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

Before we begin!

In [2]:
```
# One of the most important basics to remember about matplotlib is that
# Every situation is DIFFERENT!
# Each plot has its own parameters
# And the type of data you are using will make a difference too
# So don't muck about trying to make a barplot based on your knowledge of line plots...!
# Read and understand the documentation first :)
# The gallery provides the best point of entry: https://matplotlib.org/gallery.html
```

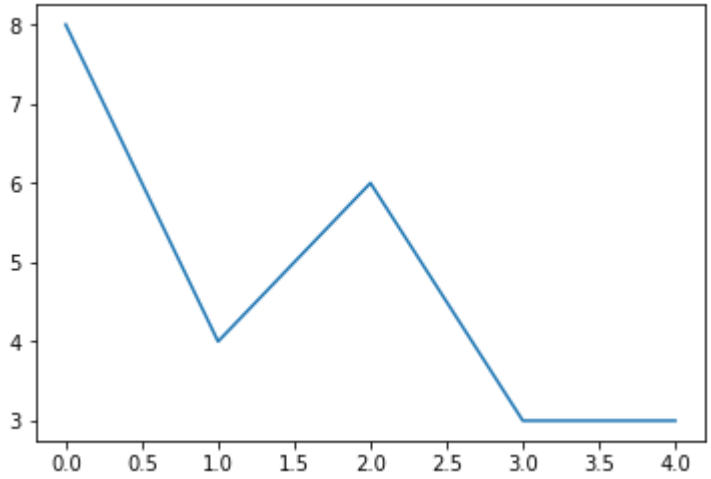
Simple plotting - line and bar charts

In [3]:
```
# Let's generate some basic values to use in the simplest charts:
values1 = np.random.randint(0, 10, 5)
values2 = np.random.randint(0, 10, 5)
values1, values2
```

Out[3]:
```
(array([8, 4, 6, 3, 3]), array([4, 8, 6, 1, 4]))
```

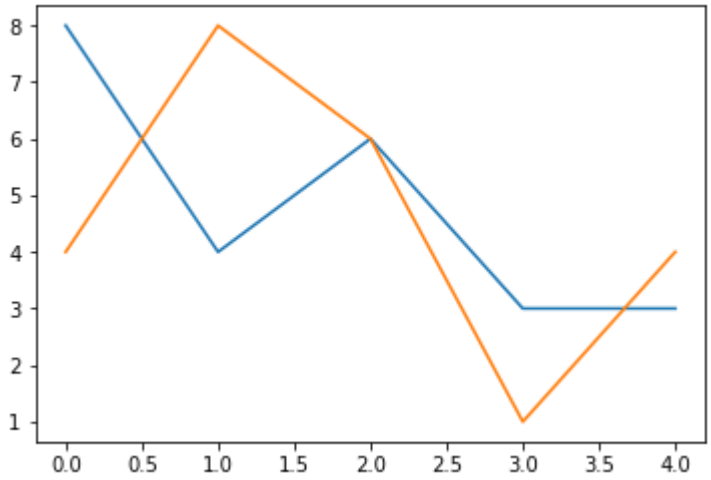
In [4]:
```
plt.plot(values1) # A very simple line chart with 1 set of values
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x1141cc6d8>]
```



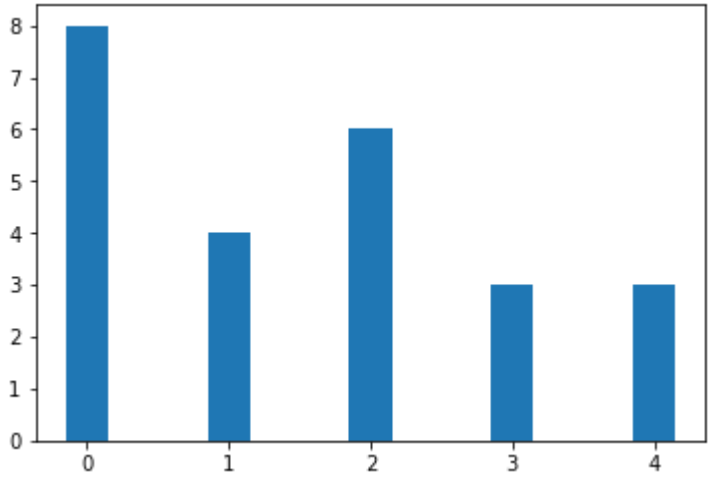
In [5]:
```
plt.plot(values1) # If we have 2 sets of values we can plot the second set like this
plt.plot(values2)
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x1142d30f0>]
```



