Numpy Exercise

September 5, 2023

1 NumPy Exercises 21BCE2512

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

[ ]:
```
%pip install numpy
import numpy as np
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: numpy in
/home/kyojuro/.local/lib/python3.10/site-packages (1.25.2)
Note: you may need to restart the kernel to use updated packages.
```

Create an array of 10 zeros

[ ]:
```
arr_zeroes = np.zeros(10)
display (arr_zeroes)
```

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

Create an array of 10 ones

[ ]:
```
arr_ones = np.ones(10, dtype = int)
display(arr_ones)
```

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

Create an array of 10 fives

[ ]:
```
arr_five = np.full(10,5,dtype = int)
display(arr_five)
```

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

Create an array of the integers from 10 to 50

[ ]:
```
arr_10_50 = np.arange(10,51)
display(arr_10_50)
```

```
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

[ ]:
```
arr_even_10_50 = np.arange(10,51,2)
display(arr_even_10_50)
# print(type(arr_10_50))
```

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

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

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

[ ]:
```
arr_0_8 = np.arange(0,9,1).reshape(3,3)
display(arr_0_8)
```

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

Create a 3x3 identity matrix

[ ]:
```
arr_3_identity = np.identity(3)
display(arr_3_identity)
```

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

Use NumPy to generate a random number between 0 and 1

[ ]:
```
random_number = np.random.rand()
print(random_number)
```

```
0.6051528058257452
```

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

[ ]:
```
normal_dist_arr = np.random.randn(25)
display(normal_dist_arr)
```

```
array([-0.00903023, -1.34567575, -0.64467222,  0.1951025 ,  0.95713059,
        0.20085406,  0.53002494,  0.16311673,  0.89439882,  0.97385788,
        0.13067791,  0.62701358,  0.93374976, -0.90650531,  0.17514234,
        0.81118621, -0.36769895,  2.25677267,  0.36169802,  0.08317119,
        1.15627475, -0.14599534,  1.53250424, -1.53581598, -0.21094977])
```

Create the following matrix:

[ ]:
```
# linearly_sdpaced_points_100 = np.linspace(0.01,1,100).reshape(10,10)
#or
linearly_sdpaced_points_100 = np.array([[0.01*(i*10+j+1) for j in range(10)]␣
 ↪for i in range(10)] )
display(linearly_sdpaced_points_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:

[ ]:
```
linearly_spaced_points_20 = np.linspace(0,1,20)
display(linearly_spaced_points_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.        ])
```

1.1 Numpy Indexing and Selection

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

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

```
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
```

[ ]:
```
display(mat[2:, 1:])
```

```
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
```

```
[ ]: mat[3,4]
```

```
[ ]: 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
```

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

```
[ ]: 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
```

```
[ ]: mat[4]
```

```
[ ]: 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
```

```
[ ]: mat[3:]
```

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

1.1.1 Now do the following

Get the sum of all the values in mat

```
[ ]: display(sum(sum(mat[0:])))
```

```
325
```

Get the standard deviation of the values in mat

```
[ ]: mat.std()
```

[ ]: 7.211102550927978

Get the sum of all the columns in mat

[ ]:
```
mat.sum(axis=1)
```

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