DATA 690 Homework 4 (60 points - Due on Sunday, October 9, 2022 by 11:00 pm ET)

The output of this assignment for submission should be in PDF format **AND** .py or .ipynb. The name of the file should be as follows: Lastname_Firstname_Homework4.pdf (example: Thomas_Sunela_Homework4.pdf) **AND** Lastname_Firstname_Homework4.ipynb (example: Thomas_Sunela_Assignment4.ipynb. In short, you are submitting the python notebook as well as the pdf of that notebook. Do **NOT** submit .html file, the system will give you an error.

Incorrect file name will cost you points!

Instructions for converting a Jupyter Python notebook to PDF: Go to the menu and choose, File --> Download As --> html. Open that html file and print it to PDF. Submit the PDF file **NOT** the html file.

If you are using Google Colab, remember to review the PDF before submitting to ensure that all cells and answers are displayed in the PDF.

Things to note:

- Each cell should display an output
- Use only the basic Python concepts and methods
- Use both Markdown and code comments in the Jupyter Notebook as needed

The results of your code should match my results. Both our outputs should be displayed**

READ the IMPORTANT NOTE below

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.

IMPORTANT NOTE! Make sure you don't run the cells directly above the example output shown, otherwise you will end up writing over the example output!

1. Import NumPy as np

In [87]:
```
#Importing numpy as np
import numpy as np
```

2. Create an array of 10 zeros

In [88]:
```
# YOUR CODE HERE

#Zeros() generate array of 0's
np.zeros(10)
```

Out[88]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [2]:
```
# DON'T WRITE HERE
```

Out[2]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

3. Create an array of 10 ones

In [4]:
```
# YOUR CODE HERE
#Ones() generate array of 1's
np.ones(10)
```

Out[4]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [3]:
```
# DON'T WRITE HERE
```

Out[3]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

4. Create an array of 10 fives

In [6]:
```
# YOUR CODE HERE
#Multiplying array of 1's with 5
np.ones(10)*5
```

Out[6]:
```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

In [4]:
```
# DON'T WRITE HERE
```

Out[4]:
```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

5. Create an array of the integers from 10 to 50

In [8]:
```
# YOUR CODE HERE
#arange() generates array of the integers
np.arange(10,51)
```

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])
```

In [5]:
```
# DON'T WRITE HERE
```

Out[5]:
```
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])
```

6. Create an array of all the even integers from 10 to 50

In [9]:
```
# YOUR CODE HERE
#arange() generates array of the integers with interval
np.arange(10,51,2)
```

Out[9]:
```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48, 50])
```

In [6]:
```
# DON'T WRITE HERE
```

Out[6]:
```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48, 50])
```

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

In [12]:
```
# YOUR CODE HERE
#Creating a 3x3 matrix using reshape()
np.arange(0,9).reshape(3,3)
```

Out[12]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [7]:
```
# DON'T WRITE HERE
```

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

8. Create a 3x3 identity matrix

In [13]:
```
# YOUR CODE HERE
 np.identity(3)
```

Out[13]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [89]:
```
np.eye(3)
```

Out[89]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

identity() or eye() returns identity matrix

In [8]:
```
# DON'T WRITE HERE
```

Out[8]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

9. Use NumPy to generate a random number between 0 and 1

NOTE: Your result's value should be different from the one shown below.

In [15]:
```
# YOUR CODE HERE
#Using rand(1) to  generate a random number between 0 and 1
np.random.rand(1)
```

Out[15]:
```
array([0.07128285])
```

In [96]:
```
#Using randint(0,1) to  generate a random number between 0 and 1
np.random.randint(0,1)
```

Out[96]:
```
0
```

In [94]:
```
# DON'T WRITE HERE
#Output given here was array([0.65248055])
```

array([0.65248055])

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

NOTE: Your result's values should be different from the ones shown below.

In [98]:
```
# YOUR CODE HERE
#using randn to generate an array of 25 random numbers sampled from a standard normal distribution

