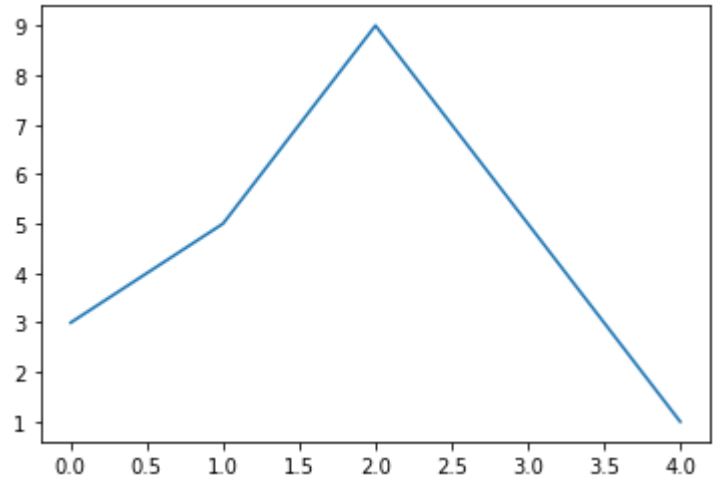
In [ ]:
```
!pip install matplotlib
```

In [1]:
```
import matplotlib.pyplot as plt
# %matplotlib inline
```

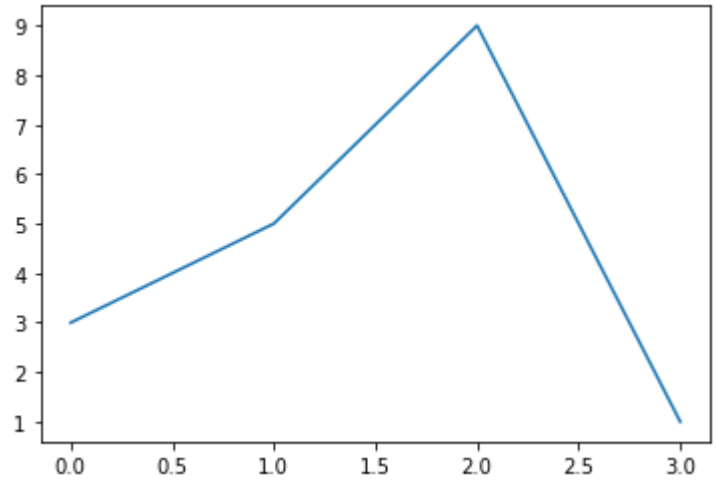
In [ ]:
```

```

In [2]:
```
x_values = [0, 1, 2, 4]
y_values = [3, 5, 9, 1]
plt.plot(x_values ,y_values)
plt.show()
```



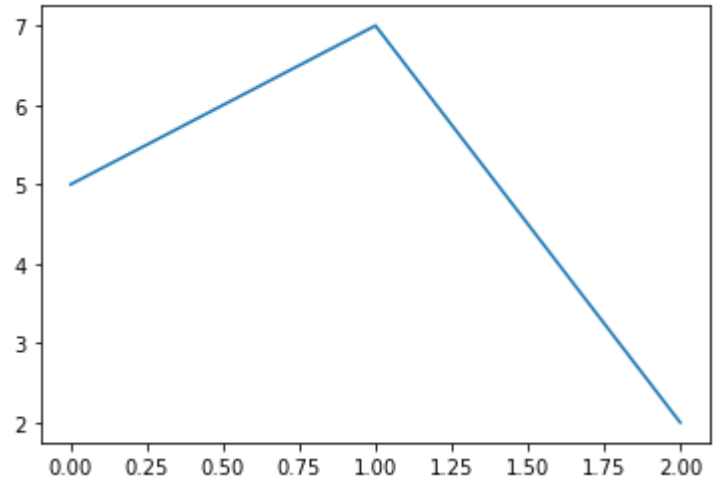
In [ ]:
```

```

In [3]:
```
plt.plot(y_values)
plt.show()
```



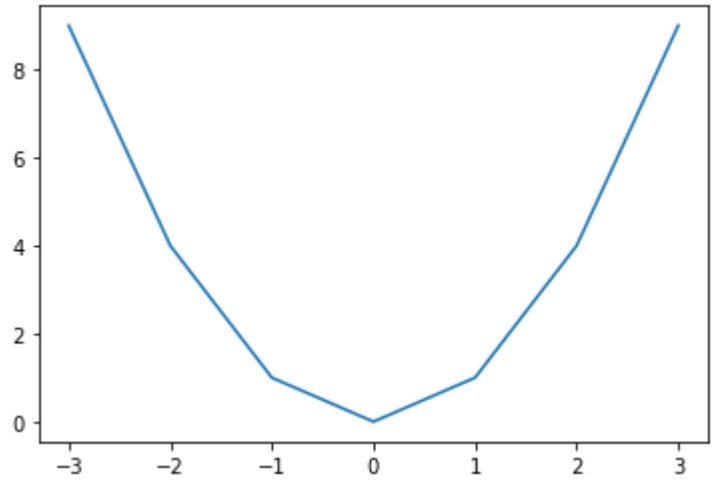
In [5]:
```
x = [10,20,30]
y = [5, 7, 2]
plt.plot(x, y)
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x7fc231ae7430>]
```



