ROI TRAINING
MAXIMIZE YOUR TRAINING INVESTMENT

COURSE 811M

PYTHON FOR DATA SCIENTISTS

EXERCISE MANUAL

The following course materials are copyright protected materials. They may not be reproduced or distributed and may only be used by students attending the *Python for Data Scientists* course.

Table of Contents

This page intentionally left blank.

Exercise 2.1: Array Creation

The aim of this exercise is to gain some experience of working with NumPy arrays.

1. Start IPython.
2. Define an `ndarray` containing the integer numbers 0 to 9.
3. Print the type of the array to the console.
4. Print the following properties of the array (they are accessible in the same way as `ntype`):
    a. `ndim`
    b. `shape`
    c. `size`
    d. `itemsize`
5. Define a 3 x 3 NumPy array containing all 1's and display it to the console.
6. Print the four properties of Step 4 on the array defined in Step 5.

This page intentionally left blank.

Exercise 2.2: Array Basic Operations

The aim of this exercise is to gain some experience of working with basic operations on NumPy arrays.

1. Define a 3 x 3 array with the integers 1 through 9 named `array1`.
2. Define a second 3 x 3 array with the number 2 in each cell named `array2`.
3. Now, perform the following operations using the two arrays:
    a. `array1+array2`
    b. `array1-array2`
    c. `array1/array2`
    d. `array1*5`
4. Print elements 4 to 6 of array 1 using a slice operation.
5. Create a new single-dimensioned array named `array3` with the numbers 0 through 19 in it.
6. Take a slice of elements 5 to 15 of array 3 and assign the slice to a variable named `aslice` and print the variable.
7. Modify the contents of the first and last elements of the slice by writing the value 99 into these elements.
8. Print the contents of the slice `aslice` and the array `array3`. Are the contents what you expect of both arrays?

    _____

This page intentionally left blank.

Exercise 3.1: Pandas `Series`

The aim of this exercise is to gain some experience of working with the NumPy `Series` data structure.

1. Define a `Series` object holding the values 1 to 10.
2. Display the data values of the `Series` object defined in Step 1.
3. Display the index values of the `Series` object defined in Step 1.
4. Define a new `Series` object holding the values 1 to 10, with the corresponding index values set 'a' through to 'j'.
5. Display the data values and index of the `Series` object of Step 4.
6. Access the third and fifth elements of the `Series` object using their index.
7. Define the following dictionary: `{'Dublin': 200000, 'Athlone': 15000, 'Galway': 700000}`.
8. Define the following array: `['Dublin', 'Athlone', 'Waterford']`.
9. Now, construct a `Series` object using the dictionary in Step 7 and the index in Step 8.
10. Display the `Series` object defined in Step 9.
11. Use the `Series` `notnull()` and `isnull()` methods to display which elements are not null and null, respectively, for the `Series` object defined in Step 9.

This page intentionally left blank.

Exercise 3.2: Pandas `DataFrame`

The aim of this exercise is to gain some experience of working with NumPy `DataFrame` data structure.

1. Use the following dictionary to create a `DataFrame`:

   a.
```
{'team':['Leicester', 'Manchester City', 'Arsenal'],
player':['Vardy', 'Aguero', 'Sanchez'],'goals':[24,22,19]}
```

2. Display the above `DataFrame` to the console.
3. What values are assigned for the index and columns?

   _____

4. Use the dictionary from Step 1 and create a second `DataFrame` with index values `'one'`, `'two'`, `'three'`, respectively, and columns `team`, `player`, `goals`, `played`. Display the `DataFrame` to the console.

This page intentionally left blank.

Exercise 3.3: Working with `DataFrame` and `Series`

The aim of this exercise is to gain some more experience of working with NumPy `Series` and `DataFrame` data structures.

1. Define and display a `Series` object with the following data: `[1.1,2.2,3.3,4.4]` and `index=['d','b','c','a']`.
2. Reindex the data with the index `index=['a','b','c','d','e','f']` and display the `Series` object. Are there any missing values? _____
   If so, reindex again and zero-fill the missing values.
3. Display only those values in the series that have a data value >2.
4. Define a 3 x 3 `DataFrame` with columns A, B, C and index a, b, c. You choose the integer values of the data for each cell. Display the `DataFrame`.
5. Reindex the `DataFrame` of Step 3 with the index a, b, c, d, e. Display the `DataFrame`. Is it as you expect? What order are the rows?

   _____
6. For your `DataFrame`, display the data but only include those rows whose data in Column B is >2.
7. Define a 4 x 4 `DataFrame` of integer numbers. Define a 3 x 3 `DataFrame` of integer numbers. Add the two `DataFrame`'s together. What is the result?

