21BAI1105_AnandMisra

September 6, 2023

1 Assignment 1

1.1 Anand Misra - 21BAI1105

2 NumPy Exercises

Now that we've learned about NumPy let's test your knowledge. We'll start off with a few simple tasks, and then you'll be asked some more complicated questions.

Import NumPy as np

[2]:
```
import numpy as np
```

Create an array of 10 zeros

[3]:
```
arr_0 = np.zeros(10)
arr_0
```

[3]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Create an array of 10 ones

[5]:
```
arr_1 = np.ones(10)
arr_1
```

[5]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Create an array of 10 fives

[7]:
```
arr_5 = np.ones(10) * 5
arr_5
```

[7]:
```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

Create an array of the integers from 10 to 50

[9]:
```
arr_10to50 = np.arange(10,51)
arr_10to50
```

[9]:
```
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 the even integers from 10 to 50

[10]:
```
arr_even = np.arange(10,51,2)
arr_even
```

[10]:
```
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

[11]:
```
matrix_3x3 = np.arange(9).reshape(3, 3)
matrix_3x3
```

[11]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Create a 3x3 identity matrix

[12]:
```
id_matrix_3x3 = np.eye(3)
id_matrix_3x3
```

[12]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Use NumPy to generate a random number between 0 and 1

[13]:
```
rand = np.random.rand()
rand
```

[13]:
```
0.364865756037417
```

Use NumPy to generate an array of 25 random numbers sampled from a standard normal distribution

[15]:
```
rand_sample = np.random.randn(25)
rand_sample
```

[15]:
```
array([ 0.79520668,  1.73029785,  0.50144765, -0.06138078, -2.28399832,
        0.8607485 ,  0.84008663,  0.90509332, -0.9899608 , -1.37568289,
        0.24905431, -0.09195265,  1.2824688 ,  0.05616804, -0.25514416,
       -0.41962915,  0.81113355,  0.25674028,  1.18758735, -0.00540194,
       -2.21112132, -0.59796574, -0.54279296, -2.06703142, -1.77173925])
```

Create the following matrix:

[16]:
```
matrix_100 = np.arange(0.01, 1.01, 0.01).reshape(10, 10)
matrix_100
```

[16]:
```
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:

[17]:
```
equispaced_20 = np.linspace(0, 1, 20)
equispaced_20
```

[17]:
```
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.        ])
```

2.1 Numpy Indexing and Selection

Now you will be given a few matrices, and be asked to replicate the resulting matrix outputs:

[18]:
```
mat = np.arange(1,26).reshape(5,5)
mat
```

[18]:
```
array([[ 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]])
```

[ ]:
```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

[25]:
```
mat_q1 = mat[2:,1:]
mat_q1
```

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

[ ]:
```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

[27]:
```
mat_q2 = mat[3,4]
mat_q2
```

[27]: 20

[ ]:
```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

[32]:
```
mat_q3 = mat[:3,1].reshape(3, 1)
mat_q3
```

[32]:
```
array([[ 2],
       [ 7],
       [12]])
```

[ ]:
```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

[33]:
```
mat_q4 = mat[4,]
mat_q4
```

[33]:
```
array([21, 22, 23, 24, 25])
```

[ ]:
```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

[35]:
```
mat_q5 = mat[3:,]
mat_q5
```

[35]:
```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

2.1.1 Now do the following

Get the sum of all the values in mat

[37]:
```
sum = np.sum(mat)
sum
```

[37]: 325

Get the standard deviation of the values in mat

[38]:
```
sd = np.std(mat)
sd
```

[38]: 7.211102550927978

Get the sum of all the columns in mat

[41]:
```
sum_cols = np.sum(mat, axis=0)
sum_cols
```

[41]:
```
array([55, 60, 65, 70, 75])
```