = open('file.txt','r')
data = f.read()
print(data)
f.close()
```

```
Hello everyone
Good morning
Welcome to python workshop
Have a good day
```

In [7]:

```
f = open('file.txt','r')
data = f.read(20)
print(data)
f.close()
```

```
Hello everyone
Good
```

```
- seek()
    - Used to change cursor position from one place to another

- tell()
    - Used to know the cursor position
```

In [11]:

```
f1 = open('file.txt')
data = f1.read(5)
r = f1.tell()
print(r)
f1.seek(20)
r1 = f1.tell()
print(r1)
f1.close()
```

```
5
20
```

In [12]:

```
with open('file.txt') as f:
    data = f.read()
    print(data)
```

```
Hello everyone
Good morning
Welcome to python workshop
Have a good day
```

In [13]:

```
def ReadFile(filename):
    with open(filename) as f:
        filedata = f.read()
    return filedata

ReadFile('file.txt')

```

Out[13]:

```
'Hello everyone\nGood morning\nWelcome to python workshop \nHave a good day'
```

In [15]:

```
def no_lines(filename):
    with open(filename) as f1:
        res = len(ReadFile(filename).split('\n'))
    return res
no_lines('file.txt')
```

Out[15]:

```
4
```

In [16]:

```
def no_words(filename):
    w_c = 0
    with open(filename) as f2:
        filedata = f2.readlines()
        for line in filedata:
            words_list = line.split()
            w_c += len(words_list)
    return  w_c
no_words('file.txt')

```

Out[16]:

```
12
```

Comprehensions

- It is a process of creating new sequence from existed sequence

```
- Types Of Comprehensions

  - List Comprehension
  - Dictionary Comprehension
  - Set Comprehension

```

```
- Advantages
    - Less code
    - Less complexity
```

In [ ]:

```
# [23,56,78,11,99]
# [23,11,99]
```

In [21]:

```
s=[23,56,78,11,99]
for i in s:
    if(i%2==1):
        print(i,end=' ')
```

```
23 11 99
```

In [ ]:

```
syntax for list comprehension:

        - [output_variable loop condition]
```

In [22]:

```
j=[i for i in s if i%2==1]
j
```

Out[22]:

```
[23, 11, 99]
```

In [23]:

```
# natural numbers upto 10
k=[i for i in range(10)]
k
```

Out[23]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [25]:

```
k1=[i**2 for i in range(10)]
k1
```

Out[25]:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [32]:

```
f=["good afternoon everyone","welcome to python programming"]
# o/p: 'good','afternoon','everyone','welcome','to','python',
    # 'programming'
f1=[]
for i in f:
    for j in i.split():
        f1.append(i)
print(f1)
```

```
['good afternoon everyone', 'good afternoon everyone', 'good afternoon every
one', 'welcome to python programming', 'welcome to python programming', 'wel
come to python programming', 'welcome to python programming']
```

In [1]:

```
f=['good afternoon everyone','welcome to python programming']
f1=[]
for i in f:
    for j in i.split():
        f1.append(j)
print(f1)
```

```
['good', 'afternoon', 'everyone', 'welcome', 'to', 'python', 'programming']
```

In [2]:

```
q=[j for i in f for j in i.split()]
q
```

Out[2]:

```
['good', 'afternoon', 'everyone', 'welcome', 'to', 'python', 'programming']
```

In [4]:

```
# Dictionary Comprehension
# o/p: {1:1,2:4,3:9,4:16,5:25}
k={i:i**2 for i in range(6)}
k
```

Out[4]:

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

In [5]:

```
# i/p: [2,3,4,5,1,8,9,7]
# o/p: {2:8,3:27,4:64,5:125,.....7:343}
m=[2,3,4,5,1,8,9,7]
n={i:i**3 for i in m}
n
```

Out[5]:

```
{2: 8, 3: 27, 4: 64, 5: 125, 1: 1, 8: 512, 9: 729, 7: 343}
```

Set Comprehension

In [ ]:

```
i/p: [5,6,7,3,4,9]
o/p: {25 49 9 81}
```

In [9]:

```
s=[5,6,7,3,4,9]
f=set()
for i in s:
    if(i%2!=0):
        f.add(i**2)
f
```

Out[9]:

```
{9, 25, 49, 81}
```

In [10]:

```
l={i**2 for i in s if(i%2!=0)}
l
```

Out[10]:

```
{9, 25, 49, 81}
```

Functional Programming

In [ ]:

```
- It is used to create small functions
- We can call it as single line functions
- Anonymous Functions
1.lambda()
    - syntax: lambda arguments:condition
2.map()
    - syntax:  map(function,sequence)
3.filter()
    - syntax:  filter(function,sequence)
```

In [11]:

```
# addition of 2 numbers by using function
def add(a,b):
    print(a+b)
add(2,3)
```

```
5
```

In [13]:

```
k=(lambda a,b:a+b)
k(4,5)
```

Out[13]:

```
9
```

In [17]:

```
k1=(lambda a,b,c:a*b*c)
k1(4,5,6)
```

Out[17]:

```
120
```

In [18]:

```
# map()
def mul(a):
    return a*a
m=map(mul,[2,3,5])
print(list(m))
```

```
[4, 9, 25]
```

In [ ]:

```
i/p: ["RAJU","RAVI","RAM"]
o/p: ["raju","ravi","ram"]

```

In [19]:

```
c=["RAJU","RAVI","RAM"]
res=list(map(str.lower,c))
res
```

Out[19]:

```
['raju', 'ravi', 'ram']
```

In [ ]:

```
# filter()
i/p: [23,56,11,67,55,235]
o/p: 56,67,55,235
```

In [24]:

```
d=[23,56,11,67,55,235]
def greater(d):
    if(d>=55):
        return True
    else:
        return False
n=filter(greater,d)
for i in n:
    print(i,end=' ')
```

```
56 67 55 235
```

In [ ]:

```

```