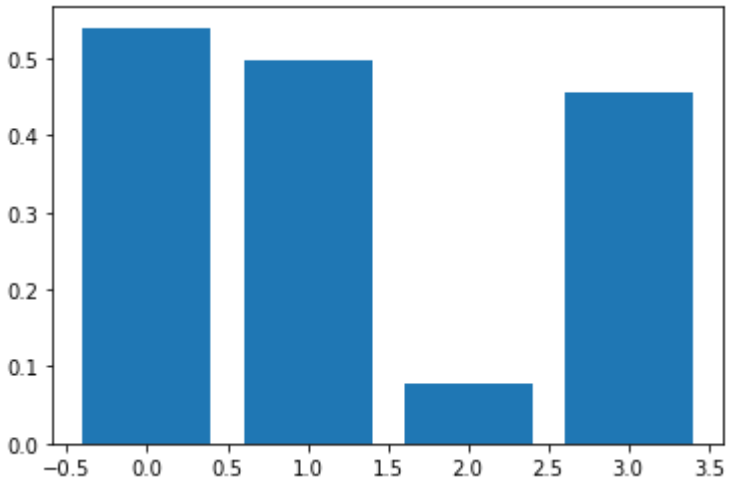
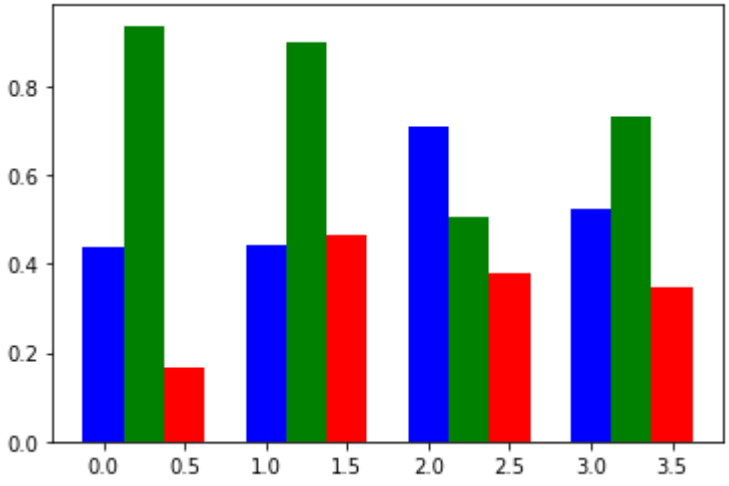
In [1]:

```
# A bar graph is a representation of discrete and categorical data items with bars.
# You can represent the data with vertical or horizontal bars. The height or length
# of bars is always in proportion to the magnitude of the data. You can use bar charts
# or bar graphs when you have discrete categorical data. The following is a simple examp
# of a bar graph:

import numpy as np
import matplotlib.pyplot as plt
x = np.arange(4)
y = np.random.rand(4)
plt.bar(x, y)
plt.show()
```



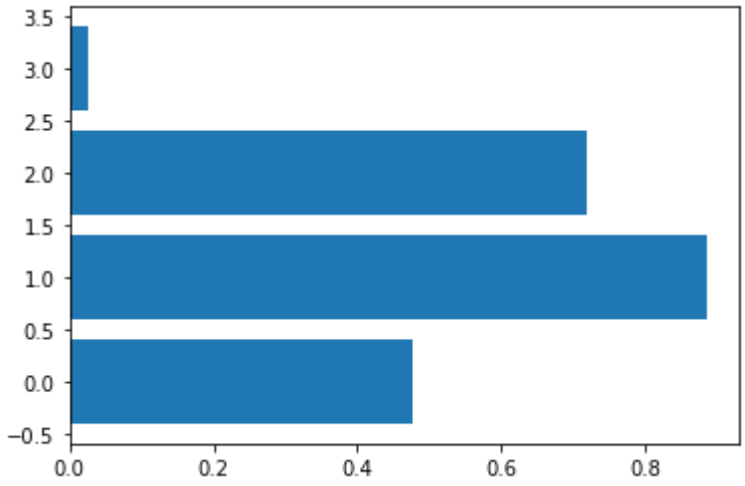
In [2]:

```
# You can have a combined bar graph as follows:

import numpy as np
import matplotlib.pyplot as plt
y = np.random.rand(3, 4)
plt.bar(x + 0.00, y[0], color = 'b', width = 0.25)
plt.bar(x + 0.25, y[1], color = 'g', width = 0.25)
plt.bar(x + 0.50, y[2], color = 'r', width = 0.25)
plt.show()
```



