In [2]:
```
import numpy as np
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(a)
print(type(a))
print(a[0])
print(a[1])
print(a[2])
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
<class 'numpy.ndarray'>
[1 2 3]
[4 5 6]
[7 8 9]
```

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

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

In [3]:
```
print(np.ones((1,2)))
print(np.zeros((2,2)))
print(np.full((2,2),6))
```

```
[[1. 1.]]
[[0. 0.]
 [0. 0.]]
[[6 6]
 [6 6]]
```

In [5]:
```
print(np.eye(5))
print(np.random.random((3,2)))
```

```
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
[[0.57723649 0.48552283]
 [0.92808777 0.45374634]
 [0.4395876  0.82239667]]
```

In [6]:
```
b=a[:2,1:3]
print(b)
row_r1=a[1,:]
row_r2=a[1:2,:]
row_r3=a[-1:,:]
print(row_r1,row_r1.shape)
print(row_r2,row_r2.shape)
print(row_r3,row_r3.shape)
```

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

In [7]:
```
print(np.array([a[0,0],a[1,1],a[0,2]]))
```

```
[1 5 3]
```

In [8]:
```
col_r1=a[:,1]
col_r2=a[:,1:2]
print(col_r1,col_r1.shape)
print(col_r2,col_r2.shape)
```

```
[2 5 8] (3,)
[[2]
 [5]
 [8]] (3, 1)
```

In [11]:
```
b=np.array([0,2,1])
print(a[np.arange(3),b])
```

```
[1 6 8]
```

In [12]:
```
a=np.array([[1,2],[3,4],[5,6]])
bool_idx=(a>2)
print(bool_idx)
```

```
[[False False]
 [ True  True]
 [ True  True]]
```

In [13]:
```
print(a[[a>2]])
```

```
[3 4 5 6]
```

```
<ipython-input-13-4c91deed23eb>:1: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  print(a[[a>2]])
```

In [17]:
```
x=np.array([[1,2],[3,4]],dtype=np.float64)
y=np.array([[5,6],[7,8]],dtype=np.float64)
print('x',x)
print('y',y)
print(x+y)
print(np.add(x,y))
print(x-y)
print(np.subtract(x,y))
print(x*y)
print(np.multiply(x,y))
print(x/y)
print(np.divide(x,y))
print(np.sqrt(x))
print(np.sqrt(y))
```

```
x [[1. 2.]
 [3. 4.]]
y [[5. 6.]
 [7. 8.]]
[[ 6.  8.]
 [10. 12.]]
[[ 6.  8.]
 [10. 12.]]
[[-4. -4.]
 [-4. -4.]]
[[-4. -4.]
 [-4. -4.]]
[[ 5. 12.]
 [21. 32.]]
[[ 5. 12.]
 [21. 32.]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[1.         1.41421356]
 [1.73205081 2.        ]]
[[2.23606798 2.44948974]
 [2.64575131 2.82842712]]
```

In [18]:
```
v=np.array([9,10])
w=np.array([11,12])
print(np.dot(v,w))
```

```
219
```

In [19]:
```
print(x.dot(v))
```

```
[29. 67.]
```

In [20]:
```
print(np.sum(x))
print(np.sum(x,axis=0))
print(np.sum(x,axis=1))
```

```
10.0
[4. 6.]
[3. 7.]
```

In [21]:
```
print(x)
print(x.T)
```

```
[[1. 2.]
 [3. 4.]]
[[1. 3.]
 [2. 4.]]
```

In [22]:
```
x=np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
v=np.array([1,0,1])
y=np.empty_like(x)

for i in range(4):
    y[i,:]=x[i,:]+v
print(y)
```

```
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

In [24]:
```
vv=np.tile(v,(4,1))
print(vv)
```

```
[[1 0 1]
 [1 0 1]
 [1 0 1]
 [1 0 1]]
```

In [25]:
```
y=x+vv
print(y)
```

```
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

In [34]:
```
x=np.array([[1,2,3]])
w=np.array([4,5,6])
print(np.reshape(v,(3,1))*w)
```

```
[[ 4  5  6]
 [ 8 10 12]
 [12 15 18]]
```

In [33]:
```
x=np.array([[1,2,3],[4,5,6]])
print(v+x)
```

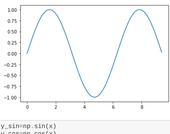
```
[[2 4 6]
 [5 7 9]]
```

In [35]:
```
print(x*2)
```

```
[[ 2  4  6]
 [ 8 10 12]]
```

In [3]:
```
import matplotlib.pyplot as plt
x=np.arange(0,3*np.pi,0.1)
y=np.sin(x)
plt.plot(x,y)
```

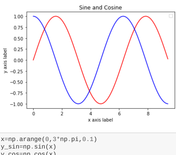
Out[3]:
```
[<matplotlib.lines.Line2D at 0x19817f34a90>]
```



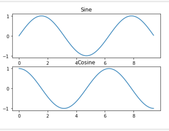
In [6]:
```
y_sin=np.sin(x)
y_cos=np.cos(x)
plt.plot(x,y_sin,'-r')
plt.plot(x,y_cos,'b')
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend('Sine','Cosine')
```

```
<ipython-input-6-aa1cdae5ec25>:8: UserWarning: Legend does not support 'S' instances.
A proxy artist may be used instead.
See: http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for-adding-to-the-legend-aka-proxy-artists
  plt.legend('Sine','Cosine')
<ipython-input-6-aa1cdae5ec25>:8: UserWarning: Legend does not support 'i' instances.
A proxy artist may be used instead.
See: http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for-adding-to-the-legend-aka-proxy-artists
  plt.legend('Sine','Cosine')
<ipython-input-6-aa1cdae5ec25>:8: UserWarning: Legend does not support 'n' instances.
A proxy artist may be used instead.
See: http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for-adding-to-the-legend-aka-proxy-artists
  plt.legend('Sine','Cosine')
<ipython-input-6-aa1cdae5ec25>:8: UserWarning: Legend does not support 'e' instances.
A proxy artist may be used instead.
See: http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for-adding-to-the-legend-aka-proxy-artists
  plt.legend('Sine','Cosine')
```

Out[6]:
```
<matplotlib.legend.Legend at 0x19817f5dc70>
```



In [8]:
```
x=np.arange(0,3*np.pi,0.1)
y_sin=np.sin(x)
y_cos=np.cos(x)
plt.subplot(2,1,1)
plt.plot(x,y_sin)
plt.title('Sine')
plt.subplot(2,1,2)
plt.plot(x,y_cos)
plt.title('Cosine')
plt.show()
```



In [ ]: