Mathematical function

Q1.SIN() - This function helps user to calculate trigonometric sine for all x(being the array elements) ¶

In [16]:
```
# sin() function

import numpy as np


array = [0, math.pi / 2, np.pi / 3, np.pi]
print ("Input array : \n", array)

Sin_Values = np.sin(array)
print ("\nSine values : \n", Sin_Values)
```

```
Input array :
 [0, 1.5707963267948966, 1.0471975511965976, 3.141592653589793]

Sine values :
 [0.00000000e+00 1.00000000e+00 8.66025404e-01 1.22464680e-16]
```

we can use np also as well as math also to use pi functions. here pi value is defined in the package as 3.14159

In [14]:
```
import numpy as np
import math

array = [0, math.pi / 2, math.pi / 3, math.pi]
print ("Input array : \n", array)

Sin_Values = np.sin(array)
print ("\nSine values : \n", Sin_Values)
```

```
Input array :
 [0, 1.5707963267948966, 1.0471975511965976, 3.141592653589793]

Sine values :
 [0.00000000e+00 1.00000000e+00 8.66025404e-01 1.22464680e-16]
```

Q2. COS() - This mathematical function helps user to calculate trignmetric cosine for all x(being the array elements).

In [25]:
```
# Python program explaining
# cos() function

import numpy as np
import math

array = [0, np.pi / 2, np.pi / 3, np.pi]
print ("Input array : \n", array)

cos_Values = np.cos(array)
print ("\nCosine values : \n", cos_Values)
```

```
Input array :
 [0, 1.5707963267948966, 1.0471975511965976, 3.141592653589793]

Cosine values :
 [ 1.000000e+00  6.123234e-17  5.000000e-01 -1.000000e+00]
```

Q3. TAN() - This mathematical function helps user to calculate trigonometric tangent for all x(being the array elements).

2pi Radians = 360 degrees

tan(x) = sin(x) / cos(x)

An array with trigonometric sine of x for all x i.e. array elements

In [26]:
```
import numpy as np
import math

in_array = [0, math.pi / 4, 3*np.pi / 2, math.pi/6]
print ("Input array : \n", in_array)

tan_Values = np.tan(in_array)
print ("\nTan values : \n", tan_Values)
```

```
Input array :
 [0, 0.7853981633974483, 4.71238898038469, 0.5235987755982988]

Tan values :
 [0.00000000e+00 1.00000000e+00 5.44374645e+15 5.77350269e-01]
```

Q4. conversion from Radian to degrees

In [27]:
```
# rad2deg() function

import numpy as np
import math

array = [0, math.pi / 2, np.pi / 3, np.pi]
print ("Radian values : \n", array)

conversion = np.rad2deg(array)
print ("\nDegree values : \n", conversion)
```

```
Radian values :
 [0, 1.5707963267948966, 1.0471975511965976, 3.141592653589793]

Degree values :
 [  0.  90.  60. 180.]
```

In [28]:
```
#2nd method to get radians converted to degrees
# degrees() function
import numpy as np
import math

in_array = [0, math.pi / 2, np.pi / 3, np.pi]
print ("Radian values : \n", in_array)

degree_Values = np.degrees(in_array)
print ("\nDegree values : \n", degree_Values)
```

```
Radian values :
 [0, 1.5707963267948966, 1.0471975511965976, 3.141592653589793]

Degree values :
 [  0.  90.  60. 180.]
```

Q5. conversion from Degrees to Radian

In [29]:
```
# degrees() function

import numpy as np
import math

in_array = np.arange(10.) * 90
print ("Degree values : \n", in_array)

radian_Values = np.radians(in_array)
print ("\nRadian values : \n", radian_Values)
```

```
Degree values :
 [  0.  90. 180. 270. 360. 450. 540. 630. 720. 810.]

Radian values :
 [ 0.          1.57079633  3.14159265  4.71238898  6.28318531  7.85398163
  9.42477796 10.99557429 12.56637061 14.13716694]
```

In [30]:
```
#method2

# degrees() function

import numpy as np
import math

in_array = np.arange(10.) * 90
print ("Degree values : \n", in_array)

radian_Values = np.radians(in_array)
print ("\nRadian values : \n", radian_Values)
```

```
Degree values :
 [  0.  90. 180. 270. 360. 450. 540. 630. 720. 810.]

Radian values :
 [ 0.          1.57079633  3.14159265  4.71238898  6.28318531  7.85398163
  9.42477796 10.99557429 12.56637061 14.13716694]
```

Q6. Hyperbolic Functions

This mathematical function helps user to calculate hypotenuse for the right angled triangle, given its side and perpendicular.

In [31]:
```
# hypot() function

import numpy as np

leg1 = [12, 3, 4, 6]
print ("leg1 array : ", leg1)


leg2 = [5, 4, 3, 8]
print ("leg2 array : ", leg2)

result = np.hypot(leg1, leg2)
print("\nHypotenuse is as follows :")
print(result)
```

```
leg1 array :  [12, 3, 4, 6]
leg2 array :  [5, 4, 3, 8]

Hypotenuse is as follows :
[13.  5.  5. 10.]
```