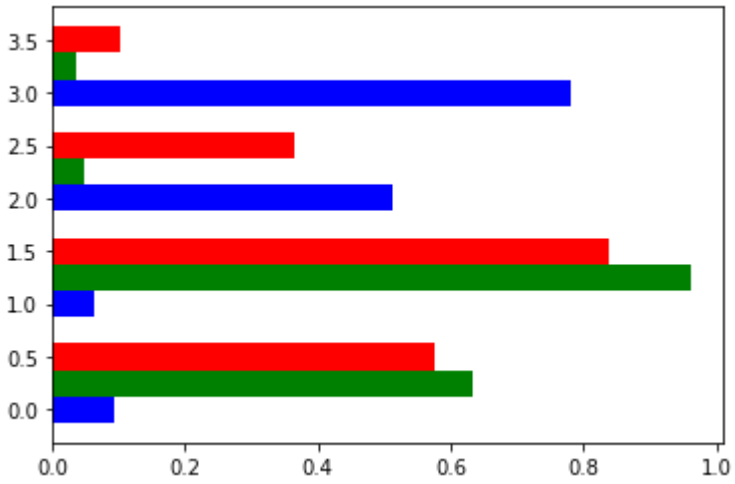
In [3]:

```
# The previous graphs were examples of vertical bar graphs. Similarly, you can have
# horizontal bar graphs as follows:

import numpy as np
import matplotlib.pyplot as plt
x = np.arange(4)
y = np.random.rand(4)
plt.barh(x, y)
plt.show()
```



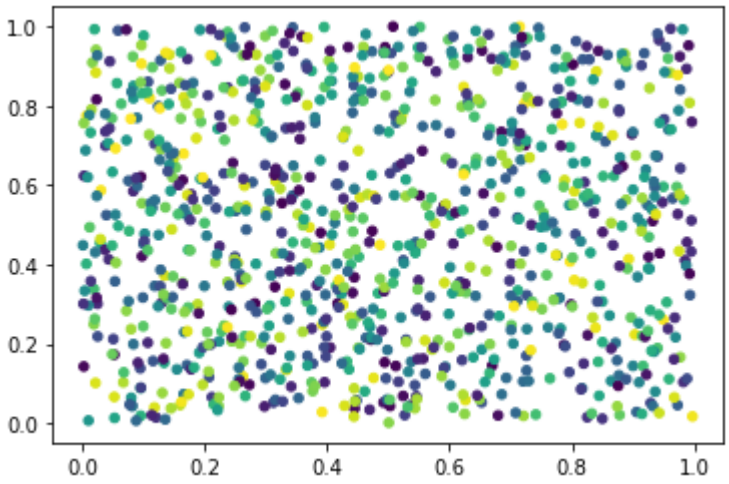
In [4]:

```
# You can also have combined horizontal bar graphs as follows:

import numpy as np
import matplotlib.pyplot as plt
y = np.random.rand(3, 4)
plt.barh(x + 0.00, y[0], color = 'b', height=0.25)
plt.barh(x + 0.25, y[1], color = 'g', height=0.25)
plt.barh(x + 0.50, y[2], color = 'r', height=0.25)
plt.show()
```