np.random.randn(5,5)
```

Out[98]:
```
array([[ 1.01251548, -0.91386915, -1.02953021,  1.20979645,  0.5018723 ],
       [ 0.13884618,  0.64076111,  0.52733267, -1.15436024, -2.21333348],
       [-1.68175651, -1.78809425, -2.21853495, -0.64743078, -0.52840432],
       [-0.03920917,  0.21497595, -0.3843588 , -0.25390408,  0.07325207],
       [-0.99720384, -0.71385629,  0.03541635, -0.67794537, -0.57188106]])
```

In [10]:
```
# DON'T WRITE HERE
```

Out[10]:
```
array([ 1.80076712, -1.12375847, -0.98524305,  0.11673573,  1.96346762,
        1.81378592, -0.33790771,  0.85012656,  0.0100703 , -0.91005957,
        0.29064366,  0.69906357,  0.1774377 , -0.61958694, -0.45498611,
       -2.0804685 , -0.06778549,  1.06403819,  0.4311884 , -1.09853837,
        1.11980469, -0.48751963,  1.32517611, -0.61775122, -0.00622865])
```

11. Create the following matrix:

In [113...
```
# YOUR CODE HERE
```

Method1:

In [111...
```
#Creating matrix from a list using np array and reshape
my_list=[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.
my_array=np.array(my_list)
my_matrix= my_array.reshape(10,10)
my_matrix
```

Out[111...
```
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.  ]])
```

Method2:

In [115...
```
#Creating matrix from matrix array
my_matrix = [[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,
np.array(my_matrix)
```

Out[115...
```
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.  ]])
```

In [11]:
```
# DON'T WRITE HERE
```

Out[11]:
```
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.  ]])
```

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

In [20]:
```
# YOUR CODE HERE
#Using linspace to create an array of 20 linearly spaced points between 0 and 1
np.linspace(0,1,20)
```

Out[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.        ])
```

In [12]:
```
# DON'T WRITE HERE
```

Out[12]:
```
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 starting matrix (be sure to run the cell below!), and be asked to replicate the resulting matrix outputs:

In [23]:
```
# RUN THIS CELL - THIS IS YOUR STARTING MATRIX
mat = np.arange(1,26).reshape(5,5)
mat
```

Out[23]:
```
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]])
```

13. Write code that reproduces the output shown below.

Be careful not to run the cell immediately above the output, otherwise you won't be able to see the output any more.

In [103...
```
# YOUR CODE HERE
mat[2:,1:]
```

Out[103...
```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

In [104...
```
mat[2:5,1:5]
```

Out[104...
```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

In [14]:
```
# DON'T WRITE HERE
```

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

14. Write code that reproduces the output shown below.

In [55]:
```
# YOUR CODE HERE
mat[3][4]
```

Out[55]:
```
20
```

In [57]:
```
mat[3,4]
```

Out[57]:
```
20
```

In [15]:
```
# DON'T WRITE HERE
```

Out[15]:
```
20
```

15. Write code that reproduces the output shown below.

In [65]:
```
# YOUR CODE HERE
mat[0:3,1:2]
```

Out[65]:
```
array([[ 2],
       [ 7],
       [12]])
```

In [16]:
```
# DON'T WRITE HERE
```

Out[16]:
```
array([[ 2],
       [ 7],
       [12]])
```

16. Write code that reproduces the output shown below.

In [67]:
```
# YOUR CODE HERE
mat[4,:]
```

Out[67]:
```
array([21, 22, 23, 24, 25])
```

In [17]:
```
# DON'T WRITE HERE
```

Out[17]:
```
array([21, 22, 23, 24, 25])
```

17. Write code that reproduces the output shown below.

In [105...
```
# YOUR CODE HERE
mat[3:]
```

Out[105...
```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [18]:
```
# DON'T WRITE HERE
```

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

NumPy Operations

18. Get the sum of all the values in mat

In [81]:
```
# YOUR CODE HERE
mat.sum()
```

Out[81]:
```
325
```

In [19]:
```
# DON'T WRITE HERE
```

Out[19]:
```
325
```

19. Get the sum of all the columns in mat

In [82]:
```
# YOUR CODE HERE
mat.sum(axis=0)
```

Out[82]:
```
array([55, 60, 65, 70, 75])
```

In [21]:
```
# DON'T WRITE HERE
```

Out[21]:
```
array([55, 60, 65, 70, 75])
```

20. We worked a lot with random data with numpy, but is there a way we can insure that we always get the same random numbers? If so, write the code

In [85]:
```
# YOUR CODE HERE
np.random.seed(12)
np.random.rand(3)
```

Out[85]:
```
array([0.15416284, 0.7400497 , 0.26331502])
```

In [86]:
```
np.random.seed(12)
np.random.rand(3)
```

Out[86]:
```
array([0.15416284, 0.7400497 , 0.26331502])
```

seed() always generates the same random numbers by setting the random state