Dr AQ khan Institute of Computer Science and Information

Lab no.2

Artificial Intelligence (Lab)

Submitted To:

Engr.Zia Ur Rehman

Submitted By:

Muhammad Abul Hassan

Registration No:

202101004

Department:

BSCE

Semester:

7

Date:

October 2, 2023

Lab Rubrics:

Turn in on time	Schematic/Simulation	Presentation	Coding	Outcome	Total Marks
4	4	4	4	4	20

Artificial Intelligence Lab 02

Muhammad Abul Hassan

BSCE-7

(Numpy, Pandas, Matplotlib)

01. Numpy

Example 01

- Take 2 lists and multiply both you'll see that error occurs repeat the process but by coverting them toarray by numpy.array()

In [1]:

```
l1 = [4,4,4]
l2 = [5,5,5]

l1*l2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[1], line 4
      1 l1 = [4,4,4]
      2 l2 = [5,5,5]
----> 4 l1*l2

TypeError: can't multiply sequence by non-int of type 'list'
```

- Now, by using numpy

In [4]:

```
import numpy as np

l1 = [1,2,3]
l2 = [4,5,6]

A1 = np.array(l1)
A2 = np.array(l2)

print(f"{A1} * {A2} = {A1*A2}")
```

```
[1 2 3] * [4 5 6] = [ 4 10 18]
```

Example 02

- Demonstrate the use of numpy.dtype and numpy.shape() functions

In [5]:

```
import numpy as np

l1 = [1,2,3]
l2 = [4,5,6]

A1 = np.array(l1)
A2 = np.array(l2)

A = A1*A2

print(f"{A1} * {A2} = {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")
```

```
[1 2 3] * [4 5 6] = [ 4 10 18]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: int32
The dimension of an array: (3,)
```

Example 03

- The size of an array created with numpy.array() is int32 convert it to int 8

In [6]:

```
import numpy as np

l1 = [1,2,3]
l2 = [4,5,6]

A1 = np.array(l1, np.int8)
A2 = np.array(l2, np.int8)

A = A1*A2

print(f"{A1} * {A2} = {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")
```

```
[1 2 3] * [4 5 6] = [ 4 10 18]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: int8
The dimension of an array: (3,)
```

Example 04

- Demonstrate the use of numpy.size() functions

In [8]:

```
import numpy as np

l1 = [1,2,3]
l2 = [4,5,6]

A1 = np.array(l1, np.int8)
A2 = np.array(l2, np.int8)

A = A1*A2

print(f"{A1} * {A2} = {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
[1 2 3] * [4 5 6] = [ 4 10 18]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: int8
The dimension of an array: (3,)
The size of an array: 3
```

Example 05

- Create a 2D array using numpy.array()

In [13]:

```
import numpy as np

l1 = [1,2,3]
l2 = [4,5,6]

A = np.array((l1, l2))
print(f" The 2D array is : \n {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
 The 2D array is :
 [[1 2 3]
 [4 5 6]]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: int32
The dimension of an array: (2, 3)
The size of an array: 6
```

Example 06

- Create a 1 D array by passing a list

In [15]:

```python
A = np.array(([1,2,3,4,5]))
print(f" The 1D array is : \n {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
 The 1D array is : 
 [1 2 3 4 5]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: int32
The dimension of an array: (5,)
The size of an array: 5
```

Example 07

- Create a 2 D array by passing lists

In [18]:

```python
import numpy as np

A  = np.array(([1,2,3,4,5],  [2,3,4,5,6]))
print(f" The 2D array is : \n {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
 The 2D array is : 
 [[1 2 3 4 5]
 [2 3 4 5 6]]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: int32
The dimension of an array: (2, 5)
The size of an array: 10
```

Example 08

- Create 4 x 4 Matrix

In [21]:

```python
import numpy as np

r1  = [1,2,3,4]
r2  = [3,6,3,4]
r3  = [1,2,9,4]
r4  = [1,4,5,4]

A = np.array((r1,r2,r3,r4))
print(f" The array is : \n {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a
print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
 The array is :
 [[1 2 3 4]
 [3 6 3 4]
 [1 2 9 4]
 [1 4 5 4]]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: int32
The dimension of an array: (4, 4)
The size of an array: 16
```

Example 09

- Replace 2nd row 3rd element of above 4x4 matrix with 10

In [23]:

```
import numpy as np

r1 = [1,2,3,4]
r2 = [3,6,3,4]
r3 = [1,2,9,4]
r4 = [1,4,5,4]

A = np.array((r1,r2,r3,r4))
print(f" The original array is : \n {A}")

A[1,2] = 10
print(f" The array after replacing : \n {A}")
```

```
 The original array is :
 [[1 2 3 4]
 [3 6 3 4]
 [1 2 9 4]
 [1 4 5 4]]
 The array after replacing :
 [[ 1  2  3  4]
 [ 3  6 10  4]
 [ 1  2  9  4]
 [ 1  4  5  4]]
```

Example 10

- Create a 5 x 5 matrix of all zeros by setting values of both rows and column

In [24]:

```
import numpy as np

A = np.zeros([5,5])
print(f" The array is : \n {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
 The array is : 
 [[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: float64
The dimension of an array: (5, 5)
The size of an array: 25
```

Example 11

- Create a 5 x 5 matrix of all zeros by passing only 1 argument

In [29]:

```python
import numpy as np

A = np.zeros([5])
print(f" The array is : \n {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
 The array is : 
 [0. 0. 0. 0. 0.]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: float64
The dimension of an array: (5,)
The size of an array: 5
```

Example 12

- Create an array from 1 to 100 by numpy.arrange()

In [32]:

```python
import numpy as np

A = np.arange(1,100)
print(f" The array is : \n {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
 The array is : 
 [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96
 97 98 99]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: int32
The dimension of an array: (99,)
The size of an array: 99
```

Example 13

- Create an array from 1 to 100 by numpy.arrange() with a stepsize of 10

In [33]:

```python
import numpy as np

A = np.arange(1,100,10)
print(f" The array is : \n {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
 The array is : 
 [ 1 11 21 31 41 51 61 71 81 91]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: int32
The dimension of an array: (10,)
The size of an array: 10
```

Example 14

- Create an array of 100 elements ranging from 2 to 3

In [34]:

```python
import numpy as np

A = np.linspace(2,3,100)
print(f" The array is : \n {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
 The array is :
 [2.         2.01010101 2.02020202 2.03030303 2.04040404 2.05050505
 2.06060606 2.07070707 2.08080808 2.09090909 2.1010101  2.11111111
 2.12121212 2.13131313 2.14141414 2.15151515 2.16161616 2.17171717
 2.18181818 2.19191919 2.2020202  2.21212121 2.22222222 2.23232323
 2.24242424 2.25252525 2.26262626 2.27272727 2.28282828 2.29292929
 2.3030303  2.31313131 2.32323232 2.33333333 2.34343434 2.35353535
 2.36363636 2.37373737 2.38383838 2.39393939 2.4040404  2.41414141
 2.42424242 2.43434343 2.44444444 2.45454545 2.46464646 2.47474747
 2.48484848 2.49494949 2.50505051 2.51515152 2.52525253 2.53535354
 2.54545455 2.55555556 2.56565657 2.57575758 2.58585859 2.5959596
 2.60606061 2.61616162 2.62626263 2.63636364 2.64646465 2.65656566
 2.66666667 2.67676768 2.68686869 2.6969697  2.70707071 2.71717172
 2.72727273 2.73737374 2.74747475 2.75757576 2.76767677 2.77777778
 2.78787879 2.7979798  2.80808081 2.81818182 2.82828283 2.83838384
 2.84848485 2.85858586 2.86868687 2.87878788 2.88888889 2.8989899
 2.90909091 2.91919192 2.92929293 2.93939394 2.94949495 2.95959596
 2.96969697 2.97979798 2.98989899 3.        ]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: float64
The dimension of an array: (100,)
The size of an array: 100
```

Example 15

- Create identity matrix

In [35]:

```python
import numpy as np

A = np.identity(5)
print(f" The array is : \n {A}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
 The array is :
 [[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: float64
The dimension of an array: (5, 5)
The size of an array: 25
```

Example 16

- Create a 4 x 4 matrix and find the sum of all columns

In [37]:

```python
import numpy as np

r1 = [1,2,3,4]
```

```
r2 = [3,6,3,4]
r3 = [1,2,9,4]
r4 = [1,4,5,4]

A = np.array((r1,r2,r3,r4))
print(f" The Matrix is : \n {A}")

print(f" The row wise sum is : {A.sum(axis=1)}")
print(f" The column wise sum is : {A.sum(axis=0)}")

print(f"The type of array using type: {type(A)}")
print(f"The type of array using dtype: {A.dtype}") # no () with dtype because it is a

print(f"The dimension of an array: {A.shape}")

print(f"The size of an array: {A.size}") # The size attrubute counts the total elemen
```

```
 The Matrix is :
 [[1 2 3 4]
 [3 6 3 4]
 [1 2 9 4]
 [1 4 5 4]]
 The row wise sum is : [10 16 16 14]
 The column wise sum is : [ 6 14 20 16]
The type of array using type: <class 'numpy.ndarray'>
The type of array using dtype: int32
The dimension of an array: (4, 4)
The size of an array: 16
```

Example 17

- Find the transpose of a Matrix

In [38]:

```
import numpy as np

r1 = [1,2,3,4]
r2 = [3,6,3,4]
r3 = [1,2,9,4]
r4 = [1,4,5,4]

A = np.array((r1,r2,r3,r4))
print(f" The Matrix is : \n {A}")

print(f" The transpose is : \n {A.T}")
```

```
 The Matrix is :
 [[1 2 3 4]
 [3 6 3 4]
 [1 2 9 4]
 [1 4 5 4]]
 The transpose is :
 [[1 3 1 1]
 [2 6 2 4]
 [3 3 9 5]
 [4 4 4 4]]
```

Example 18

- Use reshape command to convrt 4 x 4 matrix to 8 x 2

In [41]:
```
import numpy as np

r1 = [1,2,3,4]
r2 = [3,6,3,4]
r3 = [1,2,9,4]
r4 = [1,4,5,4]

A = np.array((r1,r2,r3,r4))
print(f" The 4x4 Matrix is : \n {A}")

print(f" The 8x2 matrix: \n {A.reshape(8,2)}")
```

```
The 4x4 Matrix is :
[[1 2 3 4]
[3 6 3 4]
[1 2 9 4]
[1 4 5 4]]
The 8x2 matrix:
[[1 2]
[3 4]
[3 6]
[3 4]
[1 2]
[9 4]
[1 4]
[5 4]]
```

Example 19

- Demonstrate the use of numpy.ravel()

In [43]:
```
import numpy as np

r1 = [1,2,3,4]
r2 = [3,6,3,4]
r3 = [1,2,9,4]
r4 = [1,4,5,4]

A = np.array((r1,r2,r3,r4))
print(f" The 4x4 Matrix is : \n {A}")

print(f" The 1D array from above matrix using ravel: \n {A.ravel()}")
```

```
The 4x4 Matrix is :
[[1 2 3 4]
[3 6 3 4]
[1 2 9 4]
[1 4 5 4]]
The 1D array from above matrix using ravel:
[1 2 3 4 3 6 3 4 1 2 9 4 1 4 5 4]
```

Example 20

- Demonstrate the use of argmax, argmin,argsort

In [44]:

```
import numpy as np

a = [1, 16, 31, 4]

A = np.array(a)
print(f"The original array: {A}")

print(f"The index of maximum value in array is: {A.argmax()}")
print(f"The index of minimum value in array is: {A.argmin()}")
print(f"Sorted Indexes: {A.argsort()}")
```

```
The original array: [ 1 16 31  4]
The index of maximum value in array is: 2
The index of minimum value in array is: 0
Sorted Indexes: [0 3 1 2]
```

Example 21

- Demostrate the use of numpy.full(),vstack(),hstack(),column_stack()

In [3]:

```
import numpy as np

f1=np.full((2,2),5)
f1
```

Out[3]:

```
array([[5, 5],
       [5, 5]])
```

In [4]:

```
f2 = np.full((2,2), 3)
f2
```

Out[4]:

```
array([[3, 3],
       [3, 3]])
```

In [6]:

```
a = np.vstack([f1, f2])
a
```

Out[6]:

```
array([[5, 5],
       [5, 5],
       [3, 3],
       [3, 3]])
```

In [7]:

```
b = np.hstack([f1, f2])
b
```

Out[7]:

```
array([[5, 5, 3, 3],
       [5, 5, 3, 3]])
```

In [8]:

```
a = np.column_stack([f1, f2])
a
```

Out[8]:

```
array([[5, 5, 3, 3],
       [5, 5, 3, 3]])
```

Example 22

- Save and load a matrix in the memory

In [9]:

```
import numpy as np

a = np.full((2,3), 5)
a
```

Out[9]:

```
array([[5, 5, 5],
       [5, 5, 5]])
```

In [10]:

```
np.save("untitled.npy",  a)
```

In [11]:

```
savedMatrix  =  np.load('untitled.npy')
savedMatrix
```

Out[11]:

```
array([[5, 5, 5],
       [5, 5, 5]])
```

Example 23

- Demonstrate the use of numoy.dot() and compare it with simple multiplication

In [12]:

```
import numpy as np

f1=np. full((2,2),5)
print("\nf1 = \n",f1)

f2=np.full((2,2),  3)
print("\nf2 = \n", f2)

print("point to point multiplication = ",f1*f2)

print("point to point multiplication = ", np.dot(f1,f2))
```

```
f1 =
 [[5 5]
 [5 5]]

f2 =
 [[3 3]
 [3 3]]
point to point multiplication = [[15 15]
 [15 15]]
point to point multiplication = [[30 30]
 [30 30]]
```

02. Pandas

Example 01

- `Create a Dictionary and convert them into data frames also check its datatype`

In [14]:

```
#create a dictionary

StuDict={"Name": ["Aqsa","Esha", "Ayesha", "Ayra", "Arfa", "Afsa", "Abdul", "Saadia",
"ID": ["SID-1","SID-2", "SID-3", "SID-4", "SID-5","SID-6", "SID-7", "SID-8", "SID-9",
"Rol1_no": [1,2,3,4,5,6,7,8,9,10],
"Semester"  :  [7,7,7,7,6,6,6,5,8,8]}

StuDict
```

Out[14]:

```
{'Name': ['Aqsa',
  'Esha',
  'Ayesha',
  'Ayra',
  'Arfa',
  'Afsa',
  'Abdul',
  'Saadia',
  'Abu Bakar',
  'Atif'],
 'ID': ['SID-1',
  'SID-2',
  'SID-3',
  'SID-4',
  'SID-5',
  'SID-6',
  'SID-7',
  'SID-8',
  'SID-9',
  'SID-10'],
 'Rol1_no': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
 'Semester': [7, 7, 7, 7, 6, 6, 6, 5, 8, 8]}
```

In [16]:

```
#convert into data frames

import pandas as pd

data=pd.DataFrame  (StuDict)

print(data)

print("\n\nThe data type of above given syntax is :",type (data))
```

```
        Name     ID  Rol1_no  Semester
0       Aqsa   SID-1        1         7
1       Esha   SID-2        2         7
2     Ayesha   SID-3        3         7
3       Ayra   SID-4        4         7
4       Arfa   SID-5        5         6
5       Afsa   SID-6        6         6
6      Abdul   SID-7        7         6
7     Saadia   SID-8        8         5
8  Abu Bakar   SID-9        9         8
9       Atif  SID-10       10         8


The data type of above given syntax is : <class 'pandas.core.frame.DataFrame'>
```

Example 02

- `Demonstrate the use of describe function for a data frame`

In [21]:

```
print(data.describe())
```

```
        Roll_no   Semester
count  10.00000  10.000000
mean    5.50000   6.700000
std     3.02765   0.948683
min     1.00000   5.000000
25%     3.25000   6.000000
50%     5.50000   7.000000
75%     7.75000   7.000000
max    10.00000   8.000000
```

Example 03

- `Demonstrate the use of head function for a data frame`

In [22]:
```
print(data.head())
```

```
     Name     ID  Roll_no  Semester
0    Aqsa  SID-1        1         7
1    Esha  SID-2        2         7
2  Ayesha  SID-3        3         7
3    Ayra  SID-4        4         7
4    Arfa  SID-5        5         6
```

Example 04

- `Demonstrate the use of tail function for a data frame`

In [23]:
```
print(data.tail())
```

```
        Name      ID  Roll_no  Semester
5       Afsa   SID-6        6         6
6      Abdul   SID-7        7         6
7     Saadia   SID-8        8         5
8  Abu Bakar   SID-9        9         8
9       Atif  SID-10       10         8
```

Example 05

- `Demonstrate the use of info function for a data frame`

In [24]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Name      10 non-null     object
 1   ID        10 non-null     object
 2   Roll_no   10 non-null     int64 
 3   Semester  10 non-null     int64 
dtypes: int64(2), object(2)
memory usage: 448.0+ bytes
None
```

Example 06

- Convert the data frame in a variable to CSV file

In [25]:
```
data.to_csv('student.csv')
```

Example 07

- Remove the indexes from the csv file

In [27]:
```
data.to_csv('Without_index.csv',  index=False)
```

Example 08

- Read from csv file

In [28]:
```
df  =  pd.read_csv('student.csv')
df
```

Out[28]:

	Unnamed: 0	Name	ID	Rol1_no	Semester
0	0	Aqsa	SID-1	1	7
1	1	Esha	SID-2	2	7
2	2	Ayesha	SID-3	3	7
3	3	Ayra	SID-4	4	7
4	4	Arfa	SID-5	5	6
5	5	Afsa	SID-6	6	6
6	6	Abdul	SID-7	7	6
7	7	Saadia	SID-8	8	5
8	8	Abu Bakar	SID-9	9	8
9	9	Atif	SID-10	10	8

Example 09

- Use describe,head,tail and info function for CSV file

In [1]:
```
import pandas as pd
df  =  pd.read_csv('student.csv')

print(f"Describe Function \n {df.describe()}, \n head Function \n {df.head()} \n tail
print(f"\n info Function \n {df.info()}")
```

```
Describe  Function
        Unnamed: 0   Roll_no   Semester
count    10.00000  10.00000  10.000000
mean      4.50000   5.50000   6.700000
std       3.02765   3.02765   0.948683
min       0.00000   1.00000   5.000000
25%       2.25000   3.25000   6.000000
50%       4.50000   5.50000   7.000000
75%       6.75000   7.75000   7.000000
max       9.00000  10.00000   8.000000,
 head Function
    Unnamed: 0    Name     ID  Roll_no  Semester
0           0    Aqsa  SID-1        1         7
1           1    Esha  SID-2        2         7
2           2  Ayesha  SID-3        3         7
3           3    Ayra  SID-4        4         7
4           4    Arfa  SID-5        5         6
 tail Function
    Unnamed: 0       Name      ID  Roll_no  Semester
5           5       Afsa   SID-6        6         6
6           6      Abdul   SID-7        7         6
7           7     Saadia   SID-8        8         5
8           8  Abu Bakar   SID-9        9         8
9           9       Atif  SID-10       10         8
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  10 non-null     int64 
 1   Name        10 non-null     object
 2   ID          10 non-null     object
 3   Roll_no     10 non-null     int64 
 4   Semester    10 non-null     int64 
dtypes: int64(3), object(2)
memory usage: 528.0+ bytes

 info Function
 None
```

Example 10

- `Access a column by its name`

In [2]:

```python
import pandas as pd

df['Name']
```

Out[2]:

```
0          Aqsa
1          Esha
2        Ayesha
3          Ayra
4          Arfa
5          Afsa
6         Abdul
7        Saadia
8     Abu Bakar
9          Atif
Name: Name, dtype: object
```

Example 11

- `Access the 1st element of a column`

In [3]:

```
df['Name'][0]
```

Out[3]:

```
'Aqsa'
```

Example 12

- `Update the value in the column`

In [4]:

```
df['Name'][0] = 'Saddam'
df
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_17156\2832195598.py:1:  SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df['Name'][0] = 'Saddam'
```

Out[4]:

	Unnamed: 0	Name	ID	Rol1_no	Semester
0	0	Saddam	SID-1	1	7
1	1	Esha	SID-2	2	7
2	2	Ayesha	SID-3	3	7
3	3	Ayra	SID-4	4	7
4	4	Arfa	SID-5	5	6
5	5	Afsa	SID-6	6	6
6	6	Abdul	SID-7	7	6
7	7	Saadia	SID-8	8	5
8	8	Abu Bakar	SID-9	9	8
9	9	Atif	SID-10	10	8

Example 13

- `Find the columns and indexes in a data frame`

```
In [5]: df.columns
```

```
Out[5]: Index(['Unnamed: 0', 'Name', 'ID', 'Rol1_no', 'Semester'], dtype='object')
```

```
In [6]: df.index
```

```
Out[6]: RangeIndex(start=0, stop=10, step=1)
```

Example 14

- `Create a series of 50 random numbers and check their data type and shape`

```
In [9]: import pandas as pd
        import numpy as np

        s = pd.Series(np.random.rand(50))
        print(s)
        print(f"Using dtype: {s.dtype}")
        print(f"Using type: {type(s)}")
        print(f"Using Shape: {s.shape}")
```

```
0     0.213985
1     0.705265
2     0.355021
3     0.522076
4     0.945126
5     0.889899
6     0.153129
7     0.923423
8     0.569371
9     0.036794
10    0.021892
11    0.968794
12    0.095629
13    0.690430
14    0.658842
15    0.227521
16    0.239709
17    0.993304
18    0.864950
19    0.362236
20    0.439025
21    0.898234
22    0.198995
23    0.713161
24    0.456129
25    0.526069
26    0.498385
27    0.838366
28    0.958630
29    0.812052
30    0.245112
31    0.874729
32    0.440818
33    0.563841
34    0.456232
35    0.523567
36    0.637054
37    0.036077
38    0.326362
39    0.511615
40    0.242937
41    0.787437
42    0.764949
43    0.166757
44    0.842915
45    0.685571
46    0.902296
47    0.889085
48    0.832561
49    0.093487
dtype: float64
Using dtype: float64
Using type: <class 'pandas.core.series.Series'>
Using Shape: (50,)
```

Example 15

- `Create a 50 x  5 data set from random values`

In [12]:

```
import pandas as pd
import numpy as np

dataf =  pd.DataFrame(np.random.rand(50,5))
print(s)
```

```
           0         1         2         3         4
0   0.095117  0.639494  0.662132  0.377261  0.484351
1   0.619275  0.943618  0.842647  0.882023  0.789404
2   0.341192  0.423504  0.457024  0.182026  0.034171
3   0.341754  0.709385  0.585795  0.407479  0.102380
4   0.068927  0.212324  0.611905  0.775162  0.103636
5   0.847370  0.306612  0.444425  0.632644  0.659969
6   0.317868  0.846814  0.728211  0.204996  0.125151
7   0.790399  0.043253  0.376310  0.094241  0.215075
8   0.136808  0.813166  0.611163  0.436034  0.227827
9   0.196890  0.428134  0.534169  0.007336  0.372561
10  0.993245  0.883609  0.091237  0.928452  0.020587
11  0.535516  0.687310  0.810303  0.787431  0.318259
12  0.602856  0.067509  0.395043  0.777689  0.543671
13  0.728186  0.257312  0.067979  0.815434  0.677408
14  0.911252  0.149094  0.898748  0.514953  0.545707
15  0.202250  0.901579  0.828142  0.552332  0.841038
16  0.918471  0.168144  0.495997  0.994389  0.854160
17  0.158534  0.908556  0.026688  0.575473  0.382548
18  0.759819  0.625431  0.845625  0.496324  0.368966
19  0.661029  0.160153  0.874547  0.206498  0.644983
20  0.810375  0.390970  0.306229  0.676295  0.722247
21  0.539418  0.330170  0.877379  0.271891  0.246110
22  0.211602  0.603702  0.120888  0.673845  0.652311
23  0.464343  0.657081  0.223643  0.240461  0.137519
24  0.236285  0.591518  0.631231  0.930020  0.405097
25  0.146758  0.741755  0.602359  0.651188  0.165919
26  0.651339  0.785964  0.344340  0.335508  0.891862
27  0.337528  0.042312  0.402397  0.803161  0.392351
28  0.181088  0.814396  0.192070  0.359722  0.747379
29  0.058017  0.231529  0.869340  0.861270  0.260224
30  0.205536  0.524932  0.544692  0.026514  0.717713
31  0.242184  0.397659  0.448239  0.463087  0.196380
32  0.753762  0.738208  0.984093  0.957331  0.489630
33  0.038807  0.378607  0.590429  0.042695  0.838390
34  0.412880  0.390730  0.873042  0.699334  0.544883
35  0.339604  0.313319  0.298470  0.340717  0.464802
36  0.697489  0.039384  0.527194  0.575181  0.138330
37  0.949106  0.033361  0.382915  0.408636  0.668237
38  0.854683  0.236989  0.882661  0.641276  0.557344
39  0.725617  0.698783  0.925393  0.690543  0.428553
40  0.981666  0.497764  0.030554  0.042278  0.732673
41  0.627341  0.598191  0.529094  0.991350  0.832340
42  0.011232  0.085394  0.488852  0.939697  0.876394
43  0.207493  0.628804  0.484087  0.440481  0.019162
44  0.766026  0.786058  0.121711  0.811946  0.572203
45  0.470483  0.758412  0.748143  0.933412  0.080471
46  0.608353  0.949183  0.307380  0.700421  0.271150
47  0.557307  0.275719  0.221070  0.116993  0.390716
48  0.204027  0.894675  0.428996  0.865814  0.513925
49  0.482776  0.495380  0.469520  0.907205  0.610760
```

Example 16

- `Find the minimum maximum and mean values column wise in a dataset`

In [13]:

```
dataf.min()
```

Out[13]:

```
0    0.059367
1    0.042292
2    0.037593
3    0.017715
4    0.016974
dtype: float64
```

In [15]:

```
dataf.max()
```

Out[15]:

```
0    0.994831
1    0.988014
2    0.977716
3    0.983931
4    0.984856
dtype: float64
```

In [16]:

```
dataf.mean()
```

Out[16]:

```
0    0.545592
1    0.493583
2    0.505099
3    0.467980
4    0.499710
dtype: float64
```

Example 17

- `Find the maximum value in 1st column`

In [17]:

```
dataf[0].max()
```

Out[17]:

```
0.9948314451530725
```

Example 18

- `Convert the dataset into numpy array and also take transpose of it`

In [18]:

```
d1 = dataf.to_numpy()
d1
```

```
Out[18]: array([[0.7393515 , 0.1410113 , 0.95232446, 0.27968595, 0.52121743],
       [0.67105707, 0.95720862, 0.76273975, 0.20318665, 0.7837203 ],
       [0.16949225, 0.90963359, 0.0566862 , 0.05422452, 0.34295278],
       [0.85294082, 0.29990849, 0.32708384, 0.4506501 , 0.07945654],
       [0.38973178, 0.14341489, 0.52974695, 0.81087479, 0.14011717],
       [0.3783606 , 0.06620189, 0.40777707, 0.93989169, 0.38610238],
       [0.33478993, 0.95365972, 0.63622107, 0.34372466, 0.11275837],
       [0.64980654, 0.84741828, 0.97771612, 0.76349159, 0.39733983],
       [0.97340478, 0.59362464, 0.82962848, 0.87709887, 0.35612845],
       [0.33623943, 0.97435094, 0.61054717, 0.55461407, 0.84265851],
       [0.70403398, 0.15963808, 0.04998997, 0.14036765, 0.17432382],
       [0.53277448, 0.28288425, 0.20784172, 0.31909872, 0.13089884],
       [0.90445384, 0.57488759, 0.19792259, 0.19024968, 0.71321779],
       [0.4837902 , 0.63158316, 0.43378344, 0.74442296, 0.73244945],
       [0.9862479 , 0.24817927, 0.51331763, 0.6551637 , 0.79715009],
       [0.42253708, 0.24474893, 0.27963863, 0.01771509, 0.89211518],
       [0.57335614, 0.61300471, 0.40717674, 0.08549452, 0.04244156],
       [0.96529077, 0.87946914, 0.71806047, 0.52545259, 0.96131564],
       [0.57968664, 0.97142878, 0.09439642, 0.28490671, 0.98485552],
       [0.23957182, 0.97737104, 0.80234145, 0.73288291, 0.87055257],
       [0.44346164, 0.48075562, 0.90135794, 0.12484963, 0.40437861],
       [0.09197983, 0.81247544, 0.12394999, 0.46068291, 0.21052968],
       [0.13353714, 0.46723784, 0.04622671, 0.66609222, 0.97940835],
       [0.78364204, 0.85035609, 0.42707098, 0.58358839, 0.20409372],
       [0.94083201, 0.17714531, 0.40497237, 0.91795843, 0.9730511 ],
       [0.9266498 , 0.36106879, 0.40496567, 0.41363171, 0.53998853],
       [0.61342117, 0.57908669, 0.21074144, 0.84376987, 0.2154542 ],
       [0.1358717 , 0.49319834, 0.41513576, 0.52636498, 0.48799033],
       [0.48202982, 0.56045508, 0.50024961, 0.83471884, 0.48079867],
       [0.69537486, 0.98801376, 0.35801536, 0.35385361, 0.61993073],
       [0.70524398, 0.90077411, 0.50958275, 0.49006846, 0.0509138 ],
       [0.44209512, 0.04985223, 0.89191521, 0.3294331 , 0.01908167],
       [0.99483145, 0.68281723, 0.9477649 , 0.81598522, 0.25123   ],
       [0.31887917, 0.17682515, 0.63375265, 0.85561304, 0.65583027],
       [0.62085445, 0.0993042 , 0.89092473, 0.19893132, 0.51804008],
       [0.70757475, 0.0422924 , 0.69668739, 0.17156558, 0.27929757],
       [0.05936676, 0.04748593, 0.03759287, 0.84477152, 0.45075669],
       [0.43157103, 0.29389433, 0.90453111, 0.07385892, 0.17262269],
       [0.49990029, 0.85349312, 0.53201861, 0.13193771, 0.93942281],
       [0.40277789, 0.19770235, 0.26095425, 0.65594109, 0.38914842],
       [0.83846413, 0.16109425, 0.66724733, 0.72232729, 0.86240368],
       [0.78686072, 0.39704197, 0.86007725, 0.98393064, 0.71524632],
       [0.54865491, 0.66156762, 0.34710651, 0.0920543 , 0.75170471],
       [0.44323088, 0.14043856, 0.37472918, 0.16227455, 0.05968647],
       [0.2906618 , 0.07186828, 0.22060287, 0.26710715, 0.96606535],
       [0.12394171, 0.79599579, 0.54657473, 0.18324681, 0.22362947],
       [0.27601601, 0.38690341, 0.53768255, 0.5685669 , 0.62568555],
       [0.59315427, 0.32012885, 0.39707109, 0.46937379, 0.97891412],
       [0.33792893, 0.39239046, 0.76698353, 0.50889495, 0.68145491],
       [0.72386212, 0.76788152, 0.64350337, 0.17438467, 0.01697399]])
```

```
In [19]: d1.T
```

Example 19

```
Out[19]:
array([[0.7393515 , 0.67105707, 0.16949225, 0.85294082, 0.38973178,
         0.5783606 , 0.33478993, 0.64980654, 0.97340478, 0.33623943,
         0.70403398, 0.53277448, 0.90445384, 0.4837902 , 0.9862479 ,
         0.42253708, 0.57335614, 0.96529077, 0.57968664, 0.23957182,
         0.44346164, 0.09197983, 0.13353714, 0.78364204, 0.94083201,
         0.9266498 , 0.61342117, 0.1358717 , 0.48202982, 0.69537486,
         0.70524398, 0.44209512, 0.99483145, 0.31887917, 0.62085445,
         0.70757475, 0.05936676, 0.43157103, 0.49990029, 0.40277789,
         0.83846413, 0.78686072, 0.54865491, 0.44323088, 0.2906618 ,
         0.12394171, 0.27601601, 0.59315427, 0.33792893, 0.72386212],
        [0.1410113 , 0.95720862, 0.90963359, 0.29990849, 0.14341489,
         0.06620189, 0.95365972, 0.84741828, 0.59362464, 0.97435094,
         0.15963808, 0.28288425, 0.57488759, 0.63158316, 0.24817927,
         0.24474893, 0.61300471, 0.87946914, 0.97142878, 0.97737104,
         0.48075562, 0.81247544, 0.46723784, 0.85035609, 0.17714531,
         0.36106879, 0.57908669, 0.49319834, 0.56045508, 0.98801376,
         0.90077411, 0.04985223, 0.68281723, 0.17682515, 0.0993042 ,
         0.0422924 , 0.04748593, 0.29389433, 0.85349312, 0.19770235,
         0.16109425, 0.39704197, 0.66156762, 0.14043856, 0.07186828,
         0.79599579, 0.38690341, 0.32012885, 0.39239046, 0.76788152],
        [0.95232446, 0.76273975, 0.0566862 , 0.32708384, 0.52974695,
         0.40777707, 0.63622107, 0.97771612, 0.82962848, 0.61054717,
         0.04998997, 0.20784172, 0.19792259, 0.43378344, 0.51331763,
         0.27963863, 0.40717674, 0.71806047, 0.09439642, 0.80234145,
         0.90135794, 0.12394999, 0.04622671, 0.42707098, 0.40497237,
         0.40496567, 0.21074144, 0.41513576, 0.50024961, 0.35801536,
         0.50958275, 0.89191521, 0.9477649 , 0.63375265, 0.89092473,
         0.69668739, 0.03759287, 0.90453111, 0.53201861, 0.26095425,
         0.66724733, 0.86007725, 0.34710651, 0.37472918, 0.22060287,
         0.54657473, 0.53768255, 0.39707109, 0.76698353, 0.64350337],
        [0.27968595, 0.20318665, 0.05422452, 0.4506501 , 0.81087479,
         0.93989169, 0.34372466, 0.76349159, 0.87709887, 0.55461407,
         0.14036765, 0.31909872, 0.19024968, 0.74442296, 0.6551637 ,
         0.01771509, 0.08549452, 0.52545259, 0.28490671, 0.73288291,
         0.12484963, 0.46068291, 0.66609222, 0.58358839, 0.91795843,
         0.41363171, 0.84376987, 0.52636498, 0.83471884, 0.35385361,
         0.49006846, 0.3294331 , 0.81598522, 0.85561304, 0.19893132,
         0.17156558, 0.84477152, 0.07385892, 0.13193771, 0.65594109,
         0.72232729, 0.98393064, 0.0920543 , 0.16227455, 0.26710715,
         0.18324681, 0.5685669 , 0.46937379, 0.50889495, 0.17438467],
        [0.52121743, 0.7837203 , 0.34295278, 0.07945654, 0.14011717,
         0.38610238, 0.11275837, 0.39733983, 0.35612845, 0.84265851,
         0.17432382, 0.13089884, 0.71321779, 0.73244945, 0.79715009,
         0.89211518, 0.04244156, 0.96131564, 0.98485552, 0.87055257,
         0.40437861, 0.21052968, 0.97940835, 0.20409372, 0.9730511 ,
         0.53998853, 0.2154542 , 0.48799033, 0.48079867, 0.61993073,
         0.0509138 , 0.01908167, 0.25123   , 0.65583027, 0.51804008,
         0.27929757, 0.45075669, 0.17262269, 0.93942281, 0.38914842,
         0.86240368, 0.71524632, 0.75170471, 0.05968647, 0.96606535,
         0.22362947, 0.62568555, 0.97891412, 0.68145491, 0.01697399]])
```

- `Change names of the columns.`

```
In [20]: dataf.columns = ['A', 'B', 'C', 'D', 'E']
         dataf
```

```
Out[20]:
           A         B         C         D         E
 0  0.739351  0.141011  0.952324  0.279686  0.521217
 1  0.671057  0.957209  0.762740  0.203187  0.783720
 2  0.169492  0.909634  0.056686  0.054225  0.342953
 3  0.852941  0.299908  0.327084  0.450650  0.079457
 4  0.389732  0.143415  0.529747  0.810875  0.140117
 5  0.378361  0.066202  0.407777  0.939892  0.386102
 6  0.334790  0.953660  0.636221  0.343725  0.112758
 7  0.649807  0.847418  0.977716  0.763492  0.397340
 8  0.973405  0.593625  0.829628  0.877099  0.356128
 9  0.336239  0.974351  0.610547  0.554614  0.842659
10  0.704034  0.159638  0.049990  0.140368  0.174324
11  0.532774  0.282884  0.207842  0.319099  0.130899
12  0.904454  0.574888  0.197923  0.190250  0.713218
13  0.483790  0.631583  0.433783  0.744423  0.732449
14  0.986248  0.248179  0.513318  0.655164  0.797150
15  0.422537  0.244749  0.279639  0.017715  0.892115
16  0.573356  0.613005  0.407177  0.085495  0.042442
17  0.965291  0.879469  0.718060  0.525453  0.961316
18  0.579687  0.971429  0.094396  0.284907  0.984856
19  0.239572  0.977371  0.802341  0.732883  0.870553
20  0.443462  0.480756  0.901358  0.124850  0.404379
21  0.091980  0.812475  0.123950  0.460683  0.210530
22  0.133537  0.467238  0.046227  0.666092  0.979408
23  0.783642  0.850356  0.427071  0.583588  0.204094
24  0.940832  0.177145  0.404972  0.917958  0.973051
25  0.926650  0.361069  0.404966  0.413632  0.539989
26  0.613421  0.579087  0.210741  0.843770  0.215454
27  0.135872  0.493198  0.415136  0.526365  0.487990
28  0.482030  0.560455  0.500250  0.834719  0.480799
29  0.695375  0.988014  0.358015  0.353854  0.619931
30  0.705244  0.900774  0.509583  0.490068  0.050914
31  0.442095  0.049852  0.891915  0.329433  0.019082
32  0.994831  0.682817  0.947765  0.815985  0.251230
```

	A	B	C	D	E
33	0.318879	0.176825	0.633753	0.855613	0.655830
34	0.620854	0.099304	0.890925	0.198931	0.518040
35	0.707575	0.042292	0.696687	0.171566	0.279298
36	0.059367	0.047486	0.037593	0.844772	0.450757
37	0.431571	0.293894	0.904531	0.073859	0.172623
38	0.499900	0.853493	0.532019	0.131938	0.939423
39	0.402778	0.197702	0.260954	0.655941	0.389148
40	0.838464	0.161094	0.667247	0.722327	0.862404
41	0.786861	0.397042	0.860077	0.983931	0.715246
42	0.548655	0.661568	0.347107	0.092054	0.751705
43	0.443231	0.140439	0.374729	0.162275	0.059686
44	0.290662	0.071868	0.220603	0.267107	0.966065
45	0.123942	0.795996	0.546575	0.183247	0.223629
46	0.276016	0.386903	0.537683	0.568567	0.625686
47	0.593154	0.320129	0.397071	0.469374	0.978914
48	0.337929	0.392390	0.766984	0.508895	0.681455
49	0.723862	0.767882	0.643503	0.174385	0.016974

Example 20

- `Display column B and C from the dataset and also use head function`

In [21]:

```
dataf[['B', 'C']]
```

Out[21]:

	B	C
0	0.141011	0.952324
1	0.957209	0.762740
2	0.909634	0.056686
3	0.299908	0.327084
4	0.143415	0.529747
5	0.066202	0.407777
6	0.953660	0.636221
7	0.847418	0.977716
8	0.593625	0.829628
9	0.974351	0.610547
10	0.159638	0.049990
11	0.282884	0.207842
12	0.574888	0.197923
13	0.631583	0.433783
14	0.248179	0.513318
15	0.244749	0.279639
16	0.613005	0.407177
17	0.879469	0.718060
18	0.971429	0.094396
19	0.977371	0.802341
20	0.480756	0.901358
21	0.812475	0.123950
22	0.467238	0.046227
23	0.850356	0.427071
24	0.177145	0.404972
25	0.361069	0.404966
26	0.579087	0.210741
27	0.493198	0.415136
28	0.560455	0.500250
29	0.988014	0.358015
30	0.900774	0.509583
31	0.049852	0.891915
32	0.682817	0.947765

	B	C
33	0.176825	0.633753
34	0.099304	0.890925
35	0.042292	0.696687
36	0.047486	0.037593
37	0.293894	0.904531
38	0.853493	0.532019
39	0.197702	0.260954
40	0.161094	0.667247
41	0.397042	0.860077
42	0.661568	0.347107
43	0.140439	0.374729
44	0.071868	0.220603
45	0.795996	0.546575
46	0.386903	0.537683
47	0.320129	0.397071
48	0.392390	0.766984
49	0.767882	0.643503

In [22]:
```
dataf.head()
```

Out[22]:

	A	B	C	D	E
0	0.739351	0.141011	0.952324	0.279686	0.521217
1	0.671057	0.957209	0.762740	0.203187	0.783720
2	0.169492	0.909634	0.056686	0.054225	0.342953
3	0.852941	0.299908	0.327084	0.450650	0.079457
4	0.389732	0.143415	0.529747	0.810875	0.140117

Example 21

- `Demonstrate the use of iloc function`

In [23]:
```
dataf.iloc[:, 0:2]   # : means all rows and 0:2 means cloumns till 2
```

Out[23]:

	A	B
0	0.739351	0.141011
1	0.671057	0.957209
2	0.169492	0.909634
3	0.852941	0.299908
4	0.389732	0.143415
5	0.378361	0.066202
6	0.334790	0.953660
7	0.649807	0.847418
8	0.973405	0.593625
9	0.336239	0.974351
10	0.704034	0.159638
11	0.532774	0.282884
12	0.904454	0.574888
13	0.483790	0.631583
14	0.986248	0.248179
15	0.422537	0.244749
16	0.573356	0.613005
17	0.965291	0.879469
18	0.579687	0.971429
19	0.239572	0.977371
20	0.443462	0.480756
21	0.091980	0.812475
22	0.133537	0.467238
23	0.783642	0.850356
24	0.940832	0.177145
25	0.926650	0.361069
26	0.613421	0.579087
27	0.135872	0.493198
28	0.482030	0.560455
29	0.695375	0.988014
30	0.705244	0.900774
31	0.442095	0.049852
32	0.994831	0.682817

	A	B
33	0.318879	0.176825
34	0.620854	0.099304
35	0.707575	0.042292
36	0.059367	0.047486
37	0.431571	0.293894
38	0.499900	0.853493
39	0.402778	0.197702
40	0.838464	0.161094
41	0.786861	0.397042
42	0.548655	0.661568
43	0.443231	0.140439
44	0.290662	0.071868
45	0.123942	0.795996
46	0.276016	0.386903
47	0.593154	0.320129
48	0.337929	0.392390
49	0.723862	0.767882

Example 22

- `Print column A to C and fimd the value on 0,0`

```
In [26]: dataf.loc[:, 'A':'C'] # loc function use to specify the columns label or name
```

Out[26]:

	A	B	C
0	0.739351	0.141011	0.952324
1	0.671057	0.957209	0.762740
2	0.169492	0.909634	0.056686
3	0.852941	0.299908	0.327084
4	0.389732	0.143415	0.529747
5	0.378361	0.066202	0.407777
6	0.334790	0.953660	0.636221
7	0.649807	0.847418	0.977716
8	0.973405	0.593625	0.829628
9	0.336239	0.974351	0.610547
10	0.704034	0.159638	0.049990
11	0.532774	0.282884	0.207842
12	0.904454	0.574888	0.197923
13	0.483790	0.631583	0.433783
14	0.986248	0.248179	0.513318
15	0.422537	0.244749	0.279639
16	0.573356	0.613005	0.407177
17	0.965291	0.879469	0.718060
18	0.579687	0.971429	0.094396
19	0.239572	0.977371	0.802341
20	0.443462	0.480756	0.901358
21	0.091980	0.812475	0.123950
22	0.133537	0.467238	0.046227
23	0.783642	0.850356	0.427071
24	0.940832	0.177145	0.404972
25	0.926650	0.361069	0.404966
26	0.613421	0.579087	0.210741
27	0.135872	0.493198	0.415136
28	0.482030	0.560455	0.500250
29	0.695375	0.988014	0.358015
30	0.705244	0.900774	0.509583
31	0.442095	0.049852	0.891915
32	0.994831	0.682817	0.947765

	A	B	C
33	0.318879	0.176825	0.633753
34	0.620854	0.099304	0.890925
35	0.707575	0.042292	0.696687
36	0.059367	0.047486	0.037593
37	0.431571	0.293894	0.904531
38	0.499900	0.853493	0.532019
39	0.402778	0.197702	0.260954
40	0.838464	0.161094	0.667247
41	0.786861	0.397042	0.860077
42	0.548655	0.661568	0.347107
43	0.443231	0.140439	0.374729
44	0.290662	0.071868	0.220603
45	0.123942	0.795996	0.546575
46	0.276016	0.386903	0.537683
47	0.593154	0.320129	0.397071
48	0.337929	0.392390	0.766984
49	0.723862	0.767882	0.643503

Example 23

- `Print 1st 12 elements of column 2 and 4`

```
In [27]: dataf.iloc[0:12, 2:4]
```

Out[27]:

	C	D
0	0.952324	0.279686
1	0.762740	0.203187
2	0.056686	0.054225
3	0.327084	0.450650
4	0.529747	0.810875
5	0.407777	0.939892
6	0.636221	0.343725
7	0.977716	0.763492
8	0.829628	0.877099
9	0.610547	0.554614
10	0.049990	0.140368
11	0.207842	0.319099

03. Matplotlib

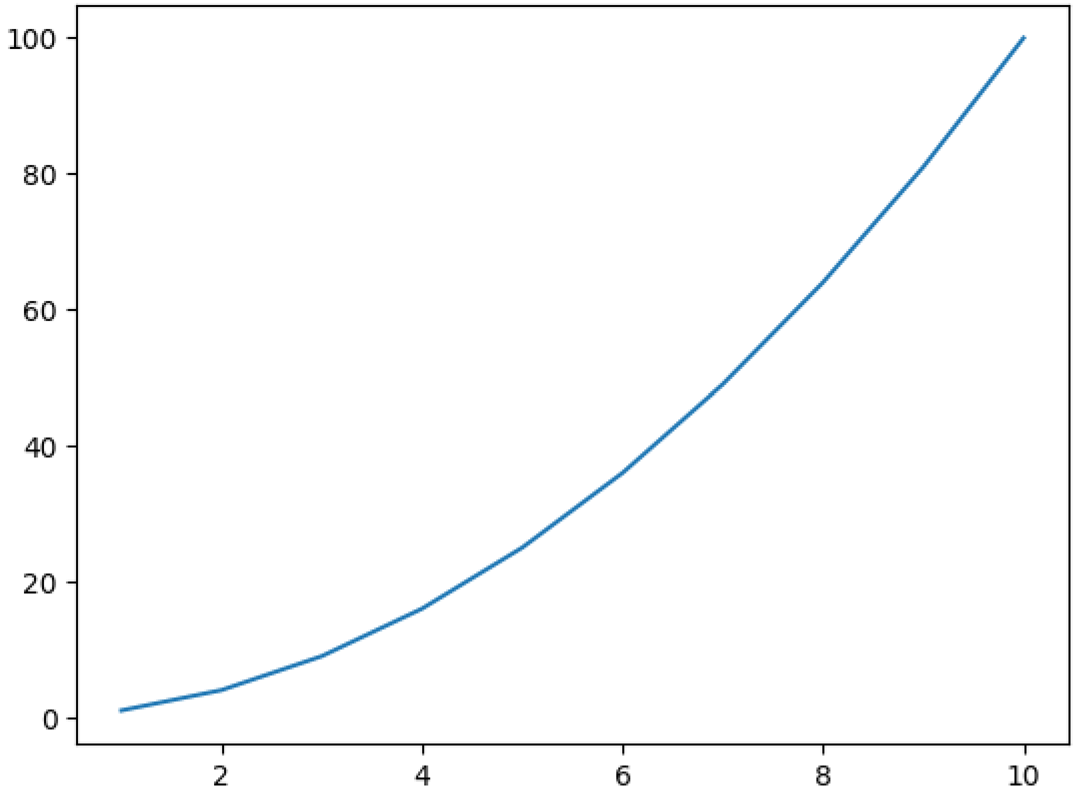
Example 01

- `Use plot and show function to create and show the graph`

In [28]:

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

x  =  np.array((1,2,3,4,5,6,7,8,9,10))
y = x**2

print(x)

plt.plot(x,y)
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```



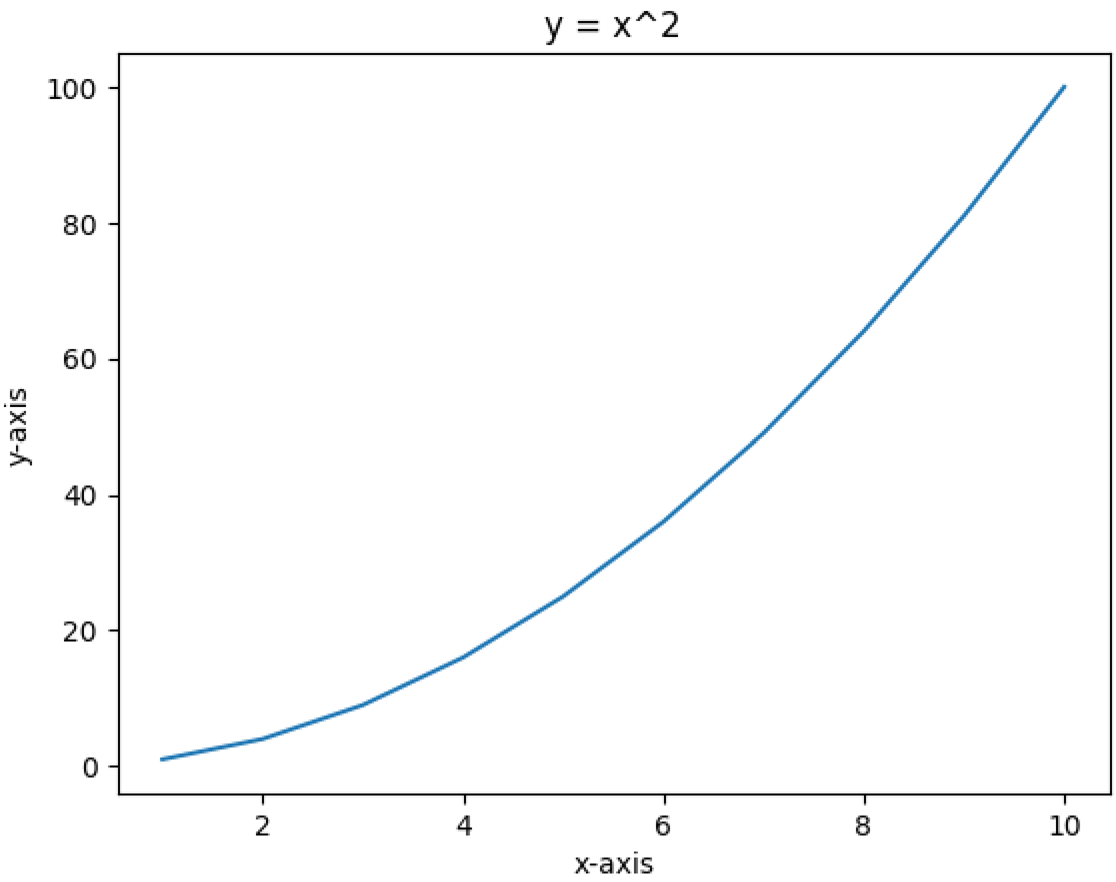
Example 02

- `Add labels and tittle to the graph`

In [29]:

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

x  =  np.array((1,2,3,4,5,6,7,8,9,10))
y = x**2

print(x)

plt.plot(x,y)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title("y = x^2")
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

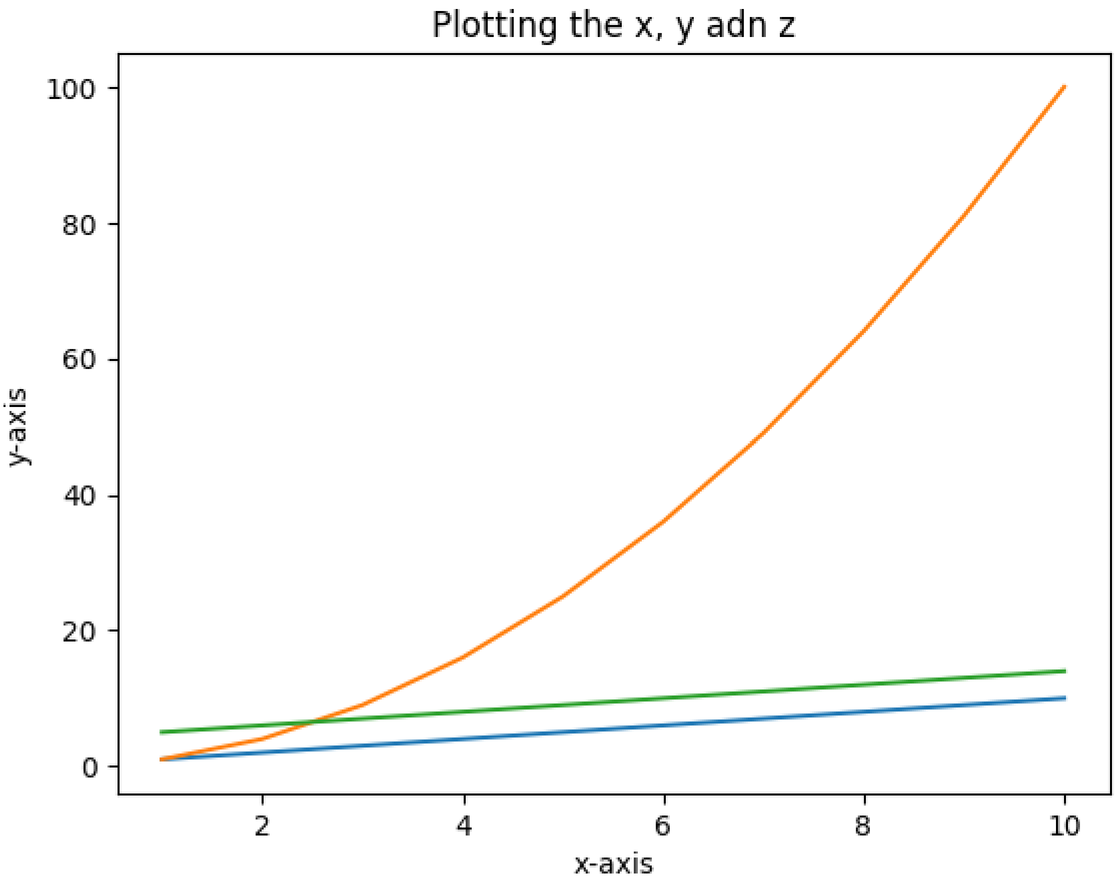


Exampel 03

- `Plot 3 variables on a single graph`

In [31]:
```
import numpy as np
import matplotlib.pyplot as plt

x  =  np.array((1,2,3,4,5,6,7,8,9,10))
y  = x**2
z  = x + 4

print(f"{x}\n{y}\n{z}")

plt.plot(x,x)
plt.plot(x,y)
plt.plot(x,z)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title("Plotting the x, y adn z")
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[  1   4   9  16  25  36  49  64  81 100]
[ 5  6  7  8  9 10 11 12 13 14]
```



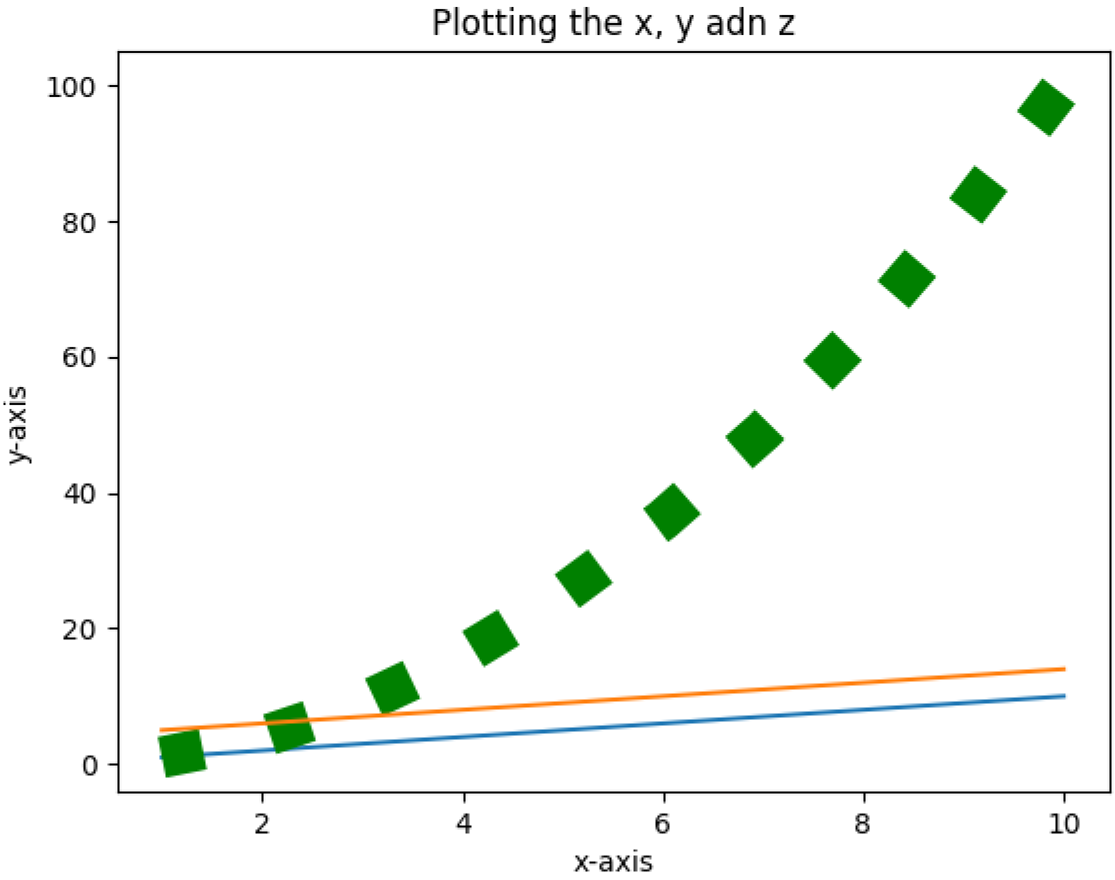
Example 04

- `Change the color linestyle and linewidth of the graph`

In [6]:

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

x  =  np.array((1,2,3,4,5,6,7,8,9,10))
y = x**2
z = x + 4

print(f"{x}\n{y}\n{z}")

plt.plot(x,x)
plt.plot(x,y, color='g', linestyle = ':', linewidth=15)
plt.plot(x,z)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title("Plotting the x, y adn z")
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[  1   4   9  16  25  36  49  64  81 100]
[ 5  6  7  8  9 10 11 12 13 14]
```



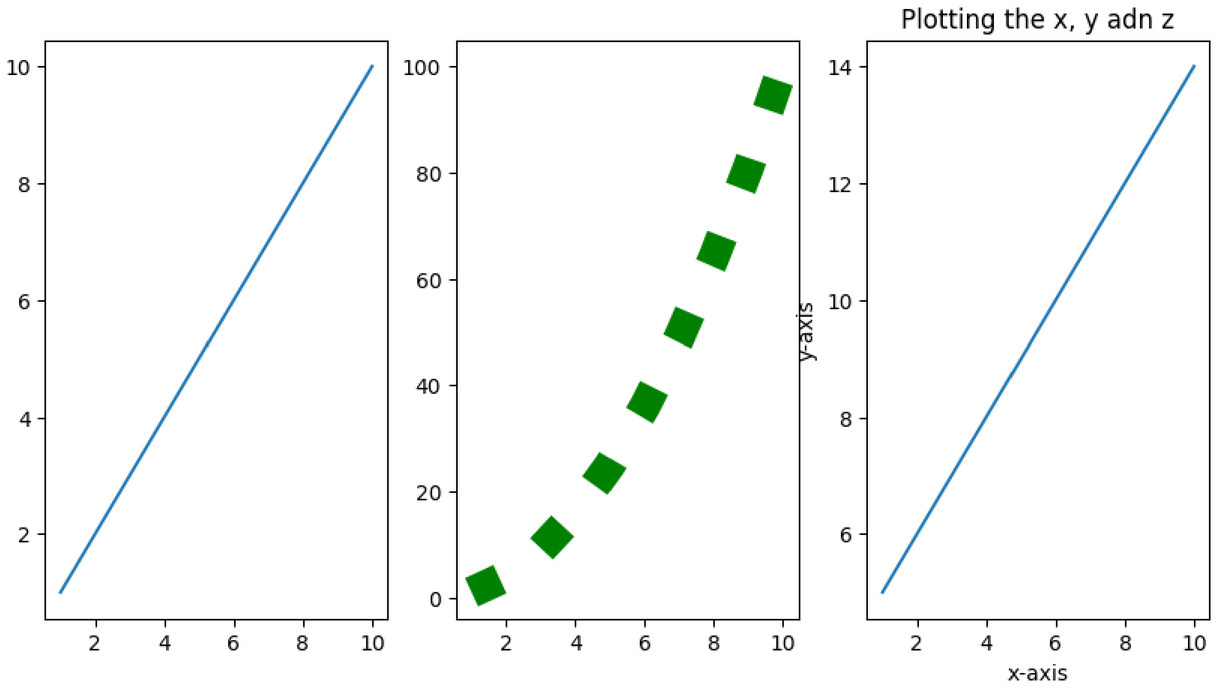
Example 05

- `Plot using subplot`

In [10]:

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

x  =  np.array((1,2,3,4,5,6,7,8,9,10))
y = x**2
z = x + 4

print(f"{x}\n{y}\n{z}")

plt.figure(figsize=(10,5))

plt.subplot(1,3,1)
plt.plot(x,x)

plt.subplot(1,3,2)
plt.plot(x,y, color='g', linestyle = ':', linewidth=15)

plt.subplot(1,3,3)
plt.plot(x,z)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title("Plotting the x, y adn z")
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[  1   4   9  16  25  36  49  64  81 100]
[ 5  6  7  8  9 10 11 12 13 14]
```

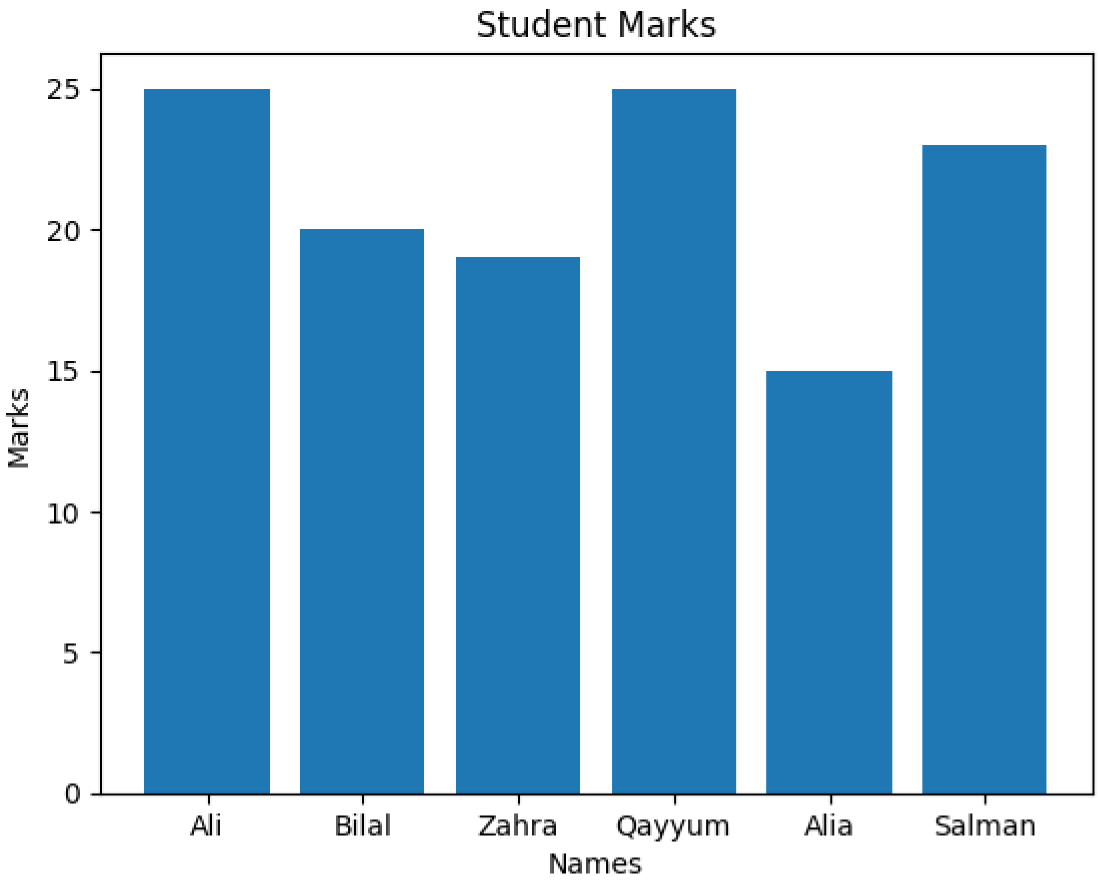


Example 06

- `Print the marks of students w.r.t their names using Dictionary`

In [14]:
```
stuMarks = {"Ali": 25, "Bilal": 20, "Zahra": 19, "Qayyum": 25, "Alia": 15, "Salman":
print(stuMarks)

k = stuMarks.keys()
v = stuMarks.values()

plt.title("Student  Marks")
plt.xlabel("Names")
plt.ylabel("Marks")
plt.bar(k,v)
plt.show()
```

```
{'Ali': 25, 'Bilal': 20, 'Zahra': 19, 'Qayyum': 25, 'Alia': 15, 'Salman': 23}
```



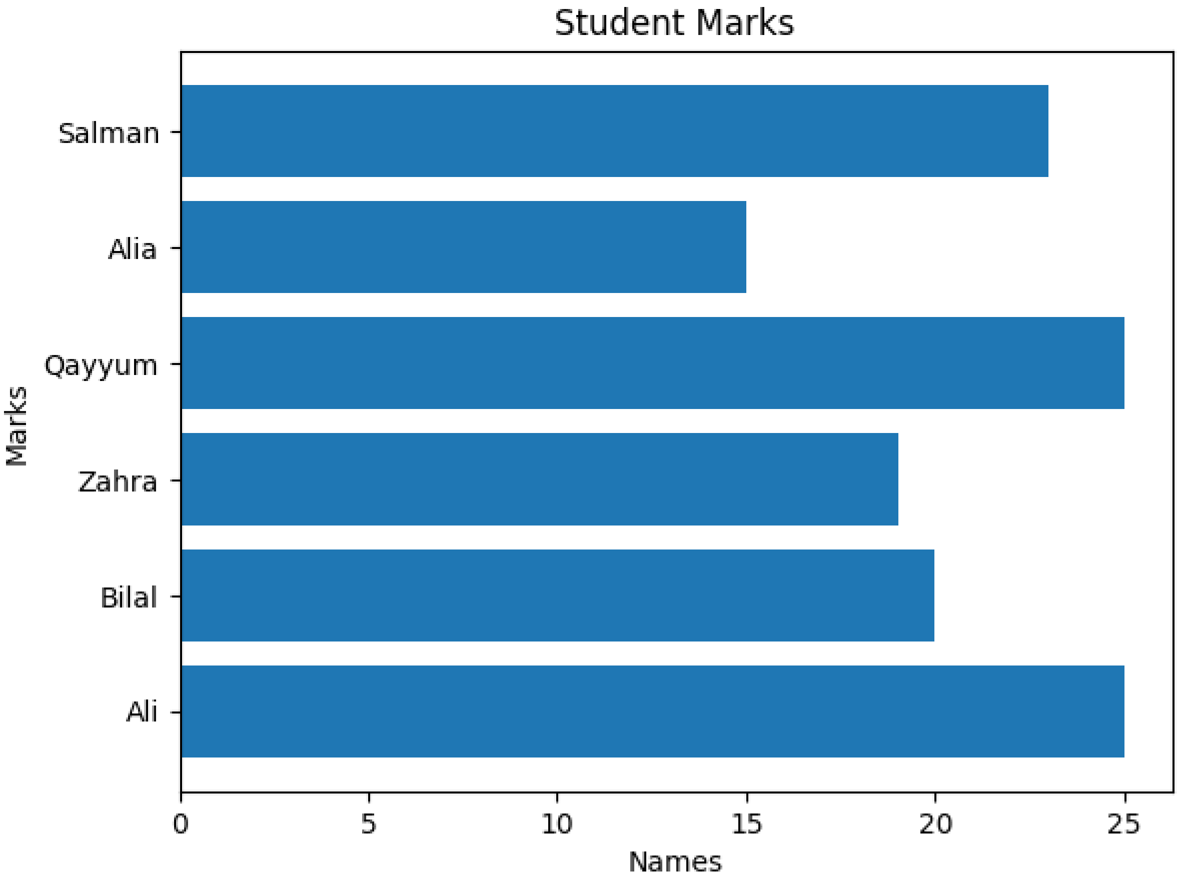
Example 07

- `Plot horizonatal bar garaph`

In [16]:

```
stuMarks = {"Ali": 25, "Bilal": 20, "Zahra": 19, "Qayyum": 25, "Alia": 15, "Salman":
print(stuMarks)

k = list(stuMarks.keys())
v = list(stuMarks.values())

plt.title("Student  Marks")
plt.xlabel("Names")
plt.ylabel("Marks")
plt.barh(k,v)
plt.show()
```

```
{'Ali': 25, 'Bilal': 20, 'Zahra': 19, 'Qayyum': 25, 'Alia': 15, 'Salman': 23}
```

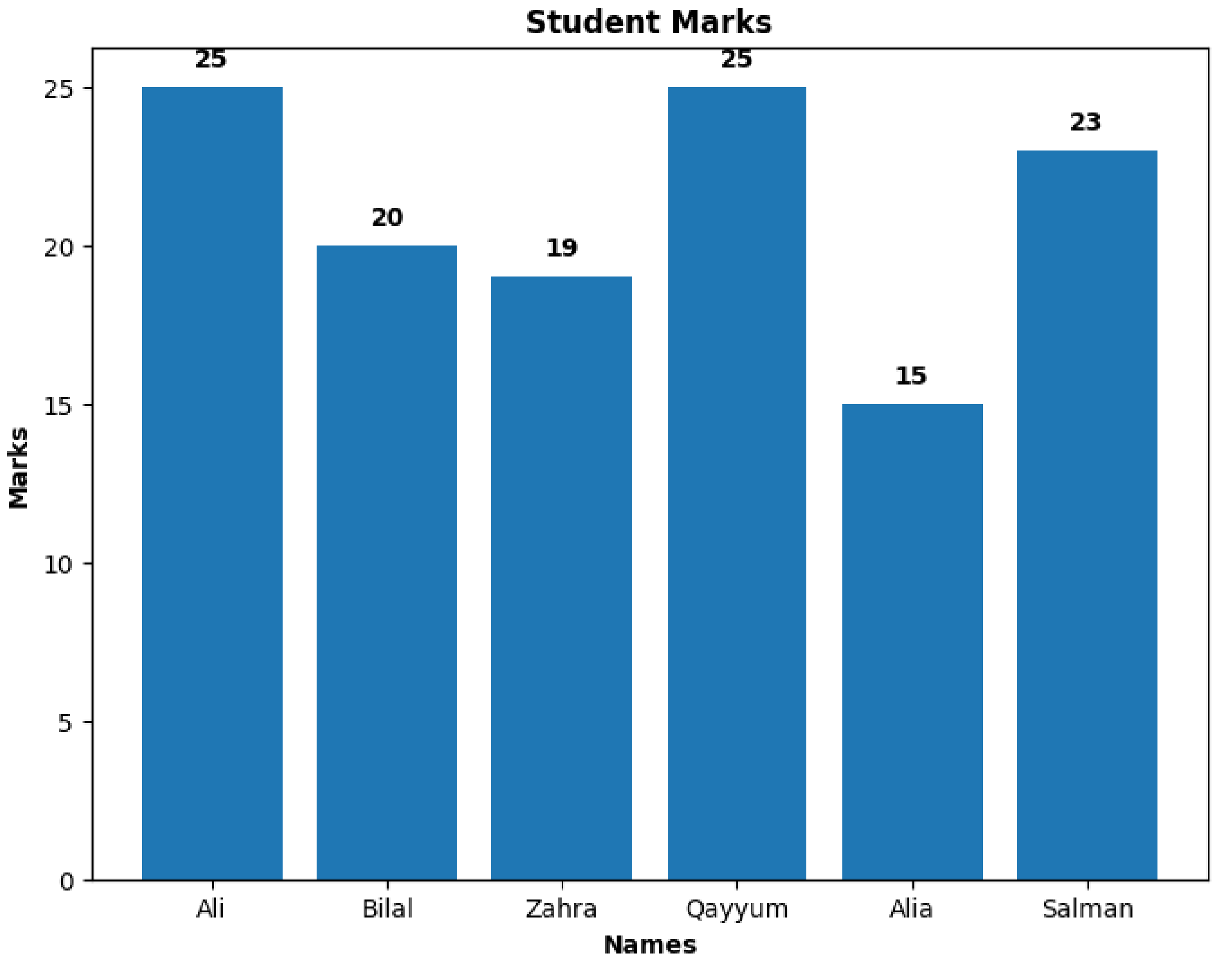


Example 08

- `Bold the tittle and xlabel y label and also show the value of yaxis on top of bars`

In [20]:
```
import matplotlib.pyplot as plt

stuMarks = {"Ali": 25, "Bilal": 20, "Zahra": 19, "Qayyum": 25, "Alia": 15, "Salman":
print(stuMarks)

k = stuMarks.keys()
v = stuMarks.values()

plt.figure(figsize=(8, 6))  # Set the figure size

# Plot the bar chart
plt.bar(k, v)

# Customize the plot
plt.title("Student Marks", fontweight="bold")  # Make the title bold
plt.xlabel("Names", fontweight="bold")  # Make xlabel bold
plt.ylabel("Marks", fontweight="bold")  # Make ylabel bold

# Annotate the values on top of the bars
for key, value in stuMarks.items():
    plt.text(key, value + 0.5, str(value), ha='center', va='bottom', fontweight='bold

plt.show()
```

```
{'Ali': 25, 'Bilal': 20, 'Zahra': 19, 'Qayyum': 25, 'Alia': 15, 'Salman': 23}
```



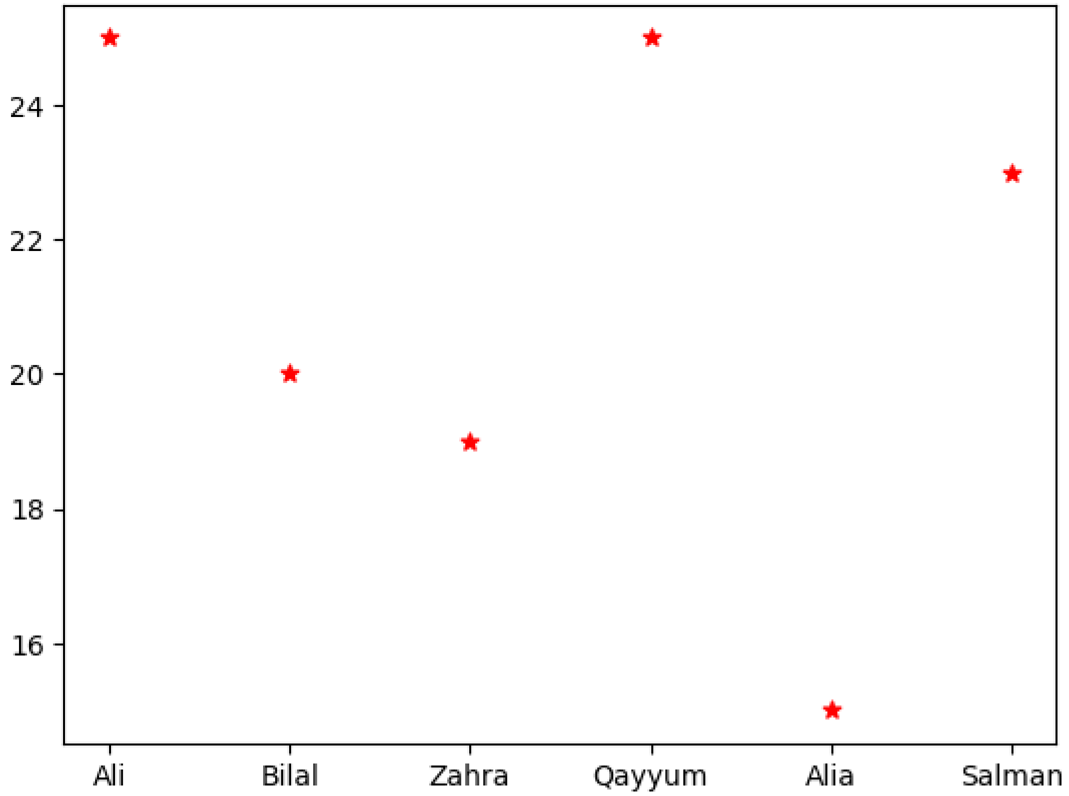
Example 09

- `Plot using scatter function`

In [23]:

```
import matplotlib.pyplot as plt

stuMarks = {"Ali": 25, "Bilal": 20, "Zahra": 19, "Qayyum": 25, "Alia": 15, "Salman":
print(stuMarks)

k = stuMarks.keys()
v = stuMarks.values()

plt.scatter(k,v, color = 'r', marker="*", s = 40)
plt.show()
```

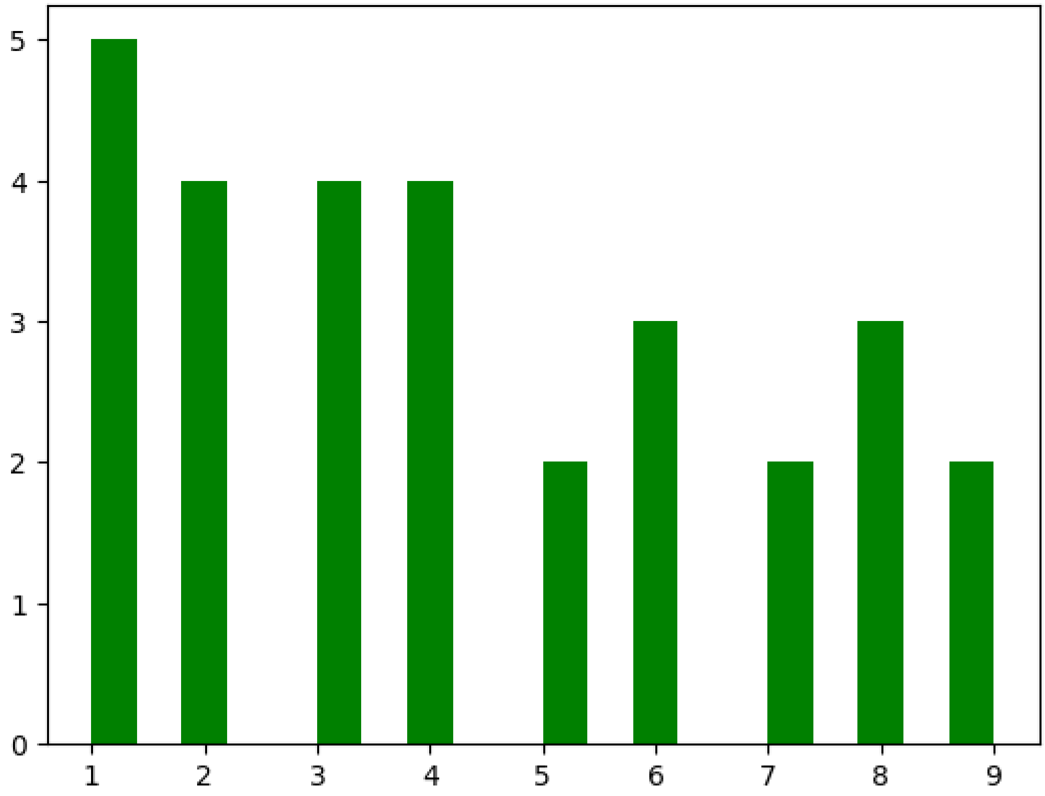
```
{'Ali': 25, 'Bilal': 20, 'Zahra': 19, 'Qayyum': 25, 'Alia': 15, 'Salman': 23}
```



Example 10

- `Plot a histogram`

In [27]:
```
a  =  [1,2,3,4,5,6,7,8,9,4,6,8,2,3,1,1,6,8,9,3,4,2,1,1,2,3,4,5,7]

plt.hist(a, bins=20, color='g')
plt.show()
```

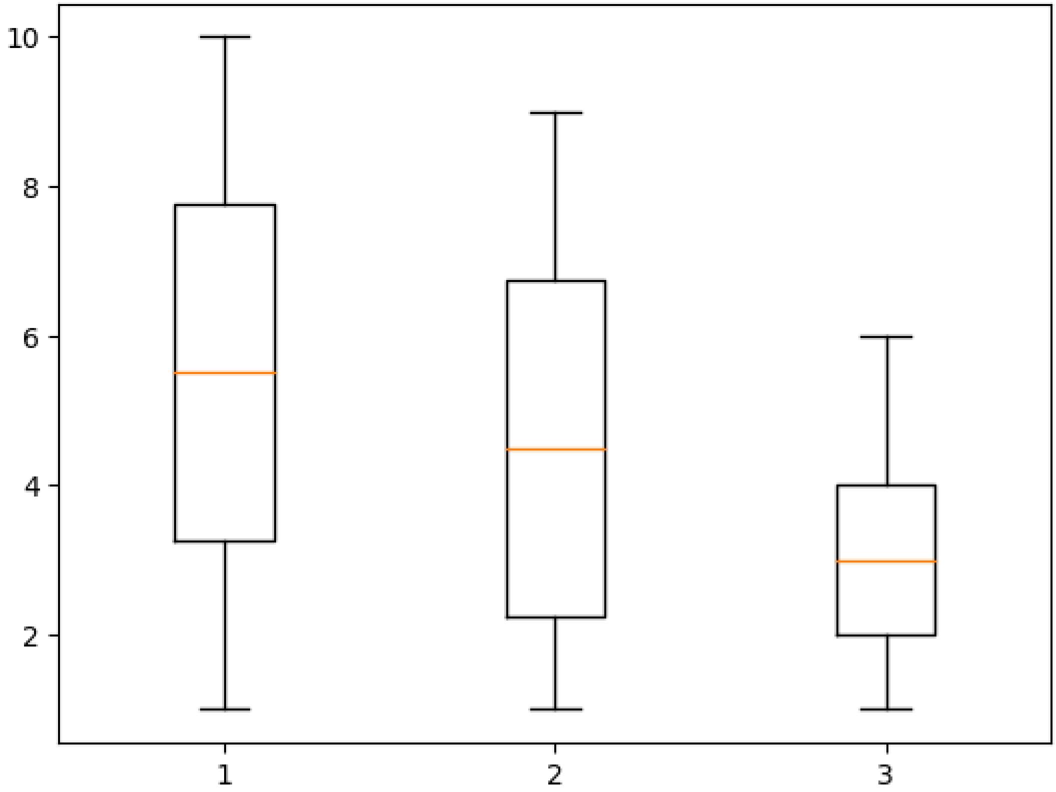


Example 11

- `Demonstrate the use of Box Plot`

A Box Plot is also known as Whisker plot is created to display the summary of the set of data values having properties like minimum, first quartile, median, third quartile and maximum. In the box plot, a box is created from the first quartile to the third quartile, a vertical line is also there which goes through the box at the median. Here x-axis denotes the data to be plotted while the y-axis shows the frequency distribution.

In [28]:
```
l1 = [1,2,3,4,5,6,7,8,9,10]
l2  =  [3,4,5,6,7,1,2,8,9,1]
l3  =  [1,2,3,4,1,2,3,4,5,6]

data = list([l1,l2,l3])

plt.boxplot(data)
plt.show()
```

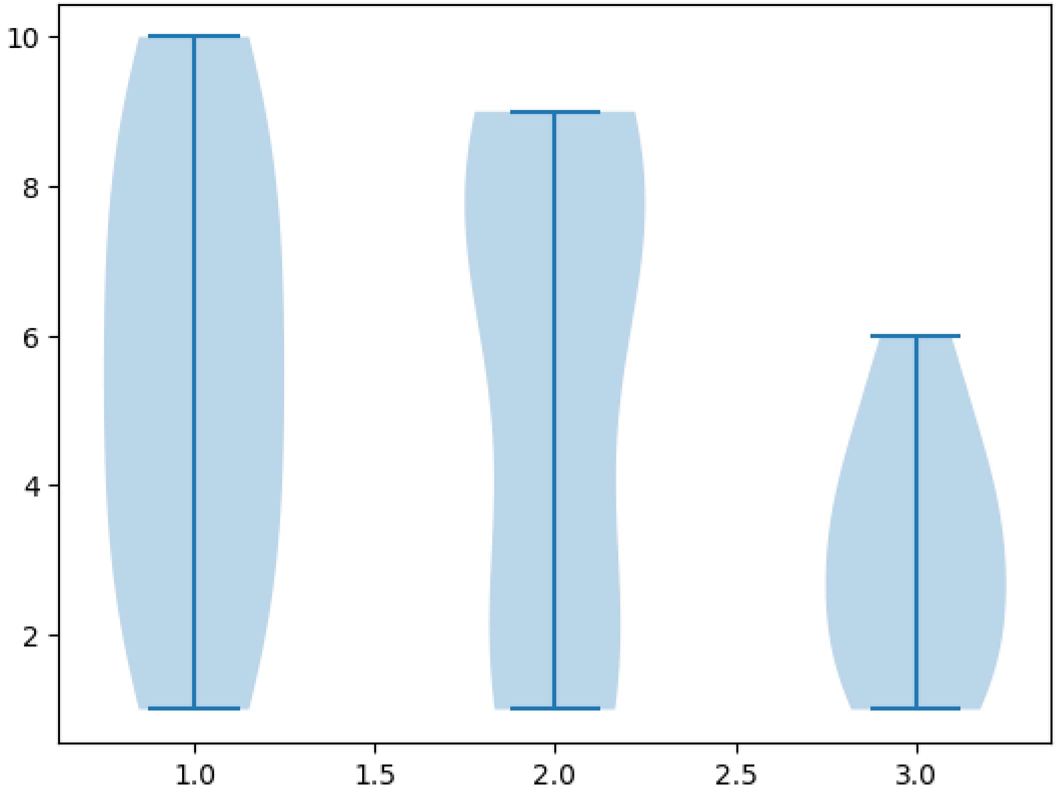


Example 12

- `Demonstrate the use of violin plot`

In [30]:

```
l1 = [1,2,3,4,5,6,7,8,9,10]
l2  =  [3,8,9,6,7,1,2,8,9,1]
l3  =  [1,2,3,4,1,2,3,4,5,6]

data = list([l1,l2,l3])

plt.violinplot(data)
plt.show()
```



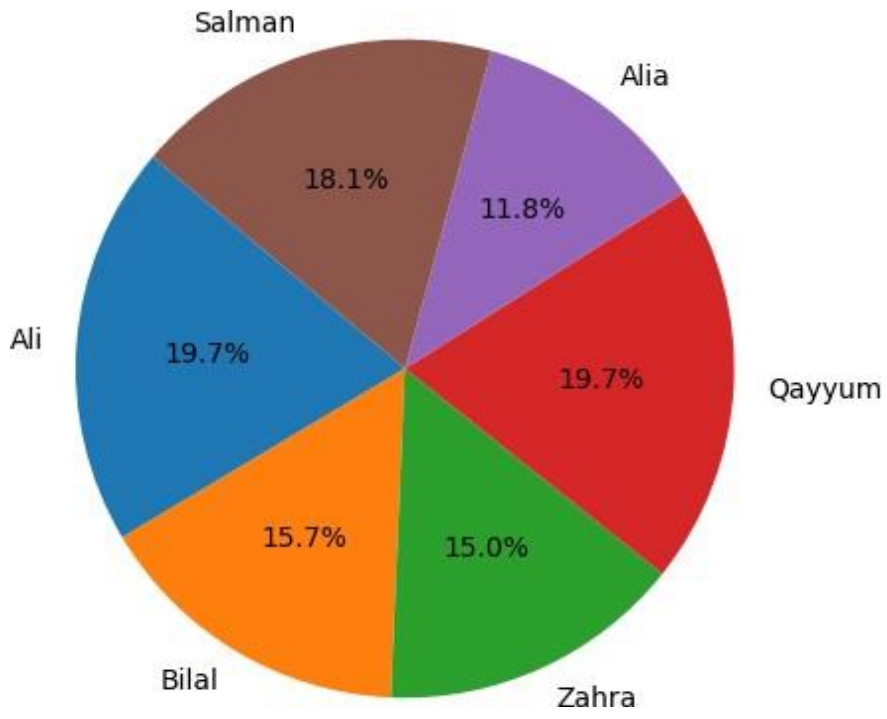
Example 13

- `Show the example of pie plot`

In [33]:

```
import matplotlib.pyplot as plt

stuMarks = {"Ali": 25, "Bilal": 20, "Zahra": 19, "Qayyum": 25, "Alia": 15, "Salman":
print(stuMarks)

k = stuMarks.keys()
v = stuMarks.values()

plt.pie(v,labels=k,  autopct='%1.1f%%',  startangle=140)
plt.axis('equal')

plt.show()
```

```
{'Ali': 25, 'Bilal': 20, 'Zahra': 19, 'Qayyum': 25, 'Alia': 15, 'Salman': 23}
```


Salman
18.1%
Alia
11.8%
Ali
19.7%
19.7%
Qayyum
15.7%
15.0%
Bilal
Zahra