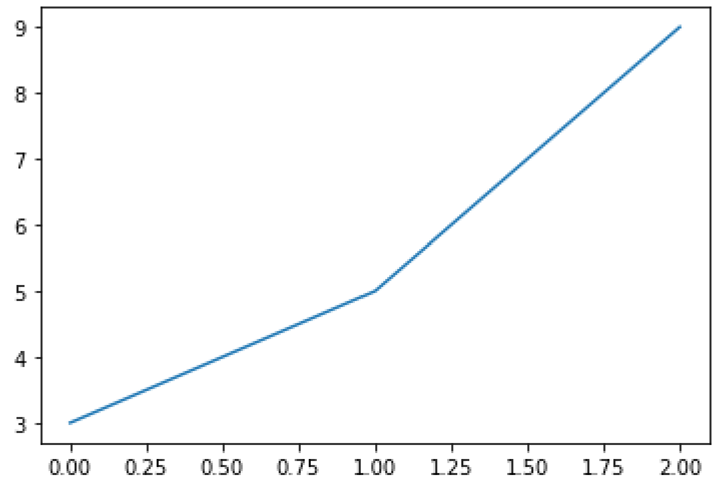
```
#!pip install matplotlib
# pyplot submodule
# no business case today
# schedule programming mock interviews as soon as we are done with done module
# Netflix case is due in 7 days
# Colab Link - https://colab.research.google.com/drive/1gzxy_wvDS4tCzM4-95dsMUpjUC8
```

```
import matplotlib.pyplot as plt
```

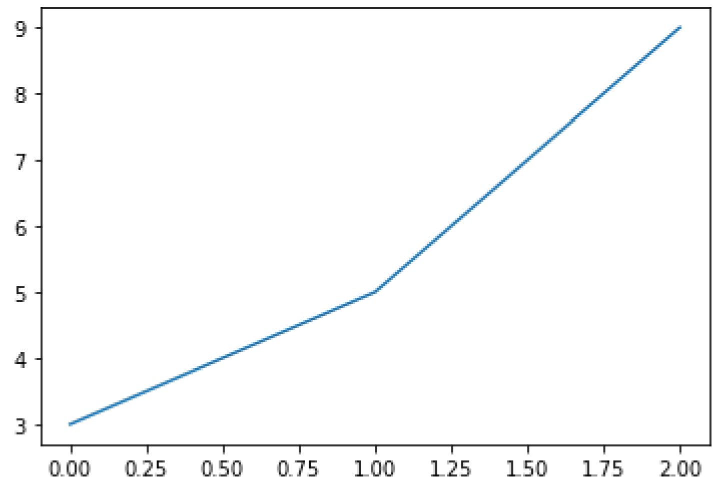
```
# (0, 3), (1, 5), (2, 9)
```

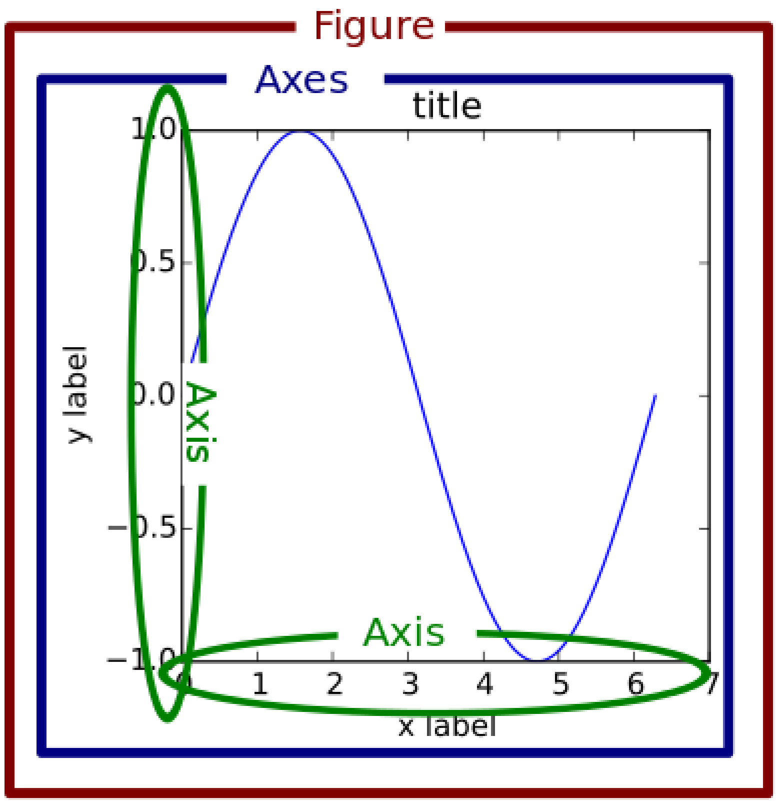
```
x_val = [0, 1, 2]
y_val = [3, 5, 9]
plt.plot(x_val, y_val)
```

```
[<matplotlib.lines.Line2D at 0x7ff22579e710>]
```

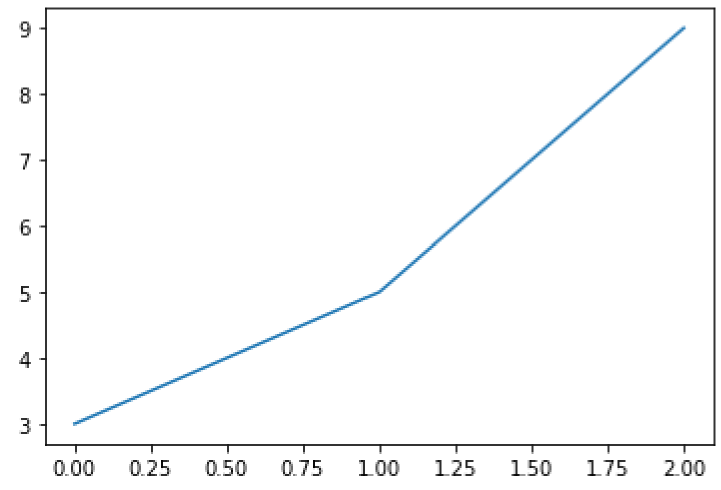


```
x_val = [0, 1, 2]
y_val = [3, 5, 9]
plt.plot(x_val, y_val)
plt.show()
```

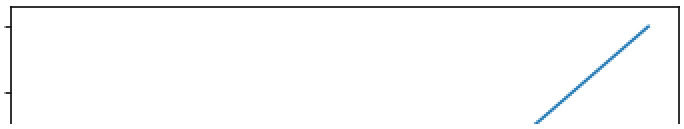




```
plt.plot(y_val)
plt.show()
```

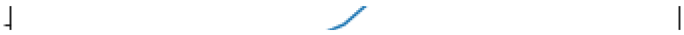


```
plt.plot(y_val)
plt.savefig("first-plot.jpg")
```