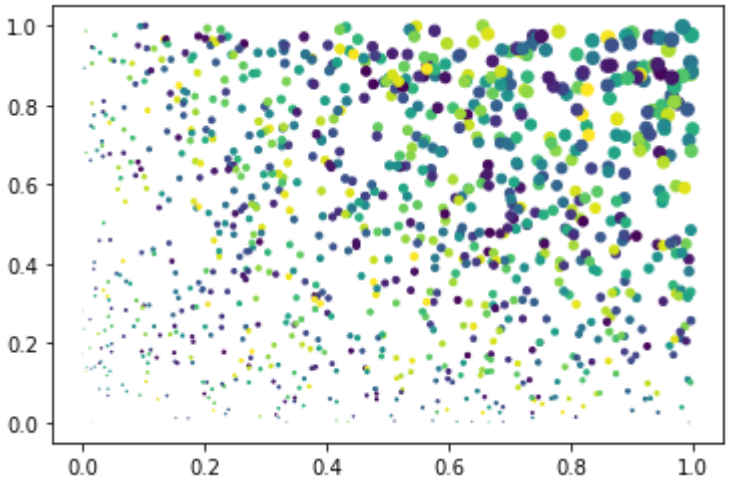
In [5]:

```
# You can also visualize your data with scatter plots.
# You will usually visualize a set of two variables with a scatter plot.
# One variable is assigned to the x-axis, and another is assigned to the y-axis.
# Then you draw a point for the values of x-y pairs. The size of x and y must be same
# (they are always one-dimensional arrays). You can show additional
# variables by manipulating the colors and sizes of the points.
# In that case, the sizes of the one-dimensional arrays representing x, y, the color,
# and the size must be the same.
# In the following example, we are assigning random x- and y-axes values and colors to
# 1,000 points. All points are of size 20.

import numpy as np
import matplotlib.pyplot as plt
N = 1000
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
size = (20)
plt.scatter(x, y, s=size, c=colors, alpha=1)
plt.show()
```



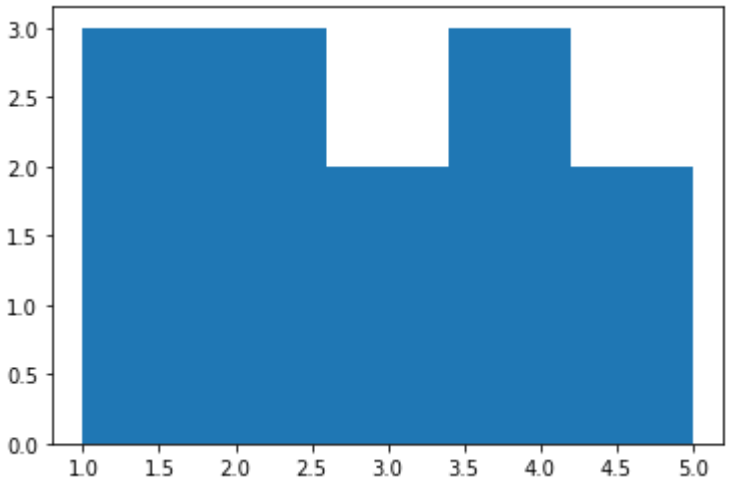
In [6]:

```
# The size of the points is fixed in this example. You can also set the size per the pla
# on the graph (which depends on the values of the x and y coordinates). Here is an
# example:

import numpy as np
import matplotlib.pyplot as plt
N = 1000
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
size = (50 * x * y)
plt.scatter(x, y, s=size, c=colors, alpha=1)
plt.show()
```



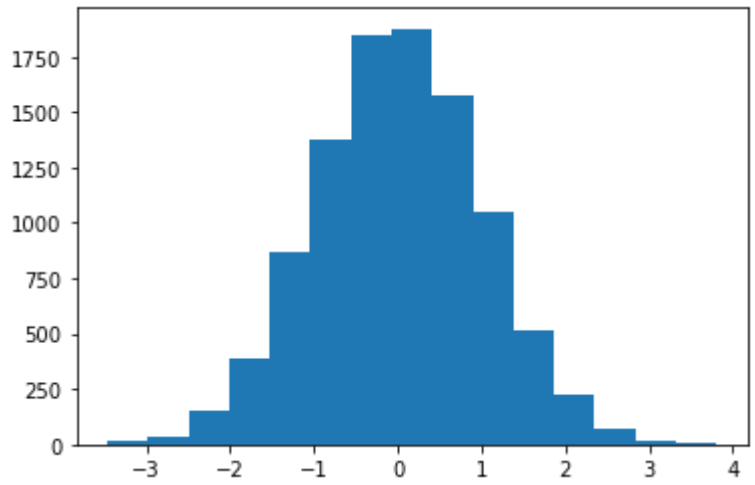
In [7]:

```
# Suppose you have a set of members with various values.
# You can create a table that has various buckets of ranges
# of values in a column. Each bucket must have at least one value.
# Then you can count the number of members that fall into that bucket
# and note those counts against the buckets.
# create a dataset and define the number of buckets equal to the
# cardinality (number of distinct elements) of the set.

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
x = [1, 3, 5, 1, 2, 4, 4, 2, 5, 4, 3, 1, 2]
n_bins = 5
plt.hist(x, bins=n_bins)
plt.show()
```



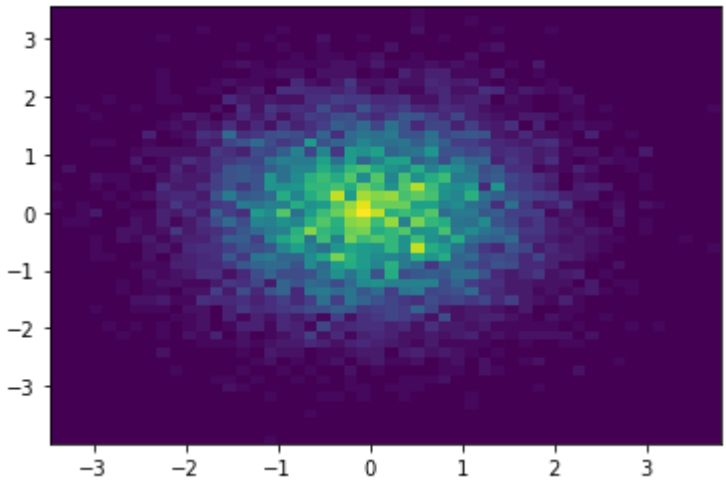
In [8]:

```
# Normal (or Gaussian) distribution is a type of continuous probability distribution. It
# is usually a bell-shaped curve. Let's create a histogram with a normal distribution cu
# To create the data, we will use a NumPy routine. Let's draw a histogram of random data
# with normal distribution as follows:

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(31415)
n_points = 10000
n_bins = 15
x = np.random.randn(n_points)
plt.hist(x, bins=n_bins)
plt.show()
```



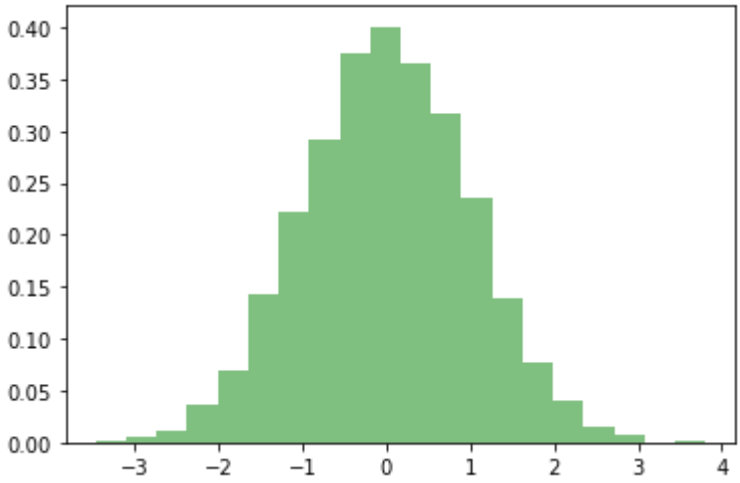
In [11]:

```
# The histogram of one-dimensional data is a 2D figure.
# When you want to create a histogram of 2D data, you have to create a 3D figure
# with the data variables on the x- and y-axes and the histogram on the z-axis.
# In other words, you can use 2D coordinates to show this 3D visualization and
# view the histogram from the top (top view). The bars can be color coded to signify
# their magnitude.

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(31415)
n_points = 10000
x = np.random.randn(n_points)
y = np.random.randn(n_points)
plt.hist2d(x, y, bins=50)
plt.show()
```



In [12]:

```
# You can customize the histogram by setting the transparency and the color as
# follows:

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(31415)
n_points = 10000
x = np.random.randn(n_points)
plt.hist(x, 20, density=True,
         histtype='stepfilled',
         facecolor='g', alpha=0.5)
plt.show()
```



In [13]:

```
# You can also show just the outline of the histogram as follows:

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(31415)
n_points = 10000
x = np.random.randn(n_points)
plt.hist(x, 20, density=True,
         histtype='step')
plt.show()
```