Programming in Python for Business Analytics (BMAN73701)

Lab Session: Lab 6 - Numpy

Introduction

This week we will be working with numpy, an extremely popular package for scientific computing. The core data structure here are arrays. If you've ever done any maths involving matrix, then you should have a grasp on what an array is.

There are some things to keep in mind when working with arrays and numpy. The first is that, you should rarely (if ever) need to use a for loop. If you find yourself writing a loop, take a step back and think about it. When we worked with lists, we iterated/looped over this list and did something with each individual element. With numpy, you should be thinking more that we work with the whole thing at the same time. This is much, much faster.

As always, we put our import statements are the top of any script. The convention for numpy is to write:

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

So when using numpy functions like sum, we access it through np.sum().

Exercise 1: Slicing/Indexing

Fortunately, this is quite similar to what we did with lists. You should be comfortable with this by now. Try to write the code that gives the output shown.

Note: Python uses zero-based indexing, which means that the first element is at index 0, the second element at index 1, and so on.

In [2]:
```
import numpy as np

a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
# Print the whole array
print(a)
# Print all rows, skip 1st column
print(a[:, 1:])
# First two rows, 1st and 4th columns
print(a[:2, [0,3]])
# Every 2nd row, reversed order
print(a[::2, ::-1])
# Print a boolean matrix with True if the value in a at that position is sma
print(a < 7)
# Print all values from matrix 'a' that are larger than 4.
print(a[a > 4])
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[ 2  3  4]
 [ 6  7  8]
 [10 11 12]]
[[1 4]
 [5 8]]
[[ 4  3  2  1]
 [12 11 10  9]]
[[ True  True  True  True]
 [ True  True False False]
 [False False False False]]
[ 5  6  7  8  9 10 11 12]
```

Exercise 2: Working with arrays

Whenever you think about doing something with an array, there's a good chance that there is a numpy function for that. Always Google or check the docs

In [3]:
```
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print(np.mean(a)) # Print mean (average) of whole matrix
print(np.mean(a, axis = 0)) # Print mean of each column
print(np.mean(a, axis = 1)) # Print mean of each row
print(np.std(a)) # Print standard deviation of the whole matrix
```

```
6.5
[5. 6. 7. 8.]
[ 2.5  6.5 10.5]
3.452052529534663
```

Exercise 3: Broadcasting

In [4]:
```
a = np.arange(0, 51, 10)
print(a)
b = np.arange(0, 10, 1.5)
print(b)

print(a.reshape((a.shape[0], 1))) # Print 'a' as a column vector
# print(a[:,np.newaxis])

print(a.reshape((2,3))) # Print 'a' as a matrix (2 x 3)
print(a.reshape((a.shape[0],1)) + b) # Calculate a matrix X, where X_ij = a_
```

```
[ 0 10 20 30 40 50]
[0.  1.5 3.  4.5 6.  7.5 9. ]
[[ 0]
 [10]
 [20]
 [30]
 [40]
 [50]]
[[ 0 10 20]
 [30 40 50]]
[[ 0.   1.5  3.   4.5  6.   7.5  9. ]
 [10.  11.5 13.  14.5 16.  17.5 19. ]
 [20.  21.5 23.  24.5 26.  27.5 29. ]
 [30.  31.5 33.  34.5 36.  37.5 39. ]
 [40.  41.5 43.  44.5 46.  47.5 49. ]
 [50.  51.5 53.  54.5 56.  57.5 59. ]]
```

Exercise 4: Sorting

In [5]:
```
np.random.seed(42)
# Create and print a uniformly random matrix with 3 rows and 4 columns
a = np.random.rand(3,4)
print(a)

# axis = 1: perform operation horizontally
# axis = 0: perform operation vertically

# Sort each row
print(np.sort(a, axis = 1))

# Sort each column in descending order
print(-np.sort(-a, axis = 0))

# Print the order of the sorted elements of each row (indirect sorting)
print(np.argsort(a, axis = 1))
```

```
[[0.37454012 0.95071431 0.73199394 0.59865848]
 [0.15601864 0.15599452 0.05808361 0.86617615]
 [0.60111501 0.70807258 0.02058449 0.96990985]]
[[0.37454012 0.59865848 0.73199394 0.95071431]
 [0.05808361 0.15599452 0.15601864 0.86617615]
 [0.02058449 0.60111501 0.70807258 0.96990985]]
[[0.60111501 0.95071431 0.73199394 0.96990985]
 [0.37454012 0.70807258 0.05808361 0.86617615]
 [0.15601864 0.15599452 0.02058449 0.59865848]]
[[0 3 2 1]
 [2 1 0 3]
 [2 0 1 3]]
```

Exercise 5: Data Exploration

Download the file "sales_dept_store.csv" from Blackboard. In this file, values are the volume of sales, where each row corresponds to a store and each column to a different department.

Write code to print the following information:
A.Number of stores and number of departments
B.The number of stores that have at least one department with negative sales
C.The five departments with the highest mean sales (and their mean sales)

In [6]:
```
import numpy as np

# Read in the data using Numpy. This creates a NumPy matrix
sales = np.genfromtxt("sales_dept_store.csv", delimiter = ',')

# import pandas as pd
# sales_df = pd.read_csv("sales_dept_store.csv")
# sales = sales_df.to_numpy()

sales
```

Out[6]:
```
array([[3.21940518e+06, 6.59259893e+06, 1.88051836e+06, ...,
        5.03465088e+06, 1.69137122e+06, 1.13253700e+04],
       [4.40125125e+06, 9.42554792e+06, 2.49914856e+06, ...,
        5.81970018e+06, 2.00706232e+06, 1.99876700e+04],
       [1.04799281e+06, 2.40837392e+06, 7.87830010e+05, ...,
        4.80812300e+04, 1.10055000e+03, 0.00000000e+00],
       ...,
       [1.07952259e+06, 2.96336776e+06, 1.42949790e+05, ...,
        2.76704856e+06, 1.37206605e+06, 1.05000000e+02],
       [1.15114890e+06, 1.34095004e+06, 8.16553900e+04, ...,
        9.49014840e+05, 4.95649020e+05, 7.01000000e+00],
       [2.53766599e+06, 5.11953048e+06, 1.35964614e+06, ...,
        9.24775550e+05, 7.57672700e+04, 0.00000000e+00]])
```

A. Number of stores and number of departments

In [7]:
```
print("There are", sales.shape[0], "stores and", sales.shape[1], "department
```

```
There are 45 stores and 81 departments
```

B. The number of stores that have at least one department with negative sales

In [8]:
```
# Solution 1
print("There are", np.sum(np.min(sales, axis=1) < 0), "stores reporting nega
# Solution 2
print("There are", np.sum(np.any(sales < 0, axis=1)), "stores reporting nega
```

```
There are 26 stores reporting negative sales
There are 26 stores reporting negative sales
```

C. The five departments with the highest mean sales (and their mean sales)

In [9]:
```
best_num = 5
mean_sales_by_dept = np.mean(sales, axis=0)

# Solution: sort decreasing, select the first best_num
best_dept = np.argsort(-mean_sales_by_dept)[:best_num]
print("The", best_num, "departments with the highest mean sales (and their m
print(best_dept)
print(mean_sales_by_dept[best_dept])
```

```
The 5 departments with the highest mean sales (and their mean sales):

[73 76 36 60 71]
[10754296.486       9984892.50044444  8735958.59822222  6793892.27133333
  6468188.08177778]
```

Further Resources

Numpy documentation

Scipy Lectures: Numpy chapter

Python for Data Science Handbook: Introduction to Numpy

In [ ]: