W. P. Carey School of Business
Arizona State University

CIS 325: Programming for Business Analytics

Assignment 5 (Individual work)

Submission Instructions

- Submit your Python script according to the guidelines in this document as a .py file with the following naming convention: "A5-[ASURITE ID].py", where you will replace the text "[ASURITE ID]" with your ASURITE ID (= your ASU login ID).

  For example, if your ASURITE ID is "abcd12", then your submission file name would be "A5-abcd12.py".

- On the top (header) of your submission .py file, add your name and email address as shown below:

```
"""
@author: [YOUR FULL NAME]
@email: [YOUR EMAIL ADDRESS]"""
```

- Add Python comment statements (such as "*# Answer to Question 1* ") in your submission .py file to separate your answers between questions.
- Not following the above submission instructions will result in a **0.2 point reduction** from your grade.

Question 1 (1 points)

Research the numpy lecture materials and numpy online help to create a random sample of numbers from a *standard normal distribution* using the numpy randn statement. Create enough random sample numbers to populate a multidimensional numpy array with a shape of 5 elements by 3 elements.

For instance, the output to the Python console might look like this:

```
array([[ 0.65876993,  0.2655893 ,  1.13088307],
       [ 0.68855403, -0.49286531,  1.39454068],
       [ 0.41191895,  0.20237657,  0.20445781],
       [-1.11698295,  1.97901341,  0.30472651],
       [-0.10527436,  0.13313341,  0.71505922]])
```

Obviously, since these are random numbers, for your case, your random numbers would be different than what you see in the screenshot above.

Your last step in your script is to multiply each of your created random sample numbers in the numpy multidimensional array by 100 using the numpy mathematical batch operations covered in our lectures. For instance, the output to the Python console would look like this:

```
array([[  65.87699257,   26.55893022,  113.08830708],
       [  68.85540332,  -49.28653121,  139.45406795],
       [  41.19189503,   20.23765654,   20.44578078],
       [-111.69829511,  197.90134137,   30.47265108],
       [ -10.52743603,   13.31334084,   71.50592177]])
```

You must use the numpy features mentioned in this question to receive credit for this question.

Question 2 (1 points)

Create an a numpy unidimensional (one dimension) array of sequential numbers between 10 and including 20 in increments (numpy step) of 2 using the numpy arange statement. Then use the numpy reshape statement to transform this one-dimensional array to a two-dimensional array of shape 2 elements by 3 elements. Then output the two-dimensional array to Python console using a print statement.

You must use the numpy features mentioned in this question to receive credit for this question.

Question 3 (1 points)

Create a numpy array of sequential numbers from 0 to and including 99. Ensure that the sequential numbers you have created are stored in a multidimensional numpy array with a shape of 20 elements by 5 elements.

You must use the numpy features mentioned in this question to receive credit for this question.

Question 4 (1 points)

