In [1]:
```
import numpy as np
data = np.random.randn(2, 3)
print(data)

print(data.shape)

print(data.dtype)
```

```
[[ 1.10508595 -0.70052575  1.60853955]
 [ 0.52545898  0.88914865 -2.11464584]]
(2, 3)
float64
```

In [2]:
```
arr = np.arange(10)
print(arr)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

In [3]:
```
print(data + data)
```

```
[[ 2.21017189 -1.40105149  3.2170791 ]
 [ 1.05091795  1.7782973  -4.22929167]]
```

In [4]:
```
print(data * 10)
```

```
[[ 11.05085947  -7.00525747  16.08539551]
 [  5.25458977   8.89148652 -21.14645835]]
```

In [5]:
```
print(data * data)
```

```
[[1.22121495 0.49073632 2.58739949]
 [0.27610714 0.79058532 4.47172701]]
```

Creating and converting to ndarrays

In [6]:
```
data1 = [6, 7.5, 8, 0, 1]
arr1 = np.array(data1)

print(arr1)
```

```
[6.  7.5 8.  0.  1. ]
```

In [7]:
```
print(arr1.ndim)
```

```
1
```

In [8]:

```
arr2 = np.array([[1,2,3,4],[5,6,7,8]])
print(arr2)
print(arr2.ndim)
```

```
[[1 2 3 4]
 [5 6 7 8]]
2
```

In [9]:

```
zero_matrix = np.zeros((3, 6))
print(zero_matrix)
```

```
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
```

In [10]:

```
ones_matrix = np.ones((4,5))
print(ones_matrix)
```

```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

Indexing and Slicing

In [11]:

```
eight_m = ones_matrix * 8
print(eight_m)
```

```
[[8. 8. 8. 8. 8.]
 [8. 8. 8. 8. 8.]
 [8. 8. 8. 8. 8.]
 [8. 8. 8. 8. 8.]]
```

In [12]:

```
print(arr)
print(arr[5:8])
```

```
[0 1 2 3 4 5 6 7 8 9]
[5 6 7]
```

In [13]:

```
arr[2:5] = 12
print(arr)
```

```
[ 0  1 12 12 12  5  6  7  8  9]
```

In [14]:

```
arr[0:] = 5
print(arr)
```

```
[5 5 5 5 5 5 5 5 5 5]
```

In [15]:

```
arr[:] = 9
print(arr)
```

```
[9 9 9 9 9 9 9 9 9 9]
```

In [16]:

```
# 3-D slicing
arr3d = np.arange(12)
print(arr3d)
arr3d = arr3d.reshape(2,2,3)
print(arr3d)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11]
[[[ 0  1  2]
  [ 3  4  5]]

 [[ 6  7  8]
  [ 9 10 11]]]
```

In [17]:

```
arr3d = arr3d * 2
print(arr3d)
arr3d.ndim
```

```
[[[ 0  2  4]
  [ 6  8 10]]

 [[12 14 16]
  [18 20 22]]]
```

Out[17]:

```
3
```

In [18]:

```
arr3d[0]
```

Out[18]:

```
array([[ 0,  2,  4],
       [ 6,  8, 10]])
```

In [19]:

```
arr3d[:, :, 2]
```

Out[19]:

```
array([[ 4, 10],
       [16, 22]])
```

linear Algebra

In [20]:

```
x = np.random.randint(0, 9, 20)
print(x)
x = x.reshape(4,5)
```

```
[4 4 6 3 2 7 3 2 8 1 1 1 5 6 0 1 6 8 7 6]
```

```
In [21]: x

Out[21]: array([[4, 4, 6, 3, 2],
                [7, 3, 2, 8, 1],
                [1, 1, 5, 6, 0],
                [1, 6, 8, 7, 6]])
```

```
In [22]: # Transpose
         x.T

Out[22]: array([[4, 7, 1, 1],
                [4, 3, 1, 6],
                [6, 2, 5, 8],
                [3, 8, 6, 7],
                [2, 1, 0, 6]])
```

```
In [23]: #dot product
         print(x)
         print(x.T)
         dot = np.dot(x, x.T)
         dot

         [[4 4 6 3 2]
          [7 3 2 8 1]
          [1 1 5 6 0]
          [1 6 8 7 6]]
         [[4 7 1 1]
          [4 3 1 6]
          [6 2 5 8]
          [3 8 6 7]
          [2 1 0 6]]

Out[23]: array([[ 81,  78,  56, 109],
                [ 78, 127,  68, 103],
                [ 56,  68,  63,  89],
                [109, 103,  89, 186]])
```

```
In [24]: m = np.arange(10)
         m

Out[24]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [25]: np.sqrt(m)

Out[25]: array([0.        , 1.        , 1.41421356, 1.73205081, 2.        ,
                2.23606798, 2.44948974, 2.64575131, 2.82842712, 3.        ])
```

```
In [26]: np.exp(m)

Out[26]: array([1.00000000e+00, 2.71828183e+00, 7.38905610e+00, 2.00855369e+01,
                5.45981500e+01, 1.48413159e+02, 4.03428793e+02, 1.09663316e+03,
                2.98095799e+03, 8.10308393e+03])
```