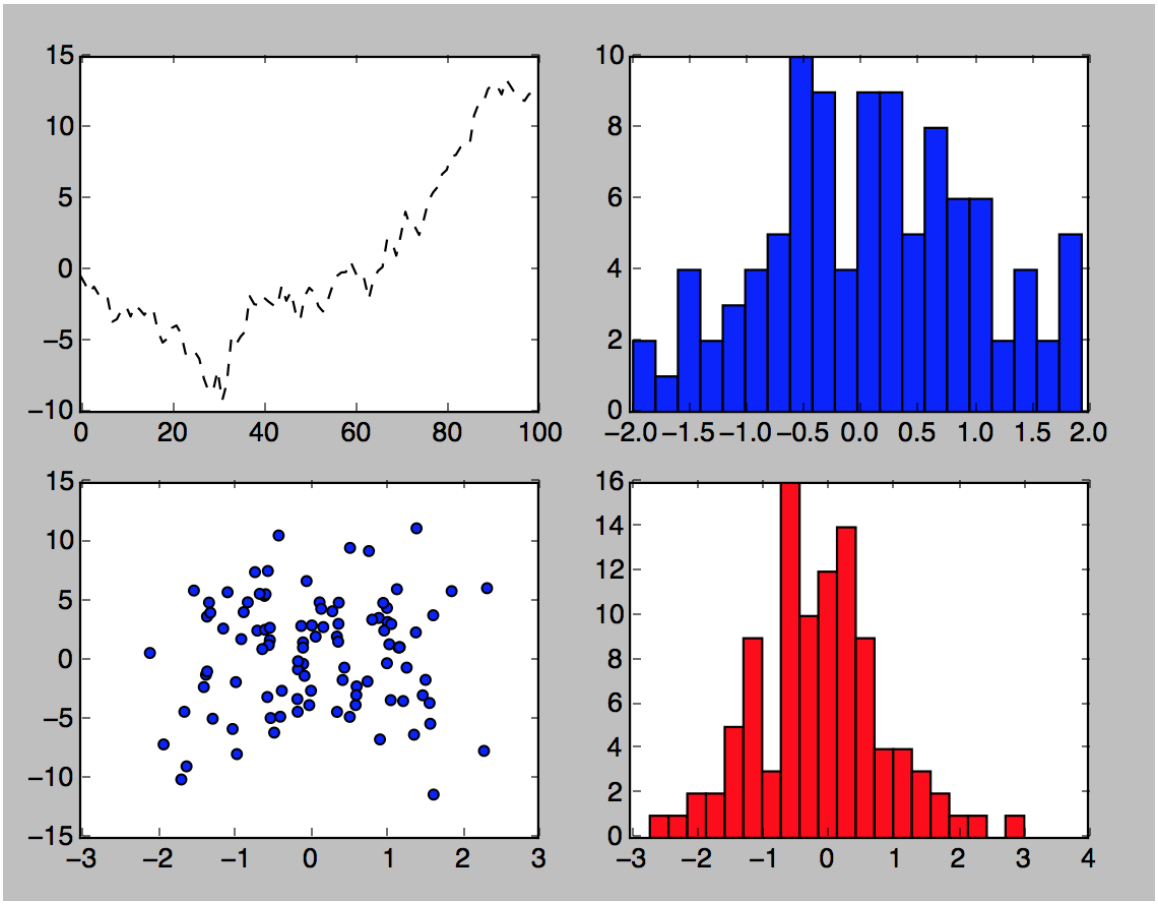
   _____
8. Repeat Step 7, but in a way that any missing values in the result are zero-filled.
9. Define the following `DataFrame`: `DataFrame(np.random.randn(4,4))`.
10. Now, define a Python Lambda function of the following form: `f = lambda x: x.max() - x.min()`.
11. Apply the function in Step 7 to each row in the `DataFrame` defined in Step 9.
12. Repeat Step 11, but for each column.
13. Sort the `DataFrame` in Step 7 firstly by column index, then repeat by row index.
14. On the `DataFrame` defined in Step 7, apply the following functions:
    a. `describe()`
    b. `sum() – do this for each axis`

This page intentionally left blank.

Exercise 5.1: Working with Matplotlib

The aim of this exercise is to gain some experience of plotting data using matplotlib.

1. Your task is to plot a chart similar to the following:



2. The data for the above can be generated as follows:
    a. For the plot: `randn(x).cumsum()`
    b. For the histogram: `randn(x), bins=y`
    c. For the scatter plot: `randn(x), randn(x)-y*randn(100)`
3. Plot the result of the cumulative sum of 2000 random numbers. The chart should have a title, x axis ticks set at 1, 500, 1000, 1500, and 2000 intervals, and x and y labels.
4. Define the following `DataFrame`:
    a. `DataFrame({'A': np.random.randn(1000) + 1,'B': np.random.randn(1000), 'C': np.random.randn(1000)-1})`
5. Plot the `DataFrame` from Step 4 as a histogram.

6. For the `DataFrame` of Step 4, plot a scatter plot for the following:
    a. Column A vs. Column B
    b. Column B vs. Column C
7. Add labels and a title to the plots in Step 6.