LAB1:

Basics of programming in Python: Basic input/output Basic data types and data structures Control flow Functions and modules Basic numerical and scientific computation Graphical visualization

LAB 1:Basics of python programming

1.1 Basic Input Output

```
b = 2

print(b)

2

string1 = "This is python programming"

print(string1[1:3])

hi

n=input("enter the number")
print(n)

enter the number12
12
```

Tuples

```
tup=("hello",1,2,3)

tup

('hello', 1, 2, 3)
```

List

```
l1=[1,2,"hello"]

l1

[1, 2, 'hello']
```

Dictionary

```
dict1={"a":1,2:"hi"}

dict1
```

```
{'a': 1, 2: 'hi'}
```

Set

```
s1=set({1,2,1,"hello"})

s1

{1, 2, 'hello'}
```

Control flow

IF else

```
a=10
b=20
if(a>b):
  print("a is greater")
else:
  print("b is greater")

b is greater

cmd="start"
match cmd:
  case "start":
    print("start the game")
  case "stop":
    print("stop the game")

start the game
```

For loop

```
for i in range(10):
  print(i)

0
1
2
3
4
5
6
7
8
9
```

Functions and modules

```
def hello():
  print("hello")

hello()

hello

import numpy as np

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

a

array([1, 2, 3, 4])

from cmath import sin
a=sin(10)
print(a)

(-0.5440211108893698-0j)

import math
a=math.sqrt(18)
print(a)

4.242640687119285
```

Error handling

```
a=-2
try:
  if a<0:
     raise ValueError("Cannot find the squareroot of negative number")

  b=math.sqrt(a)
  print(b)
except ValueError as e:
  print(e)

Cannot find the squareroot of negative number

try:
    c = 10 / 0  # This will raise a ZeroDivisionError
    raise ValueError("Cannot divide by zero")  # This won't execute
because of the above error
except ZeroDivisionError as e:
    print("Caught a ZeroDivisionError:", e)
except ValueError as e:
    print("Caught a ValueError:", e)
finally:
    print("The error is of another source or handled completely.")
```

```
Caught a ZeroDivisionError: division by zero
The error is of another source or handled completely.
```

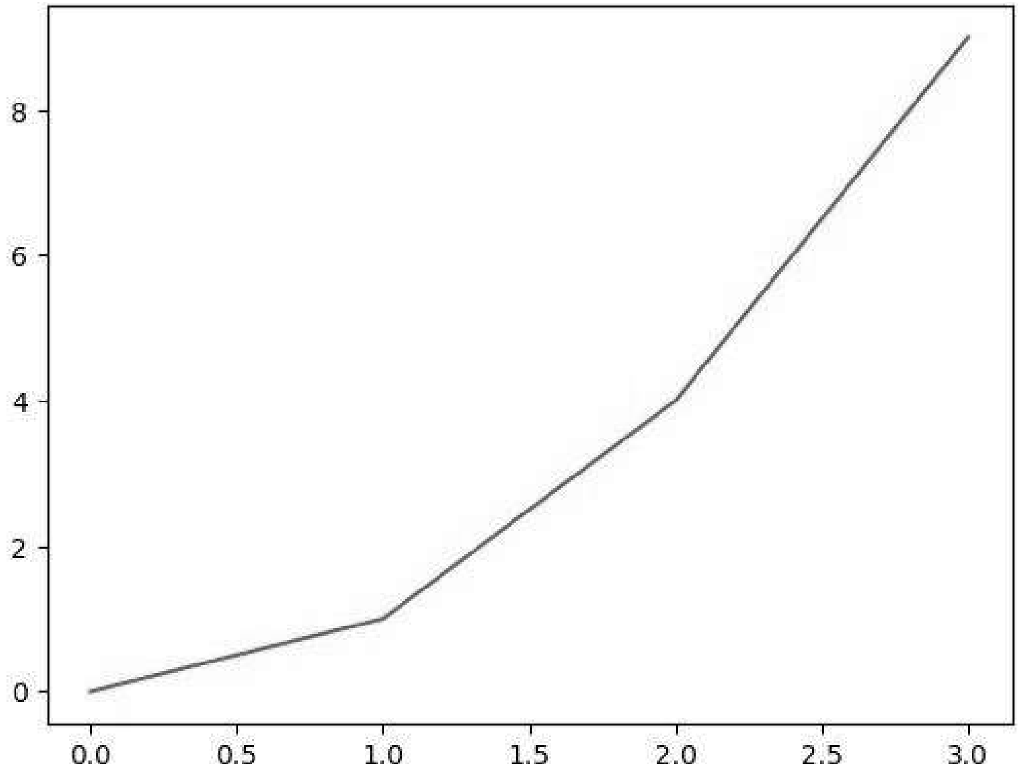
Numerical and scientific computations

```
a=np.array([[1,2,3],[5,6,7]])

a

array([[1, 2, 3],
       [5, 6, 7]])

b=np.array([[23,7,3],[8,6,9]])

b

array([[23,  7,  3],
       [ 8,  6,  9]])

c=np.array([[4,5],[6,7],[9,0]])

c

array([[4, 5],
       [6, 7],
       [9, 0]])

d=np.dot(a,c)

d

array([[ 43,  19],
       [119,  67]])

e=a@c

e

array([[ 43,  19],
       [119,  67]])

f=a+b

f

array([[24,  9,  6],
       [13, 12, 16]])

g=a*2

g
```

```
array([[ 2,  4,  6],
       [10, 12, 14]])

h=a+c.T

h

array([[ 5,  8, 12],
       [10, 13,  7]])

import numpy as np

a = np.array([1, 2, 3])
b = 5  # Scalar

result = a + b
print(result)

[6 7 8]

a = np.array([[1, 2, 3],
              [4, 5, 6]])
b = np.array([10, 20, 30])  # Shape (3,)

result = a + b
print(result)

[[11 22 33]
 [14 25 36]]

a = np.array([[1, 2, 3],
              [4, 5, 6]])
b = np.array([[10],
              [20]])  # Shape (2, 1)

result = a + b
print(result)

[[11 12 13]
 [24 25 26]]
```

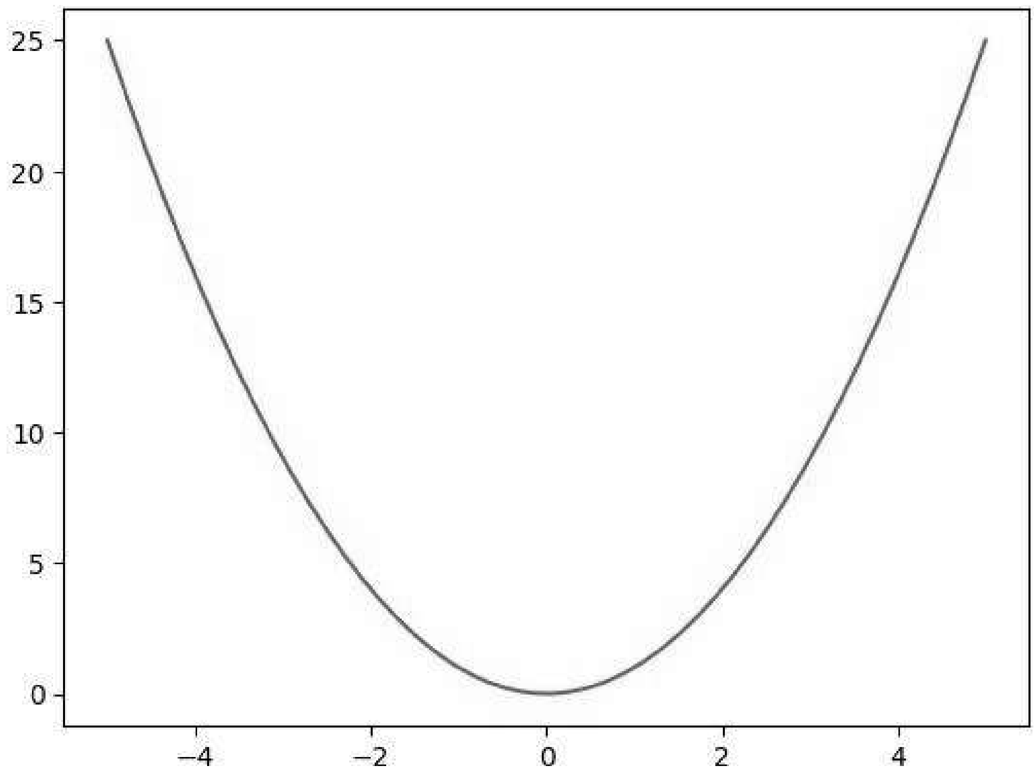
Visualization and Plotting

```
%matplotlib inline
import matplotlib.pyplot as plt

x = [0, 1, 2, 3]
y = [0, 1, 4, 9]

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



Make a plot of the function f (x) = x2 for −5 ≤ x ≤ 5

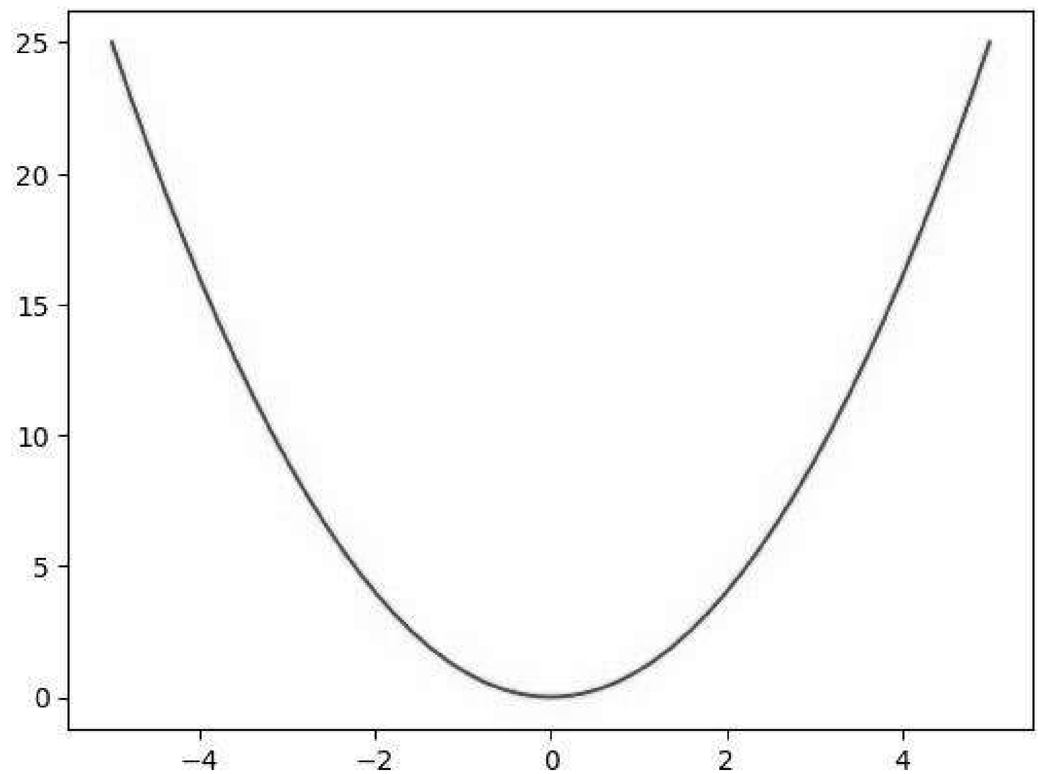
```
 %matplotlib inline
x = np.linspace(-5,5, 100)
plt.plot(x, x**2)
plt.show()
```



you can specify the color and format using the below table

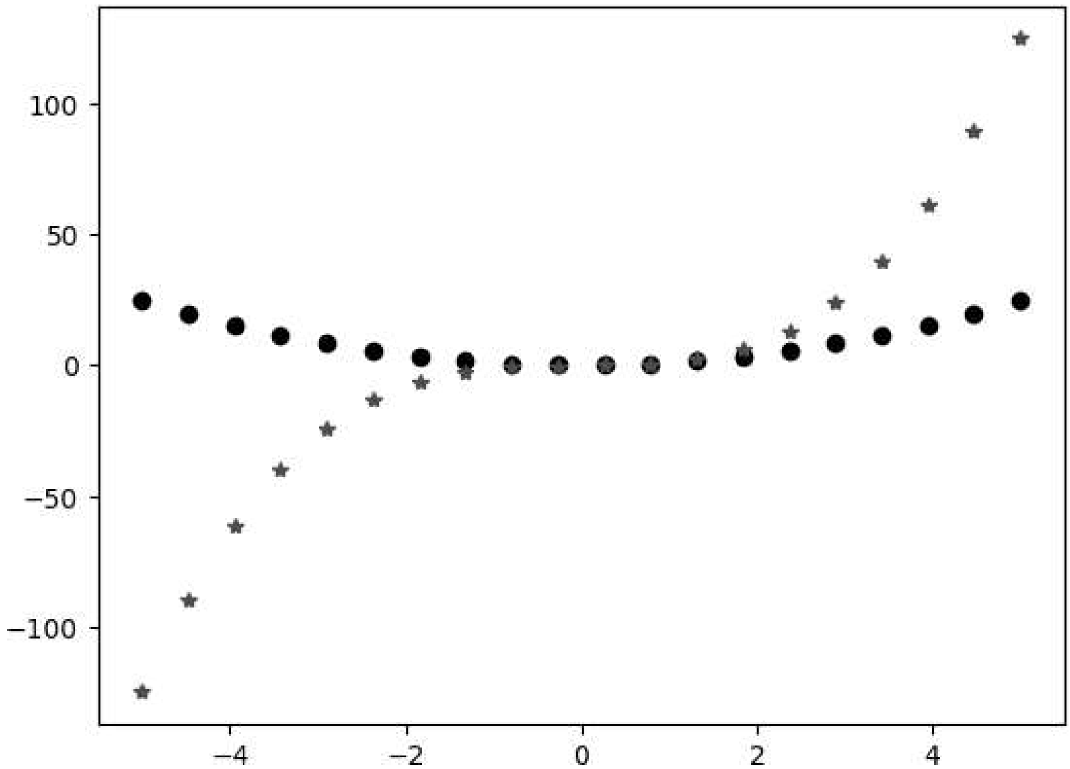
Symbol	Description	Symbol	Description
b	blue	T	T
g	green	s	square
r	red	d	diamond
c	cyan	v	triangle (down)
m	magenta	^	triangle (up)
y	yellow	<	triangle (left)
k	black	>	triangle (right)
w	white	p	pentagram
.	point	h	hexagram
o	circle	-	solid
x	x-mark	:	dotted
+	plus	-.	dashed–dotted
*	star	-	dashed

```
%matplotlib inline
x = np.linspace(-5,5, 100)
plt.plot(x, x**2,"m-")
plt.show()
```

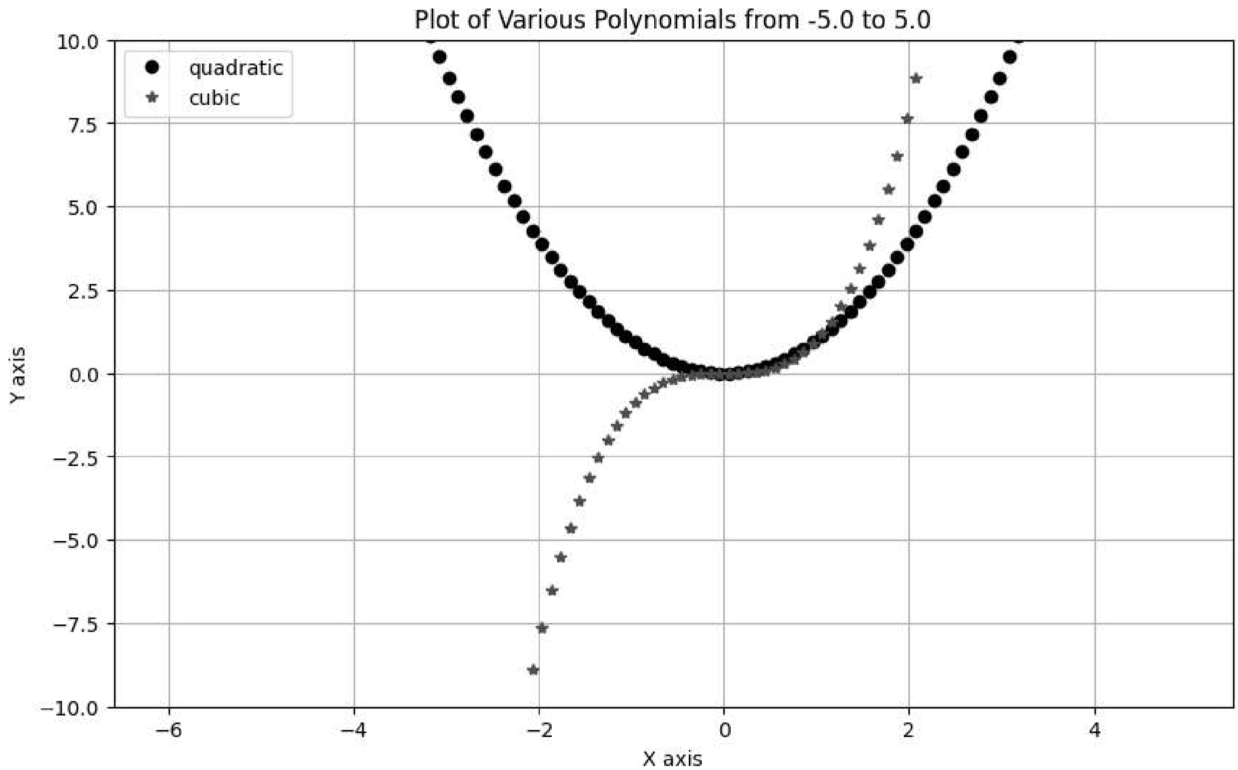


Make a plot of the function f (x) = x2 and g(x) = x3 for −5 ≤ x ≤ 5. Use different colors and markers for each function.

```
x = np.linspace(-5,5,20)
plt.plot(x, x**2, "ko")
plt.plot(x, x**3, "r*")
plt.show()
```



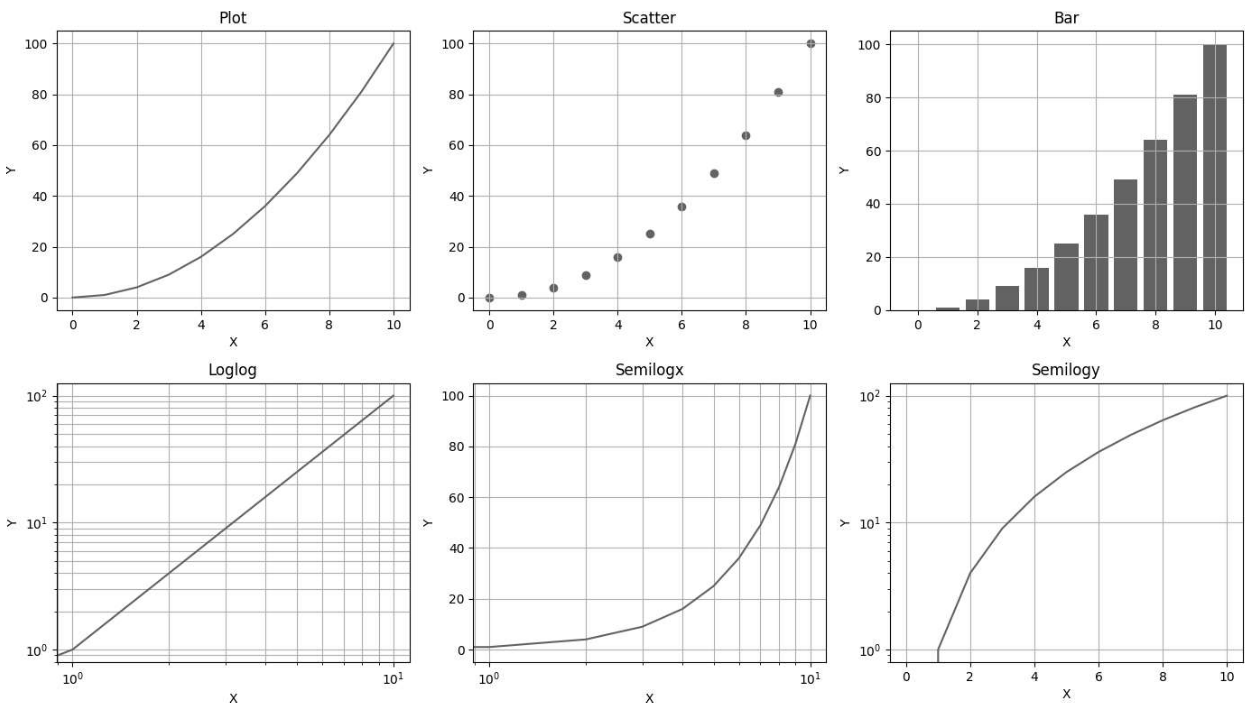

```
plt.figure(figsize = (10,6))
x = np.linspace(-5,5,100)
plt.plot(x, x**2, "ko", label = "quadratic")
plt.plot(x, x**3, "r*", label = "cubic")
plt.title(f"Plot of Various Polynomials from {x[0]} to {x[-1]}")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.legend(loc = 2)
plt.xlim(-6.6)
plt.ylim(-10,10)
plt.grid()
plt.show()
```



Given the lists x = np.arange(11) and y = x2, create a 2 × 3 subplot where each subplot plots x versus y using plot, scatter, bar, loglog, semilogx, and semilogy. Title and label each plot appropriately. Use a grid, but a legend is not necessary here.

```
x = np.arange(11)
y = x**2
plt.figure(figsize = (14, 8))
plt.subplot(2, 3, 1)
plt.plot(x,y)
plt.title("Plot")
plt.xlabel("X")
plt.ylabel("Y")
plt.grid()
plt.subplot(2, 3, 2)
plt.scatter(x,y)
plt.title("Scatter")
plt.xlabel("X")
plt.ylabel("Y")
plt.grid()
plt.subplot(2, 3, 3)
plt.bar(x,y)
plt.title("Bar")
plt.xlabel("X")
plt.ylabel("Y")
plt.grid()
plt.subplot(2, 3, 4)
```

```
plt.loglog(x,y)
plt.title("Loglog")
plt.xlabel("X")
plt.ylabel("Y")
plt.grid(which="both")
plt.subplot(2, 3, 5)
plt.semilogx(x,y)
plt.title("Semilogx")
plt.xlabel("X")
plt.ylabel("Y")
plt.grid(which="both")
plt.subplot(2, 3, 6)
plt.semilogy(x,y)
plt.title("Semilogy")
plt.xlabel("X")
plt.ylabel("Y")
plt.grid()
plt.tight_layout()
plt.show()
```



3D plotting

```
import numpy as np
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
plt.style.use("seaborn-v0_8-poster")
fig = plt.figure(figsize = (10,10))
```