numpy_lab

April 11, 2020

[51]:
```
import numpy as np  # import numpy module as np
```

[52]:
```
a=np.array([1,2,3]) # Creating 1D array
```

[53]:
```
print("Data type of a = ",a.dtype)  # return the data type of the array
```

```
Data type of a =  int32
```

[54]:
```
print("a = ",a)
```

```
a =  [1 2 3]
```

[55]:
```
matrix = np.array ([np.arange (3), [i for i in range(1 ,4)],[6 ,7 ,8]])
print("matrix = ",matrix)
```

```
matrix =  [[0 1 2]
 [1 2 3]
 [6 7 8]]
```

[56]:
```
#2.2 Initialization
```

[57]:
```
print("array of all zero of float data type = ",np.zeros((5,2,2),dtype=float))␣
↪# array of all zero of float data type
```

```
array of all zero of float data type =  [[[0. 0.]
  [0. 0.]]

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

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

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

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

[58]:
```
np.ones(4,5) # array full of one's
```

```
      ␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-58-0cc42b671757> in <module>
   ----> 1 np.ones(4,5) # array full of one's


       c:
↪\users\user\appdata\local\programs\python\python38-32\lib\site-packages\numpy\core\numeric.
↪py in ones(shape, dtype, order)
       205
       206     """
   --> 207     a = empty(shape, dtype, order)
       208     multiarray.copyto(a, 1, casting='unsafe')
       209     return a


       TypeError: data type not understood
```

[59]:
```
print("array full of one's = ",np.ones((4,5),dtype=int)) # array full of one's
```

```
array full of one's =  [[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]
```

[60]:
```
print("array which initial content is random = ",np.empty([3,4])) #array which␣
 ↪initial content is random
```

```
array which initial content is random =  [[1.50689109e-312 0.00000000e+000
8.76794447e+252 2.15895723e+227]
 [6.48224638e+170 3.67145870e+228 1.18015315e-095 9.03292329e+271]
 [9.08366793e+223 1.41075687e+232 1.16070543e-028 9.28330511e-310]]
```

[61]:
```
print("array with evenly spaced values = ",np.arange (2 ,10 ,2)) # array with␣
 ↪evenly spaced values
```

```
array with evenly spaced values =  [2 4 6 8]
```

[62]:
```
print("array with evenly spaced values = ",np.arange (2 ,10 ,1)) # array with␣
↪evenly spaced values
```

```
array with evenly spaced values =  [2 3 4 5 6 7 8 9]
```

[63]:
```
print("rearranging the size of the array = ",np.arange (2 ,10 ,1).reshape(4,2))␣
↪#rearranging the size of the array
```

```
rearranging the size of the array =  [[2 3]
 [4 5]
 [6 7]
 [8 9]]
```

[64]:
```
print("creates an array with constant values = ",np.full ([2 ,3] , 4)) #␣
↪creates an array with constant values
```

```
creates an array with constant values =  [[4 4 4]
 [4 4 4]]
```

[65]:
```
print("creates an identity matrix = ",np.eye(3)) # creates an identity matrix
```

```
creates an identity matrix =  [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

[66]:
```
print("creates an evenly spaced array within specified interval = ",np.linspace␣
↪(2 ,3,5)) # creates an evenly spaced array within specified interval
```

```
creates an evenly spaced array within specified interval =  [2.   2.25 2.5  2.75
3.  ]
```

[67]:
```
#2.3 Copying, Sorting, Slicing
```

[68]:
```
nt = np.copy(matrix) #returns the copy of the object
print("nt =\n",nt)
```

```
nt =
 [[0 1 2]
 [1 2 3]
 [6 7 8]]
```

[69]:
```
st = matrix.copy() #deep copy
print("st = \n",st)
```

```
st =
 [[0 1 2]
 [1 2 3]
 [6 7 8]]
```

[70]:
```
print("\noriginal matrix = \n",matrix)
```

```
original matrix =
 [[0 1 2]
 [1 2 3]
 [6 7 8]]
```

[71]:
```
matrix[0,2] = 5
print("\nchanged original matrix = \n",matrix)
```

```
changed original matrix =
 [[0 1 5]
 [1 2 3]
 [6 7 8]]
```

[72]:
```
print("\nafter matrix chnage \n st = \n",st)
#st copy has changed to the new values of the matrix
```

```
after matrix chnage
 st =
 [[0 1 2]
 [1 2 3]
 [6 7 8]]
```

[73]:
```
print("\nafter matrix chnage \n nt = \n",nt)
#nt has not chnaged to the new values of the matrix
```

```
after matrix chnage
 nt =
 [[0 1 2]
 [1 2 3]
 [6 7 8]]
```

[74]:
```
#again change the matrix to the original values
matrix[0,2] = 2
print("original matrix = \n",matrix)
```

```
original matrix =
 [[0 1 2]
 [1 2 3]
 [6 7 8]]
```

[75]:
```
rt = matrix.view() #shallow copy
print("rt = \n",rt)
```

```
rt =
 [[0 1 2]
 [1 2 3]
 [6 7 8]]
```

[76]:
```
matrix[0,2] = 10
print("\nchanged original matrix = \n",matrix)
```

```
changed original matrix =
 [[ 0  1 10]
 [ 1  2  3]
 [ 6  7  8]]
```

[77]:
```
print("\nafter matrix chnage \n rt = \n",rt)
#rt copy has changed to the new values of the matrix
```

```
after matrix chnage
 rt =
 [[ 0  1 10]
 [ 1  2  3]
 [ 6  7  8]]
```

[78]:
```
#change the matrix
matrix[0,0] = 20
print("changed original matrix = \n",matrix)
```

```
changed original matrix =
 [[20  1 10]
 [ 1  2  3]
 [ 6  7  8]]
```

[79]:
```
matrix.sort() # sorts in ascending order
print("changed original matrix = \n",matrix)
```

```
changed original matrix =
 [[ 1 10 20]
 [ 1  2  3]
 [ 6  7  8]]
```

[80]:
```
#change the matrix
matrix[0,0] = 30
print("changed original matrix = \n",matrix)
```

```
changed original matrix =
 [[30 10 20]
 [ 1  2  3]
```

```
 [ 6  7  8]]
```

[81]:
```
matrix.sort(axis=1) #sort along the specified axis
print("changed original matrix = \n",matrix)
```

```
changed original matrix =
 [[10 20 30]
 [ 1  2  3]
 [ 6  7  8]]
```

[82]:
```
#again change the matrix to the original values
matrix[0,0] = 0
matrix[0,1] = 1
matrix[0,2] = 2
print("original matrix = \n",matrix)
```

```
original matrix =
 [[0 1 2]
 [1 2 3]
 [6 7 8]]
```

[83]:
```
print(matrix [0: ,:1]) # 2D array slicing
```

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

[84]:
```
print(matrix [:2, 0:2])
```

```
[[0 1]
 [1 2]]
```

[85]:
```
print(matrix [:1, :])
```

```
[[0 1 2]]
```

[86]:
```
#2.3.1 Try out
```

[87]:
```
print(matrix [1,0])
```

```
1
```

[88]:
```
matrix [0] = 42
```

[89]:
```
print("matrix = ",matrix)
```

```
matrix =  [[42 42 42]
 [ 1  2  3]
 [ 6  7  8]]
```

[90]:
```
print(matrix [1:3])
```

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

[91]:
```
matrix []
```

```
  File "<ipython-input-91-7fe32dadec60>", line 1
    matrix []
           ^
SyntaxError: invalid syntax
```