In [6]:
```
plt.plot([-3, -2, -1, 0, 1, 2, 3], [9, 4, 1, 0, 1, 4, 9])
```

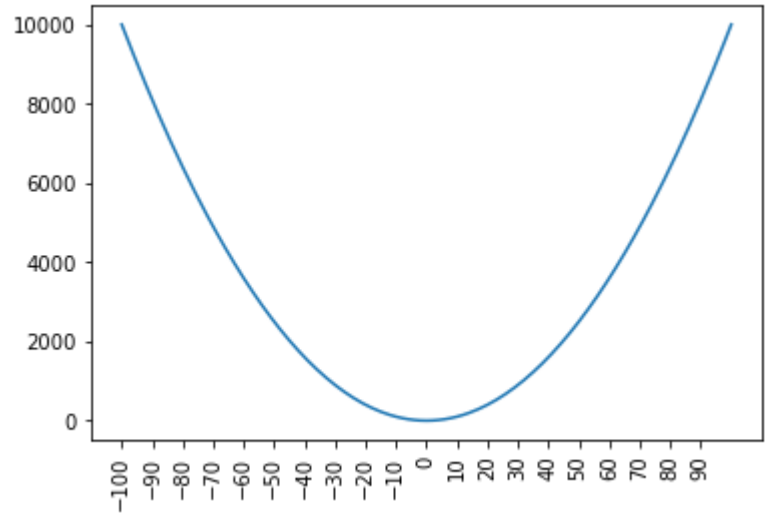
Out[6]:
```
[<matplotlib.lines.Line2D at 0x7fc220dbbe50>]
```



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

In [11]:
```
x= np.arange(-100, 101)
y = x**2
```

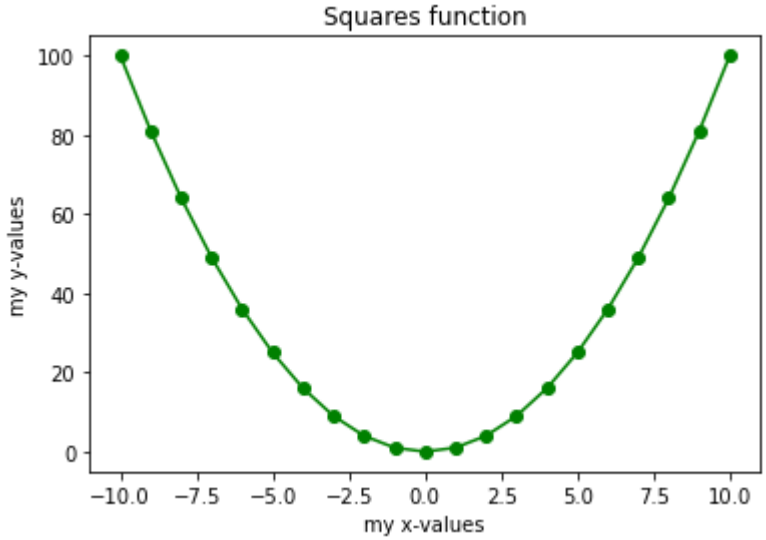
In [19]:
```
plt.plot(x, y)
plt.xticks(np.arange(-100, 100, 10), rotation=90)
# plt.yticks
plt.show()
```



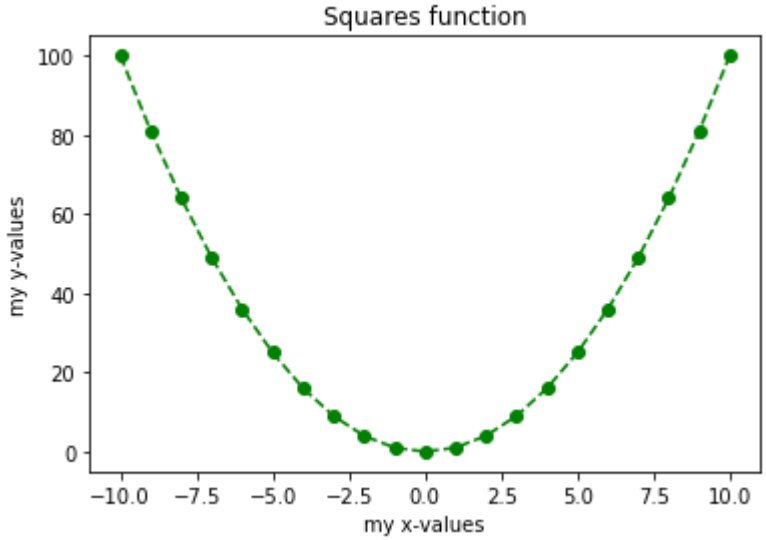
Styling and Labelling

In [31]:
```
x = np.arange(-10, 11, 1)
y = x**2
```

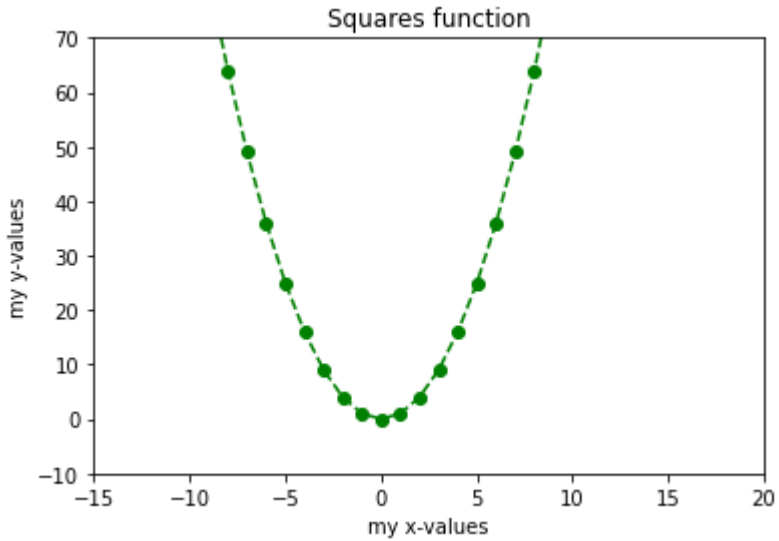
In [39]:
```
plt.plot(x, y, color='green', marker='o', linestyle='solid', )
plt.xlabel("my x-values")
plt.ylabel("my y-values")
plt.title("Squares function")
plt.show()
```



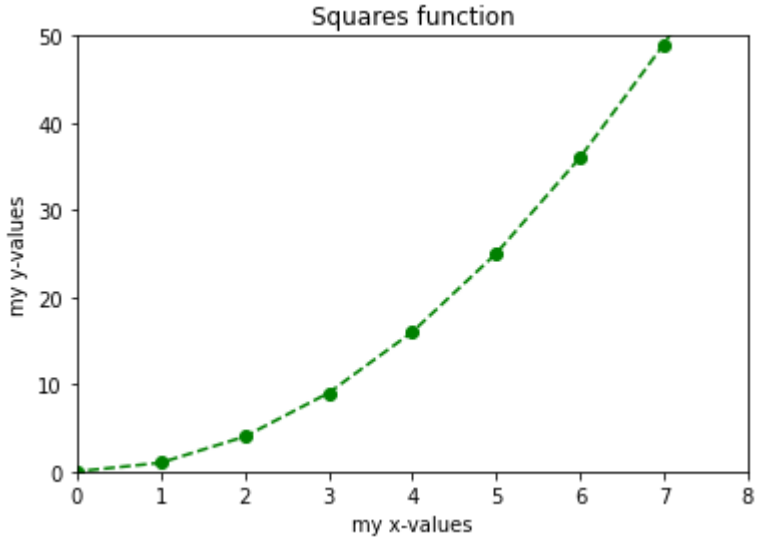
In [41]:
```
plt.plot(x, y, 'go--')
plt.xlabel("my x-values")
plt.ylabel("my y-values")
plt.title("Squares function")
plt.show()
```



In [47]:
```
plt.plot(x, y, 'go--')
plt.xlabel("my x-values")
plt.ylabel("my y-values")
plt.title("Squares function")
plt.xlim(-15, 20)
plt.ylim(-10, 70)
plt.show()
```



In [52]:
```
plt.plot(x, y, 'go--')
plt.xlabel("my x-values")
plt.ylabel("my y-values")
plt.title("Squares function")
plt.xlim(0,8)
plt.ylim(0,50)
plt.show()
```



Q-In what order should you run the following commands in order to generate and display a plot?

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')

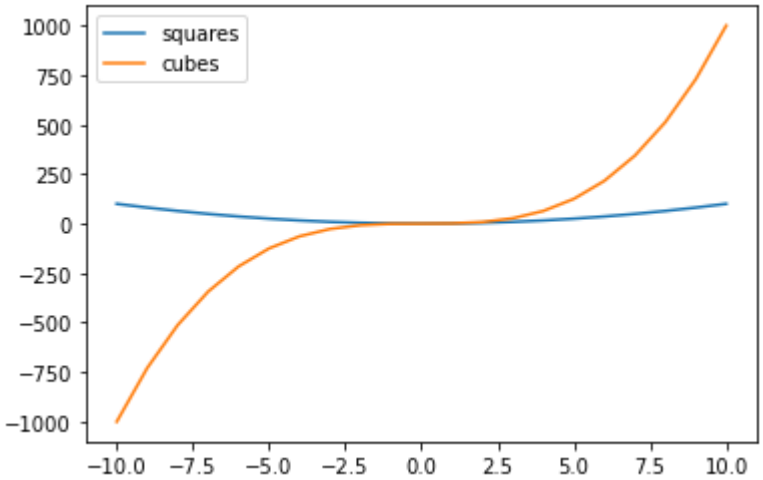
- 2-3-1-4-5
- 2-5-4-1-3
- 5-4-1-3-2
- 1-2-4-3-5

In [54]:
```
x
```

Out[54]:
```
array([-10,  -9,  -8,  -7,  -6,  -5,  -4,  -3,  -2,  -1,   0,   1,   2,
         3,   4,   5,   6,   7,   8,   9,  10])
```

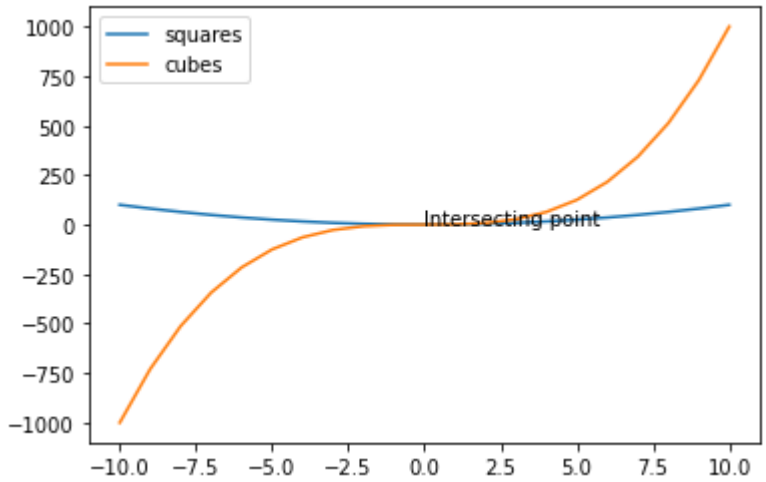
In [55]:
```
y2 = x**2
y3 = x**3
```

In [74]:
```
plt.plot(x, y2, label="squares")
plt.plot(x, y3, label="cubes")
plt.legend()
# plt.grid()
# plt.axis("off")
plt.show()
```



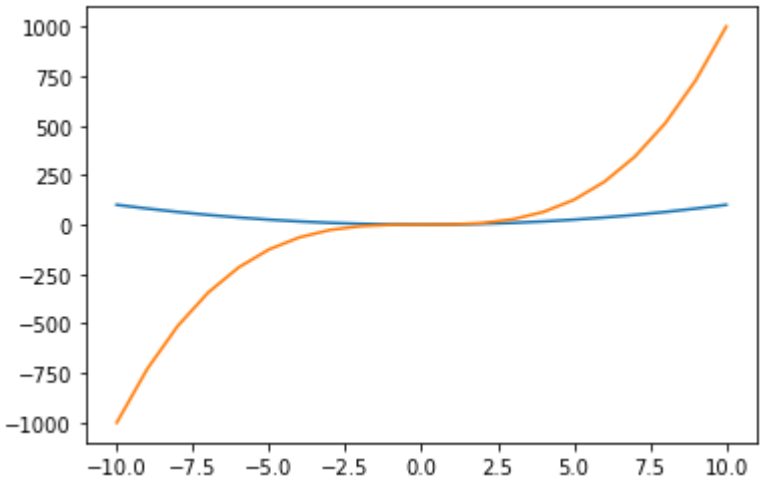
In [ ]:
```
```

In [76]:
```
plt.plot(x, y2, label="squares")
plt.plot(x, y3, label="cubes")
plt.legend()
# plt.annotate
plt.text(0,0, "Intersecting point")
plt.show()
```

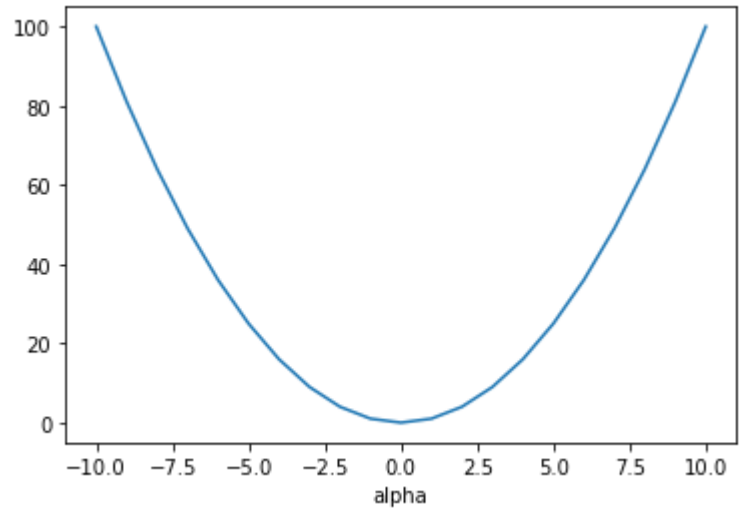


In [ ]:

