NumPy Random Library Functions

May 18, 2022

0.1 This library contains several functions to create nd array with random data

*randint -> to generate random int values in the given range

*bydefault it is int type so change the type will get error

*bt by the use of reshape can casting the type

[2]:
```
import numpy as np

np.random.randint(10,20) #within the given range
```

[2]:
```
16
```

0.2 1-D Array

[3]:
```
np.random.randint(1,9, size = 10)
```

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

[4]:
```
#from 0 to 9 2D Array
np.random.randint(10, size=(3,2))
```

[4]:
```
array([[4, 0],
       [7, 1],
       [9, 6]])
```

[5]:
```
#from 0 to 9 3D Array
np.random.randint(100, size=(2,3,2))
```

[5]:
```
array([[[79, 35],
        [ 8, 30],
        [91,  2]],

       [[47, 13],
        [23, 10],
        [ 9, 74]]])
```

*rand -> crate an array of the given shape and populate it with random samples from a uniform distribution

*a single float value

*range is always 0 to 1

*uniform ditribution with the range

[6]:
```
np.random.rand()
```

[6]:
```
0.2384815527255847
```

[7]:
```
#uniform -> customized range,uniformd distriution
#bydefault float

np.random.uniform()
```

[7]:
```
0.4077971201461119
```

[8]:
```
np.random.uniform(4) #given range
```

[8]:
```
3.1633919828671395
```

0.3 2-D Array

[9]:
```
np.random.uniform(2,4,3)
```

[9]:
```
array([3.42528638, 2.78813797, 3.73109626])
```

0.4 3-D Array

[10]:
```
np.random.uniform(3,4,size=(3,2))
```

[10]:
```
array([[3.9933388 , 3.36757568],
       [3.76592993, 3.38201394],
       [3.16827378, 3.55462665]])
```

[12]:
```
#randn -> values from normal distribution with mean 0 and standard deviation 1

a = np.random.randn()
b = np.random.randn(3) #1-D Array
c = np.random.randn(3,2) #2-D Array
d = np.random.randn(3,2,3) #3-D Array

print("a =",a,"\nb =",b,"\nc =",c,"\nd =",d)
```

```
a = 0.20218097111845543
b = [ 0.32662209  2.40234859 -0.60914865]
c = [[-0.20334519  0.05004447]
 [ 0.02048258 -0.22011091]
```

```
 [ 0.28794346  0.07423437]]
d = [[[ 0.48118823 -1.47072481 -0.54136893]
  [ 0.65841912  0.35202305 -0.59549074]]

 [[-1.53337466 -0.8227271   2.5014208 ]
  [ 0.8570757  -1.47305731 -0.74924317]]

 [[-0.14151686 -0.91402197 -0.01195046]
  [ 0.09490856  1.11938873  0.26728949]]]
```

[13]:
```
#normal -> customised mean and standard deviation
#by deafult mean 0.0 standard deviation 1.0

a = np.random.normal()
b = np.random.normal(3)
c = np.random.normal(3,2)
d = np.random.normal(3,2,4)

print(a,b,c,d)
```

```
0.8264339094410303 3.093722919437927 1.8040351203433627 [2.02115291 2.82440226
2.86036974 2.22810279]
```

[14]:
```
#Shuffle -> Modify a sequence in-place by shuffling its content

a = np.arange(9) #1D Array
np.random.shuffle(a)
a
```

[14]:
```
array([3, 6, 0, 5, 8, 1, 7, 2, 4])
```

[15]:
```
#2D Array
#only axis-0 will be change for multidimensinnal array

a = np.random.randint(1,101,size=(3,4))
a
```

[15]:
```
array([[51, 67, 63, 11],
       [79, 53, 12, 58],
       [33, 30, 28, 98]])
```

[18]:
```
np.random.shuffle(a)
a
```

[18]:
```
array([[33, 30, 28, 98],
       [51, 67, 63, 11],
       [79, 53, 12, 58]])
```