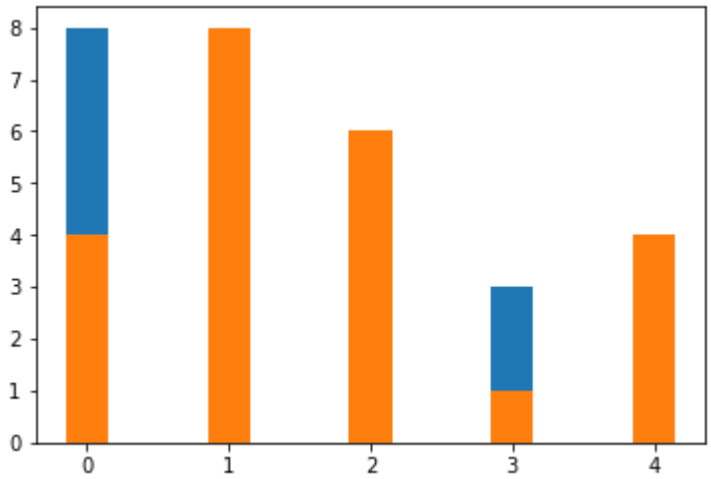
In [6]:
```
plt.bar(np.arange(5), height = values1, width = 0.3,) # A very simple bar chart with 1 set of values
```

Out[6]: <Container object of 5 artists>



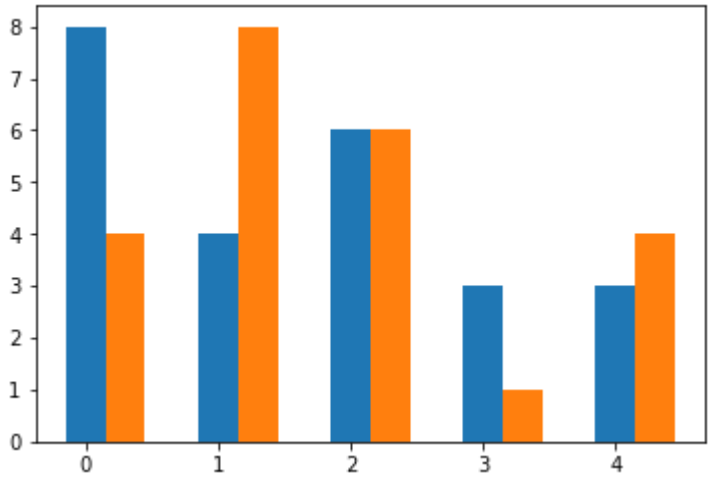
In [7]:
```
plt.bar(np.arange(5), height = values1, width = 0.3,) # If we plot again for the second set of values, howeve
r,
plt.bar(np.arange(5), height = values2, width = 0.3,) # it won't work! The 2 graphs are overlaid - urrk!
```

Out[7]: <Container object of 5 artists>



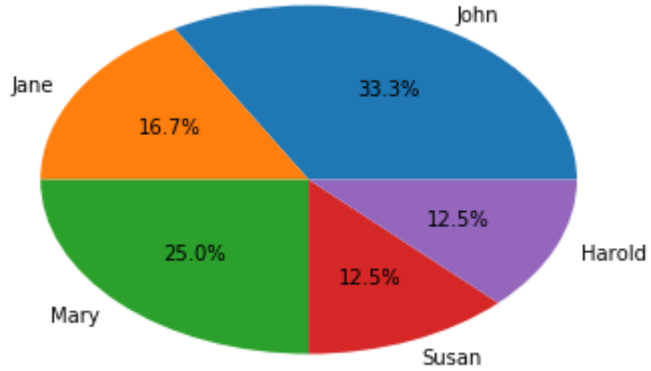
In [8]:
```
plt.bar(np.arange(len(values1)), height = values1, width = 0.3)
plt.bar(np.arange(len(values2)) + 0.3, height = values2, width = 0.3,)

# Because our first parameter is the x co-ordinates of the bars, we can adjust this so that the next bar
# starts at the end of the previous bar, and voila!
```