In [79]:
```
plt.plot(x, y2)
plt.plot(x, y3)

plt.show()
```

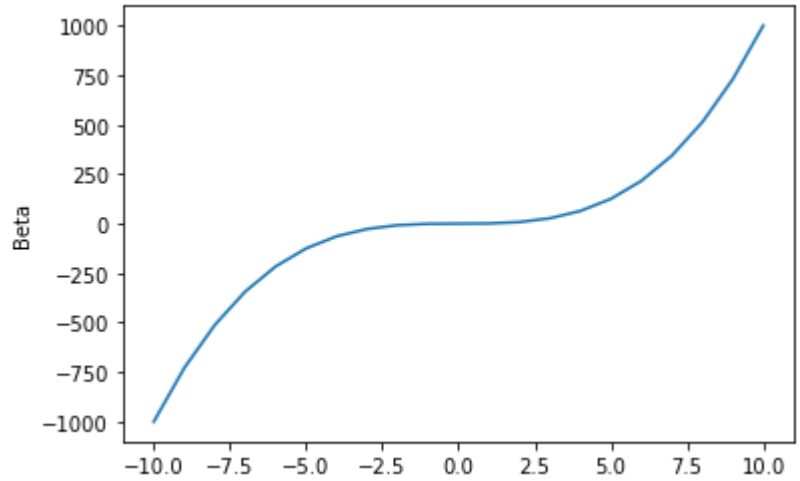


In [87]:
```
plt.plot(x, y2)
plt.xlabel("alpha")
plt.savefig("mygraph")
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```

In [85]:
```
plt.plot(x, y3)
plt.ylabel("Beta")
plt.show()
```

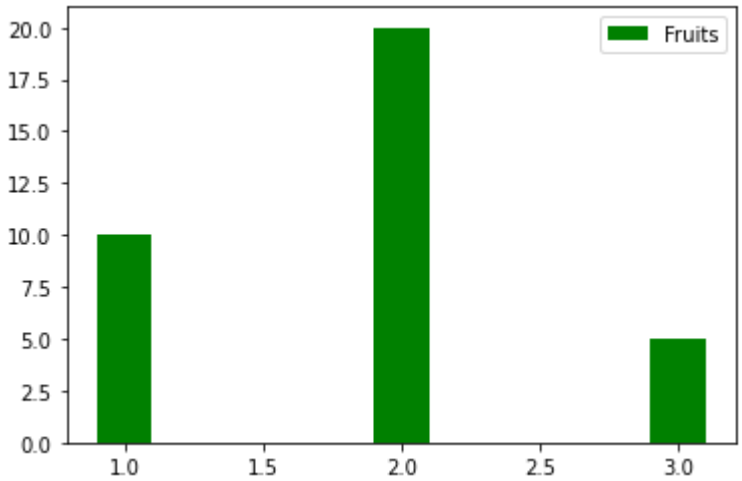


In [ ]:

Types of Graph

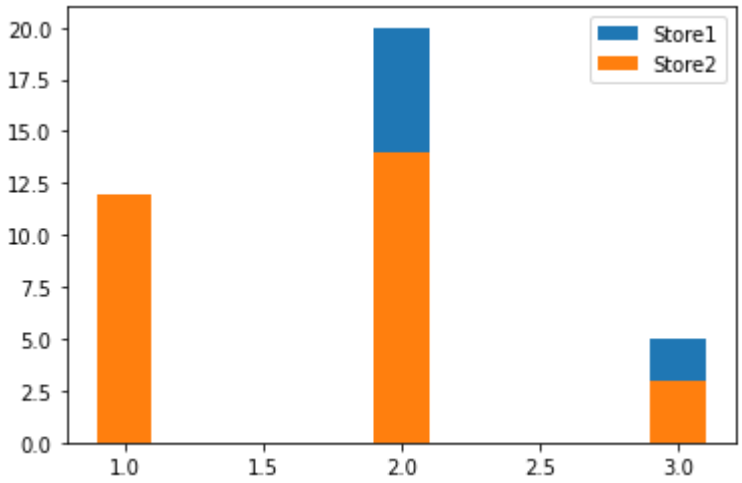
In [ ]:

In [91]:
```
plt.bar(x= [1,2,3], height=[10, 20, 5], width=0.2, label = 'Fruits', color='green')
plt.legend()

plt.show()
```

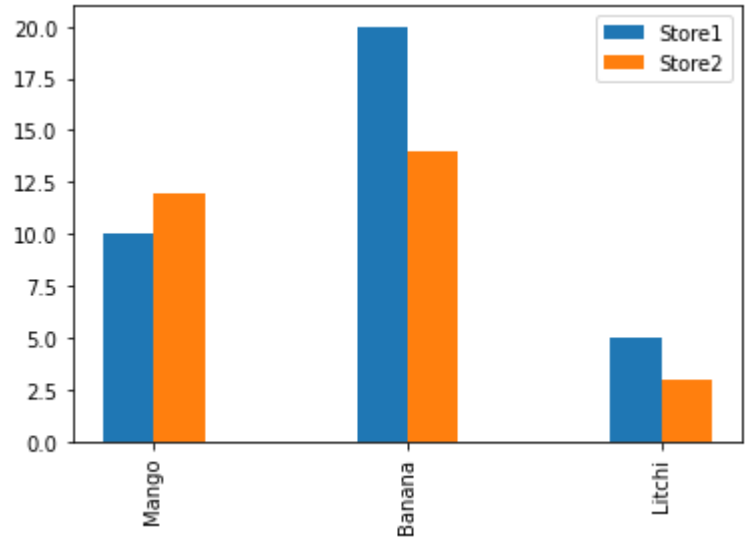


In [100]:
```
x= np.arange(1, 4)
store1 = [10, 20, 5]
store2 = [12, 14, 3]
```

In [101]:
```
plt.bar(x, store1, label="Store1", width=0.2)
plt.bar(x, store2, label="Store2", width=0.2)
plt.legend()
plt.show()
```



In [106]:
```
plt.bar(x-0.1, store1, label="Store1", width=0.2)
plt.bar(x+0.1, store2, label="Store2", width=0.2, )
plt.xticks(x, labels= ['Mango', 'Banana', 'Litchi'], rotation = 90)
plt.legend()
plt.show()
```

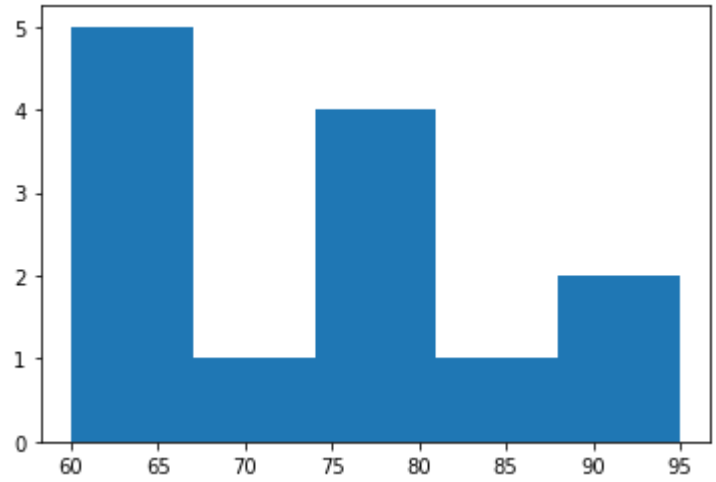


In [ ]:
```
```

Histogram

In [111]:
```
marks = np.array([70,61,60,64, 75, 80, 92, 65, 76, 74, 82, 95, 63])
```

In [113]:
```
plt.hist(marks, bins=5)
plt.show()
```

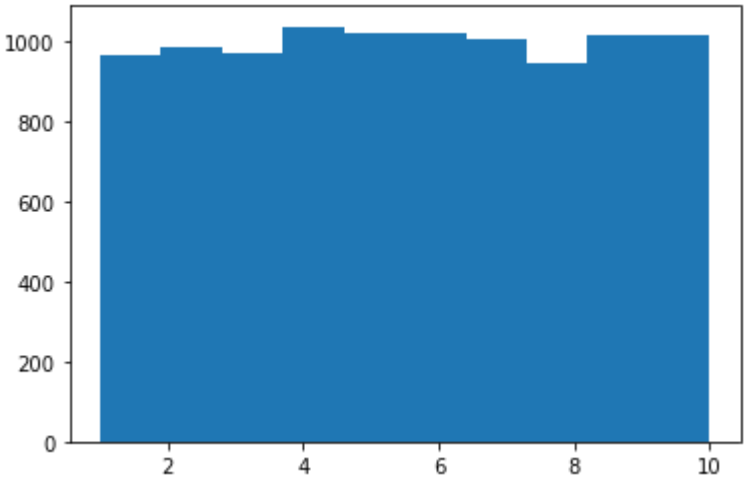


In [115]:
```
np.histogram(marks, bins = 5)
```

Out[115]:
```
(array([5, 1, 4, 1, 2]), array([60., 67., 74., 81., 88., 95.]))
```

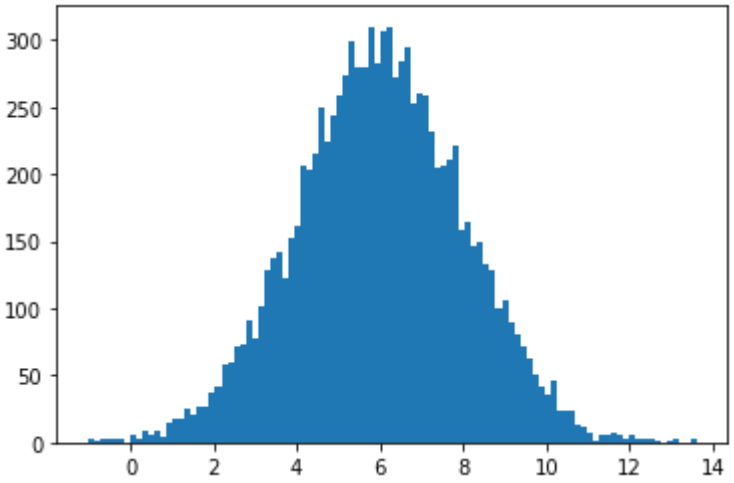
In [ ]:

In [138]:
```
vals = np.random.uniform(1, 10, size=(10000, ))
plt.hist(vals)
plt.show()
```



In [ ]:
```
```

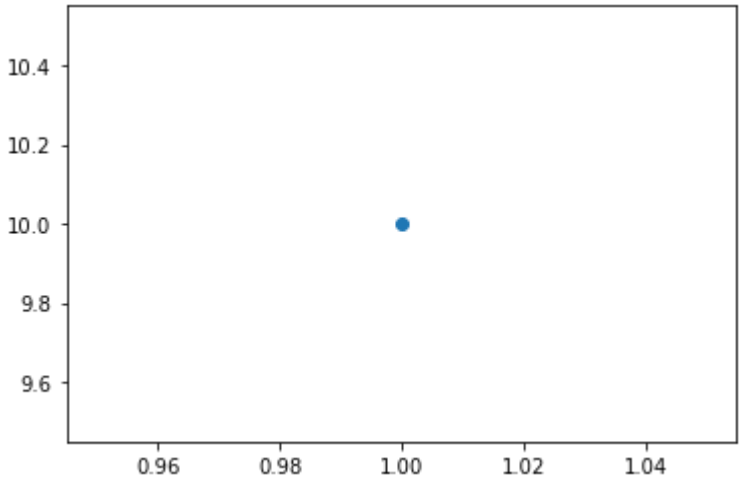
In [142]:
```
vals = np.random.normal(6, 2, size=(10000, ))
plt.hist(vals, bins=100)
plt.show()
```



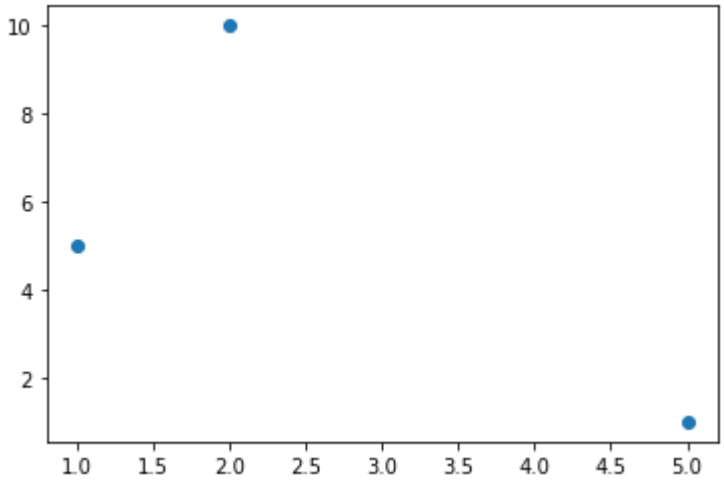
In [ ]:

Scatter plot

In [145]:
```
plt.scatter(1, 10)
plt.show()
```

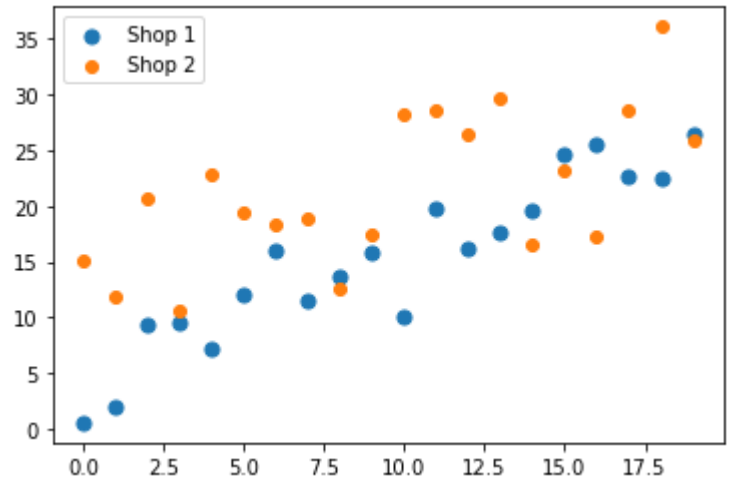


In [149]:
```
plt.scatter([2, 1, 5], [10, 5, 1])
# plt.scatter([2, 1, 5], [12, 4, 3])
plt.show()
```



In [180]:
```
x = np.arange(20)
y1 = x + np.random.rand(20)*10
y2 = x + np.random.rand(20)*20 +0.2
```

In [190]:
```
plt.scatter(x, y1, label="Shop 1",  s= 50)
plt.scatter(x, y2, label="Shop 2")
plt.legend()
plt.show()
```



In [182]:
```
y2.mean()
```

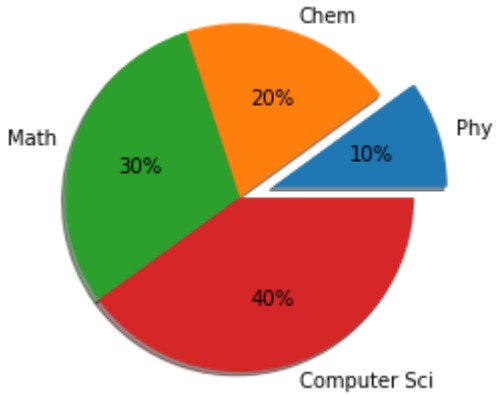
Out[182]: 21.401193585501055

In [183]:
```
y1.mean()
```

Out[183]: 15.127576700576933

In [ ]:

In [232]:
```
plt.pie([1,2,3,4],
        labels=['Phy', 'Chem', "Math", 'Computer Sci'],
        shadow=True,
        explode= (0.2,0,0,0),
        autopct="%0.0f%%"
       )
plt.show()
```

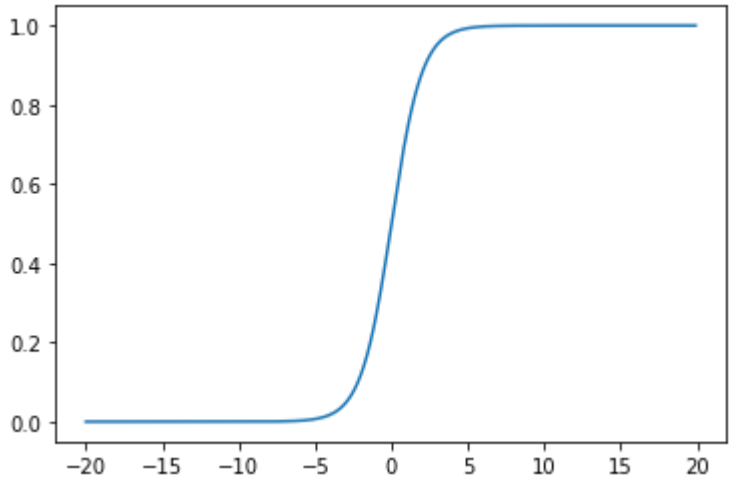


In [ ]:
```
```

In [221]:
```
x = np.arange(-20, 20, 0.1)
y = 1/(1+np.exp(-x))
```

In [222]:
```
plt.plot(x, y)
```

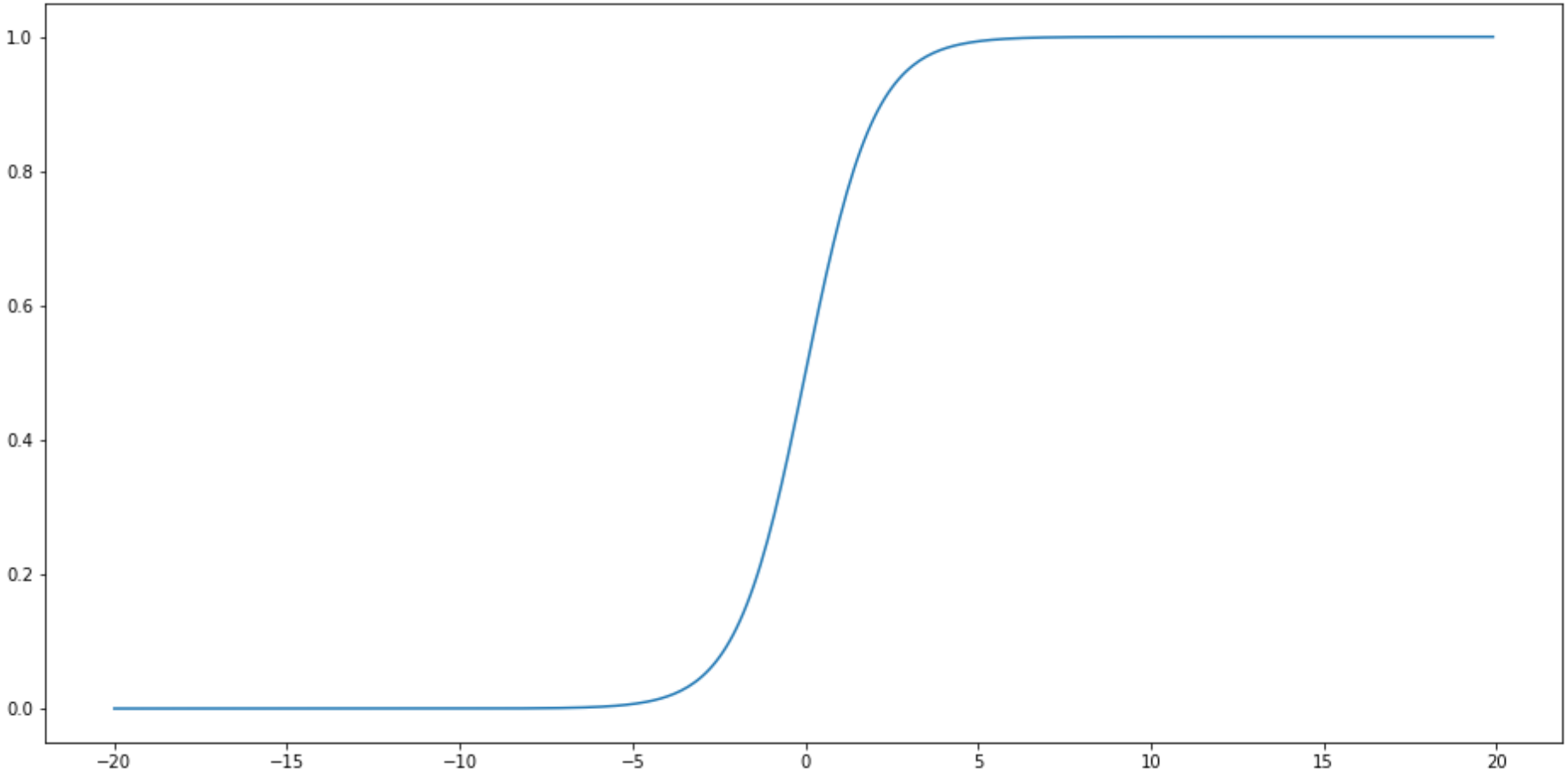
Out[222]:
```
[<matplotlib.lines.Line2D at 0x7fc1f04820d0>]
```



In [ ]:
```

```

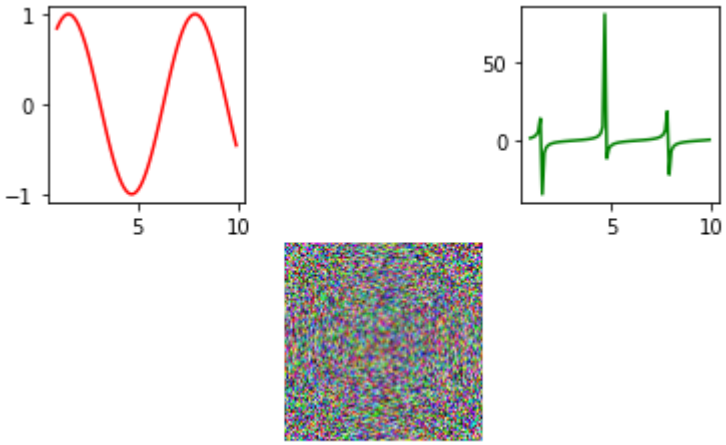
Sub-plots

In [238]:
```
plt.figure(figsize=(16,8))
plt.plot(x, y)
plt.show()
```



In [239]:
```
x = np.arange(1, 10, 0.1)
y1 = np.sin(x)
y2 = np.tan(x)
y3 = np.log(x)
```

In [251]:
```
# plt.figure()

plt.subplot(2, 3, 1)
plt.plot(x, y1, color='red')


plt.subplot(2, 3, 3)
plt.plot(x, y2, color='green')

plt.subplot(2, 3, 5)
# plt.plot(x, y3, color='yellow')
plt.imshow(np.random.rand(100,100, 3))
plt.axis("off")

plt.show()
```



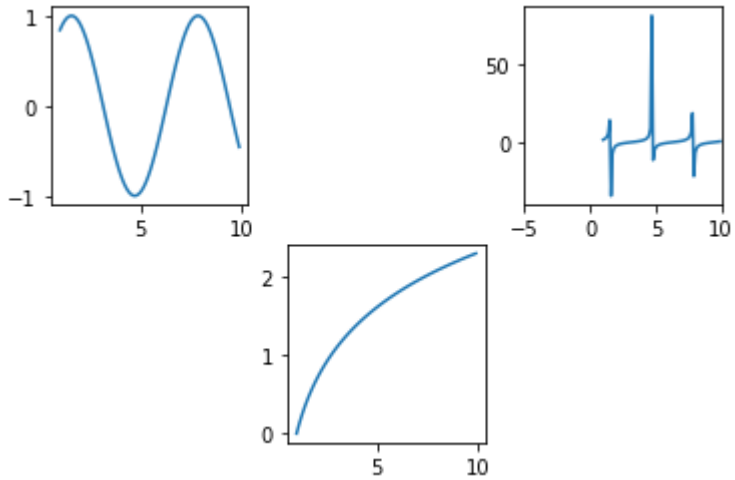
In [ ]:

In [ ]:

We did functional matplotlib programming

Object oriented Programming

In [279]:
```
fig = plt.figure()

ax231 = fig.add_subplot(2,3,1)
ax231.plot(x, y1)

ax233 = fig.add_subplot(2,3,3)
ax233.plot(x, y2)
ax233.set_xlim(-5, 10)

ax235 = fig.add_subplot(2,3,5)
ax235.plot(x, y3)

plt.show()
```



In [ ]:
```
f, (ax1, ax2, ax3) = plt.subplots(1, 3, sharey=True, figsize=(15,3))
```

3D graphs

In [ ]:
```

```