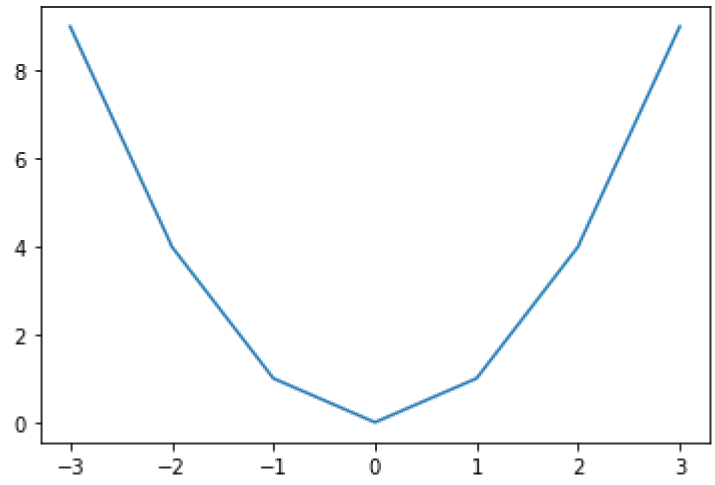
```
!ls
```

```
    first-plot.jpg  sample_data
```



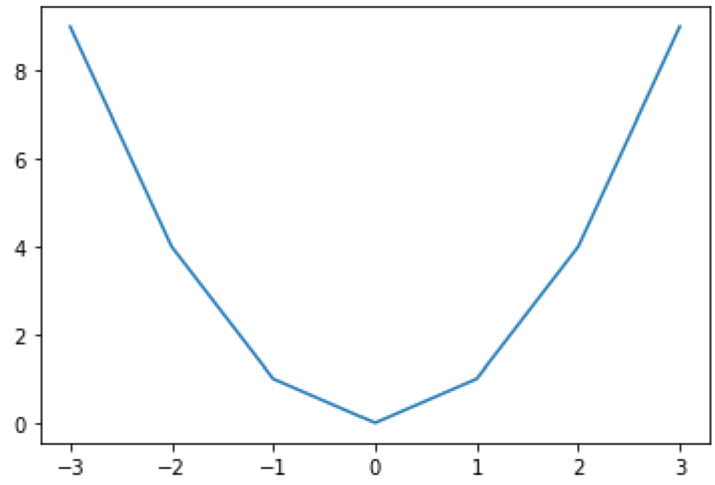
```
# y = x^2
```

```
x_val = [-3, -2, -1, 0, 1, 2, 3]
y_val = [9, 4, 1, 0, 1, 4, 9]
plt.plot(x_val, y_val)
plt.show()
```



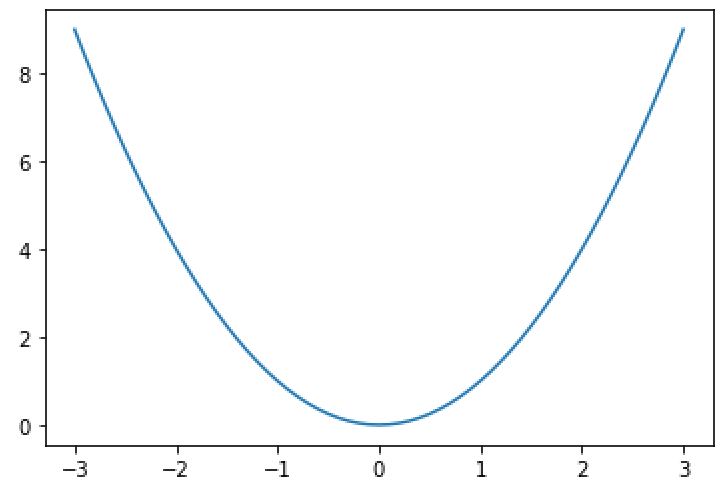
```
import numpy as np
```

```
x_val = np.arange(-3, 4, 1)
y_val = x_val ** 2
plt.plot(x_val, y_val)
plt.show()
```

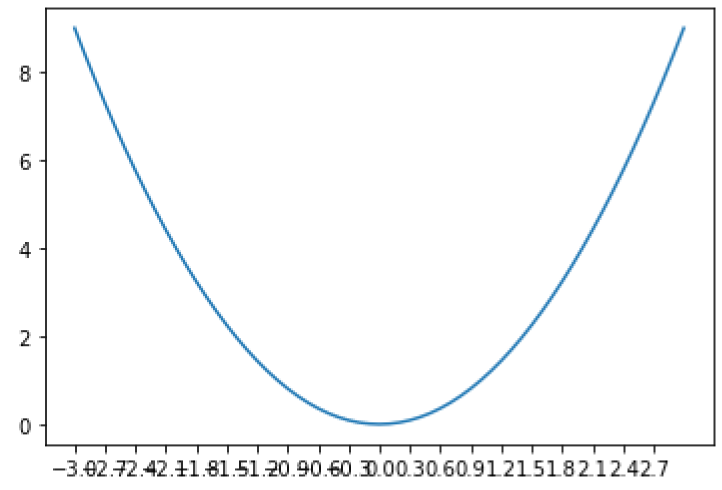


```
x_val = np.arange(-3, 3.1, 0.1)
```

```
y_val = x_val ** 2
plt.plot(x_val, y_val)
plt.show()
```

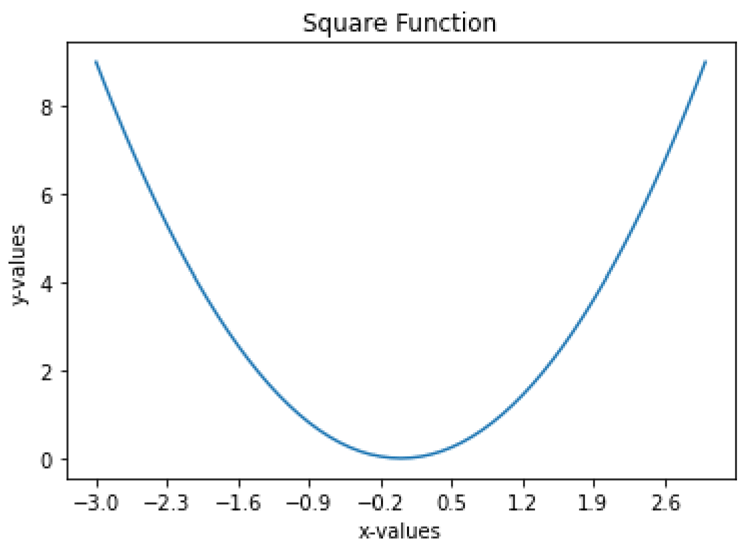


```
x_val = np.arange(-3, 3.1, 0.1)
y_val = x_val ** 2
plt.plot(x_val, y_val)
plt.xticks(np.arange(-3, 3, 0.3))
plt.show()
```

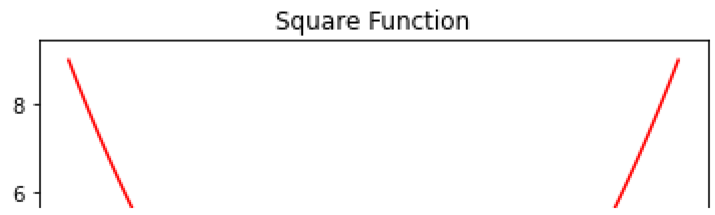


```
x_val = np.arange(-3, 3.1, 0.1)
y_val = x_val ** 2
plt.plot(x_val, y_val)
plt.xticks(np.arange(-3, 3, 0.3), rotation=90)
plt.show()
```

```
x_val = np.arange(-3, 3.1, 0.1)
y_val = x_val ** 2
plt.plot(x_val, y_val)

plt.title("Square Function")
plt.xlabel("x-values")
plt.ylabel("y-values")
plt.xticks(np.arange(-3, 3, 0.7))
plt.show()
```

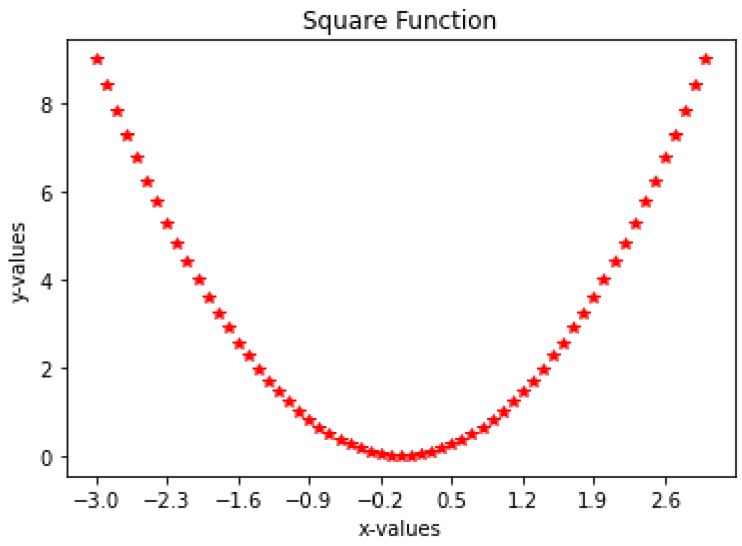


```
x_val = np.arange(-3, 3.1, 0.1)
y_val = x_val ** 2

plt.plot(x_val, y_val, color="r")

plt.title("Square Function")
plt.xlabel("x-values")
plt.ylabel("y-values")

plt.xticks(np.arange(-3, 3, 0.7))

plt.show()
```

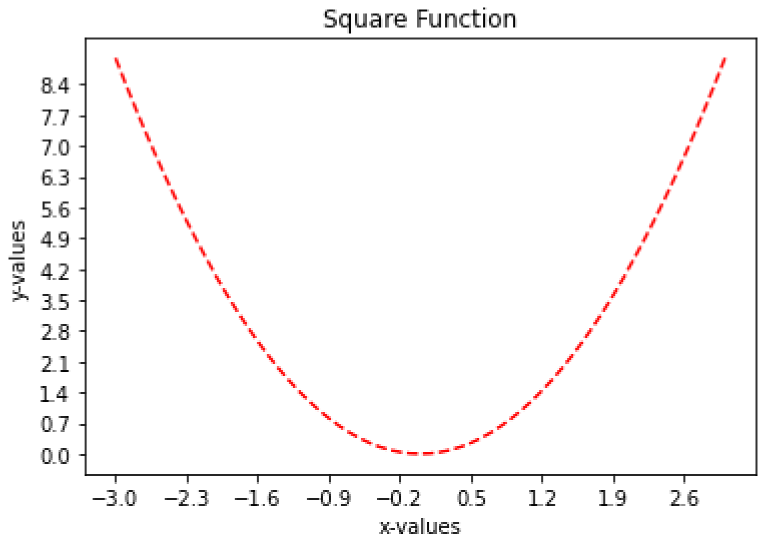


```
* for start-shaped points,
o for circular points
. for smaller points
v for upside down triangle
- for single solid line
-- for dashed line
... and so on
```

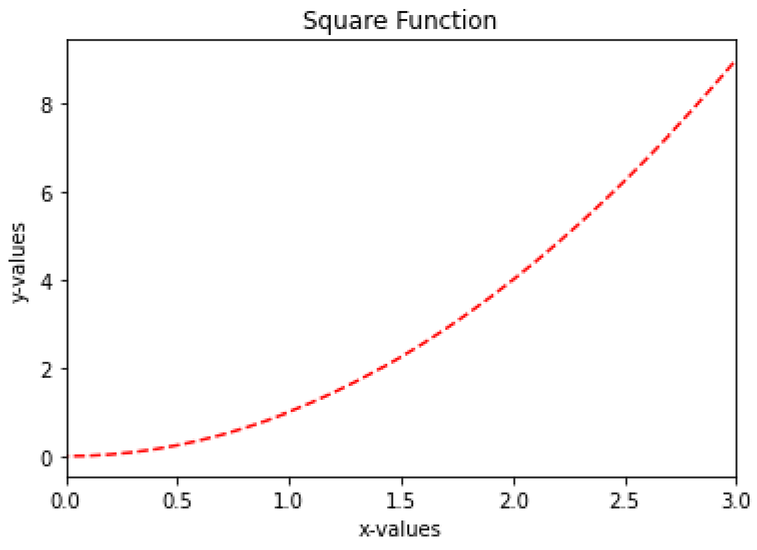
```
x_val = np.arange(-3, 3.1, 0.1)
y_val = x_val ** 2

plt.plot(x_val, y_val,'r*')

plt.title("Square Function")
plt.xlabel("x-values")
plt.ylabel("y-values")

plt.xticks(np.arange(-3, 3, 0.7))

plt.show()
```



```
x_val = np.arange(-3, 3.1, 0.1)
y_val = x_val ** 2

plt.plot(x_val, y_val,'r--')

plt.title("Square Function")
plt.xlabel("x-values")
plt.ylabel("y-values")

plt.xticks(np.arange(-3, 3, 0.7))
plt.yticks(np.arange(0, 9, 0.7))

plt.show()
```

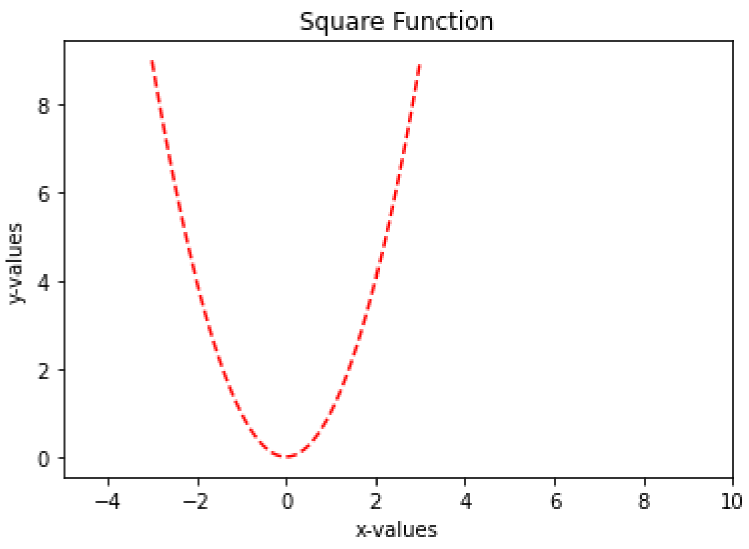


```
x_val = np.arange(-3, 3.1, 0.1)
y_val = x_val ** 2

plt.plot(x_val, y_val,'r--')

plt.title("Square Function")
plt.xlabel("x-values")
plt.ylabel("y-values")

plt.xlim(0, 3)

plt.show()
```



```
x_val = np.arange(-3, 3.1, 0.1)
y_val = x_val ** 2
```

```
plt.plot(x_val, y_val,'r--')

plt.title("Square Function")
plt.xlabel("x-values")
plt.ylabel("y-values")

plt.xlim(-5, 10)

plt.show()
```

