In [1]:
```
import numpy as np
a = np.arange(6)
a2 = a[np.newaxis, :]
a2.shape
```

Out[1]:
```
(1, 6)
```

In [3]:
```
a = np.array([1, 2, 3, 4, 5, 6])
a
```

Out[3]:
```
array([1, 2, 3, 4, 5, 6])
```

In [5]:
```
a = np.array([1, 2, 3])
a
```

Out[5]:
```
array([1, 2, 3])
```

Differnt Methods

In [32]:
```
#Zeros
np.zeros(6)
```

Out[32]:
```
array([0., 0., 0., 0., 0., 0.])
```

In [34]:
```
#Ones
np.ones(5)
```

Out[34]:
```
array([1., 1., 1., 1., 1.])
```

In [36]:
```
#Empty
np.empty(4)
```

Out[36]:
```
array([0.        , 0.        , 3.46536611, 3.31525748])
```

In [37]:
```
#Specific
np.arange(2,10)
```

Out[37]:
```
array([2, 3, 4, 5, 6, 7, 8, 9])
```

In [38]:
```
#Specific Interval
np.arange(2,30,3)
```

Out[38]:
```
array([ 2,  5,  8, 11, 14, 17, 20, 23, 26, 29])
```

In [42]:
```
#table
np.arange(3,33,3)
```

Out[42]:
```
array([ 3,  6,  9, 12, 15, 18, 21, 24, 27, 30])
```

In [54]:
```
#Line Space
np.linspace(1, 100, num=5)
```

Out[54]:
```
array([  1.  ,  25.75,  50.5 ,  75.25, 100.  ])
```

In [60]:
```
# Specific Your Data Type
np.ones(50,dtandype=np.int64)
```

Out[60]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1], dtype=int64)
```

In [63]:
```
np.ones(50, dtype=np.float64)
```

Out[63]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [ ]:
```

```

Adding, removing, and sorting elements

In [7]:
```
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
np.sort(arr)
```

Out[7]:
```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

-argsort, which is an indirect sort along a specified axis,

-lexsort, which is an indirect stable sort on multiple keys,

-searchsorted, which will find elements in a sorted array, and

-partition, which is a partial sort.

In [8]:
```
a = np.array([1, 2, 3, 4])
b = np.array([5, 6, 7, 8])
np.concatenate((a, b))
```

Out[8]:
```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

In [9]:
```
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6]])
np.concatenate((x, y), axis=0)
```

Out[9]:
```
array([[1, 2],
       [3, 4],
```

```
       [5, 6]])
```

How do you know the shape and size of an array?

In [12]:
```
array_example = np.array([[[0, 1, 2, 3],
                           [4, 5, 6, 7]],

                          [[0, 1, 2, 3],
                           [4, 5, 6, 7]],

                          [[0 ,1 ,2, 3],
                           [4, 5, 6, 7]]])
array_example.ndim
```

Out[12]:
```
(3, 2, 4)
```

In [13]:
```
array_example.size
```

Out[13]:
```
24
```

In [14]:
```
array_example.shape
```

Out[14]:
```
(3, 2, 4)
```

Can you reshape an array?¶

In [15]:
```
a = np.arange(6)
print(a)
```

```
[0 1 2 3 4 5]
```

In [16]:
```
b = a.reshape(3, 2)
```

How to convert a 1D array into a 2D array (how to add a new axis to an array)

In [17]:
```
a = np.array([1, 2, 3, 4, 5, 6])
a.shape
```

Out[17]:
```
(6,)
```

In [18]:
```
a2 = a[np.newaxis, :]
a2.shape
```

Out[18]:
```
(1, 6)
```

In [19]:
```
row_vector = a[np.newaxis, :]
row_vector.shape
```

Out[19]:
```
(1, 6)
```

In [20]:
```
col_vector = a[:, np.newaxis]
col_vector.shape
```

Out[20]:
```
(6, 1)
```

Indexing and slicing

In [23]:
```
data = np.array([1, 2, 3])

data[1]
```

Out[23]:
```
2
```

In [24]:
```
data[0:2]
```

Out[24]:
```
array([1, 2])
```

Creating matrices

In [25]:
```
data = np.array([[1, 2], [3, 4], [5, 6]])
data
```

Out[25]:
```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [26]:
```
data[0,1]
```

Out[26]:
```
2
```

In [27]:
```
data[1:3]
```

Out[27]:
```
array([[3, 4],
       [5, 6]])
```

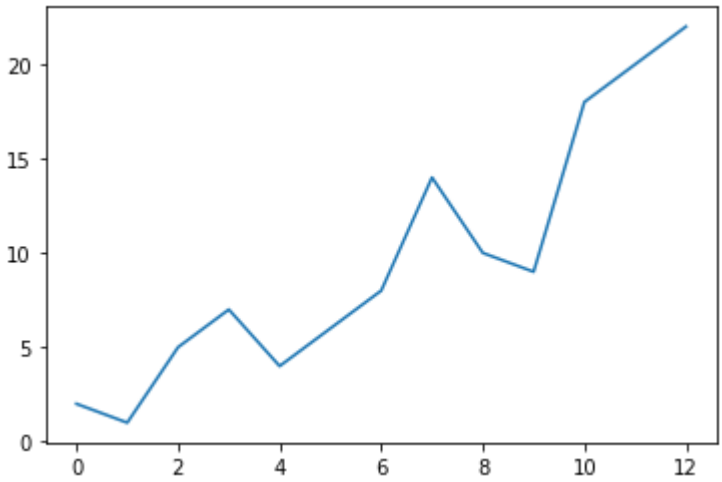
In [28]:
```
data = np.array([[1, 2], [3, 4], [5, 6]])
ones_row = np.array([[1, 1]])
data + ones_row
```

Out[28]:
```
array([[2, 3],
       [4, 5],
       [6, 7]])
```

Plotting arrays with Matplotlib

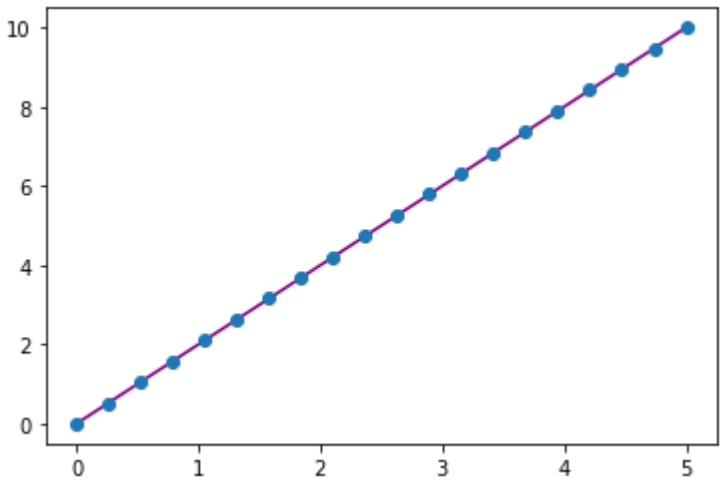
In [29]:
```
import matplotlib.pyplot as plt
a = np.array([2, 1, 5, 7, 4, 6, 8, 14, 10, 9, 18, 20, 22])

# If you're using Jupyter Notebook, you may also want to run the following
# line of code to display your code in the notebook:

%matplotlib inline
plt.plot(a)

# If you are running from a command line, you may need to do this:
# >>> plt.show()
```

Out[29]:
```
[<matplotlib.lines.Line2D at 0x221ed6aec10>]
```



In [30]:
```
x = np.linspace(0, 5, 20)
y = np.linspace(0, 10, 20)
plt.plot(x, y, 'purple') # line
plt.plot(x, y, 'o')
```

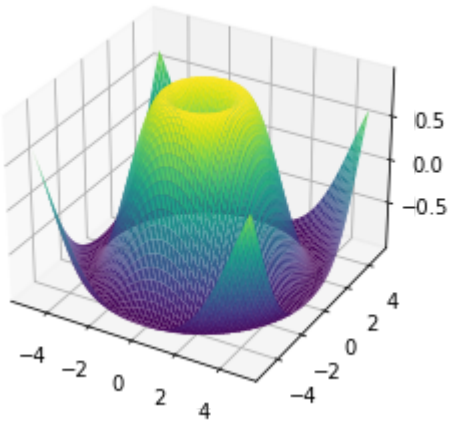
Out[30]:
```
[<matplotlib.lines.Line2D at 0x221ede5fbb0>]
```



In [31]:
```
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
X = np.arange(-5, 5, 0.15)
```

```
Y = np.arange(-5, 5, 0.15)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap='viridis')
```

Out[31]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x221edee62b0>



In [ ]: