MODULE 1 AND 2 COMBINED

1. Different ways to declare a dictionary

```
#Empty dictionary
my_dict={}
```

1. my_dict={1:'apple',2:'ball'}
   print(my_dict)
   OUTPUT: {1: 'apple', 2: 'ball'}

2. my_dict={'Name':'John',1:[2,4,3]}
   print(my_dict)
   OUTPUT: {'Name': 'John', 1: [2, 4, 3]}

3. my_dict=dict({1:'apple',2:'ball'})
   print(my_dict)
   OUTPUT: {1: 'apple', 2: 'ball'}

4. my_dict=dict([(1,'apple'),(2,'ball')])
   print(my_dict)
   OUTPUT: {1: 'apple', 2: 'ball'}

2. Dictionary functions

```
my_dict={'name':'John','age':26}
print(my_dict)
OUTPUT: {'name': 'John', 'age': 26}

#To display the value by calling its key
print(my_dict['name'])
OUTPUT: John
```

```
#Another method to display the value by calling its key
print(my_dict.get('age'))
OUTPUT: 26

#To update an existing value by calling its key\
my_dict['age']=27
print(my_dict)
OUTPUT: {'name': 'John', 'age': 27}   #The age is updated from 26 to 27

#To add a new value in the dictionary
my_dict['address']='Downtown'
print(my_dict)
OUTPUT: {'name': 'John', 'age': 27, 'address': 'Downtown'}



squares={1:1, 2:4, 3:9, 4:16, 5:25}
print(squares)
OUTPUT: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

#To pop(delete) a value out of the dictionary
print(squares.pop(4))     #The popped value will be printed
OUTPUT: 16
print(squares)               #The updated dictionary will be printed
OUTPUT: {1: 1, 2: 4, 3: 9, 5: 25}

#popitem removes the last added element from the dictionary
print(squares.popitem())
OUTPUT: (5, 25)
print(squares)                          #Prints the updated dictionary
OUTPUT: {1: 1, 2: 4, 3: 9}
```

```
#To delete all the elements in the dictionary
squares.clear()
print(squares)
OUTPUT: {}
```

3. Tuple functions

```
my_tuple=()
print(my_tuple)

my_tuple=(1, 2, 3)
print(my_tuple)

my_tuple=(1, "Hello", 3, 4)
print(my_tuple)

my_tuple=("mouse",[8, 4, 6],(1, 2, 3))
print(my_tuple)

my_tuple=3, 4.6, "dog"
print(my_tuple)

a, b, c=my_tuple
print(a)
print(b)
print(c)

my_tuple=("hello")
print(type(my_tuple))
```

```
my_tuple="hello"
print(type(my_tuple))

my_tuple=('p','e','r','m','i','t')
print(my_tuple[0])
print(my_tuple[5])

#list indices must be integers, not float

n_tuple=("mouse",[8,4,6],(1,2,3))
print(n_tuple[0][3])
print(n_tuple[1][1])

my_tuple=('p','e','r','m','i','t')
print(my_tuple[-1])
print(my_tuple[-6])

my_tuple=('p','r','o','g','r','a','m','i','z')
print(my_tuple[1:4])
print(my_tuple[:-7])
print(my_tuple[7:])
print(my_tuple[:])

my_tuple=(4,2,3,[6,5])

#my_tuple[1]=9

my_tuple[3][0]=9
print(my_tuple)

my_tuple=('p','r','o','g','r','a','m','i','z')
```

```
print(my_tuple)
print((1,2,3)+(4,5,6))
print(("repeat")*3)

my_tuple=('p','r','o','g','r','a','m','i','z')
del my_tuple
print(my_tuple)
```

4. String functions

```
string_var='Python'
string_var="Python"
string_var="""Python"""

string_var=""" This document will help you to
explore all the concepts
of Python Strings ! """

substr_var=string_var.replace("document","tutorial")
print(substr_var)

sample_str='Python String'

print(sample_str[0])

print(sample_str[-1])

print(sample_str[-2])

print(sample_str[3:5])
```

```
print(sample_str[7:])

print(sample_str[:6])

print(sample_str[7:-4])

var1='Python'
var2='String'
print(var1+var2)

var1='Python'
print(var1*3)

var1='Python'
print('n' in var1)

var1='Python'
print("N" not in var1)

for var in var1: print(var, end ="")
```

5. Set functions

```
my_set={1,2,3}
print(my_set)

my_set={1.0, "Hello", (1,2,3)}
print(my_set)
```

```
my_set={1,2,3,4,3,2}
print(my_set)

#my_set={1,2,[3,4]}
#typeerror: set cannot have mutable items(list)

my_set=set([1,2,3])
print(my_set)

#creating an empty set
a={}
print(type(a))
#says a is a dictionary

a=set()
print(type(a))

my_set={1,3}
print(my_set)
my_set.add(2)
print(my_set)

my_set.update([2,3,4])
print(my_set)

my_set.update([4,5],[1,6,8])
print(my_set)

my_set={1,3,4,5,6}
print(my_set)
```

```
my_set.discard(4)
print(my_set)

my_set.remove(6)
print(my_set)

#removing an element that is not present in my_set
my_set.discard(2)
print(my_set)

my_set=set("HelloWorld")
print(my_set)

#pop a random element
print(my_set.pop())

my_set.pop()
print(my_set)

my_set.clear()
print(my_set)

A={1,2,3,4,5}
B={4,5,6,7,8}

#union function
print(A | B)
print(A.union(B))

#intersection function
print(A & B)
```

```
print(A.intersection(B))

#difference function
print(A - B)
print(A.difference(B))

print(B - A)
print(B.difference(A))
```

6. Except functions

```
import sys
randomlist=['a',0,2]
for entry in randomlist:
    try:
        print("The entry is: ",entry)
        r=1/int(entry)
        break
    except:
        print("Oops!",sys.exc_info()[0],"occured.")
        print("Next entry ")
        print()
print("The reciprocal of ",entry,"is ",r)

try:
    a=int(input("Enter a positive integer: "))
    if a<=0:
        raise ValueError("That is not a positive number!")
except ValueError as ve:
    print(ve)
```

7. Random functions

```
import random

print("A random number from list is:",end="")
print(random.choice([1,4,8,10,3]))

print("A random number from range is:",end="")
print(random.choice([20,50,3]))

print("A random number between 0 and 1 is:",end="")
print(random.random())

random.seed(5)

print("A mapped number with 5 is:",end="")
print(random.random())

random.seed(7)

print("A mapped number with 7 is:",end="")
print(random.random())

random.seed(5)

print("A mapped number with 5 is:",end="")
print(random.random())
```

8. User defined functions

```
def add_numbers(x,y):
    sum=x+y
    return sum
num1=2
num2=8
print("The sum is", add_numbers(num1, num2))

def sum(a=4,b=2):
    print(a+b)

sum(1,2) #calling with arguments
sum()    #calling without arguments
```

9. Init functions

```
class Student:
    def __init__(self, name, roll_no):
        self.name=name
        self.roll_no=roll_no
    def myfunc(self):
        print("Name: " + self.name)
        print("Roll no: ", self.roll_no)


p1=Student("X",20198002)
p2=Student("Y",20198003)

print(p1.name)
```

```
print(p1.roll_no)


print(p1.name)
print(p1.roll_no)


p1.myfunc()
p2.myfunc()
```

10. OOPS

```
class Polygon:
    def __init__(self, no_of_sides):
        self.n=no_of_sides
        self.sides=[0 for i in range(no_of_sides)]


    def inputSides(self):
        self.sides=[float(input("Enter side"+str(i+1)+" : ")) for i in range(self.n)]


    def dispSides(self):
        for i in range(self.n):
            print("Side",i+1,"is",self.sides[i])


class Triangle(Polygon):
    def __init__(self):
        Polygon.__init__(self, 3)


    def findArea(self):
        a,b,c=self.sides
        #calculate the semi perimeter
        s=(a+b+c)/2
        area=(s*(s-a)*(s-b)*(s-c))**0.5
```

```
        print('The area of the triangle is %0.2f' %area)

class Quadrilateral(Polygon):
    def __init__(self):
        Polygon.__init__(self, 4)

    def findPeri(self):
        a,b,c,d=self.sides
        print("The perimeter of the the quadrilateral is ", a+b+c+d)

shape=int(input("Enter 1 for triangle and 2 for quadriletral: "))
if(shape==1):
    t=Triangle()
    t.inputSides()
    t.dispSides()
    t.findArea()

else:
    q=Quadrilateral()
    q.inputSides()
    q.dispSides()
    q.findPeri()
```

11. Math functions

```
import math

a=6.4
print(math.ceil(a))
print(math.floor(a))
```

```
b=-8
c=4
print(math.fabs(b))
print(math.factorial(c))

d=-3
e=8.2
print(math.copysign(e,d))

print(math.gcd(72,132))

print(math.exp(1))

print(math.log(10,10))

print(math.log2(16))

print(math.log10(1000))

print(math.pow(2,8))

print(math.sqrt(144))

f=math.pi/2

print(math.sin((f)))

print(math.cos((f)))

print(math.tan((f)))
```

```
print(math.hypot(1,2))

print(math.degrees(math.pi/180))
print(math.radians(180/math.pi))

print(math.gamma(4))

print(math.pi)

print(math.e)

if(math.isnan(math.nan)):
   print("The number is nan")
else:
   print("The number is not nan")

if(math.isinf(math.inf)):
   print("The number is positive infinity")
else:
   print("the number is not positive infinity")
```

MODULE 3: INTRODUCTION TO NUMPY

1. Importing a csv file using numpy

```
import numpy as np

firstarray=np.genfromtxt("Numpydata.csv", delimiter=",")
print(firstarray)
```

```
[[ nan  10. 100.]
 [  2.  20. 100.]
 [  3.  30. 100.]
 [  4.  40. 100.]
 [  5.  50. 100.]
 [  6.  60. 100.]
 [  7.  70. 100.]
 [  8.  80. 100.]
 [  9.  90. 100.]
 [ 10. 100. 100.]]
```

2. Saving an array as a csv file using numpy

```
import numpy as np

nparray=([[1, 2, 3],[4, 5, 6],[7, 8, 9]])
print(nparray)

#saving the array as a csv file(excel)
np.savetxt("firstarray.csv", nparray, delimiter=",")

#reading the csv into an array
firstarray=np.genfromtxt("firstarray.csv", delimiter=",")
print(firstarray)
```

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

3. Displaying the details of the array using numpy functions

```
import numpy as np
```

```
arr=np.array([[1,2,3],
              [4,5,6]])

print("Array is of type: ", type(arr))

print("No. of dimensions: ", arr.ndim)

print("Shape of the array: ", arr.shape)

print("Size of the array: ", arr.size)

print("Array stores elements of type: ", arr.dtype)
#if the values in an array is float, the data type changes
```

```
Array is of type:  <class 'numpy.ndarray'>
No. of dimensions:  2
Shape of the array:  (2, 3)
Size of the array:  6
Array stores elements of type:  int32
```

4. Numpy array functions (Creating, reshaping and flattening)

```
import numpy as np

#Creating an array from list with data type
a=np.array([[1,2,3],[4,5,6]], dtype='float')
print("Array created using passed list:\n", a)
```

```
Array created using passed list:
 [[1. 2. 3.]
 [4. 5. 6.]]
```

#Creating an array from tuples

```
b=np.array((1,2,3))

print("\nAn array created using passed tuple:\n", b)
```

```
An array created using passed tuple:
 [1 2 3]
```

#Creating a 3x4 array with all zeroes

```
c=np.zeros((3,4))

print("\nAn array initialized with all zeroes:\n", c)
```

```
An array initialized with all zeroes:
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

#Creating a constant value array of complex type

```
d=np.full((3,3), 6, dtype='complex')

print("\nAn array initialized with all 6s:"

    "(Array type is complex)\n", d)
```

```
An array initialized with all 6s:(Array type is complex)
 [[6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]]
```

#Creating anarray with random values

```
e=np.random.random((2,2))
print("\nA random array:\n", e)
```

```
A random array:
 [[0.47848641 0.94230795]
 [0.52249977 0.8406486 ]]
```

```
#Creating a sequence of integers from 0 to 5 with steps of 30
f=np.arange(0,30,5)
print("\nA sequential array with steps of 5:\n", f)
```

```
A sequential array with steps of 5:
 [ 0  5 10 15 20 25]
```

```
#Creating a sequence of 10 values in range 0 to 5
f=np.linspace(0,5,10)
print("\nA sequential array with steps of 5:\n", f)
```

```
A sequential array with steps of 5:
 [0.         0.55555556 1.11111111 1.66666667 2.22222222 2.77777778
 3.33333333 3.88888889 4.44444444 5.        ]
```

```
#Reshaping a 3x4 array to 2x2x3 array
arr=np.array([[1,2,3,4],
        [4,5,6,7],
        [6,7,8,9,]])
newarr=arr.reshape(2,2,3)
print("\nOriginal Array:\n", arr)
print("\nReshaped Array:\n", newarr)
```

```
Original Array:
 [[1 2 3 4]
 [4 5 6 7]
 [6 7 8 9]]

Reshaped Array:
 [[[1 2 3]
  [4 4 5]]

 [[6 7 6]
  [7 8 9]]]
```

#Flattening an array

arr=np.array([[1,2,3],[4,5,6]])

flarr=arr.flatten()

print("\nOriginal Array:\n", arr)

print("\nFlattened Array:\n", flarr)

```
Original Array:
 [[1 2 3]
 [4 5 6]]

Flattened Array:
 [1 2 3 4 5 6]
```

5. Arithmetic operations on arrays using numpy functions

import numpy as np

arr=np.array([1,2,3,4])

print(arr)

OUTPUT: [1 2 3 4]

#Adds 1 to each element in the array

print(arr+1)

OUTPUT: [2 3 4 5]

#Subracts 3 from each element in the array

print(arr-3)

OUTPUT: [-2 -1 0 1]

#Multiplies 10 to each element in the array

print(arr*10)

OUTPUT: [10 20 30 40]

#Squaring each element in the array

print(arr**2)

OUTPUT: [1 4 9 16]

In the above method, array is not updated after each print statement. Hence each operation is performed on the original array.

In the following method, the array is updated after each operation is executed and the further operations are performed on the updated array.

arr=np.array([1,2,3,4])

arr+=1

print(arr)

OUTPUT: [2 3 4 5]

```
arr-=3
print(arr)
OUTPUT: [-1 0 1 2]

arr*=10
print(arr)
OUTPUT: [-10 0 10 20]

arr**=2
print(arr)
OUTPUT: [100 0 100 400]
```

6. Printing the transpose of the array using numpy function

`

```
#Transposing an array
arr=np.array([[1,2,3],
[4,5,6],
[7,8,9]])
print("\nOriginal Array:\n", arr)
print("\nTranspose of the array:\n", arr.T)
```

```
Original Array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]

Transpose of the array:
 [[1 4 7]
 [2 5 8]
 [3 6 9]]
```

7. Slicing an array using numpy functions

```
import numpy as np
x=np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

print(x[1:7:2])
```

OUTPUT: [1 3 5]

```
print(x[-2:10])
```

OUTPUT: [8 9]

```
print(x[5:])
```

OUTPUT: [5 6 7 8 9]

8. Boolean indexing using numpy on arrays

```
Import numpy as np
a=np.array([10, 40, 80, 50, 100])
print(a[a>50])
```

OUTPUT: [80 100]

9. Sum functions on arrays using numpy

```
import numpy as np
b=np.array([[5,5],[4,5],[16,4]])

#Sum of the elements in the row
sumrow=b.sum(1)
print(sumrow)
```
OUTPUT: [10 9 20]

```
#elements of the row whose sum is divisible by 10
print(b[sumrow%10==0])
```
OUTPUT: [[5 5]
[16 4]]

MODULE 4: DATA MANIPULATION WITH PANDAS

1. Reading a CSV file (Refer 12. and 13. for operations on csv file)

```
import pandas as pd
df=pd.read_csv('Pandadata.csv')
print(df.to_string())
```

OR

```
import pandas as pd
df=pd.read_csv('Pandadata.csv')
print(df)
```

2. Creating a dataframe (Check 5. for another method)

```
import pandas as pd
```

```
data=[{'a':1, 'b':2},{'a':5, 'b':10, 'c':20}]
df=pd.DataFrame(data)
print(df)
```

```
   a   b     c
0  1   2   NaN
1  5  10  20.0
```

3. Naming the rows

```
import pandas as pd
data=[{'a':1, 'b':2},{'a':5, 'b':10, 'c':20}]
df=pd.DataFrame(data, index=['first','second'])
print(df)
```

```
        a   b     c
first   1   2   NaN
second  5  10  20.0
```

4. Storing a dataframe with specified index and columns

```
import pandas as pd
data=[{'a':1, 'b':2},{'a':5, 'b':10, 'c':20}]

#with two column indices, the values are  same as dictionary keys
df1=pd.DataFrame(data, index=['first','second'], columns=['a','b'])
```

print(df1)

```
        a   b
first   1   2
second  5  10
```

#when the column name is specified is not available, NaN is assigned to the unknown values

df2=pd.DataFrame(data, index=['first','second'], columns=['a','b1'])

print(df2)

```
        a  b1
first   1 NaN
second  5 NaN
```

5. Creating dataframes using series

import pandas as pd

d={'one':pd.Series([1,2,3], index=['a','b','c']),

  'two':pd.Series([1,2,3,4], index=['a','b','c','d'])}

df=pd.DataFrame(d)

print(df)

```
    one  two
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4
```

#Adding new columns

df[three]=pd.Series([10,20,30], index=[‘a’,’b’,’c’]

print(df)

```
   one  two  three
a  1.0    1   10.0
b  2.0    2   20.0
c  3.0    3   30.0
d  NaN    4    NaN
```

#Adding columns by performing operations on existing columns

df['four']=df['one']+df['three']

print(df)

```
   one  two  three  four
a  1.0    1   10.0  11.0
b  2.0    2   20.0  22.0
c  3.0    3   30.0  33.0
d  NaN    4    NaN   NaN
```

6. Generating random values

import numpy as np

import pandas as pd

#rand(uniform distribution) can also be replaced by randn(non uniform distribution)

s=pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])

print(s.index) #Prints the index values alone

print(s) #Prints the dataframe

print(s.empty) #Returns true if the dataframe is empty else returns false

```
Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
a   -0.953725
b   -1.089666
c    0.081087
d   -0.978391
e    0.861797
dtype: float64
False
```

7. Sorting data in dataframes (sorts the index) (Refer 13. For sorting data in a csv file)

```
import pandas as pd

#(5,2) indicates 5 rows and 2 columns
unsorted_df=pd.DataFrame(np.random.randn(5,2), index=[1,4,6,2,3], columns=['Column1', 'Column2'])
print(unsorted_df)

sorted_df=unsorted_df.sort_index()          #the index is sorted
print(sorted_df)
```

```
    Column1   Column2
1 -0.690842  1.493062
4 -0.046773  0.178735
6 -0.607007 -0.250675
2  0.299335  0.421626
3  0.011015  0.306079
    Column1   Column2
1 -0.690842  1.493062
2  0.299335  0.421626
3  0.011015  0.306079
4 -0.046773  0.178735
6 -0.607007 -0.250675
```

8. Queries in DataFrames (Printing required values) (refer 15. To print required values from a csv file)

```
import pandas as pd

df=pd.DataFrame({'a':[1,4,7,2], 'b':[2,0,8,7]})

#get rows where b>4
filtered_df=df.query('b>4')

print('Original DataFrame\n----------\n',df)
print('\nFiltered DataFrame\n----------\n',filtered_df)
```

```
Original DataFrame
-----------------
   a  b
0  1  2
1  4  0
2  7  8
3  2  7

Filtered DataFrame
-----------------
   a  b
2  7  8
3  2  7
```

```
#print select columns

a=df.a

print(a)
```

```
0    1
1    4
2    7
3    2
Name: a, dtype: int64
```

9. Printing the number of unique values

```
import pandas as pd

df=pd.DataFrame({"A":[14,4,5,4,1],
                 "B":[5,2,54,3,2],
                 "C":[20,20,7,3,8],
                 "D":[14,3,6,2,6]})

print(df)

#printing unique values
print(df.nunique(axis=1))     #axis = 0 is rows, axis = 1 is columns
#df.A.nunique() gives the number of unique values in A alone
```

```
    A   B   C   D
0  14   5  20  14
1   4   2  20   3
2   5  54   7   6
3   4   3   3   2
4   1   2   8   6
0    3
1    4
2    4
3    3
4    4
dtype: int64
```

10. Creating a dataframe using dictionary and finding the max and min values in rows and columns

```
import pandas as pd

#Creating a dataframe using dictionary

mydictionary={'physics':[68,74,77,78],
        'chemistry':[84,56,73,69],
        'mathematics':[78,88,82,87]}



df_marks=pd.DataFrame(mydictionary)
print('DataFrame\n----------')
print(df_marks)

#calculate max along row
mx=df_marks.max(axis=1)
print('\nMaximum Value\n------')
print(mx)
```

```
#calculate max along columns
mx=df_marks.max(axis=0)
print('\nMaximum Value\n------')
print(mx)

#calculate min along row
mn=df_marks.min(axis=1)
print('\nMinimum Value\n------')
print(mn)

#calculate max along columns
mn=df_marks.min(axis=0)
print('\nMinimum Value\n------')
print(mn)
```

```
   physics  chemistry  mathematics
0       68         84           78
1       74         56           88
2       77         73           82
3       78         69           87

Maximum Value
------
0    84
1    88
2    82
3    87
dtype: int64

Maximum Value
------
physics        78
chemistry      84
mathematics    88
dtype: int64

Minimum Value
------
0    68
1    56
2    73
3    69
dtype: int64

Minimum Value
------
physics        68
chemistry      56
mathematics    78
dtype: int64
```

11. Iteration in dataframes (Printing elements in a desired row)

```
import pandas as pd

df=pd.DataFrame([['a','b','c'],
                 ['d','e','f'],
                 ['g','h','i'],
                 ['j','k','l']])
print(df)

#point the index to the row that needs to be printed
row=df.iloc[1]          #index=1 => second row
length=row.size
for i in range(length):
   print(row[i])
```

```
   0  1  2
0  a  b  c
1  d  e  f
2  g  h  i
3  j  k  l
d
e
f
```

12. Operations on CSV file

```
import pandas as pd

#change the index_col values to get the mentioned column as the first column
df=pd.read_csv("Pandadata.csv", index_col=1)

#Everytime the index value is changed, the data is over written in the excel file that is newly saved
df.to_excel("pdata.xlsx")                    #Saves the data as a new excel file
print(df.dtypes)
print(df)
```

```
ROLL NO       int64
THEORY        int64
PRACTICALS    int64
dtype: object
      ROLL NO  THEORY  PRACTICALS
NAME
Aa          1      97          10
Bb          2      98          10
Cc          3      93           0
Dd          4      94          10
Ee          5      92          10
Ff          6      98           0
Gg          7      97          10
Hh          8      93           0
Ii          9      90          10
Jj         10     100          10
Kk         11      96          10
Ll         12      95           0
Mm         13     100          10
Nn         14      91          10
Oo         15      92          10
Gg          7      97          10
Ii          9      90          10
Bb          2      98          10
```

#To remove the duplicates

print(df.shape) #Prints the shape of the dataframe before removing the duplicates

df=df.drop_duplicates() #Columns can be mentioned within the parantheses to remove duplicates in that column.

Eg: df.drop_duplicates(["THEORY"])

print(df.shape) #Prints the shape of the dataframe after removing the duplicates

print(df) #Prints without the duplicates(The last three entries are removed)

```
(18, 3)
(15, 3)
      ROLL NO  THEORY  PRACTICALS
NAME
Aa          1      97          10
Bb          2      98          10
Cc          3      93           0
Dd          4      94          10
Ee          5      92          10
Ff          6      98           0
Gg          7      97          10
Hh          8      93           0
Ii          9      90          10
Jj         10     100          10
Kk         11      96          10
Ll         12      95           0
Mm         13     100          10
Nn         14      91          10
Oo         15      92          10
```

print(df.info()) #Prints complete information regarding the dataframe

```
<class 'pandas.core.frame.DataFrame'>
Index: 15 entries, Aa to Oo
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   ROLL NO     15 non-null     int64
 1   THEORY      15 non-null     int64
 2   PRACTICALS  15 non-null     int64
dtypes: int64(3)
memory usage: 480.0+ bytes
None
```

#the head command by default prints the first 5 rows.
#A value has to be mentioned within the head parantheses to print the desired rows

```
print(df.head())
print(df.head(2))
```

```
      ROLL NO  THEORY  PRACTICALS
NAME
Aa          1      97          10
Bb          2      98          10
Cc          3      93           0
Dd          4      94          10
Ee          5      92          10
      ROLL NO  THEORY  PRACTICALS
NAME
Aa          1      97          10
Bb          2      98          10
```

#the tail command by default prints the last 5 rows.
#A value has to be mentioned within the head parentheses to print the desired rows

```
print(df.tail())
print(df.tail(7))
```

```
      ROLL NO  THEORY  PRACTICALS
NAME
Nn         14      91          10
Oo         15      92          10
Gg          7      97          10
Ii          9      90          10
Bb          2      98          10
      ROLL NO  THEORY  PRACTICALS
NAME
Ll         12      95           0
Mm         13     100          10
Nn         14      91          10
Oo         15      92          10
Gg          7      97          10
Ii          9      90          10
Bb          2      98          10
```

13. Filtering data from a CSV file (Refer 7. for other sorting operations) (Refer 15. for filters)

```
import pandas as pd

data=pd.read_csv("Pandadata.csv")

#sorting dataframe
data.sort_values("NAME",inplace=True)          #Sorts the specified column

#making boolean series for a name
filter=data["PRACTICALS"]==10

#filtering data
data.where(filter, inplace=True)

print(data)
```

	ROLL NO	NAME	THEORY	PRACTICALS
0	1.0	Aa	97.0	10.0
17	2.0	Bb	98.0	10.0
1	2.0	Bb	98.0	10.0
2	NaN	NaN	NaN	NaN
3	4.0	Dd	94.0	10.0
4	5.0	Ee	92.0	10.0
5	NaN	NaN	NaN	NaN
6	7.0	Gg	97.0	10.0
15	7.0	Gg	97.0	10.0
7	NaN	NaN	NaN	NaN
16	9.0	Ii	90.0	10.0
8	9.0	Ii	90.0	10.0
9	10.0	Jj	100.0	10.0
10	11.0	Kk	96.0	10.0
11	NaN	NaN	NaN	NaN
12	13.0	Mm	100.0	10.0
13	14.0	Nn	91.0	10.0
14	15.0	Oo	92.0	10.0

#The rows where practicals is not equal to 10 are removed.

14. Dropping columns from an existing CSV file

```
import pandas as pd

data=pd.read_csv("Pandadata.csv", index_col="ROLL NO")
print(data)
```

ROLL NO	NAME	THEORY	PRACTICALS
1	Aa	97	10
2	Bb	98	10
3	Cc	93	0
4	Dd	94	10
5	Ee	92	10
6	Ff	98	0
7	Gg	97	10
8	Hh	93	0
9	Ii	90	10
10	Jj	100	10
11	Kk	96	10
12	Ll	95	0
13	Mm	100	10
14	Nn	91	10
15	Oo	92	10
7	Gg	97	10
9	Ii	90	10
2	Bb	98	10

#Dropping the passed columns

#["Practicals","Theory] can be used to drop more columns

data.drop(["PRACTICALS"], axis=1, inplace=True) #axis=1 – Columns

print(data)

ROLL NO	NAME	THEORY
1	Aa	97
2	Bb	98
3	Cc	93
4	Dd	94
5	Ee	92
6	Ff	98
7	Gg	97
8	Hh	93
9	Ii	90
10	Jj	100
11	Kk	96
12	Ll	95
13	Mm	100
14	Nn	91
15	Oo	92
7	Gg	97
9	Ii	90
2	Bb	98

#Dropping the passed rows

#[1,2] can be used to drop more rows

data.drop([2], axis=0, inplace=True) #axis=0 – Rows

print(data)

ROLL NO	NAME	THEORY
1	Aa	97
3	Cc	93
4	Dd	94
5	Ee	92
6	Ff	98
7	Gg	97
8	Hh	93
9	Ii	90
10	Jj	100
11	Kk	96
12	Ll	95
13	Mm	100
14	Nn	91
15	Oo	92
7	Gg	97
9	Ii	90

15. Selecting and printing values from a csv file based on conditions (Refer 13. for similar operations)

```
import pandas as pd

data=pd.read_csv("Pandadata.csv")
print(data)

filter=data[“THEORY”]>0
print(filter)
```

```
0     False
1      True
2     False
3     False
4     False
5      True
6     False
7     False
8     False
9      True
10    False
11    False
12     True
13    False
14    False
15    False
16    False
17     True
Name: THEORY, dtype: bool
```

```
#Printing a row based on conditions
row=df.loc[df["ROLL NO"]>9]
print(row)
```

```
    ROLL NO NAME  THEORY  PRACTICALS
9         10   Jj     100          10
10        11   Kk      96          10
11        12   Ll      95           0
12        13   Mm     100          10
13        14   Nn      91          10
14        15   Oo      92          10
```

16. To find the rank of the dataframe

```
import pandas as pd

data=pd.read_csv("Pandadata.csv")
print(data)

#Returns the rank based on the position
print(data.rank())
```

	ROLL NO	NAME	THEORY	PRACTICALS
0	1.0	1.0	12.0	11.5
1	2.5	2.5	15.0	11.5
2	4.0	4.0	6.5	2.5
3	5.0	5.0	8.0	11.5
4	6.0	6.0	4.5	11.5
5	7.0	7.0	15.0	2.5
6	8.5	8.5	12.0	11.5
7	10.0	10.0	6.5	2.5
8	11.5	11.5	1.5	11.5
9	13.0	13.0	17.5	11.5
10	14.0	14.0	10.0	11.5
11	15.0	15.0	9.0	2.5
12	16.0	16.0	17.5	11.5
13	17.0	17.0	3.0	11.5
14	18.0	18.0	4.5	11.5
15	8.5	8.5	12.0	11.5
16	11.5	11.5	1.5	11.5
17	2.5	2.5	15.0	11.5

17. Sorting the index of a csv file

```
import pandas as pd


df=pd.read_csv("Pandadata.csv", index_col=2)

sorteddf=df.sort_index()

print(sorteddf)
```

	ROLL NO	NAME	PRACTICALS
THEORY			
90	9	Ii	10
90	9	Ii	10
91	14	Nn	10
92	5	Ee	10
92	15	Oo	10
93	3	Cc	0
93	8	Hh	0
94	4	Dd	10
95	12	Ll	0
96	11	Kk	10
97	7	Gg	10
97	1	Aa	10
97	7	Gg	10
98	6	Ff	0
98	2	Bb	10
98	2	Bb	10
100	13	Mm	10
100	10	Jj	10

18. Finding the count of values

```
import pandas as pd
df=pd.read_csv("Pandadata.csv")
print(df)

#counts the number of same rows in a dataframe
count=df.value_counts()
print(count)
```

```
ROLL NO  NAME  THEORY  PRACTICALS
2        Bb    98      10           2
7        Gg    97      10           2
9        Ii    90      10           2
1        Aa    97      10           1
3        Cc    93      0            1
4        Dd    94      10           1
5        Ee    92      10           1
6        Ff    98      0            1
8        Hh    93      0            1
10       Jj    100     10           1
11       Kk    96      10           1
12       Ll    95      0            1
13       Mm    100     10           1
14       Nn    91      10           1
15       Oo    92      10           1
dtype: int64
```

```
#Specify the column name to count the number of values used in the column
count1=df.PRACTICALS.value_counts()
print(count1)
```

```
10    14
0      4
Name: PRACTICALS, dtype: int64
```

19. Renaming the columns

```
import pandas as pd
df=pd.read_csv("Pandadata.csv")

#Can rename multiple columns at once
df=df.rename(columns={'ROLL NO':'REGISTER NO', 'PRACTICALS':'INTERNALS'})
print(df)
```

	REGISTER NO	NAME	THEORY	INTERNALS
0	1	Aa	97	10
1	2	Bb	98	10
2	3	Cc	93	0
3	4	Dd	94	10
4	5	Ee	92	10
5	6	Ff	98	0
6	7	Gg	97	10
7	8	Hh	93	0
8	9	Ii	90	10
9	10	Jj	100	10
10	11	Kk	96	10
11	12	Ll	95	0
12	13	Mm	100	10
13	14	Nn	91	10
14	15	Oo	92	10
15	7	Gg	97	10
16	9	Ii	90	10
17	2	Bb	98	10

MODULE 5: DATA CLEANING, PREPARATION AND VISUALIZATION

1. Fill functions in a csv file

```
import pandas as pd
import numpy as np

df=pd.read_csv("dsdata.csv")
print(df)
```

	Sl No	Name	Physics	Maths	Python
0	1	Aa	95	85.0	78.0
1	2	Bb	98	83.0	NaN
2	3	Cc	28	83.0	73.0
3	4	Dd	90	81.0	NaN
4	5	Ee	93	89.0	71.0
5	6	NaN	94	NaN	79.0
6	7	Gg	90	95.0	90.0
7	8	Hh	92	NaN	85.0
8	9	NaN	91	100.0	95.0
9	10	Jj	93	88.0	93.0

```
#Fills the null data with specified values
print(df.fillna(85))
```

	Sl No	Name	Physics	Maths	Python
0	1	Aa	95	85.0	78.0
1	2	Bb	98	83.0	85.0
2	3	Cc	28	83.0	73.0
3	4	Dd	90	81.0	85.0
4	5	Ee	93	89.0	71.0
5	6	85	94	85.0	79.0
6	7	Gg	90	95.0	90.0
7	8	Hh	92	85.0	85.0
8	9	85	91	100.0	95.0
9	10	Jj	93	88.0	93.0

#bfill-backfill(Fills the null value with the value after it)

print(df.fillna(method='bfill'))

	Sl No	Name	Physics	Maths	Python
0	1	Aa	95	85.0	78.0
1	2	Bb	98	83.0	73.0
2	3	Cc	28	83.0	73.0
3	4	Dd	90	81.0	71.0
4	5	Ee	93	89.0	71.0
5	6	Gg	94	95.0	79.0
6	7	Gg	90	95.0	90.0
7	8	Hh	92	100.0	85.0
8	9	Jj	91	100.0	95.0
9	10	Jj	93	88.0	93.0

#ffill-forwardfill(Fills the null values with the value before it)

print(df.fillna(method='ffill'))

	Sl No	Name	Physics	Maths	Python
0	1	Aa	95	85.0	78.0
1	2	Bb	98	83.0	78.0
2	3	Cc	28	83.0	73.0
3	4	Dd	90	81.0	73.0
4	5	Ee	93	89.0	71.0
5	6	Ee	94	89.0	79.0
6	7	Gg	90	95.0	90.0
7	8	Hh	92	95.0	85.0
8	9	Hh	91	100.0	95.0
9	10	Jj	93	88.0	93.0

```
newdata=df.fillna(method='pad') #Same as ffill
#Storing it as a seperate dataframe
print(newdata)
```

	Sl No	Name	Physics	Maths	Python
0	1	Aa	95	85.0	78.0
1	2	Bb	98	83.0	78.0
2	3	Cc	28	83.0	73.0
3	4	Dd	90	81.0	73.0
4	5	Ee	93	89.0	71.0
5	6	Ee	94	89.0	79.0
6	7	Gg	90	95.0	90.0
7	8	Hh	92	95.0	85.0
8	9	Hh	91	100.0	95.0
9	10	Jj	93	88.0	93.0

```
#Saving the dataframe as a new excel file
newdata.to_excel("newdsdata.xlsx")
```

2. Filtering null functions

```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv("dsdata.csv")
print(df)
```

	Sl No	Name	Physics	Maths	Python
0	1	Aa	95	85.0	78.0
1	2	Bb	98	83.0	NaN
2	3	Cc	28	83.0	73.0
3	4	Dd	90	81.0	NaN
4	5	Ee	93	89.0	71.0
5	6	NaN	94	NaN	79.0
6	7	Gg	90	95.0	90.0
7	8	Hh	92	NaN	85.0
8	9	NaN	91	100.0	95.0
9	10	Jj	93	88.0	93.0

```
#Filters and prints the dataframe where physics has null values
Physics_null=pd.isnull(df["Physics"])
print(df[Physics_null])
```

```
Empty DataFrame
Columns: [Sl No, Name, Physics, Maths, Python]
Index: []
```

```
#Filters and prints the dataframe where Maths has no null values
Maths_null=pd.notnull(df["Maths"])
print(df[Maths_null])
```

	Sl No	Name	Physics	Maths	Python
0	1	Aa	95	85.0	78.0
1	2	Bb	98	83.0	NaN
2	3	Cc	28	83.0	73.0
3	4	Dd	90	81.0	NaN
4	5	Ee	93	89.0	71.0
6	7	Gg	90	95.0	90.0
8	9	NaN	91	100.0	95.0
9	10	Jj	93	88.0	93.0

3. Map functions

```
import pandas as pd
import numpy as np

s=pd.Series(['Cat','Dog',np.nan,'Rabbit'])
print(s)
```

```
0       Cat
1       Dog
2       NaN
3    Rabbit
dtype: object
```

```
print(s.map({'Cat':'Kitten','Dog':'Pupp'}))
```

```
0    Kitten
1      Pupp
2       NaN
3       NaN
```

```
print(s.map('I am a {}'.format))
```

```
0       I am a Cat
1       I am a Dog
2       I am a nan
3    I am a Rabbit
dtype: object
```

```
print(s.map('I am a {}'.format, na_action='ignore'))
```

```
0       I am a Cat
1       I am a Dog
2              NaN
3    I am a Rabbit
dtype: object
```

4. Null and nill functions

```
import pandas as pd

import numpy as np


dict={'First Score':[100,90,np.nan,95],

    'Second Score':[30,45,56,np.nan],
```

```
    'Third Score':[np.nan,40,80,90]}

df=pd.DataFrame(dict)
print(df)

print(df.isnull())  #Prints true for null values

print(df.notnull()) #Prints true for non null values

print(df.fillna(5)) #Fills the null values with the specified number

#bfill-backfill(Fills the null value with the value after it)
print(df.fillna(method='bfill'))

#ffill-forwardfill(Fills the null values with the value before it)
print(df.fillna(method='ffill'))

print(df.fillna(method='pad')) #Same as ffill
```

5. Apply and apply map functions

```
import pandas as pd
import numpy as np

df=pd.DataFrame({'A':[1,2,3,4],
                 'B':[10,20,30,40],
                 'C':[20,40,60,80]},
                index=['Row1','Row2','Row3','Row4'])
print(df)
```

```
#apply() function is used to call a function in pandas to implement something on specified rows and columns
#Sum of rows
def custom_sum(row):
   return row.sum()
df['D']=df.apply(custom_sum, axis=1)
print(df)

#Sum of columns
df.loc['Row5']=df.apply(custom_sum, axis=0)
print(df)
#loc is used for columns

#Performing operations on specified column
def multiply_by_2(val):
   return val*2
df['D']=df['C'].apply(multiply_by_2)
print(df)

df['D']=df['B'].apply(multiply_by_2)+df['C'].apply(multiply_by_2)
print(df)

#applymap() is used to implement something to the entire dataframe
print(df.applymap(np.square))
```

6. Creating a line graph

```
import matplotlib.pyplot as plt

Year=[2003,2004,2005,2006,2007,2008,2009,2010]
Grade=[8.5,8.7,8.8,9.2,9.3,9.9,9.4,9.3]
```

```
plt.plot(Year, Grade)
plt.title('Year Vs Grade')
plt.xlabel('Year')
plt.ylabel('Grade')
plt.show()
```

7. Skew functions

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df=pd.read_csv("dsdata.csv")
print(df.shape)
print(df.info())
df.describe()

print(df['Physics'].skew())
print(df['Physics'].describe())

#Make sure mute inline plotting option in plots is unchecked
plt.boxplot(df["Physics"])
plt.show()

#Histogram
df.Physics.hist()

#Scatter plot
```

```
fig,ax=plt.subplots(figsize=(12,6))
ax.scatter(df['Physics'], df['Maths'])
ax.set_xlabel('Physics Mark')
ax.set_ylabel('Maths Mark')
plt.show()

#Drops all the values that are less than 0 and greater than 100
index=df[(df['Maths']>100)|(df['Maths']<0)].index
df.drop(index, inplace=True)
df['Maths'].describe()
```

8. String functions

```
import pandas as pd

#Displaying the csv file as Series
data=pd.read_csv('exercisedata.csv')
sr=data[data.columns[0]]   #Taking the first column alone

print(sr)

print(sr.str.capitalize())

print(sr.str.lower())

print(sr.str.len())

print(sr.str.split())

print(sr.str.slice(0,3))
```

```
print(sr.str.startswith('A'))


print(sr.str.endswith('e'))
```