Out[8]: <Container object of 5 artists>



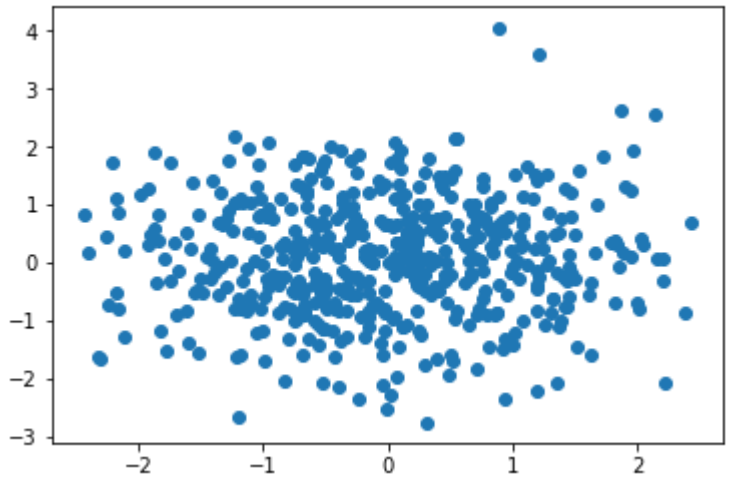
Other commonly used plots

In [15]:
```
# The pie
plt.pie(values1, labels = ["John", "Jane", "Mary", "Susan", "Harold"], autopct='%1.1f%%')
plt.show()
```

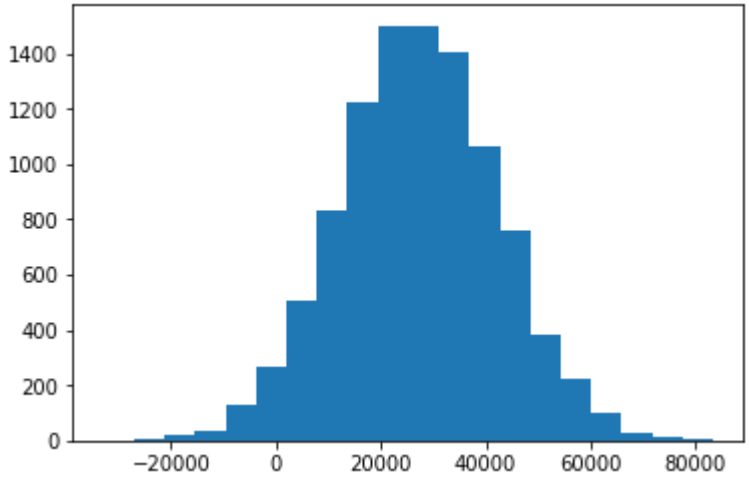


In [17]:
```
# The scatter
values3 = np.random.randn(500)
values4 = np.random.randn(500)
plt.scatter(values3, values4)
```

Out[17]: <matplotlib.collections.PathCollection at 0x11b3c87f0>



In [20]:
```
# The histogram
values5 = np.random.normal(27000, 15000, 10000)
plt.hist(values5, 20)
plt.show()
```



Making it pretty - background image

In [22]:
```
import matplotlib.image as mpimg
```

In [23]:
```
img = mpimg.imread("polar_bear.png") #.PNG as a format definitely works...
```

In [24]:
```
values3 = np.random.randint(0,10, 13)
plt.imshow(img, extent=[-0.5, 13, 0, 10]) # Play around with extent to get the image covering the right area
plt.plot(values3, color='chartreuse', linestyle='solid', marker='o',
         markerfacecolor='fuchsia', markersize=12)
plt.show()
```