[92]:
```
print(matrix [1:])
```

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

[93]:
```
print(matrix [1:100])
```

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

[94]:
```
print(matrix [:])
```

```
[[42 42 42]
 [ 1  2  3]
 [ 6  7  8]]
```

[95]:
```
print(matrix [1: ,:2])
```

```
[[1 2]
 [6 7]]
```

[96]:
```
print(matrix [:2, 1:])
```

```
[[42 42]
 [ 2  3]]
```

[97]:
```
print(matrix.ravel ())
```

```
[42 42 42  1  2  3  6  7  8]
```

[98]:
```
print(matrix [: ,1]. copy ())
```

```
[42  2  7]
```

[99]:
```
print(matrix [1]. tolist ())
```

```
[1, 2, 3]
```

[100]:
```
matrix.reshape(-1)
```

```
  File "<ipython-input-100-defe75ef5f96>", line 1
    matrix.reshape(-1)
                   ^
SyntaxError: invalid character in identifier
```

[101]:
```
print("matrix = ",matrix)
```

```
matrix =  [[42 42 42]
 [ 1  2  3]
 [ 6  7  8]]
```

[102]:
```
print(matrix.reshape(9,order='F'))
```

```
[42  1  6 42  2  7 42  3  8]
```

[103]:
```
#2.4.1 Try out
```

[104]:
```
print(np.sqrt(matrix))
```

```
[[6.4807407  6.4807407  6.4807407 ]
 [1.         1.41421356 1.73205081]
 [2.44948974 2.64575131 2.82842712]]
```

[105]:
```
print(np.exp(matrix))
```

```
[[1.73927494e+18 1.73927494e+18 1.73927494e+18]
 [2.71828183e+00 7.38905610e+00 2.00855369e+01]
 [4.03428793e+02 1.09663316e+03 2.98095799e+03]]
```

[106]:
```
print(np.min(matrix))
```

```
1
```

[107]:
```
print(np.max(matrix, axis=1))
```

```
[42  3  8]
```

[108]:
```
print(np.min(np.maximum(np.random.randn(4),np.random.randn(4))))
```

```
-0.9749816589426924
```

[109]:
```
print(np.mean(matrix))
```

```
17.0
```

[110]:
```
print(np.mean(matrix, axis=0))
```

```
[16.33333333 17.         17.66666667]
```

[111]:
```
print(np.sum(matrix))
```

```
153
```

[112]:
```
print(np.invert(matrix))
```

```
[[-43 -43 -43]
 [ -2  -3  -4]
 [ -7  -8  -9]]
```

[113]:
```
print(np.random.randn(5))
```

```
[ 1.21396917  0.45760764  0.66382144 -0.62428362 -1.75958896]
```

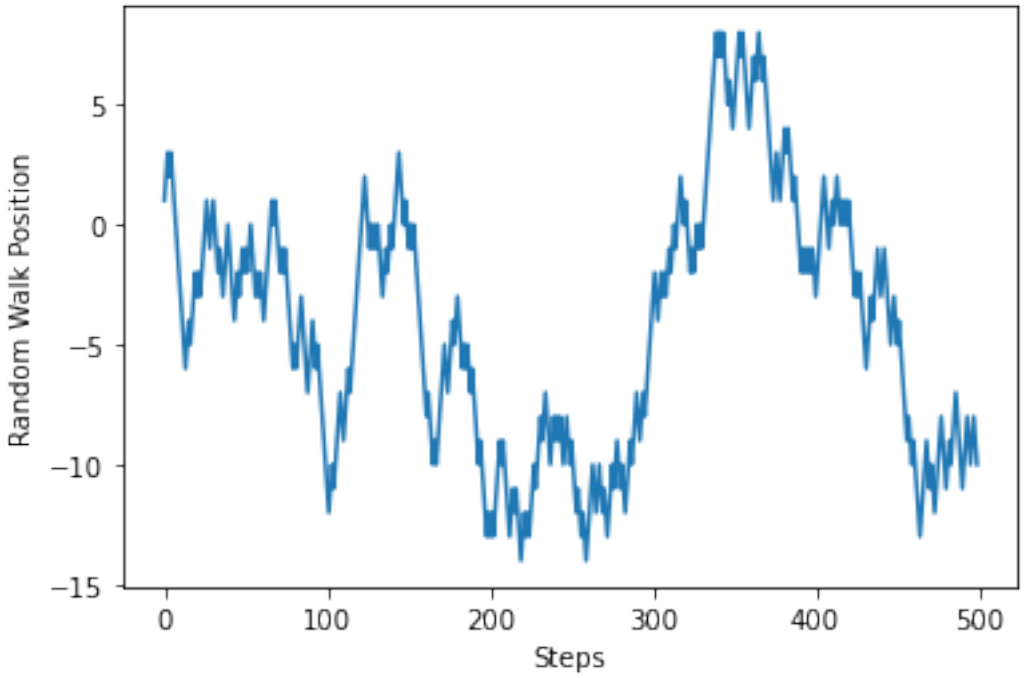
[114]:
```
print(np.trace(matrix))
```

```
52
```

