Sathwik beesetty
21BCE8881

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

In [2]:

```
import numpy as np
```

Create an array of 10 zeros

In [5]:

```
a = np.zeros(10)
a
```

Out[5]:

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

Create an array of 10 ones

In [6]:

```
b = np.ones(10)
b
```

Out[6]:

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

In [ ]:

Create an array of 10 fives

In [15]:

```
c = np.full(10,5)
c
```

Out[15]:

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

Create an array of the integers from 10 to 50

In [8]:

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

Out[8]:

```
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

In [9]:

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

Out[9]:

```
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

In [10]:

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

Out[10]:

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

Create a 3x3 identity matrix

In [11]:

```
g = np.eye(3)
g
```

Out[11]:

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

Use NumPy to generate a random number between 0 and 1

In [14]:

```
h = np.random.rand()
h
```

Out[14]:

```
0.5147187409781624
```

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

In [16]:

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

Out[16]:

```
array([-0.64208584,  0.10839438,  0.05108948, -0.84775804,  0.1428205
1,
        0.74936515,  0.69057708,  0.6963928 ,  1.1892605 ,  0.3453379
4,
        0.41224452,  2.4326844 , -1.42764703, -0.50370689, -0.6595084
8,
        0.24421014, -0.45102205, -0.8362114 , -2.06270078,  0.5413523
9,
       -0.05338208, -2.53015547, -0.44246809,  0.1466329 , -0.6664274
4])
```

Create the following matrix:

In [17]:

```
j = np.arange(0.01,1.0,0.01)
j
```

Out[17]:

```
array([0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 , 0.1
1,
       0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21, 0.2
2,
       0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32, 0.3
3,
       0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43, 0.4
4,
       0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54, 0.5
5,
       0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65, 0.6
6,
       0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.7
7,
       0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.8
8,
       0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.9
9])
```

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

In [18]:

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

Out[18]:

```
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

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

In [40]:

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

Out[40]:

```
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]])
```

In [0]:

```
# 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
```

In [41]:

```
mat[2:5,1:5]
```

Out[41]:

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

In [0]:

```
# 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
```

In [42]:

```
mat[3:4,4:5]
```

Out[42]:

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

In [0]:

```
# 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
```

In [43]:

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

Out[43]:

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

In [0]:

```
# 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
```

In [44]:

```
mat[4:5,0:5]
```

Out[44]:

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

In [0]:

```
# 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
```

In [45]:

```
mat[3:5,0:5]
```

Out[45]:

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

Now do the following

Get the sum of all the values in mat

In [46]:

```
m = np.sum(mat)
m
```

Out[46]:

```
325
```

Get the standard deviation of the values in mat

In [48]:

```
sd = np.std(mat)
sd
```

Out[48]:

```
7.211102550927978
```

Get the sum of all the columns in mat

In [49]:

```
colSum = np.sum(mat,axis=0)
colSum
```

Out[49]:

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

Type *Markdown* and LaTeX: ^{2}