

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 [17]: a=np.zeros(10)  
a
```

```
Out[17]: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Create an array of 10 ones

```
In [19]: b=np.ones(10)  
b
```

```
Out[19]: array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Create an array of 10 fives

```
In [24]: c=np.full(10,5.0)  
c
```

```
Out[24]: array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

Create an array of the integers from 10 to 50

```
In [67]: d=np.arange(10, 51)  
d
```

```
Out[67]: 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 [46]: e=np.arange(5,26)  
e*2
```

```
Out[46]: 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 [47]: f=np.arange(0,9).reshape(3,3)
f
```

```
Out[47]: array([[0, 1, 2],
               [3, 4, 5],
               [6, 7, 8]])
```

Create a 3x3 identity matrix

```
In [42]: g=np.identity(3)
g
```

```
Out[42]: array([[1., 0., 0.],
               [0., 1., 0.],
               [0., 0., 1.]])
```

Use NumPy to generate a random number between 0 and 1

```
In [40]: h=np.random.rand(1)
h
```

```
Out[40]: array([0.1579828])
```

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

```
In [50]: i=np.random.randn(5,5)
i
```

```
Out[50]: array([[ 0.69975907, -1.58531697, -0.04558781, -1.21882024, -1.17055975],
               [-0.5524359 , -0.31580087,  0.43004574, -0.28069326,  0.44643482],
               [-0.88447388,  0.31093362,  0.58288625, -0.84098706,  1.26026282],
               [-1.06562106,  0.96282645, -2.11575465, -0.94090827,  1.14050361],
               [ 0.09826363, -1.34719006, -0.34954564, -0.1523859 ,  0.09822067]])
```

Create the following matrix:

```
In [51]: j=np.arange(0.01,1.01,0.01).reshape(10,10)
j
```

```
Out[51]: 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:

```
In [52]: k=np.linspace(0,1,20)
k
```

```
Out[52]: 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 [13]: l = np.arange(1,26).reshape(5,5)
l
```

```
Out[13]: 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 [ ]: # 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 [53]: m=l[2:5, 1:5]
m
```

```
Out[53]: array([[12, 13, 14, 15],
                [17, 18, 19, 20],
                [22, 23, 24, 25]])
```

```
In [ ]: # 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 [59]: n=l[3,4]
n
```

```
Out[59]: 20
```

```
In [ ]: # 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 [58]: o=l[0:3,1:2]
o
```

```
Out[58]: array([[ 2],
                [ 7],
                [12]])
```

```
In [ ]: # 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 [60]: p=l[4,]
p
```

```
Out[60]: array([21, 22, 23, 24, 25])
```

```
In [ ]: # 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 [62]: q=l[3:5,0:5]  
q
```

```
Out[62]: 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 [63]: r=np.sum(l)  
r
```

```
Out[63]: 325
```

Get the standard deviation of the values in mat

```
In [64]: s=np.std(l)  
s
```

```
Out[64]: 7.211102550927978
```

Get the sum of all the columns in mat

```
In [65]: t=np.sum(l, axis=0)  
t
```

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