(http://csmatters.org) 4 - 4

0b100 - 0b100

File Input and Output using Python

Unit 4. Data Acquisition

Revision Date: Jan 05, 2020
Duration: 2 50-minute sessions

Lesson Summary

Summary

This lesson introduces students to reading information from an input file and writing to an output file as a functionality of Python programming as an example of using a data source to gain insight. The students will then apply these concepts to program a simple Dice Roll application to generate data. This lesson will prepare students to read and write files for use in later Data Acquisition lessons.

Learning Objectives

CSP Objectives

- *EU CRD-2 - Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.*
  - LO CRD-2.B - Explain how a program or code segment functions.
  - LO CRD-2.C - Identify input(s) to a program.
- *EU DAT-2 - Programs can be used to process data, which allows users to discover information and create new knowledge.*
  - LO DAT-2.D - Extract information from data using a program.
  - LO DAT-2.E - Explain how programs can be used to gain insight and knowledge from data.
- *EU AAP-1 - To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.*
  - LO AAP-1.A - Represent a value with a variable.
  - LO AAP-1.B - Determine the value of a variable as a result of an assignment.
- *EU AAP-2 - The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.*
  - LO AAP-2.B - Represent a step-by-step algorithmic process using sequential code statements.
  - LO AAP-2.H - For selection: a. Write conditional statements. b. Determine the result of conditional statements.
  - LO AAP-2.K - For iteration: a. Write iteration statements. b. Determine the result or side-effect of iteration statements.

Key Concepts

Outcomes

- Students will learn how to read from an input data file and write to an output file using the Python programming language.

- Students will consider how combining data sources, clustering data, and classifying data are parts of the process of using programs to gain insight and knowledge from data.
- Students will be able to apply their new knowledge to writing independent programs.

Essential Questions

- How can computation be employed to help people process data and information to gain insight and knowledge?
- How do people develop and test computer programs?

Teacher Resources

Student computer usage for this lesson is: **required**

Python for Everybody by Charles Severance, http://do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf (http://do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf).

Explanation of the CountIf function in Excel https://support.office.com/en-us/article/COUNTIF-function-e0de10c6-f885-4e71-abb4-1f464816df34 (https://support.office.com/en-us/article/COUNTIF-function-e0de10c6-f885-4e71-abb4-1f464816df34).

The mbox.txt and mbox-short.txt files are in the Lesson Resources Folder.

Lesson Plan

Session 1

Getting Started (10 min)

Think-Pair-Share

- List some pros and cons of inputting data for a program using only a keyboard.
- List some pros and cons of displaying output of a program using only video display.

Have students review their journal entries as a class and note the advantages and disadvantages on a white board.

Guided Activity (40 min)

Before Starting The Activity:

- Have the students open the book *Python for Everybody* by Charles Severance http://do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf (http://do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf) and navigate to Chapter 7 "Files" (page 79).
- Students should also open PyCharm or the Runestone coding environment and create a new .py file. (Teachers may want to project PyCharm and the textbook on the board.)
- The two files mbox.txt and mbox-short.txt are located in the Lesson Resources Folder. Students will need to have these saved in the same folder as their python file for the lesson.

Chapter 7: Files

The students should code the examples in the book as the teacher proceeds through the lessons.

1. **Section 7.1 Persistence**
   a. Emphasize the disadvantages of the loss of information when the computer is powered off and the frequent need to store data in a more permanent location.
   b. Introduce the concept of secondary memory.
2. **Section 7.2 Opening Files**
   a. After reading the brief text in Section 7.2, have students type the sample code to open the mbox.txt file and run it.
   b. Have students try to open another nonexistent file and see what kind of error they get.
   c. **Check the student's code for understanding:**

        - Were the students able to open the mbox.txt file successfully?
        - Did the students get a “No such file or directory” error message when they attempted to open a nonexistent file?
3. **Section 7.3 Text Files and Lines**
    a. Introduce the newline special character “end of line” which breaks the file into lines. In Python the newline is represented by “\n” in string constants.
    b. Have the students type in the sample code in 7.3 (page 81), which demonstrates the newline character’s function.
    c. **Check the student’s code for understanding**
        - Did the “\n” cause the string to be displayed on two lines?
4. **Section 7.4 Reading Files**
    a. Introduce the notion of reading each line in a file by using a while loop. (The students should already be familiar with for and while loops, but may need a quick review of the syntax.)
    b. Have students count the lines in the mbox.txt file by running the first sample code in 7.4 (page 82).
    c. **Check the students’ code for understanding.**
        - Does your code give the same output as in the book?
5. **Section 7.8 Writing Files**
    a. Show the students how to open a file for writing by opening it with mode 'w' as a second parameter.
    b. After reading the brief text in Section 7.8, have students type the sample code to write to the output.txt file and run it. Remind them to use the newline character, “\n”, at the end of each line.
    c. **Check the student’s code for understanding:**
        - Were the students able to write to the output.txt file successfully?
        - Did the students remember to close their output file when finished?

