Numpy Excersice

Import Numpy as np

```
import numpy as np
```

Create an array of 10 zeros:

```
np.zeros(10)
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Create an array of 10 ones

```
np.ones(10)
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Create an array of 10 fives.

```
np.ones(10)*5
```

```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

Create an of the integers from 10 to 50

```
np.arange(10,51)
```

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
       44, 45, 46, 47, 48, 49, 50])
```

Create an array of all even integers from 10 to 50

```
np.arange(10,51,2)
```

```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48, 50])
```

Create a 3x3 matrix with values ranging from 0 to 8

```
np.arange(0,9).reshape(3,3)
```

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

Create a 3x3 identity matrix

```
np.eye(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Use Numpy to generate random numbers between 0 to 1

```
np.random.rand(1)
```

```
array([0.76907018])
```

Use Numpy to generate an array of 25 random numbers sampled fom a standard normal distribution

```
np.random.randn(25)
```

```
array([ 1.44428174, -0.84042247, -1.27551624,  1.32061676, -0.96490202,
       -0.96709608, -0.48034012, -1.76522918, -1.33335097,  0.44102016,
       -1.16436776, -0.64989913, -0.23215637, -0.6341166 ,  1.18096016,
        0.35599542,  1.61733408,  1.13025844, -0.18414053, -0.50053492,
       -0.26418736, -0.06124532, -0.28831899,  1.70893584,  0.71301749])
```

Create the following matrix

```
np.linspace(0.01,1,100)
```

```
array([0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 , 0.11,
       0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21, 0.22,
       0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32, 0.33,
       0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43, 0.44,
       0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54, 0.55,
       0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65, 0.66,
       0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77,
       0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88,
       0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99,
       1.  ])
```

```
np.arange(1,101)/100
```

```
array([0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 , 0.11,
       0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21, 0.22,
       0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32, 0.33,
       0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43, 0.44,
       0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54, 0.55,
       0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65, 0.66,
       0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77,
       0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88,
       0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99,
       1.  ])
```

Create an array of 20 linearly spaced points between 0 and 1:

```
np.linspace(0,1,20)
```

```
array([0.        , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
       0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
       0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
       0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ])
```

Numpy Indexing and Selection

Create a matric of 5x5 with elements ranging from 1 to 25

```
arr = np.arange(1,26).reshape(5,5)
print(arr)
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]]
```

Create a matrix using indexing

```
arr[2:,1:]
```

```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

```
arr[1:,2:]
```

```
array([[ 8,  9, 10],
       [13, 14, 15],
       [18, 19, 20],
       [23, 24, 25]])
```

```
arr[3:4,-1:]
```

```
array([[20]])
```

```
arr[0:3,1:2]
```

```
array([[ 2],
       [ 7],
       [12]])
```

```
arr[2:3,-2]
```

```
array([14])
```

```
arr[4:]
```

```
array([[21, 22, 23, 24, 25]])
```

```
arr[3:]
```

```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

Get the sum of all the values in the matrix

```
arr.sum()
```

```
325
```

Get the standard deviation of the values in the matrix

```
arr.std()
```

```
7.211102550927978
```

Get the sum of all the columns in the matrix

```
arr.sum(axis = 0)
```

```
array([55, 60, 65, 70, 75])
```

Get the sum of al the rows in the matrix

```
arr.sum(axis = 1)
```

```
array([ 15,  40,  65,  90, 115])
```

1. Create 1-D, 2-D and 3-D numpy array

```
oned = np.array(10)
print(oned)
```

```
10
```

```
twod = np.arange(10).reshape(5,2)
print(twod)
```

```
[[0 1]
 [2 3]
 [4 5]
 [6 7]
 [8 9]]
```

```
threed = np.arange(1,13).reshape(3,2,2)
print(threed)
```

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

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
```

2. Print datatype of an array

```
print(oned.dtype)
print(twod.dtype)
print(threed.dtype)
```

```
int32
int32
int32
```

3. Print shape and dimensions of all the three different array.

```
oned.shape
```

```
()
```

```
twod.shape
```

```
(5, 2)
```

```
threed.shape
```

```
(3, 2, 2)
```

4. Create a 1-D array of 12 elements filled with zero.

```
oned = np.zeros(12)
print(oned)
```

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

5. Create a 2-D array elements filled with ones

```
twod = np.ones(12).reshape(3,4)
print(twod)
```

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

6. Create a 1-D array known as Maths Mark out of 100 having 20 elements

```
Math_Mark = np.random.randint(0,101,20)
print("Math marks : ",Math_Mark)
```

```
Math marks :  [  0  36   4  59  12  12 100  25   3  21  19  56  40  97  69  59  81   7
  68  66]
```

7. Sort the above array in ascending and descending order.

```
asc = np.sort(Math_Mark)
dsc = np.sort(Math_Mark)[::-1]
print("Ascending order : ",asc)
print("Descending order :",dsc)
```

```
Ascending order :  [  0   3   4   7  12  12  19  21  25  36  40  56  59  59  66  68
69  81
  97 100]
Descending order : [100  97  81  69  68  66  59  59  56  40  36  25  21  19  12  12
7   4
   3   0]
```

8. Create a copy of sorted array to another variable by deep copying.

```

```

9. Create a 2-D array of 5x5 which is an identity matrix and then convert it to 1-D array.

```
twod = np.eye(5)
oned = twod.flatten()
print("2-D Identity Matrix :",twod)
print("1-D Identity Matrix :",oned)
```

```
2-D Identity Matrix : [[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.]]
1-D Identity Matrix : [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.]
```

10. Create 2-D array of 3x3. Add one more row to it.

```
# twod = np.arange(9).reshape(3,3)
# print(twod)
# new_row = np.array([[65,75,85]])
# new_twod = np.concatenate((twod,new_row),axis=0)
# print(new_twod)
```

11. Split the above array created vertically and horizontally.

```
```

12. Create 4x4 identity matrix. Add 4 to each element.

```
print((np.eye(4))+4)
```

```
[[5. 4. 4. 4.]
 [4. 5. 4. 4.]
 [4. 4. 5. 4.]
 [4. 4. 4. 5.]]
```

13. Create 1-D array. Initialize with floating point values. Display its ceil, floor and round (with 2 decimal's).

```
oned = np.random.uniform(1,5,5)
print(oned)
print(np.ceil(oned))
print(np.floor(oned))
print(np.round(oned, decimals=2))
```

```
[3.16404923 1.89607821 1.67762083 1.16447892 4.09561402]
[4. 2. 2. 2. 5.]
[3. 1. 1. 1. 4.]
[3.16 1.9  1.68 1.16 4.1 ]
```

14. Create 5x5 matrix. Display min, max, variance and standard deviation column-wise.

```
mat = np.random.randint(1,100,(5,5))
print(mat)
print("Column-wise statistics.")
print("Minimum Values :",np.min(mat, axis=0))
print("Maximum Values :",np.max(mat, axis=0))
print("Variance :",np.var(mat, axis=0))
print("Standard Deviation :",np.std(mat, axis=0))
```

```
[[83 58 55 24 62]
 [92 67 95  9 57]
 [60 80 44 33 81]
 [51 37 51 23 32]
 [35 14  5 27 55]]
Column-wise statistics.
Minimum Values : [35 14  5  9 32]
Maximum Values : [92 80 95 33 81]
Variance : [434.16 542.16 822.4   62.56 245.84]
Standard Deviation : [20.83650643 23.28432949 28.67751733  7.90948797 15.6792857 ]
```

15. Create a null vector of size 10 with 6 th value filled with 200.

```
null_vector = np.zeros(10)
null_vector[5] = 200
print(null_vector)
```

```
[  0.   0.   0.   0.   0. 200.   0.   0.   0.   0.]
```

16. Create a 1-D Array/Vector with values ranging from 100 to 150 in sequence.

```
print(np.arange(100,151))
```

```
[100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150]
```

17. Reverse the above vector.

```
arr = np.arange(100,151)
print(arr[::-1])
```

```
[150 149 148 147 146 145 144 143 142 141 140 139 138 137 136 135 134 133
 132 131 130 129 128 127 126 125 124 123 122 121 120 119 118 117 116 115
 114 113 112 111 110 109 108 107 106 105 104 103 102 101 100]
```

18. Create a 5x5 matrix with values ranging from 0-24 using some function.

```
print(np.arange(25).reshape(5,5))
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

19. Find indices of non-zero elements from [1,0,2,0,0,4,0,8]

```
arr = [1,0,2,0,0,4,0,8]
print(np.nonzero(arr))
```

```
(array([0, 2, 5, 7], dtype=int64),)
```

20. Create a 10x10 matrix with 1 on the borders and 0 inside.

```
matrix = np.zeros((10,10))
matrix[0,:] = 1
matrix[-1,:] = 1
matrix[:,0] = 1
matrix[:,-1] = 1
print(matrix)
```

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

21. Create 8x8 matrix and fill with 1 and 0 in chessboard pattern.

```
matrix = np.zeros((8, 8), dtype=int)
for i in range(8):
    for j in range(8):
        if (i + j) % 2 == 0:
            matrix[i, j] = 1
print(matrix)
```

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

22. Create two 3x3 matrix with random integer values. Perform matrix addition, multiplication and subtraction.

```
mat_1 = np.random.randint(1,100,(3,3))
mat_2 = np.random.randint(1,100,(3,3))
print("Matrix A is :\n",mat_1)
print("Matrix B is :\n",mat_2)
print("The sum of the matrix A and B is :\n",(mat_1+mat_2))
print("The difference between matrix A and B is :\n",(mat_1-mat_2))
print("The difference between matrix B and A is :\n",(mat_2-mat_1))
print("The product of matrix A and B is :\n",(np.dot(mat_1,mat_2)))
```

```
Matrix A is :
 [[93 73  9]
 [14 15  5]
 [79 49 40]]
Matrix B is :
 [[45 50 25]
 [47 79  2]
 [84  8 48]]
The sum of the matrix A and B is :
 [[138 123  34]
 [ 61  94   7]
 [163  57  88]]
The difference between matrix A and B is :
 [[ 48  23 -16]
 [-33 -64   3]
 [ -5  41  -8]]
The difference between matrix B and A is :
 [[-48 -23  16]
 [ 33  64  -3]
 [  5 -41   8]]
The product of matrix A and B is :
 [[ 8372 10489  2903]
 [ 1755  1925   620]
 [ 9218  8141  3993]]
```