Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Exercise 5 - Numpy

Name: Ooi Caaron

IC Number: 990701-07-5837

Date : 24/6/23

Introduction : Using Numpy built in function to return the output

Conclusion : Still need to do a lot practice

EXERCISE 5

Numpy

In [3]:

```
import numpy as np
```

Question 1

Create a new array of 2*2 integers, without initializing entries.

In [22]:

```
array = np.ones((2, 2), dtype=int)
# array = np.array([[1, 2], [3, 4]], dtype=int)
array
```

Out[22]:

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

Question 2

Create a new array of 3*2 float numbers, filled with ones.

In [9]:

```
array = np.ones((3,2), dtype=int)
array
```

Out[9]:

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

Question 3

Create a 1-D array of 50 evenly spaced elements between 3. and 10., inclusive.

In [11]:

```
array = np.linspace(3,10,50)
array
```

Out[11]:

```
array([ 3.        ,  3.14285714,  3.28571429,  3.42857143,  3.57142857,
        3.71428571,  3.85714286,  4.        ,  4.14285714,  4.28571429,
        4.42857143,  4.57142857,  4.71428571,  4.85714286,  5.        ,
        5.14285714,  5.28571429,  5.42857143,  5.57142857,  5.71428571,
        5.85714286,  6.        ,  6.14285714,  6.28571429,  6.42857143,
        6.57142857,  6.71428571,  6.85714286,  7.        ,  7.14285714,
        7.28571429,  7.42857143,  7.57142857,  7.71428571,  7.85714286,
        8.        ,  8.14285714,  8.28571429,  8.42857143,  8.57142857,
        8.71428571,  8.85714286,  9.        ,  9.14285714,  9.28571429,
        9.42857143,  9.57142857,  9.71428571,  9.85714286, 10.        ])
```

Question 4

Create a 1-D array of 50 element spaced evenly on a log scale between 3. and 10., exclusive.

In [13]:

```
array = np.logspace(3, 10, 50, False)
array
```

Out[13]:

```
array([1.00000000e+03, 1.38038426e+03, 1.90546072e+03, 2.63026799e+03,
       3.63078055e+03, 5.01187234e+03, 6.91830971e+03, 9.54992586e+03,
       1.31825674e+04, 1.81970086e+04, 2.51188643e+04, 3.46736850e+04,
       4.78630092e+04, 6.60693448e+04, 9.12010839e+04, 1.25892541e+05,
       1.73780083e+05, 2.39883292e+05, 3.31131121e+05, 4.57088190e+05,
       6.30957344e+05, 8.70963590e+05, 1.20226443e+06, 1.65958691e+06,
       2.29086765e+06, 3.16227766e+06, 4.36515832e+06, 6.02559586e+06,
       8.31763771e+06, 1.14815362e+07, 1.58489319e+07, 2.18776162e+07,
       3.01995172e+07, 4.16869383e+07, 5.75439937e+07, 7.94328235e+07,
       1.09647820e+08, 1.51356125e+08, 2.08929613e+08, 2.88403150e+08,
       3.98107171e+08, 5.49540874e+08, 7.58577575e+08, 1.04712855e+09,
       1.44543977e+09, 1.99526231e+09, 2.75422870e+09, 3.80189396e+09,
       5.24807460e+09, 7.24435960e+09])
```

Question 5

Let x be a ndarray [10, 10, 3] with all elements set to one. Reshape x so that the size of the second dimension equals 150.

In [18]:

```
x = np.ones((10, 10, 3))  # Create ndarray with all elements set to one
x.reshape((2,150))  # Reshape x with second dimension size 150
```

Out[18]:

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

Question 6

Let x be array [[1, 2, 3], [4, 5, 6]]. Convert it to [1 4 2 5 3 6].

In [9]:

```
x = np.array([[1, 2, 3], [4, 5, 6]])
result = x.reshape(-1).reshape((-1, 2)).T.flatten()
# array([[1, 4, 2, 5, 3, 6]])
```

Out[9]:

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

Question 7

Let x be an array
[[1 2 3]
[4 5 6]].

and y be an array
[[7 8 9]
[10 11 12]].
Concatenate x and y so that a new array looks like
[[1, 2, 3, 7, 8, 9],
[4, 5, 6, 10, 11, 12]].

In [26]:

```
x = np.array([[ 1, 2, 3],[ 4, 5 ,6]])
y = np.array([[ 7 ,8 ,9],[10, 11, 12]])
array = np.hstack((x, y))
array
```

Out[26]:

```
array([[ 1,  2,  3,  7,  8,  9],
       [ 4,  5,  6, 10, 11, 12]])
```

Question 8

Let x be an array [1, 2, 3, ..., 9]. Split x into 3 arrays, each of which has 4, 2, and 3 elements in the original order.

In [28]:

```
x = np.arange(1, 10)
split_arrays = np.split(x, [4, 6])
split_arrays
# [array([1, 2, 3, 4]), array([5, 6]), array([7, 8, 9])]
```

Out[28]:

```
[array([1, 2, 3, 4]), array([5, 6]), array([7, 8, 9])]
```

Question 9

Let x be an array
[[1 2 3 4]
[5 6 7 8].
Shift elements one step to right along the second axis.

In [41]:

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

array = np.array([[1,2,3,4],[5,6,7,8]])
x = np.flip(array[0])
y = np.flip(array[1])
reversed = np.vstack((x,y))

reversed
```

Out[41]:

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

Question 10

Let x be an array [0, 1, 2]. Convert it to
[[0, 1, 2, 0, 1, 2],
[0, 1, 2, 0, 1, 2]].

In [47]:

```
x = np.array([0,1,2])
array1 = np.hstack((x,x))
array = np.vstack((array1,array1))
print(array1)
array
# array([[0, 1, 2, 0, 1, 2],
#        [0, 1, 2, 0, 1, 2]])
```

```
[0 1 2 0 1 2]
```

Out[47]:

```
array([[0, 1, 2, 0, 1, 2],
       [0, 1, 2, 0, 1, 2]])
```

In [ ]:

```

```