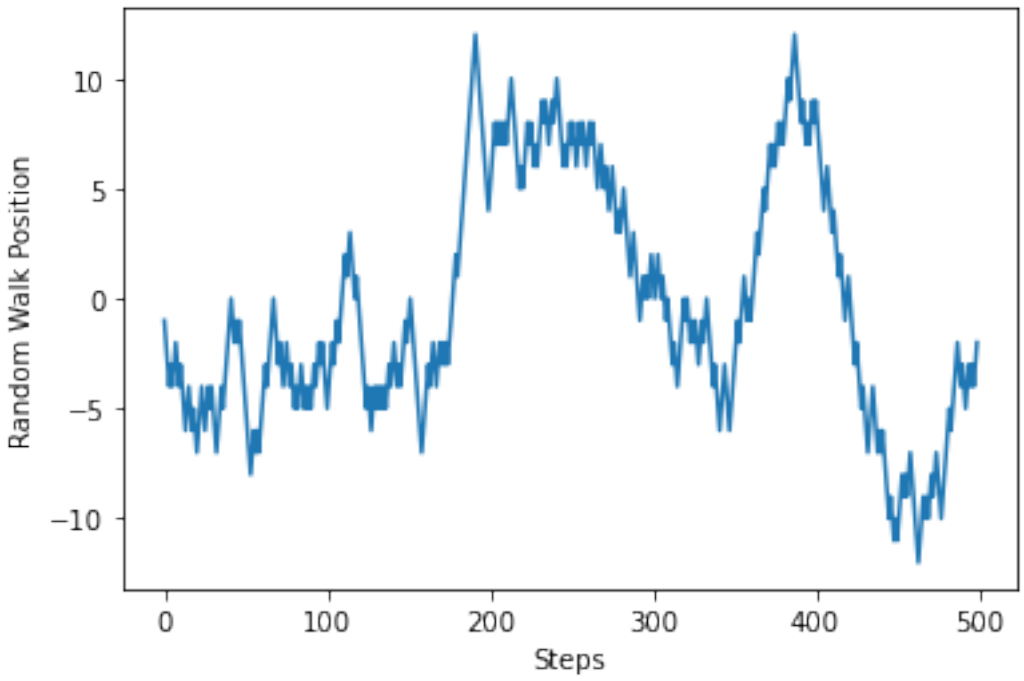
[115]:
```
#implement Random Walk

import random

def randomWalk(startPosition):
    walked_path = np.zeros(500)
    count = 0

    for i in range(startPosition,500):
        test = random.random()
        #+1 and -1 has equal probability
        if test >= 0.5:
            count = count + 1
        else:
            count = count - 1

        walked_path[i] = count

    return walked_path
```

[116]:
```
#plot the random walk
import matplotlib.pyplot as plt
```

```
steps = np.arange(500)

plt.figure()
plt.xlabel('Steps')
plt.ylabel('Random Walk Position')

plt.plot(steps,randomWalk(0))
plt.show()
```



[117]:
```
#plot the random walk
import matplotlib.pyplot as plt

steps = np.arange(500)

plt.figure()
plt.xlabel('Steps')
plt.ylabel('Random Walk Position')

plt.plot(steps,randomWalk(0))
plt.show()
```


Random Walk Position
10
5
0
−5
−10
0
100
200
300
400
500
Steps

[ ]:

[ ]: