2)**Working with arrays**

a. Create a 1D array

```
import numpy as np
x=np.array([1,2,3])
print(x)
```

```
[1 2 3]
```

b. Create a boolean array

```
x=([56,22,'hi',0,False])
y=np.array(x,dtype=bool)
print(y)
```

```
[ True  True  True False False]
```

c. Extract items that satisfy a given condition from 1D array

```
x=np.array([1,2,3,4,5,6,7,8,9])
y=np.extract(x>4,x)
print(y)
```

```
[5 6 7 8 9]
```

d. Replace items that satisfy a condition with another value in numpy array

```
x=np.array([1,2,3,4,5,6])
y=np.where(x>2,111,x)
print(y)
```

```
[  1   2 111 111 111 111]
```

e. Replace items that satisfy a condition without affecting the original array

```
x=np.arange(16)
y=np.where(x%2==0,111,x)
print(x)
print(y)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
[111   1 111   3 111   5 111   7 111   9 111  11 111  13 111  15]
```

f. Reshape an array

```
x=np.arange(25)
print(x)
print(x.reshape(5,5))
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24]
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

g. Extract all numbers between a given range from a numpy array

```
x=np.arange(25)
y=np.where((x>=10)&(x<=22))
print(x)
print(y)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24]
(array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]),)
```

3)**Multiple arrays**

a. Stack two arrays vertically

```
x=np.array([[1,2,3],[4,5,6]])
y=np.array([[11,22,33,],[44,55,66]])
z=np.vstack((x,y))
print(z)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [11 22 33]
 [44 55 66]]
```

b. Stack two arrays horizontally

```
x=np.array([[1,2,3],[4,5,6]])
y=np.array([[11,22,33,],[44,55,66]])
z=np.hstack((x,y))
print(z)
```

```
[[ 1  2  3 11 22 33]
 [ 4  5  6 44 55 66]]
```

c. Get the common items between two python numpy arrays

```
x=np.array([[1,2,3],[4,5,6]])
y=np.array([[1,2,3],[44,55,66]])
z=np.intersect1d(x,y)
print(z)
```

```
[1 2 3]
```

d. Remove from one array those items that exist in another

```
x=np.array([[1,2,3],[4,5,6]])
y=np.array([[1,2,3],[44,55,66]])
z=np.setdiff1d(x,y)
print(z)
```

```
[4 5 6]
```

e. Get the positions where elements of two arrays match

```
x=np.array([1,2,3,4,5,6])
y=np.array([1,2,3,44,55,66])
z=np.where(x==y)
print(z)
```

```
(array([0, 1, 2]),)
```

4)**Multi-dimensional arrays**

a. Convert an array of arrays into a flat 1d array

```
x=np.array([[1,2,3],[4,5,6]])
y=np.reshape(x,(1,-1))
print(y)
```

```
[[1 2 3 4 5 6]]
```

b. Swap two columns in a 2d numpy array

```
x=np.arange(9).reshape(3,3)
print(x)
x[:,[2,1,0]]
```

```
[[1 2 3]
 [4 5 6]
```

```
 [7 8 9]]
array([2, 5, 8])
```

5)**Statistical analysis**

a. Compute the mean, median, standard deviation of a numpy array

```
x=np.array([1,2,3,4,5,6,7,8,9])
mean=np.mean(x)
median=np.median(x)
std=np.std(x)
print('Mean:',mean)
print('Median:',median)
print('Std:',std)
```

```
Mean: 5.0
Median: 5.0
Std: 2.581988897471611
```

b. Find the percentile scores of a numpy array

```
x=np.arange(101)
perc=np.percentile(x,q=[50,75])
print(perc)
```

```
[50. 75.]
```

c. compute the euclidean distance between two arrays

```
x=np.array([1,2,3,4,5,6])
y=np.array([6,5,4,3,2,1])
dist=np.linalg.norm(x-y)
print(dist)
```

```
8.366600265340756
```

d. Find the correlation between two columns of a numpy array

```
import random
x=np.random.randint(0,100,500)
y=x+np.random.randint(0,100,500)
corr=np.corrcoef(x,y)
print(corr)
```

```
[[1.         0.70114853]
 [0.70114853 1.        ]]
```

e. Probabilistic sampling in numpy

```
pop=np.array(['a','b','c','d','e'])
prb=np.array([0.2,0.1,0.3,0.4,0.5])
prb=prb/prb.sum()
sample=np.random.choice(pop,size=3,p=prb)
print(sample)
```