In [27]:

```
print(x)
```

```
[[4 4 6 3 2]
 [7 3 2 8 1]
 [1 1 5 6 0]
 [1 6 8 7 6]]
```

In [28]:

```
print(x.sum())
print(x[1, 0:].sum()) # sum 1st row only
print(x[:, 3].sum()) # sum of 3rd column
print(x[3, :].sum()) # sum of 4th row
print(x[3].sum()) ## sum of 4th row
```

```
81
21
24
28
28
```

In [29]:

```
x.mean()
x[1,:].mean()
```

Out[29]:

```
4.2
```

In [30]:

```
print(x)
print(x.sum(axis = 0))
```

```
[[4 4 6 3 2]
 [7 3 2 8 1]
 [1 1 5 6 0]
 [1 6 8 7 6]]
[13 14 21 24  9]
```

In [31]:

```
print(x.sum(axis = 1))
```

```
[19 21 13 28]
```

matplotlib for ploting and visualization

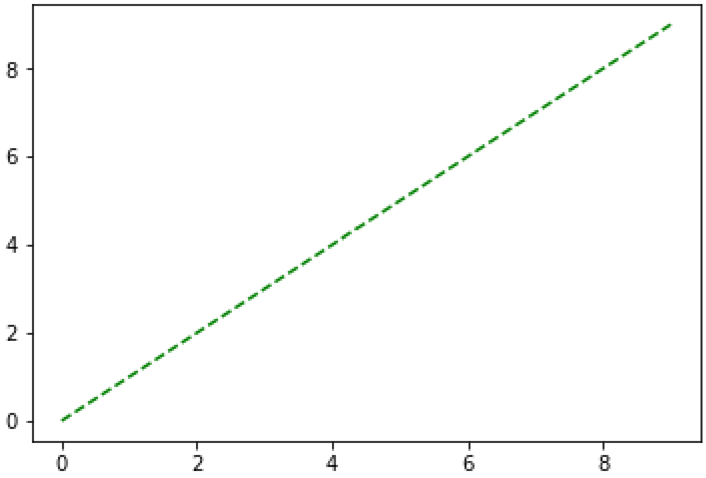
In [32]:

```
import matplotlib.pyplot as plt
arr4 = np.arange(10)
plt.plot(arr4)
plt.show()
```

```
<Figure size 640x480 with 1 Axes>
```

In [33]:

```
plt.plot(arr4, 'g--')
plt.show()
```

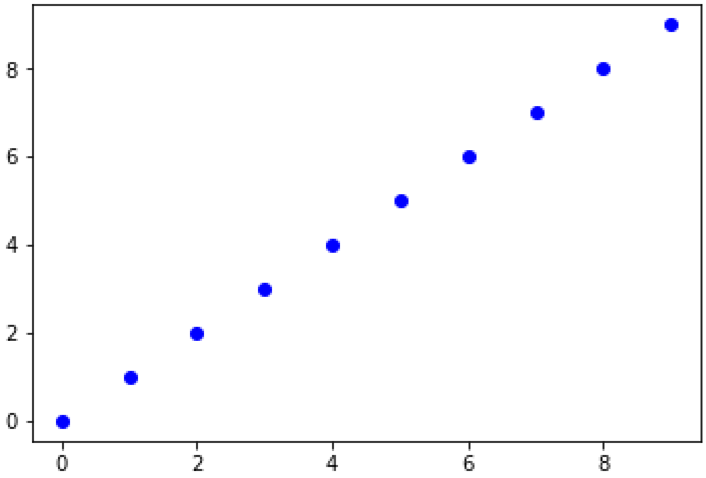


In [34]:

```
plt.plot(arr4, 'bo') #plot using blue circle markers
```

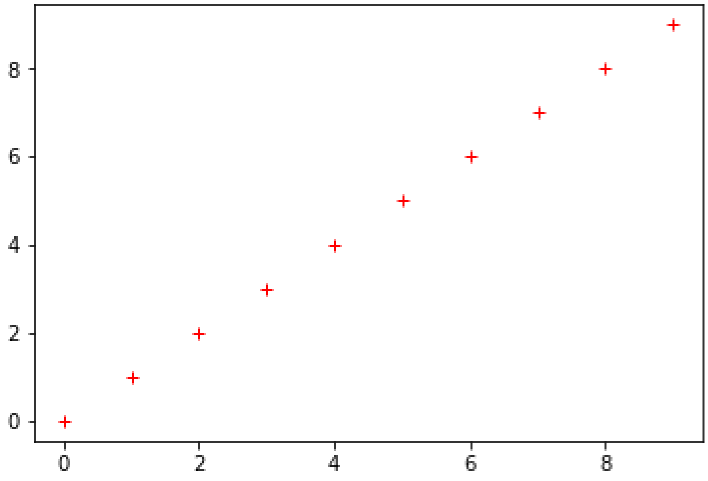
Out[34]:

```
[<matplotlib.lines.Line2D at 0x1d88d8905c0>]
```



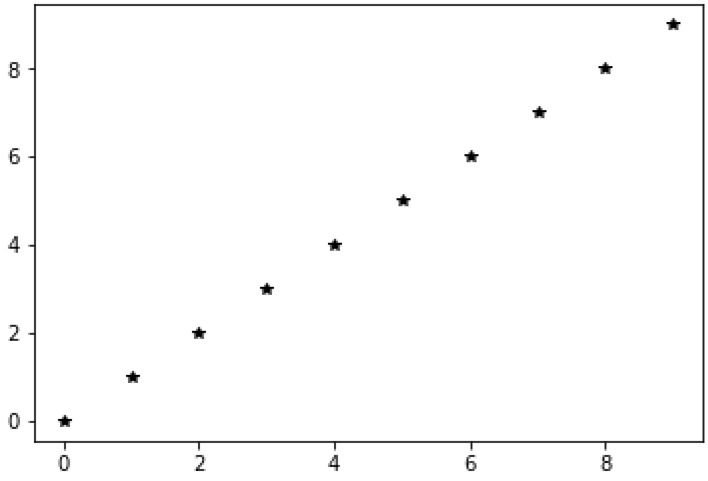
```
In [35]: plt.plot(arr4, 'r+') #using red plus '+' sign
```

```
Out[35]: [<matplotlib.lines.Line2D at 0x1d88d8ef4e0>]
```



```
In [36]: plt.plot(arr4, 'k*')
```

```
Out[36]: [<matplotlib.lines.Line2D at 0x1d88d947128>]
```

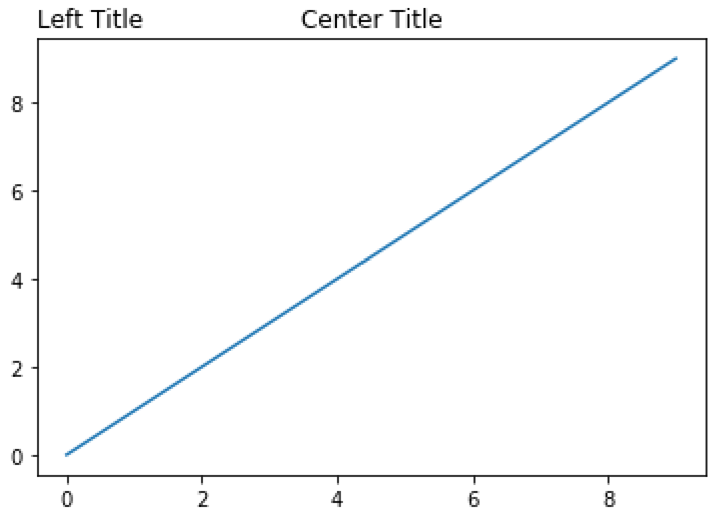


In [37]:

```
plt.title('Center Title')
plt.plot(arr4)
plt.title('Left Title', loc='left')
#plt.title('Right Title', loc='right')
```

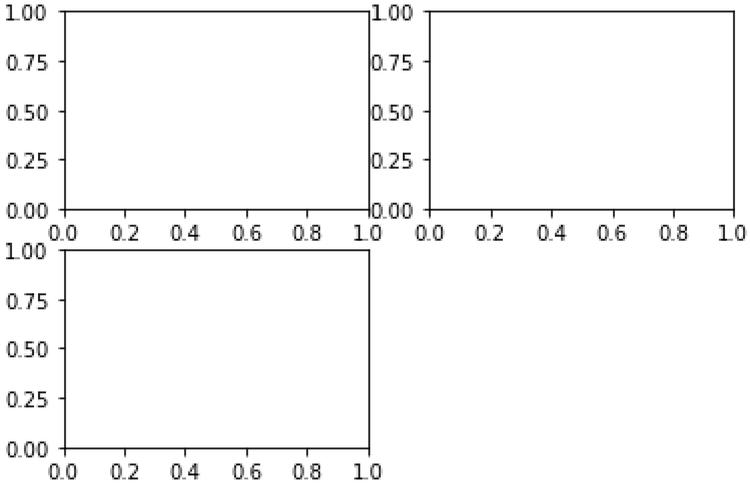
Out[37]:

```
Text(0,1,'Left Title')
```



In [38]:

```
# Figures and Subplots
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
```



In [39]:

```
'''
For more in matplotlib, Refer to:

    https://matplotlib.org/3.1.1/index.html

'''
```

Out[39]:

```
'\nFor more in matplotlib, Refer to:\n\n    https://matplotlib.org/3.1.1/inde
x.html\n\n'
```