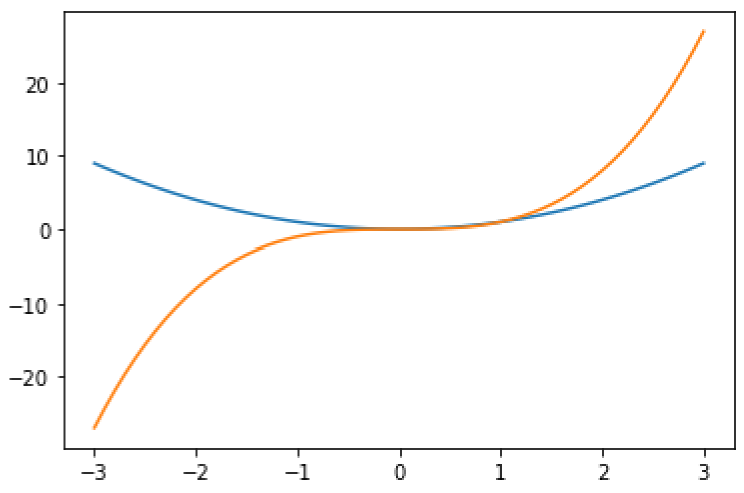


```
x_val = np.arange(-3, 3.1, 0.1)
y_val = x_val ** 2

plt.plot(x_val, y_val,'r--')

plt.title("Square Function")
plt.xlabel("x-values")
plt.ylabel("y-values")

plt.xlim(-5, 10)
plt.ylim(0, 5)

plt.show()
```

Square Function

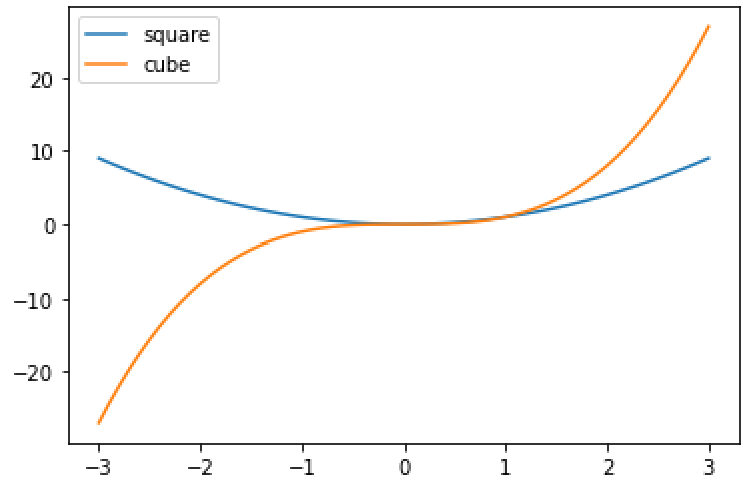
```
# 1. plt.xlim(0, 10)
# 2. x = np.linspace(1, 10, 20)
# 3. plt.show()
# 4. plt.xlabel('X-Label')
# 5. plt.plot(x, label='Example Plot')
# 2-3-1-4-5
# 2-5-4-1-3
# 5-4-1-3-2
# 1-2-4-3-5
```

```
x_val = np.arange(-3, 3.1, 0.1)
y_1 = x_val ** 2
y_2 = x_val ** 3

plt.plot(x_val, y_1, label="square")
plt.plot(x_val, y_2, label="cube")

plt.show()
```

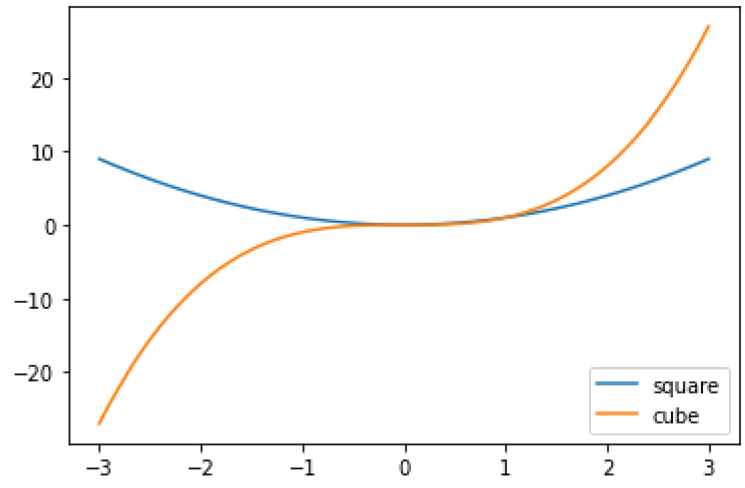


```
x_val = np.arange(-3, 3.1, 0.1)
y_1 = x_val ** 2
y_2 = x_val ** 3

plt.plot(x_val, y_1, label="square")
plt.plot(x_val, y_2, label="cube")
plt.legend()

plt.show()
```

```
x_val = np.arange(-3, 3.1, 0.1)
y_1 = x_val ** 2
y_2 = x_val ** 3

plt.plot(x_val, y_1)
plt.plot(x_val, y_2)
plt.legend(["square", "cube"])

plt.show()
```

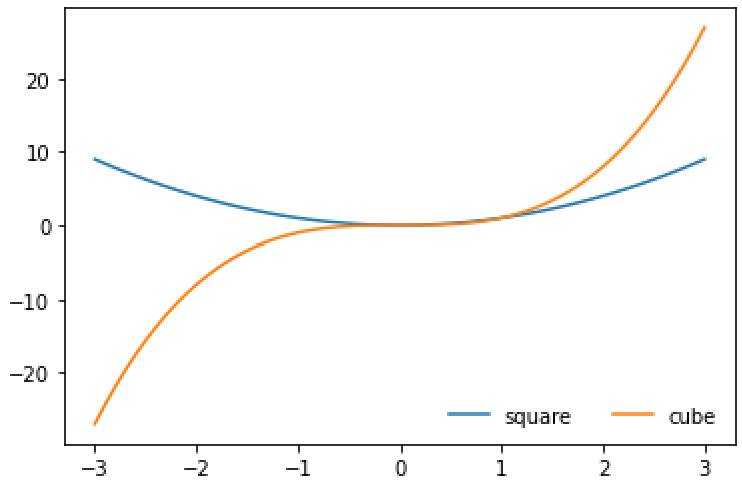


```
x_val = np.arange(-3, 3.1, 0.1)
y_1 = x_val ** 2
y_2 = x_val ** 3

plt.plot(x_val, y_1)
plt.plot(x_val, y_2)
plt.legend(["square", "cube"], loc="lower right")

plt.show()
```



