Q1 (I)

**Using Comments in Python**

*Comments* are pieces of code that are not executed by the compiler. Why are we

starting our programming chapter with commands that would not be run? Because

comments are a vital part of any programming project.

Comments are lines of code that the programmer puts there for others to

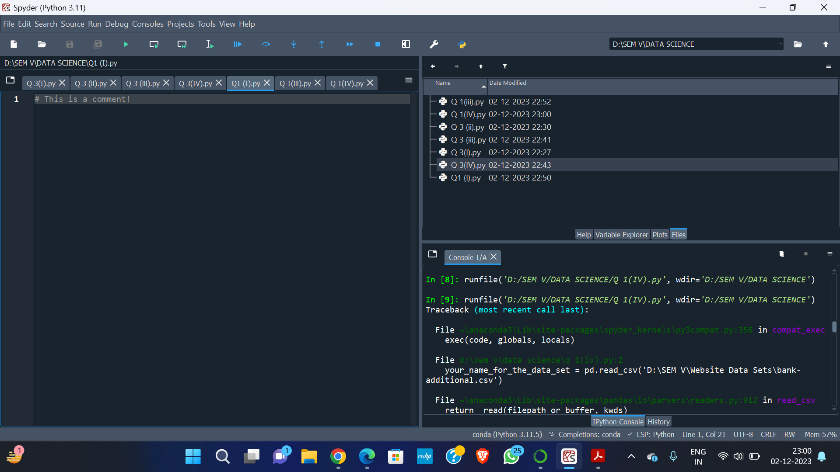
understand the code better. For example, if you want to explain what a particular

piece of code does, you may begin with a comment that explains what it does and

what the result will be.

How do we write comments in Python? Comments are lines of code that start

with a pound sign, **#**. The following is an example of a comment.



# This is a comment!

Notice that the typeface for any code, even comments, will be given in boldface

in this textbook. This applies to both Python and R code.

Q1(II) Any code you type needs to be run, or *executed*, before it will work. There are a few

different ways to execute code.

Most often, your cursor will be on one line of code, and you will want to run

that line. There is a button in Spyder to run a single line of code. There is also a

keyboard shortcut, which is revealed if you hover over that button. Both the button

and the hover text are shown in Figure 2.2.

You may want to run multiple lines of code at the same time. In that case,

highlight the relevant lines and press the “Run selection or current line” button, or

press the keyboard shortcut. All highlighted code will be run at the same time.

You can also try executing comments. As discussed previously, comments

will not be compiled, and there will be no output. They will, however, appear in the



bottom‐right window of Spyder. This tells you that the Spyder compiler has looked

at the comment, even though there is nothing for Spyder to do after reading the

comment.

Q1 (III)

While many things can be done in Python “out of the box,” meaning directly after

you download and install it, most of what we want to accomplish requires *importing*

*packages*. Packages contain specially designed code that will enable us to

perform complex data science tasks without writing the code ourselves. For

example, in Chapter 6, we need to build a classification and regression tree

(CART) model. Instead of figuring out how to build a CART model from scratch,

we will import a package that contains that code. Once the package is imported,

we can run the code to create a CART model.

Some commands are specialized, such as the **MultinomialNB()** command in

the *sklearn.naive\_bayes* package (see Chapter 10). On the other hand, there are

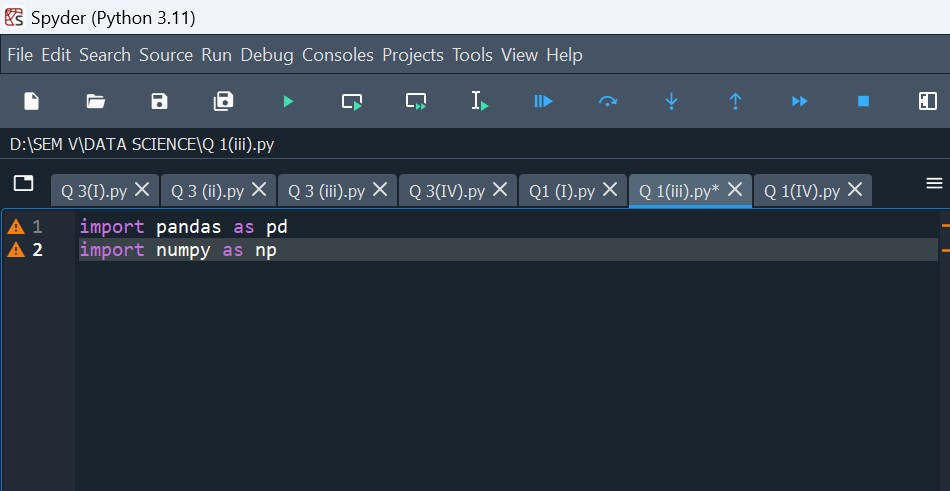
some commands that will be used in every chapter. These are the *pandas* and *numpy*

packages. To import these packages, you need to type and run the following two

lines of code:

import pandas as pd

import numpy as np



Q1 (iv) Now we will discuss how to get a data set into Python. In this text, we will use the

**read\_csv** command, using the following structure:

*your\_name\_for\_the\_data\_set* = pd.read\_csv("*the\_path\_to\_*

*the\_file*")

The command **read\_csv** comes from the *pandas* package. Following the code

in Section 2.2.3, we imported the *pandas* package as *pd*. After importing the *pandas*

package, we can access the **read\_csv** command by typing **pd**. To use the **read\_**

**csv()** command, type **pd** followed by a period, then the command **read\_csv**.

The next part of the code is the *path to the data file*, contained in double

quotes. For many Windows users, the path will start with **C:/**and end with the filename.

For example, in Chapter 4 you will need to import the **bank\_marketing\_**

**training** data set. The code to import the data set is given below.

bank\_train = pd.read\_csv("C:/.../bank\_marketing\_

training")

The programmer will substitute their own file path for the “**C:/…/**” portion of

the path given above. An example of importing a data file is given below.

bank\_train = pd.read\_csv("C:/Users/Data Science/Data/

bank\_marketing\_training")

