Course 592

Data Science with Python

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 *Data Science with Python* course.** |

**Table of Contents**

[Exercise 2.1: Array Creation 1](#_Toc479324481)

[Exercise 2.2: Array Basic Operations 3](#_Toc479324482)

[Exercise 2.3: Shape Changing and Stacking 5](#_Toc479324483)

[Exercise 2.4: Working with Stats 7](#_Toc479324484)

[Exercise 3.1: Pandas Series 9](#_Toc479324485)

[Exercise 3.2: Pandas DataFrame 11](#_Toc479324486)

[Exercise 3.3: Working with DataFrame and Series 13](#_Toc479324487)

[Exercise 4.1: Combining and Merging Data Sets 15](#_Toc479324488)

[Exercise 4.2: Transforming Data 17](#_Toc479324489)

[Exercise 4.3: String Manipulation 19](#_Toc479324490)

[Exercise 5.1: Working with Matplotlib 21](#_Toc479324491)

[Exercise 6.1: Transforming Data 23](#_Toc479324492)

[Exercise 6.2: Working with Pivot Tables 25](#_Toc479324493)

[Exercise 7.1: Working with SciPy Scientific Functions 27](#_Toc479324494)

[Exercise 7.2: Working with SciPy Image Processing 29](#_Toc479324495)

[Exercise 8.1: Supervised Learning 31](#_Toc479324496)

[Exercise 8.2: Unsupervised Learning 33](#_Toc479324497)

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):
   1. ndim
   2. shape
   3. size
   4. 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:
   1. array1+array2
   2. array1-array2
   3. array1/array2
   4. 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:
   1. {'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?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. 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?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. For your DataFrame, display the data but only include those rows whose data in Column B is >2.
2. 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?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

This page intentionally left blank.

# Exercise 4.1: Combining and Merging Data Sets

The aim of this exercise is to gain some more experience of joining and merging Series and DataFrame data structures.

1. Define two data structures as follows:
   1. DataFrame({'key': ['b','b','a','c','a','a','b'], 'data1': range(7)})
   2. DataFrame({'key': ['a','b','d'], 'data2': range(3)})
2. Now, merge the two DataFrames and display the results. Are they as you expected? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. On the two DataFrames of Step 1, perform the following joins:
   1. Left
   2. Right
   3. Outer
4. Define two data structures as follows:
   1. DataFrame({'key': [0,1,0,0,1,2], 'value': range(6)})
   2. DataFrame({'value': [2,4]})
5. Merge the above DataFrames on the following:
   1. Left key with right index
   2. Left index with right key
   3. Left and right indexes
6. Define the following three data structures:
   1. Series([0,1], index=['a','b'])
   2. Series([3,4,5], index=['c','d','e'])
   3. Series([6,7,8], index=['f','g','h'])
7. Now, concatenate the three Series objects defined in Step 6 and display the result.
8. In the result of Step 7, it is not known which Series the pieces in the result originated from. Re-run the concatenate, but in a way that shows the origin of the pieces of data in the result.
9. Use the following: np.arange(6) as the basis to define a 2 x 3 DataFrame. Perform the following operations on the DataFrame and confirm the results are as you expect:
   1. Stack
   2. Unstack

# Exercise 4.2: Transforming Data

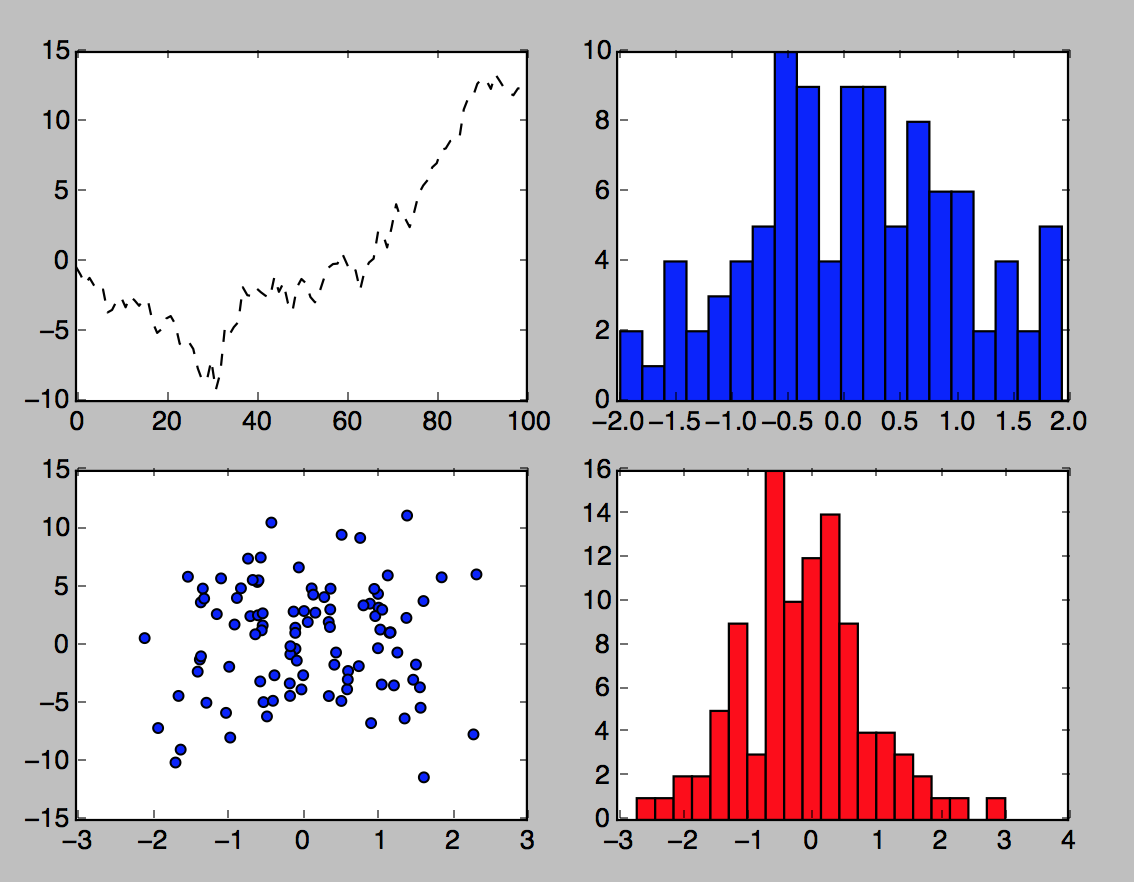
The aim of this exercise is to gain some experience of transforming DataFrame data structures.

1. Define a data structure of the following form:
   1. DataFrame({'c1' : ['a']\*2 + ['b']\*3, 'c2' : [10,10,11,12,12]})
2. For the DataFrame in Step 1, perform the following:
   1. Display it to the console.
   2. Print the indexes and whether the values at each index are duplicates or not.
   3. Print the DataFrame with duplicates removed.
3. Repeat Step 2b above, but the duplicates should only consider column c1.
4. Define a data structure of the following form:
   1. DataFrame({'player': ['sanchez','vardy','costa','lukaku'], 'goals' : [19,24,18,22]})
5. Now, transform the data in Step 5 by adding an extra column named team which represents the team each player belongs to. The data values are:
   1. sanchez Arsenal
   2. vardy Leicester
   3. costa Chelsea
   4. lukaku Everton
6. Define the following data structure:
   1. customer\_ages = [21,65,43,44,71,64,27,20,28,24,55,33,29]
7. The data above represents the ages of customers who have purchased goods from our e-commerce site. Bin these data into the following categories and display the number of people in each category:
   1. Young Adult 25-40
   2. Senior 60-100
   3. Middle Age 40-60
   4. Student 18-25

# 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:



1. The data for the above can be generated as follows:
   1. For the plot: randn(x).cumsum()
   2. For the histogram: randn(x), bins=y
   3. For the scatter plot: randn(x), randn(x)-y\*randn(100)
2. 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.
3. Define the following DataFrame:
   1. DataFrame({'A': np.random.randn(1000) + 1,'B': np.random.randn(1000), 'C': np.random.randn(1000)-1})
4. Plot the DataFrame from Step 4 as a histogram.
5. For the DataFrame of Step 4, plot a scatter plot for the following:
   1. Column A vs. Column B
   2. Column B vs. Column C
6. Add labels and a title to the plots in Step 6.

# Exercise 6.1: Transforming Data

The aim of this exercise is to gain experience of summarizing, transforming, and grouping data.

1. For this exercise, there is a data file provided. It is named phone\_data.csv and can be found in the folder python\_data\_science\ch06. The data contains data from a mobile phone log and has the following columns: index, date, duration, item, month, network, network\_type.
2. Load the file data into a DataFrame.
3. For the data, list the following:
   1. The number of rows in the data set.
   2. The longest phone call entry.
   3. How many seconds of communication there has been from the logs:
      1. In total.
      2. For phone calls.
      3. For data transfer.
   4. How many entries there are for each month.
   5. The number of unique network entries.
4. You will now use groupby to analyze the data. Data.groupby('month') will split the DataFrame by month returning a GroupBy object. The GroupBy object groups variable is a dictionary whose keys are the unique groups. The values are the axis labels. For the DataFrame, run the following commands:
   1. data.groupby(['month']).groups.keys()
   2. len(data.groupby(['month']).groups['2014-11'])
5. Functions such as min, max, mean, first, last can be applied to the groupby object. Perform the following operations:
   1. Get the first entry for each month.
   2. Get the sum of the durations per month.
   3. Get the number of dates/entries in each month.
6. Calculate the sum of the durations of calls for each network.

## Bonus

1. Here, you will use grouping on more than one variable. Determine how many calls, sms, and data entries there are for each month.
2. How many calls, sms, and data are there per month split by network type?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Exercise 6.2: Working with Pivot Tables

The aim of this exercise is to gain experience of working with pandas pivot\_table. The data represents a sales pipeline.

1. Read the file python\_data\_science\ch06\sales\_leads.csv into a DataFrame and display the first few entries
2. Define a pivot table with the name column of the data as the index.
3. Repeat Step 2, but have the name, rep, and manager as indexes.
4. Repeat Step 3, but use only the manager and rep as the index.
5. Repeat Step 4, but show only the price column.
6. In Step 5, you should see the price has been averaged. Change this so that the price is the sum for each rep (use aggfunc = np,sum).
7. Repeat Step 6, but display the mean and the number of items instead of the sum.
8. Define a pivot table for looking at the price (the sum), but per product for a manager/rep combination. (use index = manager, rep, values=price columns=product)
9. For the pivot table in Step 8, if there are any NaN values, replace these with 0.
10. Repeat Step 8, but add quantity to the values.
11. Repeat Step 10, but move product from columns and add to the index.

This page intentionally left blank.

# Exercise 7.1: Working with SciPy Scientific Functions

The aim of this exercise is to gain experience of working with SciPy scientific functions.

1. Consider the function x2 +10sin(x). Plot this function using matplotlib for the range -10 to 10 in .1 increments.
2. Using BFGS optimization, find the minimum value of the function with starting point 0, -5, and 5.
3. Repeat Step 2, but using basinhopping optimization.
4. Consider the following data set and function:
   1. xdata = np.linspace(-10,10,num=20)
   2. ydata = f(xdata)+np.random.randn(xdata.size)
   3. def f2(x,a,b):

return a\*x\*\*2 + b\*np.cos(x)

1. Perform a curve-fitting algorithm to find the best values for a and b above.
2. Plot the actual function f2 and the curve fitted result.
3. Consider the following functions:
   1. measured\_time = np.linspace(0,1,10)
   2. noise = (np.random.random(10)\*2 -1)\*1e-1
   3. measures = np.sin(2\*np.pi\*measured\_time)+noise
4. For the measures and measured\_time in Step 7, perform the following interpolation:
   1. linear
   2. cubic
5. Plot the results of Steps 8a and 8b on a chart.

This page intentionally left blank.

# Exercise 7.2: Working with SciPy Image Processing

The aim of this exercise is to gain experience of working with SciPy image processing.

1. Load the face image from SciPy with the following code:

**from** **scipy** **import** misc

face = misc.face(gray=**True**)

face = face[:512, -512:] *# crop out square on right*

**import** **numpy** **as** **np**

noisy\_face = np.copy(face).astype(np.float)

noisy\_face += face.std() \* 0.5 \* np.random.standard\_normal(face.shape)

1. For the image above, apply the following filters:
   1. Gaussian
   2. Median
   3. Weiner
2. Plot the blurred image with the three filtered versions of Step 2.
3. Plot the histograms of the filtered images above.

This page intentionally left blank.

# Exercise 8.1: Supervised Learning

The aim of this exercise is to gain experience of working with supervised learning with scikit-learn.

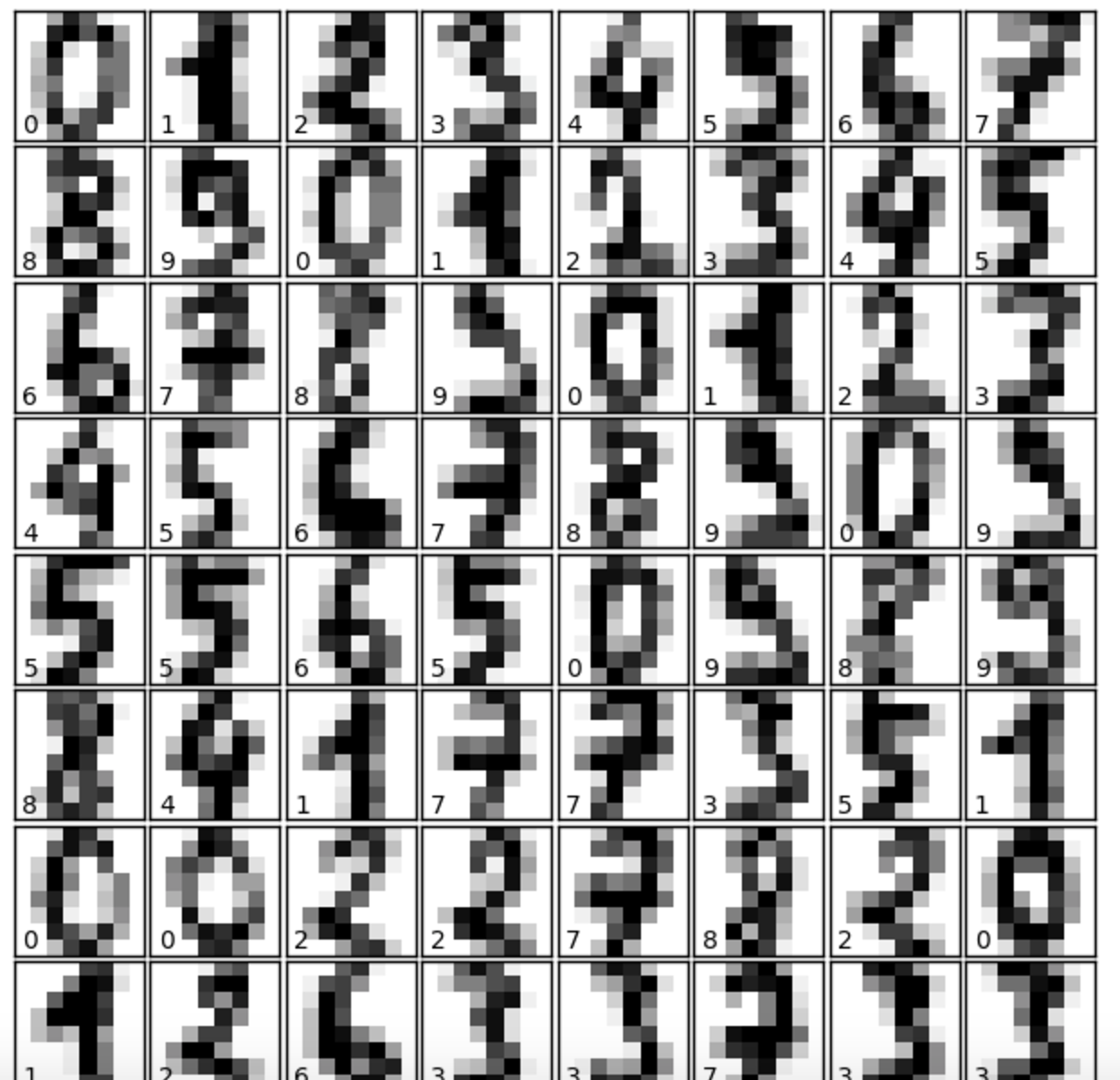
1. Open the file house\_cost.py in the folder python\_data\_science\ch08 and also the data house\_data.csv
2. Run the file and examine the results.
3. Repeat Steps 1 and 2 for the file house\_cost\_quadratic\_measures.py.

This page intentionally left blank.

# Exercise 8.2: Unsupervised Learning

The aim of this exercise is to gain experience of working with unsupervised learning with scikit-learn.

1. In this exercise, you will use Support Vector Classification (SVC) to classify the following images:



1. Load the data using datasets.load\_digits().
2. Plot the first 4 digits to verify you have the data loaded OK.
3. Using an SVC classifier, train the algorithm using the first half of the images.
4. Use the trained classifier to predict the second half of the images.
5. Plot the last 8 predictions—are they correct? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_