```
['d' 'a' 'e']
```

f. compute the moving average of a numpy array

```
x=np.random.randint(10,size=10)
movavg=np.convolve(x,np.ones(3)/3,mode='valid')
print(x)
print(movavg)
```

```
[6 1 9 3 4 6 3 0 0 8]
[5.33333333 4.33333333 5.33333333 4.33333333 4.33333333 3.
 1.         2.66666667]
```

6)**Data Cleaning**

a. Find the position of missing values in numpy array

```
x=np.array([[1,2,3],[4,5,np.nan]])
np.argwhere(np.isnan(x))
```

```
array([[1, 2]])
```

b. Drop rows that contain a missing value from a numpy array

```
x=np.array([[1,2,3],[4,5,np.nan]])
x[~np.isnan(x).any(axis=1),:]
```

```
array([[1., 2., 3.]])
```

c. Replace all missing values with 0 in a numpy array

```
x=np.array([[1,2,3],[4,5,np.nan]])
x[np.isnan(x)]=0
```

output will be shown as : array([[1., 2., 3.,4.,5.,0.]])

d. Drop all missing values from a numpy array

```
x=np.array([[1,2,3],[4,5,np.nan]])
x[~np.isnan(x)]
```

```
array([1., 2., 3., 4., 5.])
```

7)**Data Transformation**

a. Normalize an array so the values range exactly between 0 and 1

```
x=np.array([1,2,3,4,5,6])
xnorm=(x-np.min(x))/(np.max(x)-np.min(x))
print(xnorm)
```

```
[0.  0.2 0.4 0.6 0.8 1. ]
```

b. Compute the min-by-max for each row for a numpy array 2d

```
import random
x=np.random.randint(16,size=12)
x=x.reshape(4,3)
print(x)
x=x.min(axis=1)/x.max(axis=1)
print(x)
```

```
[[ 6  9  3]
 [ 7  9  5]
 [10  1  5]
 [11  6  4]]
[0.33333333 0.55555556 0.1        0.36363636]
```

8)**Pandas Basics**

a. Installing Pandas: **pip install pandas**

For importing Pandas: **import pandas as pd**

b. To check whether pandas is installed: **pip show pandas**

To know version of pandas: **import pandas as pd**

print(pd._ *version* _)

c. Create a series from a list, numpy array and dict

```
import pandas as pd
x=pd.Series([1,2,3])
print(x)
```

```
0    1
1    2
2    3
dtype: int64
```

```
import numpy as np
import pandas as pd
x=np.array([1,2,3,4,5,6,7,8,9])
y=pd.Series(x)
print(y)
```

```
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
8    9
dtype: int64
```

```
x={'Mateen':111,'Faizan':88,'Khaja':85,'Yousuf':70,'Bilal':69,'Turab':103,'Hamza':76}
y=pd.Series(x)
print(y)
```

```
Mateen    111
Faizan     88
Khaja      85
Yousuf     70
Bilal      69
Turab     103
Hamza      76
dtype: int64
```

d. Convert the index of a series into a column of a dataframe

```
x={'Mateen':111,'Faizan':88,'Khaja':85,'Yousuf':70,'Bilal':69,'Turab':103,'Hamza':76}
y=pd.Series(x)
df=y.to_frame().reset_index()
df
```

	index	0
0	Mateen	111
1	Faizan	88
2	Khaja	85
3	Yousuf	70
4	Bilal	69
5	Turab	103
6	Hamza	76

e. Combine many series to form a dataframe

```
x=pd.Series(['apple','mango','orange','pineapple'],name='Fruits')
y=pd.Series(['ladyfinger','tomatoes','potatoes','onions'],name='Vegetables')
z=pd.Series(['meat','eggs','fish','red meat'],name='Protein foods')
df=pd.concat([x,y,z],axis=1)
df
```

	Fruits	Vegetables	Protein foods
0	apple	ladyfinger	meat
1	mango	tomatoes	eggs
2	orange	potatoes	fish
3	pineapple	onions	red meat

9)**Statistical analysis in pandas**

a. Get the minimum, 25th percentile, median, 75th, and max of a numeric series

```
# Sample series data
data=pd.Series(np.random.randint(100, size=20))
minimum = np.min(data)
percentile_25 = np.percentile(data, 25)
median = np.median(data)
percentile_75 = np.percentile(data, 75)
maximum = np.max(data)
print("Minimum:", minimum)
print("25th percentile:", percentile_25)
print("Median:", median)
print("75th percentile:", percentile_75)
print("Maximum:", maximum)
```

```
Minimum: 2
25th percentile: 31.25
Median: 51.0
75th percentile: 59.75
Maximum: 87
```

b. Get frequency counts of unique items of a series

```
series=pd.Series(np.random.randint(10,size=20))
print('Series Generated:')
print(series)
print('Frequency Count:')
counts=series.value_counts()
print(counts)
```

```
Series Generated:
0     7
1     9
2     3
3     9
4     3
5     8
6     2
7     2
8     9
9     8
10    9
11    4
12    2
13    2
14    8
15    9
16    4
17    2
18    6
19    9
dtype: int64
Frequency Count:
9    6
2    5
8    3
3    2
4    2
7    1
6    1
dtype: int64
```

c. Bin a numeric series to 10 groups of equal size

```
age=pd.Series(np.random.randint(100,size=10),name='age')
bin=[0,18,25,35,55,75,100]
lab=['teen','young','adult','mid-adult','old','senior-citezen']
b=pd.cut(age,bins=bin,labels=lab)
df=pd.concat([age,b],axis=1)
df.columns=('age','age-group')
df
```

	age	age-group
0	14	teen
1	9	teen
2	78	senior-citezen
3	76	senior-citezen
4	2	teen
5	53	mid-adult
6	62	old

d. Compute the euclidean distance between two series

```
a=pd.Series([1,2,3,4,5,6])
b=pd.Series([9,8,7,6,5,6])
c=np.linalg.norm(a-b)
print(c)
```

```
10.954451150103322
```

10)**Data Preparation in pandas**

a. Normalize all columns in a dataframe

```
df=pd.DataFrame({'col1':[1,2,3],'col2':[3,4,5]})
xnorm=(df-df.min())/(df.max()-df.min())
xnorm
```

	col1	col2
0	0.0	0.0
1	0.5	0.5
2	1.0	1.0

b. Compute the correlation of each row with the suceeding row

```
import pandas as pd

# Sample DataFrame
data = {
    'A': [1, 2, 3, 4, 5],
    'B': [2, 4, 6, 8, 10],
    'C': [3, 6, 9, 12, 15]
}
df = pd.DataFrame(data)

# Compute the correlation of each row with the succeeding row
correlation_with_next_row = df.corrwith(df.shift(-1))

# Print the correlation values
print(correlation_with_next_row)
```

```
A    1.0
B    1.0
C    1.0
dtype: float64
```

c. Compute the autocorrelations of a numeric series

```
import pandas as pd
import numpy as np

# Create a numeric series
num_series = pd.Series(np.random.randint(1, 100, 100))

# Compute the autocorrelation with lag 1
auto_correlation = num_series.autocorr()

# Print the result
```

```
print("Auto-correlation with lag 1:", auto_correlation)
```

```
Auto-correlation with lag 1: -0.02803450361074562
```

ADDITIONAL PROGRAMS

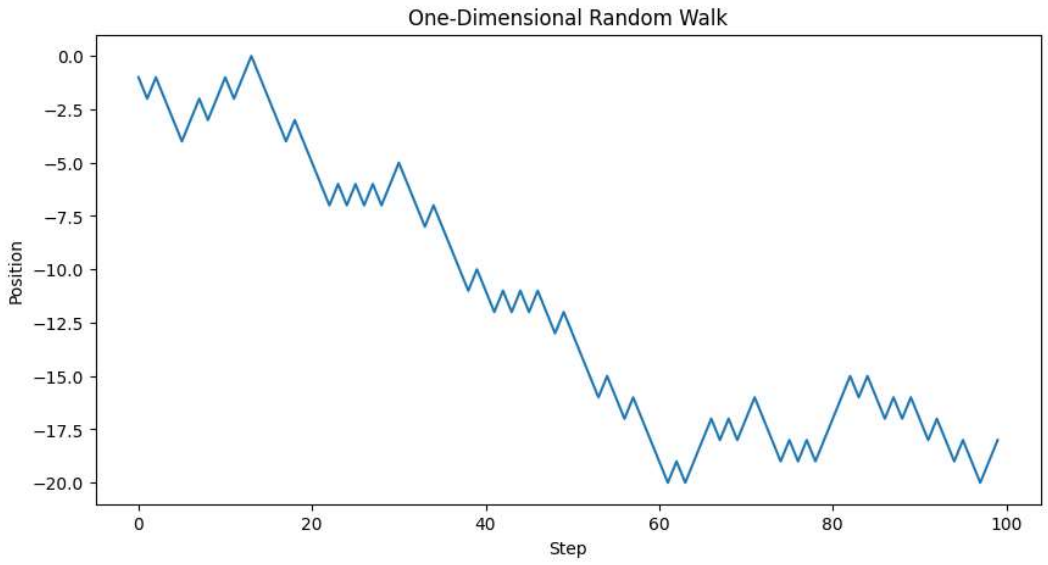
```
import numpy as np
import matplotlib.pyplot as plt

# Number of steps in the random walk
num_steps = 100

# Generate random steps (+1 or -1)
steps = np.random.choice([-1, 1], size=num_steps)

# Compute cumulative sum to get the walker's position at each step
position = np.cumsum(steps)

# Plot the random walk
plt.figure(figsize=(10, 5))
plt.plot(position)
plt.title("One-Dimensional Random Walk")
plt.xlabel("Step")
plt.ylabel("Position")
plt.show()
```




```
import numpy as np

matrix1 = np.array([[1, 2],
                    [3, 4]])
matrix2 = np.array([[5, 6],
                    [7, 8]])

result = np.dot(matrix1, matrix2)
print(result)
```

```
[[19 22]
 [43 50]]
```

```
import numpy as np

matrix = np.array([[1, -1],
                   [1,  1]])
eigenvectors = np.linalg.eig(matrix)
print(eigenvectors)
```

```
(array([1.+1.j, 1.-1.j]), array([[0.70710678+0.j        , 0.70710678-0.j        ],
       [0.        -0.70710678j, 0.        +0.70710678j]]))
```

```
import numpy as np

arr = np.array([1, 2, 3, 4])
result = np.square(arr)
print(result)
```

```
[ 1  4  9 16]
```

```
double = lambda x: x * 2
result = double(5)  # Calls the lambda function
print(result)
```

```
10
```

```
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [25, 30, 22, 28],
        'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']}

df = pd.DataFrame(data)
print(df)
```

```
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   22      Chicago
3    David   28      Houston
```

```
data={'A':[1,2,3,4,5],'B':[10,20,30,40,50],'C':[100,200,300,400,500]}
df=pd.DataFrame(data,index=['a','b','c','d','e'])
print(df)
print(df.loc['b','B'])
print(df.iloc[1,1])
```

```
   A   B    C
a  1  10  100
b  2  20  200
c  3  30  300
d  4  40  400
e  5  50  500
20
20
```

```
import pandas as pd

data = {'A': [1, 2, 3, 4, 5],
        'B': [10, 20, 30, 40, 50]}
df = pd.DataFrame(data)

summary = df.describe()
print(summary)
```

```
              A          B
count  5.000000   5.000000
mean   3.000000  30.000000
std    1.581139  15.811388
min    1.000000  10.000000
25%    2.000000  20.000000
50%    3.000000  30.000000
75%    4.000000  40.000000
max    5.000000  50.000000
```

```
import pandas as pd

df = pd.DataFrame({'A': [1, 2, None, 4], 'B': [5, None, 7, 8]})
df.dropna()  # Remove rows with missing values
df.dropna(axis=1)  # Remove columns with missing values
```

```
0

1

2

3
```

```
df.fillna(0)  # Replace missing values with 0
```

	A	B
0	1.0	5.0
1	2.0	0.0
2	0.0	7.0
3	4.0	8.0

```
import pandas as pd

# Sample data: ages of individuals
data = {'Age': [25, 18, 30, 42, 38, 20, 28, 55, 60, 22, 29, 35, 40, 19]}
df = pd.DataFrame(data)

# Define bin edges and labels
bin_edges = [18, 25, 35, 45, 60]
bin_labels = ['18-24', '25-34', '35-44', '45-60']

# Create a new column with the age bins
df['Age Group'] = pd.cut(df['Age'], bins=bin_edges, labels=bin_labels)

# Display the transformed dataframe
print(df)
```

```
    Age Age Group
0    25     18-24
1    18       NaN
2    30     25-34
3    42     35-44
4    38     35-44
5    20     18-24
6    28     25-34
7    55     45-60
8    60     45-60
9    22     18-24
10   29     25-34
11   35     25-34
12   40     35-44
13   19     18-24
```