Session 2

Python Lab (50 min)

Discussion:

Discuss what possible data sources might be used as input to programs. Brainstorm ways that combining data sources, clustering data, and classifying data are parts of the process of using programs to gain insight and knowledge from data. Point out to students that data doesn't always fall from trees in exactly the quantity, quality, and format needed for your program. A programmer needs to be a critical consumer of data and know when to use multiple sources, clean the data, or sort through to find the right breadth and variety of data needed.

Preparation:

- Before doing the lab/homework students need to know how to use the `countIf` function in Excel.
- **Example:** `this = COUNTIF(A1:A1000,1)` counts how many 1s are in the range A1 to A1000. You can show the example on the Microsoft office help website. https://support.office.com/en-us/article/COUNTIF-function-e0de10c6-f885-4e71-abb4-1f464816df34 (https://support.office.com/en-us/article/COUNTIF-function-e0de10c6-f885-4e71-abb4-1f464816df34)

Part 1:

- Open the Python program(s) for rolling two six sided and one twelve-sided dice (students should have saved this from lesson 4-3).
- Edit the code so it rolls the dice 1000 times and write the results of each roll to an output file named diceRolls.dat.

Part 2:

- Import the diceRolls.dat file into your spreadsheet software (such as Excel) either as text or cvs file type.
- Make use of the `countif` function to compare the distribution of the rolls for how many times each number 2 through 12 was rolled with the pair of six-sided dice to the distribution for the 12-sided die.
- Show the comparison visually with an appropriate chart in your spreadsheet software.

Options for Differentiated Instruction

Have students work in pairs as the new concepts are introduced and practiced.

For a class needing more scaffolding: Work as a group. Have students take turns around the room to read aloud the brief text in each section in Chapter 7. Do the short exercises together with a "row captain" assigned to each row (or group) in the classroom who is in charge of checking that everybody in their row has completed each short task and has gotten the help needed to finish. Row captains help each other until the entire class has successfully completed each task. Report out on what challenges were encountered, recording problems and solutions at the front of the classroom as the class works. Rotate the role of row captain for each section.

For more independent students: Introduce/demonstrate the key ideas first and then allow student to work through Chapter 7 at their own pace.

Evidence of Learning

Formative Assessment

The teacher will check the student’s code for understanding.

The teacher will check for understanding as each new concept is introduced.

Summative Assessment

Exercise 7.1 Write a program to read through a file and print the contents of the file (line by line) all in upper case. Executing the program will look as follows:

python shout.py

Enter a file name: mbox-short.txt

FROM STEPHEN.MARQUARD@UCT.AC.ZA SAT JAN 5 09:14:16 2008

RETURN-PATH: <POSTMASTER@COLLAB.SAKAIPROJECT.ORG>

RECEIVED: FROM MURDER (MAIL.UMICH.EDU [141.211.14.90])

BY FRANKENSTEIN.MAIL.UMICH.EDU (CYRUS V2.3.8) WITH LMTPA;

SAT, 05 JAN 2008 09:14:16 -0500

You can download the sample input file from https://www.py4e.com/code3/mbox-short.txt (https://www.py4e.com/code3/mbox-short.txt)

Exercise 7.2 Write a program to prompt for a file name, and then read through the file and look for lines of the form:

X-DSPAM-Confidence: 0.8475

When you encounter a line that starts with “X-DSPAM-Confidence:” pull apart the line to extract the floating point number on the line. Count these lines and the compute the total of the spam confidence values from these lines. When you reach the end of the file, print out the average spam confidence.

Enter the file name: mbox.txt

Average spam confidence: 0.894128046745

Enter the file name: mbox-short.txt

Average spam confidence: 0.750718518519

Test your file on the mbox.txt and mbox-short.txt files.

(http://www.umbc.edu/) (http://www.umd.edu/) (http://www.nsf.gov/)

Authored by: CS Matters in Maryland
Website: csmatters.org (http://csmatters.org)
Email: csmattersinmaryland@gmail.com (mailto:csmattersinmaryland@gmail.com)