# tu07_re_BasicNumpy_HW

February 8, 2023

## 1 Numpy review homework

1. Make a numpy matrix from a Python list of lists…

[1]:
```
import numpy as np
```

[2]:
```
my_list = [[3.3, 2.3, 2.2], [1.2, 7.8, 8.7], [4.8, 2.2, 2], [1.5, 7.5, 9.5], [5.
↪9, 1.6, 7.7]]
my_list
```

[2]:
```
[[3.3, 2.3, 2.2],
 [1.2, 7.8, 8.7],
 [4.8, 2.2, 2],
 [1.5, 7.5, 9.5],
 [5.9, 1.6, 7.7]]
```

2. Make a 3D numpy matrix from a Python list of lists of lists!

[3]:
```
np_my_list = np.array(my_list)
np_my_list
```

[3]:
```
array([[3.3, 2.3, 2.2],
       [1.2, 7.8, 8.7],
       [4.8, 2.2, 2. ],
       [1.5, 7.5, 9.5],
       [5.9, 1.6, 7.7]])
```

3. Create a 5x3 array of Gaussian random numbers.

[4]:
```
my_array = np.random.randn(5,3)
my_array
```

[4]:
```
array([[-0.33820352,  1.59909091, -0.66335956],
       [ 0.25773159,  1.97895929, -1.00451743],
       [ 0.28112694,  0.42241326, -0.29371339],
       [ 0.65074025,  1.24530189,  2.34068734],
       [ 0.39261436,  0.98898607, -0.3008753 ]])
```

4. Write a script to go through the array created in 3. and announce (print) the value and its row and column indexes.

Hint: Use nested `for` loops - one to loop through the rows and one to loop through the columns.

[5]:
```
# get row 1
my_array[0:1]
```

[5]:
```
array([[-0.33820352,  1.59909091, -0.66335956]])
```

[6]:
```
# get row 2
my_array[1:2]
```

[6]:
```
array([[ 0.25773159,  1.97895929, -1.00451743]])
```

[7]:
```
# get row 3
my_array[2:3]
```

[7]:
```
array([[ 0.28112694,  0.42241326, -0.29371339]])
```

[8]:
```
# get row 4
my_array[3:4]
```

[8]:
```
array([[0.65074025, 1.24530189, 2.34068734]])
```

[9]:
```
# get row 5
my_array[4:5]
```

[9]:
```
array([[ 0.39261436,  0.98898607, -0.3008753 ]])
```

[10]:
```
for i in range(0,5):
    print('Row:',i+1)
    y = i+1
    for row in my_array[i:y]:
        for col in row:
            print('Column: ', col, sep ='')

# I think this code looks nice, however, I cannot find a way to make an idex␣
 ↪for the column.
```

```
Row: 1
Column: -0.33820351906755025
Column: 1.59909091331389
Column: -0.663359555780775
Row: 2
Column: 0.25773158517683564
Column: 1.9789592913430925
Column: -1.0045174320793877
```

```
Row: 3
Column: 0.28112693955010865
Column: 0.4224132553884828
Column: -0.2937133854713962
Row: 4
Column: 0.6507402510377125
Column: 1.2453018883263984
Column: 2.340687343928523
Row: 5
Column: 0.39261436224298796
Column: 0.9889860672445113
Column: -0.3008753014297306
```

[11]:
```
# To make an index of column, I changed some of the code to have 2 loops.
# To be honest, I still think that code from above is easier to understand.
# But, this one just works better T^T.

for i in range(0,5):
    print('Row:',i+1)
    y = i+1
    for j in range(len(my_array[i])):
        print('Column ', j+1,':',' ',
              my_array[i][j],
             sep = '')
```

```
Row: 1
Column 1: -0.33820351906755025
Column 2: 1.59909091331389
Column 3: -0.663359555780775
Row: 2
Column 1: 0.25773158517683564
Column 2: 1.9789592913430925
Column 3: -1.0045174320793877
Row: 3
Column 1: 0.28112693955010865
Column 2: 0.4224132553884828
Column 3: -0.2937133854713962
Row: 4
Column 1: 0.6507402510377125
Column 2: 1.2453018883263984
Column 3: 2.340687343928523
Row: 5
Column 1: 0.39261436224298796
Column 2: 0.9889860672445113
Column 3: -0.3008753014297306
```

5. Make an new array out of your random numbers such that the mean is 10 and the standard deviation is 3.

[12]:
```
q5_array = np.random.normal(loc = 10, scale = 3, size = (10,5))
q5_array
```

[12]:
```
array([[ 7.57207348, 12.41728612,  9.67953067, 10.60122053, 11.30240103],
       [ 8.03446949,  8.61424446,  6.89370789,  9.53868766,  7.82472472],
       [ 8.38882514,  5.47127827,  9.13903854, 11.18963763, 15.02495855],
       [ 5.83984102, 10.19706299,  5.582831  ,  9.03014818, 11.63640941],
       [ 8.23055952,  9.50983567, 14.49560709, 10.30149231, 11.71899217],
       [ 9.38564684,  9.3373273 , 16.43647903,  6.4582413 ,  8.21013349],
       [ 5.01314831, 13.07413609,  9.64191935,  7.22139146,  8.85900617],
       [10.65204958,  3.99149691,  6.65111615, 10.37318318,  8.6795788 ],
       [13.35478454, 10.67370195, 11.40610163,  8.80257491, 12.45352332],
       [12.42403097, 11.58679553,  8.84050164,  4.3927071 ,  8.45543076]])
```

6. Count the number of values in your new array that are below 7.

[13]:
```
q5_array < 7
```

[13]:
```
array([[False, False, False, False, False],
       [False, False,  True, False, False],
       [False,  True, False, False, False],
       [ True, False,  True, False, False],
       [False, False, False, False, False],
       [False, False, False,  True, False],
       [ True, False, False, False, False],
       [False,  True,  True, False, False],
       [False, False, False, False, False],
       [False, False, False,  True, False]])
```

[14]:
```
# Count the total number that less than 7 on each column.
sum(q5_array < 7)
```

[14]:
```
array([2, 2, 3, 2, 0])
```

[15]:
```
# Total number of all value that less than 7.
sum(sum(q5_array < 7))
```

[15]:
```
9
```

7. Make a numpy sequence that has the even numbers from 2 up to (and including) 20.

[16]:
```
q7_np_r1 = np.arange(2,22,2)
q7_np_r1
```

[16]:
```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

8. Get the second and third rows of your array.

[17]:
```
# Row 2 from the first 5 numbers from question 7
q7_np_r2 = q7_np_r1[0:5]
q7_np_r2
```

[17]:
```
array([ 2,  4,  6,  8, 10])
```

[18]:
```
# Row 3 from the first 5 numbers from question 7
q7_np_r3 = q7_np_r1[5:10]
q7_np_r3
```

[18]:
```
array([12, 14, 16, 18, 20])
```

[19]:
```
# Add all numbers (add to question 5's array)
new_array = np.vstack((q5_array, q7_np_r2, q7_np_r3))
new_array
```

[19]:
```
array([[ 7.57207348, 12.41728612,  9.67953067, 10.60122053, 11.30240103],
       [ 8.03446949,  8.61424446,  6.89370789,  9.53868766,  7.82472472],
       [ 8.38882514,  5.47127827,  9.13903854, 11.18963763, 15.02495855],
       [ 5.83984102, 10.19706299,  5.582831  ,  9.03014818, 11.63640941],
       [ 8.23055952,  9.50983567, 14.49560709, 10.30149231, 11.71899217],
       [ 9.38564684,  9.3373273 , 16.43647903,  6.4582413 ,  8.21013349],
       [ 5.01314831, 13.07413609,  9.64191935,  7.22139146,  8.85900617],
       [10.65204958,  3.99149691,  6.65111615, 10.37318318,  8.6795788 ],
       [13.35478454, 10.67370195, 11.40610163,  8.80257491, 12.45352332],
       [12.42403097, 11.58679553,  8.84050164,  4.3927071 ,  8.45543076],
       [ 2.        ,  4.        ,  6.        ,  8.        , 10.        ],
       [12.        , 14.        , 16.        , 18.        , 20.        ]])
```

9. Compute the mean of the columns of your array.

[20]:
```
# Mean of the each row
new_array.mean(axis=1)
```

[20]:
```
array([10.31450237,  8.18116685,  9.84274762,  8.45725852, 10.85129735,
        9.96556559,  8.76192027,  8.06948492, 11.33813727,  9.1398932 ,
        6.        , 16.        ])
```

[21]:
```
# Mean of the each column
new_array.mean(axis=0)
```

[21]:
```
array([ 8.57461907,  9.40609711, 10.06390275,  9.49244036, 11.18042987])
```