a.) Use number to create a random vector of 12 elements (values) from a uniform distribution.

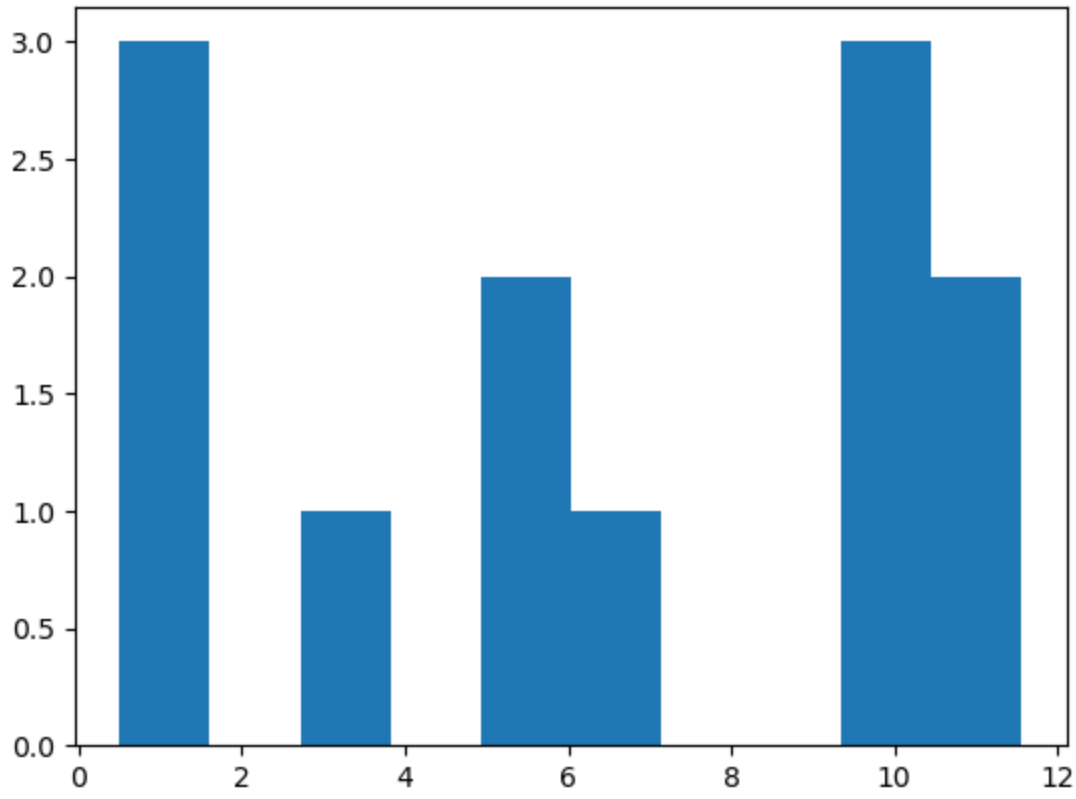
In [9]:
```
import numpy as np
dir(np.random) # used to find uniform member function
?np.random.uniform # used to read Docstring
# Docstring:
# uniform(low=0.0, high=1.0, size=None)

a = np.random.uniform(0,12,12)

import matplotlib.pyplot as plt
plt.hist(a)
```

```
Object `np.random.uniform # used to read Docstring` not found.
```

Out[9]:
```
(array([3., 0., 1., 0., 2., 1., 0., 0., 3., 2.]),
 array([ 0.5046394 ,  1.61012877,  2.71561815,  3.82110752,  4.9265969 ,
         6.03208627,  7.13757565,  8.24306503,  9.3485544 , 10.45404378,
        11.55953315]),
 <BarContainer object of 10 artists>)
```



b.) Sort the array above in forward and reverse order, using NumPy functions

In [15]:
```
Forward_order = np.sort(a)
Forward_order
```

Out[15]:
```
array([ 0.5046394 ,  0.89905768,  1.0230039 ,  2.9018233 ,  5.38190973,
        5.43935508,  6.66618655,  9.53861375,  9.6962365 , 10.0099538 ,
       10.80018594, 11.55953315])
```

In [16]:
```
Reverse_order = np.flip(Forward_order)
Reverse_order
```

Out[16]:
```
array([11.55953315, 10.80018594, 10.0099538 ,  9.6962365 ,  9.53861375,
        6.66618655,  5.43935508,  5.38190973,  2.9018233 ,  1.0230039 ,
        0.89905768,  0.5046394 ])
```

```
c.) Resize your matrix from a as 4x3 and as a 2 x 3 x2 matrices
```

In [17]:
```
a.reshape([4,3])
```

Out[17]:
```
array([[ 9.53861375,  0.89905768,  0.5046394 ],
       [ 6.66618655, 11.55953315,  5.43935508],
       [ 1.0230039 , 10.0099538 ,  5.38190973],
       [10.80018594,  2.9018233 ,  9.6962365 ]])
```

In [18]:
```
a.reshape([2,3,2])
```

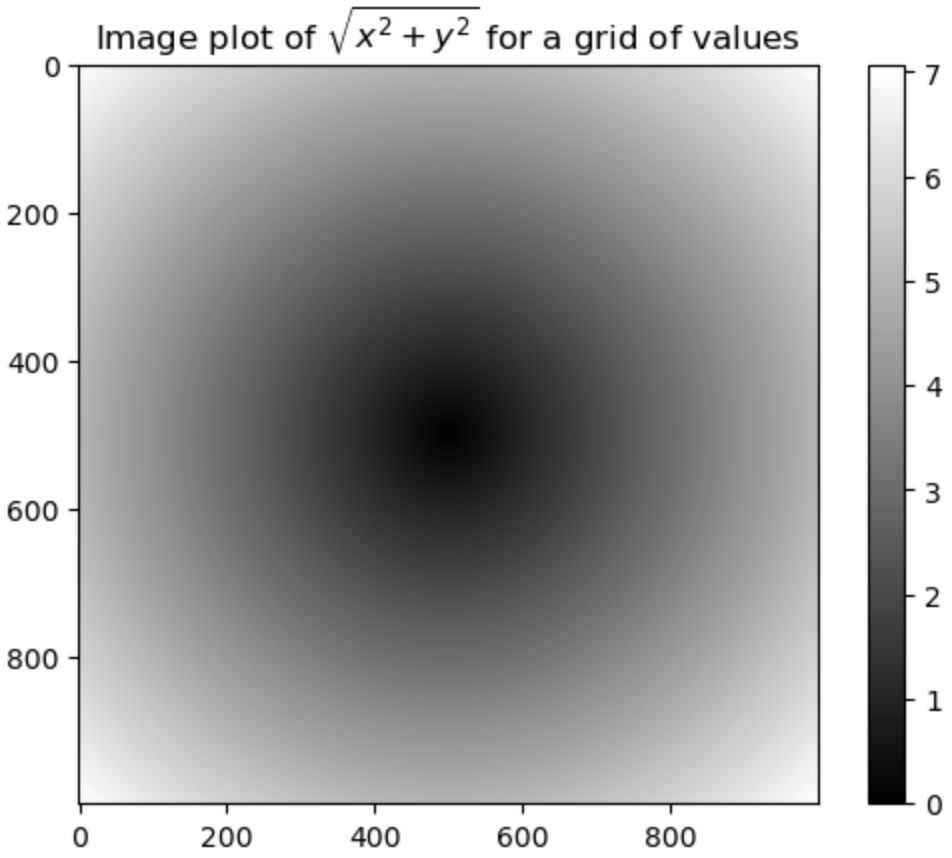
Out[18]:
```
array([[[ 9.53861375,  0.89905768],
        [ 0.5046394 ,  6.66618655],
        [11.55953315,  5.43935508]],

       [[ 1.0230039 , 10.0099538 ],
        [ 5.38190973, 10.80018594],
        [ 2.9018233 ,  9.6962365 ]]])
```

```
d.) Run the example shown in section 4.3 of McKinney, of the image plot of the
NumPy array of square root of x^2+y^2.
```

In [22]:
```
points = np.arange(-5, 5, 0.01)
xs, ys = np.meshgrid(points, points)
z = np.sqrt(xs ** 2 + ys ** 2)
plt.imshow(z, cmap=plt.cm.gray); plt.colorbar()
plt.title("Image plot of $\sqrt{x^2 + y^2}$ for a grid of values")
```

Out[22]:
```
Text(0.5, 1.0, 'Image plot of $\\sqrt{x^2 + y^2}$ for a grid of values')
```



e.) This example is from the NumPy site at
https://numpy.org/doc/stable/user/absolute_beginners.html
use the plot_surface() function to show your results in d. Look up contour plots on matplotlib and show the data from d as a contour plot.

In [25]:
```
fig = plt.figure()

ax = fig.add_subplot(projection='3d')

points = np.arange(-5, 5, 0.01)
xs, ys = np.meshgrid(points, points)

R = np.sqrt(xs**2 + ys**2)

Z = np.sin(R)

ax.plot_surface(xs, ys, Z, rstride=1, cstride=1, cmap='viridis')
```

Out[25]:
```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x2061c3be050>
```