Create a random sample of 100 numbers from a continuous *uniform* distribution using the numpy random_sample statement. Store your 100 generated numbers in a multidimensional numpy array named array6, which has a shape of 20 elements by 5 elements. Then using Python, make a copy of that array with a name array7. In your array7 go ahead multiply each of the generated random numbers by 100 using the batch mathematical operations covered in class. Your last step should be to print the first array named array6 and your second array named array7 with their corresponding Python console statements. The example Python console outputs would be as follows:

```
In [38]: print(array6)
[[ 0.23096153 -0.95014775 -0.3974851   0.74049128 -0.90788301]
 [ 0.63294189  0.28640453  1.02142514  0.46715438  1.47006986]
 [-0.22531676 -0.40152316  0.88293253  1.19693035  1.16204248]
 [-0.04351909  0.33360627 -0.18435086 -1.64361496 -1.52806907]
 [ 1.02351348  0.33291727  0.60167601  0.11314964 -0.71498584]
 [ 0.04005516  1.02502183  0.50410099 -0.57385475 -1.81634723]
 [ 2.88472304  1.02378579  1.11679259  1.2233063  -0.64792559]
 [-1.25420741  0.07216941 -1.18526081  0.72026167  0.26568224]
 [ 0.86633965  1.30241698 -0.43158384 -0.26015274  0.24987318]
 [ 0.71219518  1.55838786  0.2052889   0.00740975 -0.70513255]
 [ 0.29826842  0.5807734  -1.35365023 -0.12842839 -0.00383946]
 [-0.31648202 -1.22841927  0.64318094 -0.45697991 -1.15556314]
 [ 0.58339963 -1.83246107  0.27065774  0.66430213  0.70224192]
 [-0.1638261  -1.65077555 -0.99893297  0.63609     0.02123261]
 [-2.71617797  0.17348817 -0.04809974 -0.63589918 -0.17991206]
 [ 1.23142604 -0.46556525  0.21415121  0.39292797  0.27397305]
 [ 1.66573595  0.03688715 -1.994527   -0.84968444  0.55714424]
 [ 0.40095668 -1.34432746 -0.50420563  0.89464131  1.75696102]
 [-0.7594821   1.02792984 -1.1129573  -1.45853968 -0.73613162]
 [-0.73428829 -0.12885993 -2.46961496  1.186626    0.62245524]]
```

```
In [39]: print(array7)
[[  23.09615281  -95.01477461  -39.74851029   74.04912765  -90.7883011 ]
 [  63.29418876   28.6404535   102.14251381   46.71543794  147.00698641]
 [ -22.53167608  -40.15231634   88.29325313  119.69303486  116.20424791]
 [  -4.35190941   33.36062655  -18.43508614 -164.36149631 -152.80690651]
 [ 102.35134844   33.29172675   60.1676014    11.31496442  -71.49858378]
 [   4.00551598  102.50218336   50.41009892  -57.38547486 -181.63472345]
 [ 288.47230361  102.37857942  111.67925885  122.33063048  -64.79255925]
 [-125.42074115    7.21694091 -118.52608123   72.02616733   26.56822369]
 [  86.6339649   130.2416984   -43.15838419  -26.01527379   24.98731808]
 [  71.21951831  155.83878636   20.52889013    0.74097548  -70.5132555 ]
 [  29.82684218   58.0773402  -135.3650234   -12.84283858   -0.38394556]
 [ -31.64820246 -122.84192697   64.31809442  -45.69799068 -115.55631366]
 [  58.33996318 -183.24610744   27.06577397   66.4302128    70.22419227]
 [ -16.38260955 -165.077555    -99.89329675   63.60899995    2.12326061]
 [-271.61779672   17.34881692   -4.80997389  -63.58991806  -17.99120562]
 [ 123.14260351  -46.55652543   21.41512149   39.29279678   27.39730491]
 [ 166.57359544    3.68871474 -199.45269958  -84.96844449   55.71442433]
 [  40.09566825 -134.43274582  -50.42056287   89.46413103  175.69610211]
 [ -75.94820954  102.79298362 -111.29572991 -145.85396816  -73.61316197]
 [ -73.42882905  -12.88599265 -246.9614964   118.66260032   62.24552351]]
```

Obviously, since these are random numbers, for your case, your random numbers in both arrays would be different than what you see in the screenshots above.

Question 5 (2 points)

Create a Pandas Series data structure named obj7 with the following column names:

Column 0
Column 1
Column 2
Column 3
Column 4
Column 5

Now for each of the columns mentioned above, populate your Pandas Series data structure with values as follows

0
100
200
300
400
500

So, for instance, for Column 0 the value would be 0. For Column 1 the value would be 100. For Column 2 the value would be 200. For Column 3 the value would be 300. For Column 4 the value would be 400. For Column 5 the value would be 500.

Your last step should be to use a Python print statement to print your data structured named obj7 to the Python console. Your Python console output should look as follows:

```
In [42]: print(obj7)
Column 0      0
Column 1    100
Column 2    200
Column 3    300
Column 4    400
Column 5    500
dtype: int64
```

You must use the Pandas features mentioned in this question to receive credit for this question.

Question 6 (2 points)

Consider the following Pandas dataframe output to the Python console:

```
In [51]: print(frame10)
     state  population  year
0     Ohio         1.5  2000
1     Ohio         1.7  2001
2  Arizona         3.6  2000
3  Arizona         4.1  2002
```

Can you complete (highlighted in yellow) the Python dictionary named data10 in the Starter Code Template, so that your script would generate the following output to the Python console:

Starter Code Template:

```
data10 = {}
frame10 = pd.DataFrame(data10)
print(frame10)
```

Output:

```
In [51]: print(frame10)
     state  population  year
0     Ohio         1.5  2000
1     Ohio         1.7  2001
2  Arizona         3.6  2000
3  Arizona         4.1  2002
```

You must use the Pandas features mentioned in this question to receive credit for this question.