1 Introduction to Computers and Python

Objectives

In this chapter you'll:

- Learn about exciting recent developments in computing.
- Learn computer hardware, software and Internet basics.
- Understand the data hierarchy from bits to databases.
- Understand the different types of programming languages.
- Understand object-oriented programming basics.
- Understand the strengths of Python and other leading programming languages.
- Understand the importance of libraries.
- Be introduced to key Python and data-science libraries you'll use in this book.
- Test-drive the IPython interpreter's interactive mode for executing Python code.
- Execute a Python script that animates a bar chart.
- Create and test-drive a web-browser-based Jupyter Notebook for executing Python code.
- Learn how big "big data" is and how quickly it's getting even bigger.
- Read a big-data case study on a mobile navigation app.
- Be introduced to artificial intelligence—at the intersection of computer science and data science.

Note: Throughout the Instructor Solutions Manual, solutions are ***not provided*** for project, research and challenge exercises—many of which are substantial and appropriate for term projects, directed-study projects, capstone-course projects and thesis topics. **Before assigning a particular exercise for homework, instructors should check the IRC to be sure the solution is available.** These Instructor Solutions Manual PDFs contain **only answers to short-answer exercises and any discussion questions asked in other exercises**. Code corresponding to programming exercises can be found in the **`solutions`** folder's chapter-specific subfolder—e.g., **`ch01`** for Chapter1, **`ch02`** for Chapter 2, etc. Code generally is provided both in **Python source-code files** (**`.py`**) and **Jupyter Notebooks** (**`.ipynb`**).

Exercises

1.1 *(IPython Session)* Using the techniques you learned in Section 1.10.1, execute the following expressions. Which, if any, produce a runtime error?

a) `10 / 3`
b) `10 // 3`
c) `10 / 0`
d) `10 // 0`
e) `0 / 10`
f) `0 // 10`

Answer: (c) and (d) produce `ZeroDivisionError`s because division-by-zero is not allowed in Python.

1.2 *(IPython Session)* Using the techniques you learned in Section 1.10.1, execute the following expressions. Which, if any, produce a runtime error?

a) `10 / 3 + 7`
b) `10 // 3 + 7`
c) `10 / (3 + 7)`
d) `10 / 3 - 3`
e) `10 / (3 - 3)`
f) `10 // (3 - 3)`

Answer: (e) and (f) produce `ZeroDivisionError`s because division-by-zero is not allowed in Python.

1.3 *(Creating a Jupyter Notebook)* Using the techniques you learned in Section 1.10.3, create a Jupyter Notebook containing cells for the previous exercise's expressions and execute those expressions.

Answer: Open the file `ex03_03.ipynb` in Jupyter.

1.4 *(Computer Organization)* Fill in the blanks in each of the following statements:

a) The logical unit that receives information from outside the computer for use by the computer is the _____.

Answer: The input unit.

b) _____ is a logical unit that sends information which has already been processed by the computer to various devices so that it may be used outside the computer.

Answer: The output unit.

c) _____ and _____ are logical units of the computer that retain information.

Answer: The memory unit, the secondary storage unit.

d) _____ is a logical unit of the computer that performs calculations.
Answer: The arithmetic and logic unit (ALU).
e) _____ is a logical unit of the computer that makes logical decisions.
Answer: The arithmetic and logic unit (ALU).
f) _____ is a logical unit of the computer that coordinates the activities of all the other logical units.
Answer: The central processing unit (CPU).

1.5 ***(Clock as an Object)*** Clocks are among the world's most common objects. Discuss how each of the following terms and concepts applies to the notion of a clock: class, object, instantiation, instance variable, reuse, method, inheritance (consider, for example, an alarm clock), superclass, subclass.

Answer: Class—A **class** `Clock` would define the **instance variables** and **methods** that represent a clock's data and functionality. A clock's instance variables (attributes) might store the clock's time, style (digital or analog), etc. The behaviors of the clock include setting the time and getting the time.

Object/Instantiation—You'd create an **object** of class `Clock` (known as **instantiation**) to represent a clock in software. The entire clock is an object that is composed of many other objects (such as the moving parts, the face, etc.).

Reuse—You can reuse class `Clock` to create as many clock object's as you need.

Inheritance, Superclass, Subclass—There are specific types of clocks (such as an alarm clock or a watch). With that in mind, it is possible that other classes, such as `Watch` and `AlarmClock`, could inherit the features in class `Clock`. In this case, `Clock` would be the **superclass** (also called the **base class**) and `Watch` and `AlarmClock` would be the **subclasses** (also called **derived classes**). Using class `Clock` as a superclass is another form of **reuse**.

1.6 ***(Gender Neutrality)*** Write down the steps of a manual procedure for processing a paragraph of text and replacing gender-specific words with gender-neutral ones. Assuming that you've been given a list of gender-specific words and their gender-neutral replacements (for example, replace "wife" or "husband" with "spouse," replace "man" or "woman" with "person," replace "daughter" or "son" with "child," and so on), explain the procedure you'd use to read through a paragraph of text and manually perform these replacements. How might your procedure generate a strange term like "woperchild" and how might you modify your procedure to avoid this possibility? In Chapter 3, you'll learn that a more formal computing term for "procedure" is "algorithm," and that an algorithm specifies the *steps* to be performed and the *order* in which to perform them.

Answer: Search through the entire paragraph for a word such as "wife" and replace every occurrence with "spouse." Repeat this searching process for every gender specific word in the list. You could accidentally get a word like "woperchild" if you are not careful about how you perform replacements. For example, the word "man" can be part of a larger word, like "woman." So, replacing every occurrence of "man" can yield strange results. Consider the process of replacing "man" with "person" then replacing "son" with "child." If you encounter the

word “woman,” which contains the word “man,” you’d replace “man” with “person” resulting in the word “woperson.” In a subsequent pass you’d encounter “woperson” and replace “son” with “child” resulting in the “woperchild.”