Untitled4

June 24, 2020

In [1]:
```
import numpy as np
a = np.array([1, 2, 3])
print(a)
```

```
[1 2 3]
```

In [5]:
```
#Prints 'a' array
print(type(a))
```

```
<class 'numpy.ndarray'>
```

In [3]:
```
print(type(a))
```

```
<class 'numpy.ndarray'>
```

In [4]:
```
#Prints shape of an array
print(np.shape(a))
```

```
(3,)
```

In [43]:
```
b = np.array([[1,2], [3, 4]])
 #Prints array 'b'
 print(b)
 #Prints shape of array'b'
 print(np.shape(b))
```

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

In [7]:
```
#Create a two dimensional array
b = np.array([[1,2], [3, 4]])
#Prints array 'b'
print(b)
#Prints shape of array'b'
print(np.shape(b))
```

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

```
In [8]: #to access particular values of an array
        b[1,0]
```

```
Out[8]: 3
```

```
In [45]: c = np.zeros((3,2))
```

```
In [12]: #creates 3 * 2 null matrix
         c = np.zeros((3,2))
         print(c)
```

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

```
In [13]: d = np.ones((3,2))
         print(d)
```

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

```
In [14]: #constant array with specified value
         e = np.full((3,2), 9)
         print(e)
```

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

```
In [16]: # creates a 3 by 3 array conatining randomly generating values between 0 and 1 contai
         g = np.random.random((3,3))
         print(g)
```

```
[[0.29125203 0.4933862  0.9869859 ]
 [0.72567183 0.35158399 0.10827599]
 [0.10753116 0.76955505 0.38322366]]
```

```
In [18]: h = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
         print(h)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

```
In [19]: #slicing ans accessong numpy array
         h1 = h[:2,1:3]
         # :22 means that access first row and stop before 2 ie 3rd row
         # separed by comma it means that now column must be accessed
         #1:3 means now start with 2nd column and end before 4th column ie you access 2nd and
         print(h1)
```

```
[[2 3]
 [6 7]]
```

```
In [20]: i = h[:2,1:3]
         print(i)
```

```
[[2 3]
 [6 7]]
```

```
In [21]: #modify slice i
         i[0,0] = 500
         #if any array is modified in sliced part (here is i) same is modified in the parentr
         print(h)
```

```
[[  1 500   3   4]
 [  5   6   7   8]
 [  9  10  11  12]]
```

```
In [26]: #data type recognition
         j = np.array([1,2])
         print(j)
         print(j.dtype) #it is integer type

         #floating data type
         k = np.array([1.0,2.0])
         print(k)
         print(k.dtype)

         #to convert a floating datatype array to integer array
         l = np.array([1.0, 2.0], dtype = np.int64)
         print(l)
         print(l.dtype)
```

```
[1 2]
int64
```

```
[1. 2.]
float64
[1 2]
int64
```

```
In [28]: #mathematical operations on numpy(add, subtract, multiply, divide)
         x = np.array([[1,2], [3,4]], dtype = np.float64)
         y = np.array([[5,6], [7,8]], dtype = np.float64)
         print(x + y)
         print(x-y)
         print(x * y)
         print(x / y)
         #squarerot of each and every element of array
         print(np.sqrt(x))
```

```
[[ 6.  8.]
 [10. 12.]]
[[-4. -4.]
 [-4. -4.]]
[[ 5. 12.]
 [21. 32.]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[1.         1.41421356]
 [1.73205081 2.        ]]
```

```
In [31]: #sum of all the elements of array
         np.sum(x)

         #sum of the elements of the first row
         np.sum(x, axis = 0)

         # sum of the elements of the second row
         np.sum(x, axis = 1)
```

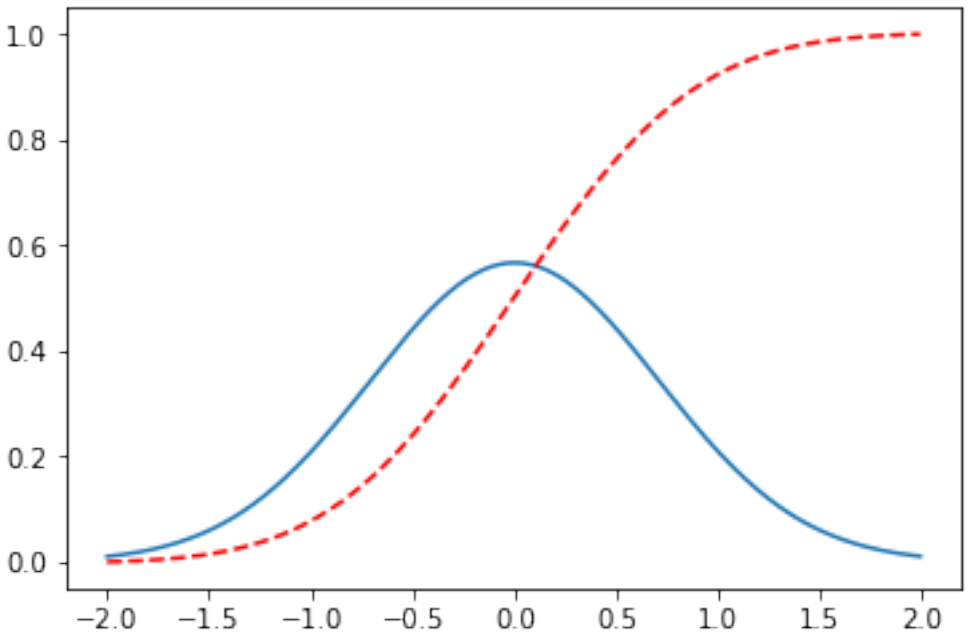
```
Out[31]: array([3., 7.])
```

```
In [32]: #use of SciPy.stats library and NumPy
         from scipy import stats
         import numpy as np
```

```
In [35]: #generating 10 ramdom element sized list using stats
         print(stats.norm.rvs(size = 10))
```

```
[ 0.39959027 -2.24074009  1.43400416 -0.49754881  0.41996935  0.16019778
  0.91381089 -0.30288283  0.29651274 -0.98456169]
```

In [38]:
```
from pylab import *

# Create some test data
dx = .01
X  = np.arange(-2,2,dx)
Y  = exp(-X**2)

# Normalize the data to a proper PDF
Y /= (dx*Y).sum()

# Compute the CDF
CY = np.cumsum(Y*dx)

# Plot both
plot(X,Y)
plot(X,CY,'r--')

show()
```



In [39]:
```
### Compute the Normal CDF of certain values
print(stats.norm.cdf(np.array([-1.5, -.5, 1, 1.5, 2, 6])))
```

```
[0.0668072  0.30853754 0.84134475 0.9331928  0.97724987 1.        ]
```

In [42]:
```
# Generate 1000 Students T continuous random variables.
np.random.seed(1234)
```

```
x = stats.t.rvs(10, size = 1000)
# equivalent to np.min(x)
print(x.min())
# equivalent to np.min(x)
print(x.max())
# equivalent to np.mean(x)
print(x.mean())
# equivalent to np.var(x)
print(x.var())
stats.describe(x)
```

```
-3.7081318682862507
5.489762351653657
0.03301501424446721
1.216841589019068
```

Out[42]: DescribeResult(nobs=1000, minmax=(-3.7081318682862507, 5.489762351653657), mean=0.033