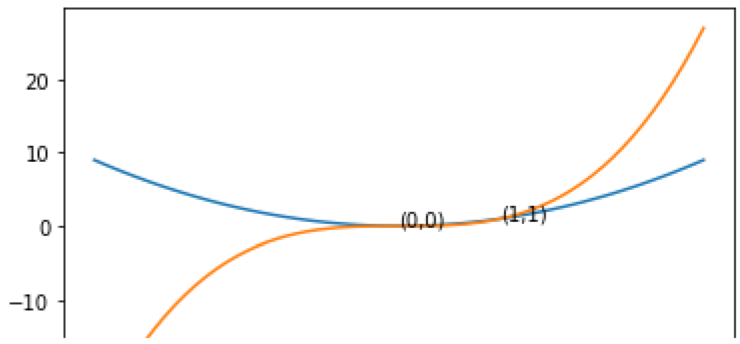
```
upper center
upper left
upper right
lower right ... etc
```

```
x_val = np.arange(-3, 3.1, 0.1)
y_1 = x_val ** 2
y_2 = x_val ** 3

plt.plot(x_val, y_1)
plt.plot(x_val, y_2)
plt.legend(["square", "cube"], loc="lower right", ncol=2, frameon = False)

plt.show()
```

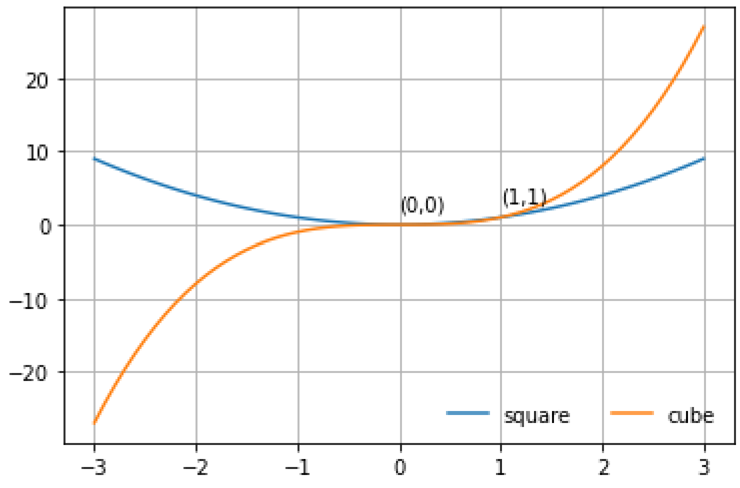


```
# (0, 0), (1, 1)
```

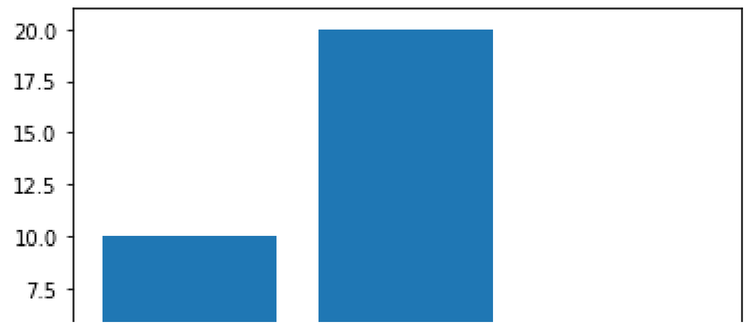
```
x_val = np.arange(-3, 3.1, 0.1)
y_1 = x_val ** 2
y_2 = x_val ** 3

plt.plot(x_val, y_1)
plt.plot(x_val, y_2)

plt.text(0, 0, "(0,0)")
plt.text(1, 1, "(1,1)")

plt.legend(["square", "cube"], loc="lower right", ncol=2, frameon = False)

plt.show()
```



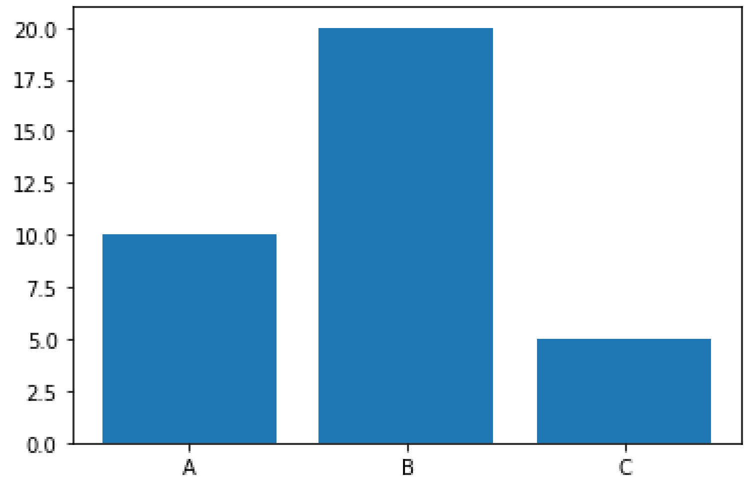
```
x_val = np.arange(-3, 3.1, 0.1)
y_1 = x_val ** 2
y_2 = x_val ** 3

plt.plot(x_val, y_1)
plt.plot(x_val, y_2)

plt.text(0, 0+2, "(0,0)")
plt.text(1, 1+2, "(1,1)")

plt.legend(["square", "cube"], loc="lower right", ncol=2, frameon = False)
plt.grid()

plt.show()
```



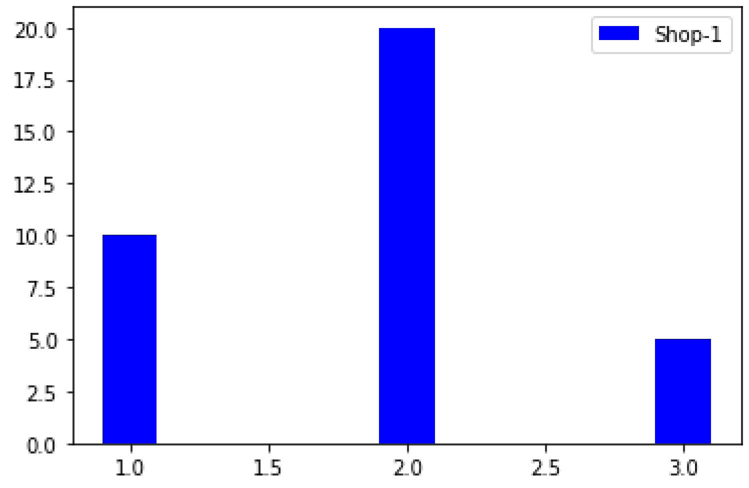
```
# types of plots


# Fruit1 costs 10rs
# Fruit2 costs 20rs
# Fruit3 costs 5rs


plt.bar([1, 2, 3], [10, 20, 5])
plt.show()
```



```
plt.bar(["A", "B", "C"], [10, 20, 5])
plt.show()
```

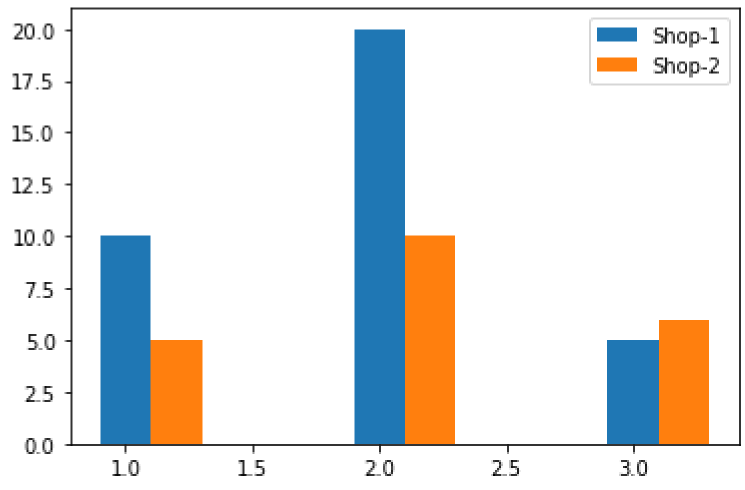


```
plt.bar([1, 2, 3], [10, 20, 5], width=0.2, color="b")
plt.legend(["Shop-1"])
plt.show()
```

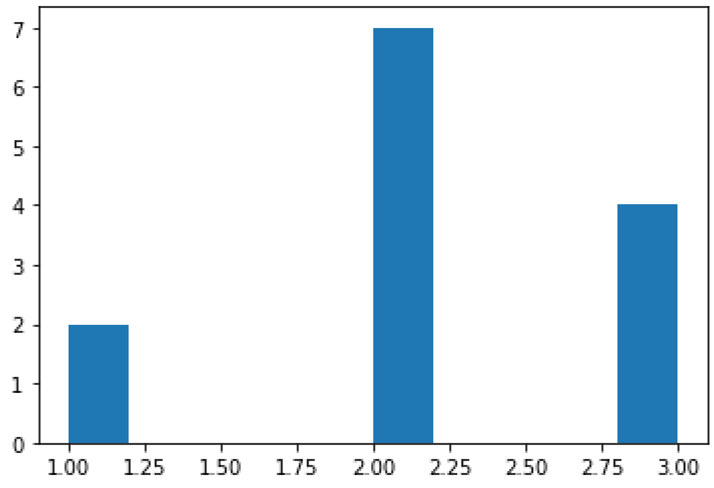


```
# Fruit1 costs 5rs
# Fruit2 costs 10rs
# Fruit3 costs 6rs
```

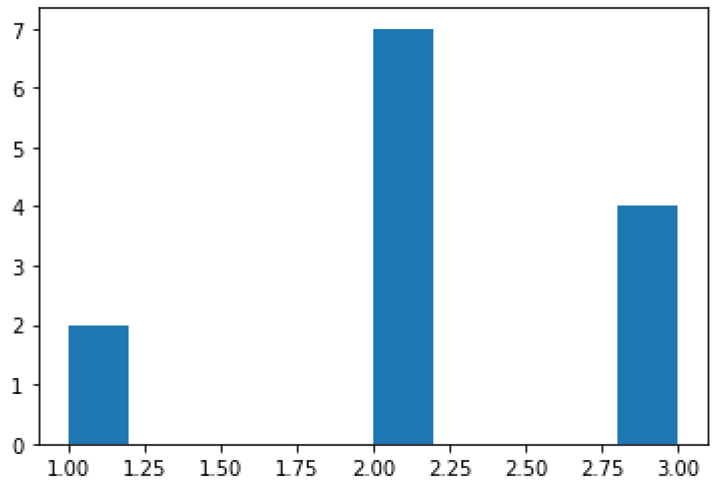
```
plt.bar([1, 2, 3], [10, 20, 5], width=0.2)
plt.bar(np.array([1, 2, 3])+0.2, [5, 10, 6], width=0.2)
plt.legend(["Shop-1", "Shop-2"])
plt.show()
```



```
vals = [1, 2, 3, 1, 2, 3, 2, 3, 2, 3, 2, 2, 2]
plt.hist(vals)
plt.show()
```



```
vals = [1, 2, 3, 1, 2, 3, 2, 3, 2, 3, 2, 2, 2]
counts, bins, patches = plt.hist(vals)
plt.show()
```



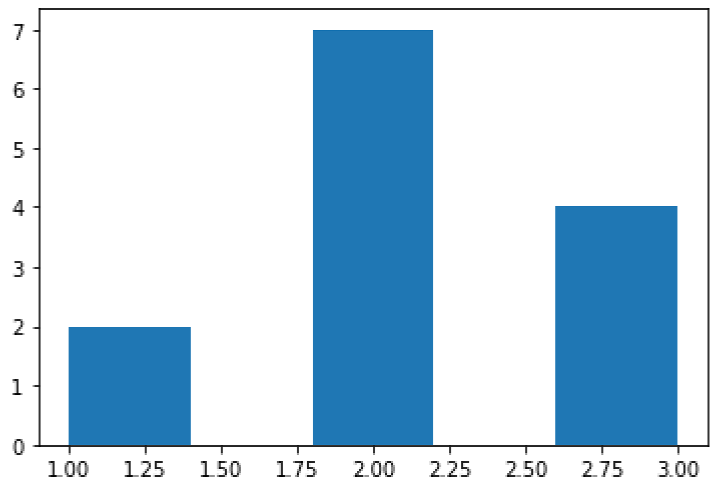
```
len(counts)
```

```
    10
```

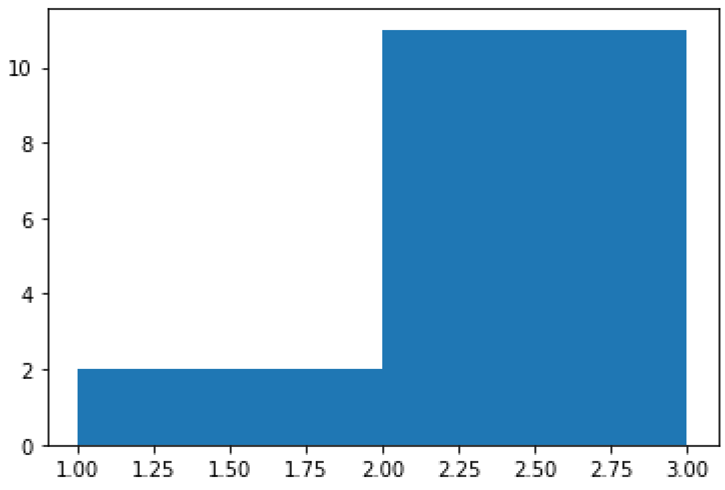
```
bins
```

```
    array([1. , 1.2, 1.4, 1.6, 1.8, 2. , 2.2, 2.4, 2.6, 2.8, 3. ])
```

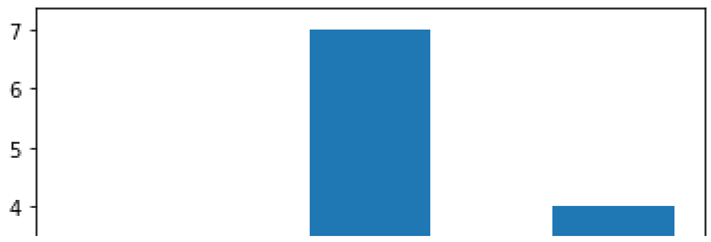
```
vals = [1, 2, 3, 1, 2, 3, 2, 3, 2, 3, 2, 2, 2]
plt.hist(vals, bins=5)
plt.show()
```



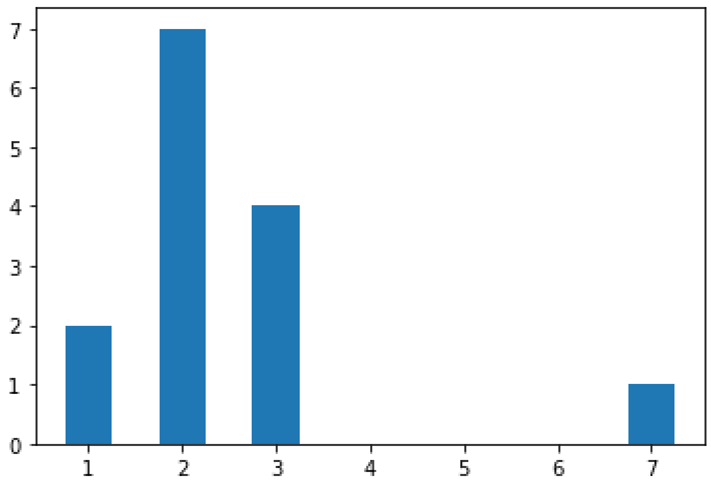
```
vals = [1, 2, 3, 1, 2, 3, 2, 3, 2, 3, 2, 2, 2]
plt.hist(vals, bins=2)
plt.show()
```



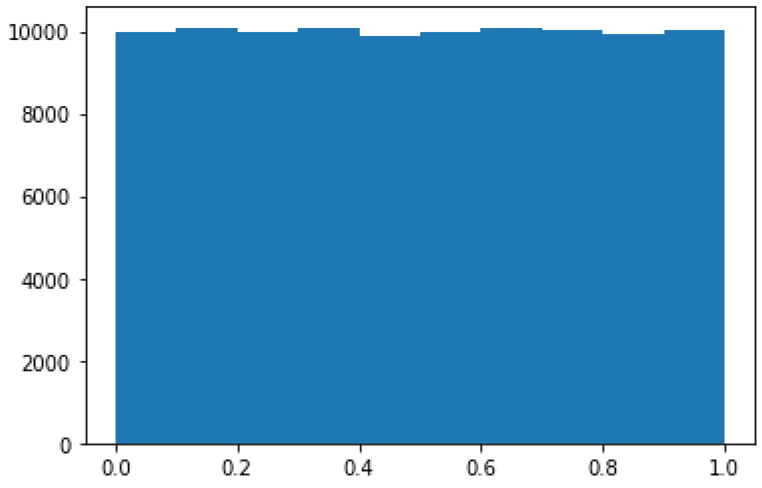
```
vals = [1, 2, 3, 1, 2, 3, 2, 3, 2, 3, 2, 2, 2]
plt.hist(vals, [0.75, 1.25, 1.75, 2.25, 2.75, 3.25])
plt.show()
```



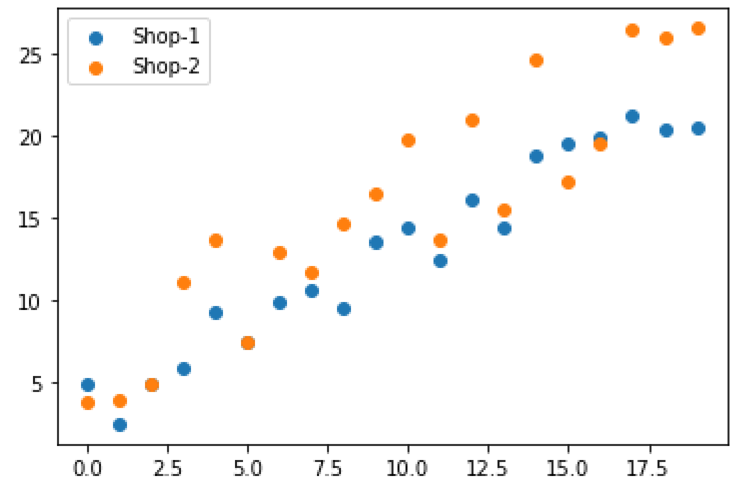
```
vals = [1, 2, 3, 1, 2, 3, 2, 3, 2, 3, 2, 2, 2, 7]
plt.hist(vals, [0.75, 1.25, 1.75, 2.25, 2.75, 3.25, 6.75, 7.25])
plt.show()
```



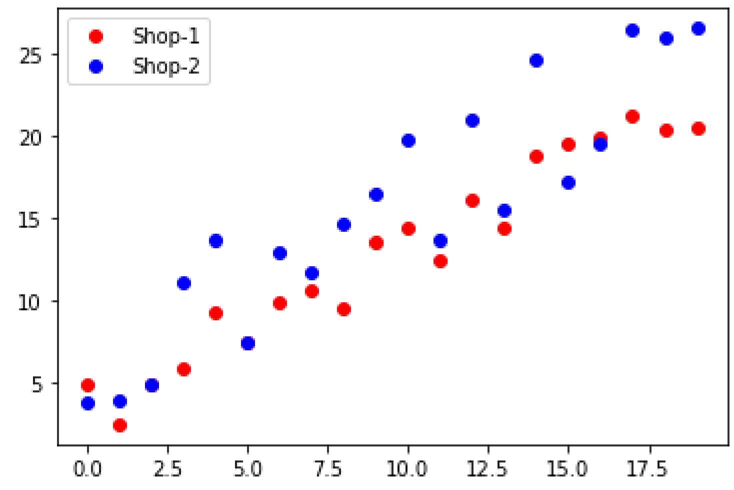
```
# where we want to see frequency/data distribution


vals = np.random.rand(100000)
plt.hist(vals)
plt.show()
```



```
x = np.arange(20)
y1 = np.linspace(1, 20, 20) + np.random.rand(20)*5
y2 = np.linspace(1, 20, 20) + np.random.rand(20)*10
plt.scatter(x, y1)
plt.scatter(x, y2)
plt.legend(["Shop-1", "Shop-2"])
plt.show()
```

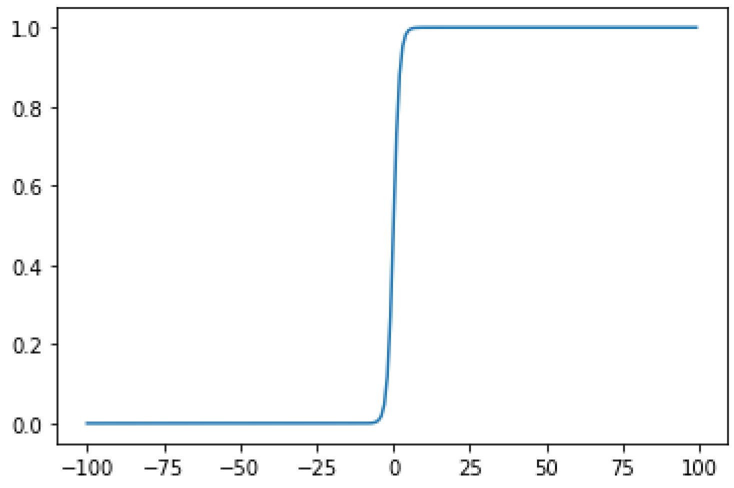


```
plt.plot(x, y1, "ro")
plt.plot(x, y2, "bo")
plt.legend(["Shop-1", "Shop-2"])
plt.show()
```

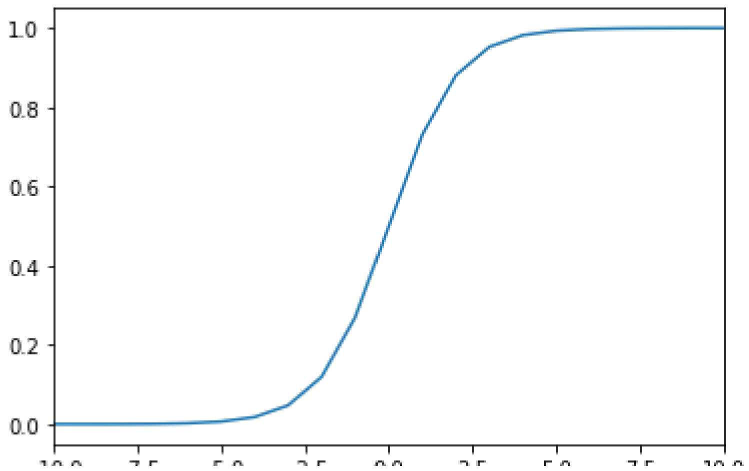


```
x = np.arange(20)
y1 = np.linspace(1, 20, 20) + np.random.rand(20)*5
y2 = np.linspace(1, 20, 20) + np.random.rand(20)*10
plt.scatter(x, y1, s=y1*5)
plt.scatter(x, y2, s=y2*5)
plt.legend(["Shop-1", "Shop-2"])
plt.show()
```

```
# sigmoid funtion / logistic

import math
sigmoid = lambda x : 1/(1 + math.exp(-x))
x = np.arange(-100, 100)
y = [sigmoid(i) for i in x]
plt.plot(x, y)
plt.show()
```

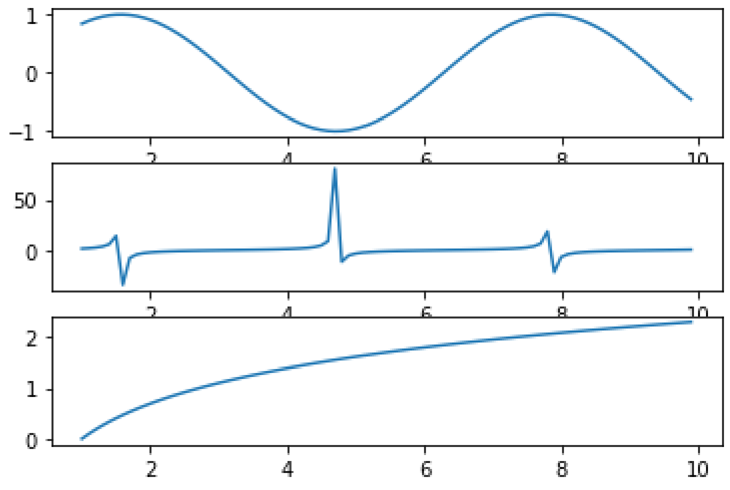


```
import math
sigmoid = lambda x : 1/(1 + math.exp(-x))
x = np.arange(-100, 100)
y = [sigmoid(i) for i in x]
plt.plot(x, y)
plt.xlim(-10, 10)
plt.show()

sigmoid = lambda x : 1/(1 + math.exp(-x))
x = np.arange(-100, 100)
y = [sigmoid(i) for i in x]
plt.plot(x, y)
plt.xlim(-10, 10)
plt.show()
```

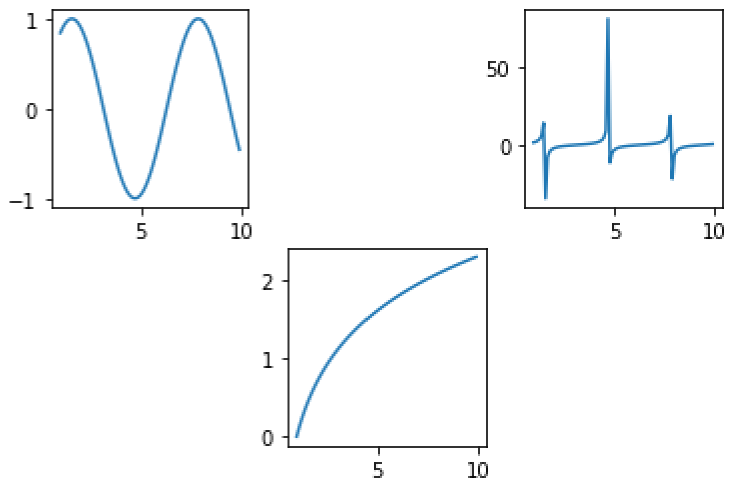


```
# subplots
```

```
x = np.arange(1, 10, 0.1)
y1 = np.sin(x)
y2 = np.tan(x)
y3 = np.log(x)


fig, ax = plt.subplots(3, 1)

ax[0].plot(x, y1)
ax[1].plot(x, y2)
ax[2].plot(x, y3)

plt.show()
```



```
x = np.arange(1, 10, 0.1)
y1 = np.sin(x)
y2 = np.tan(x)
y3 = np.log(x)

plt.figure()

plt.subplot(2, 3, 1) # remember that indexing starts from 1 here
plt.plot(x, y1)
plt.subplot(2, 3, 3)
plt.plot(x, y2)

plt.subplot(2, 3, 5)
plt.plot(x, y3)

plt.show()
```



```
fig = plt.figure()
ax231 = fig.add_subplot(231)
ax231.plot(x, y1)
ax233 = fig.add_subplot(233)
ax233.plot(x, y2)
ax235 = fig.add_subplot(235)
ax235.plot(x, y3)
plt.show()
```

✓ 0s completed at 00:12