420-SN1-RE Programming in Science - Lab Exercise 14

October 16, 2024

Introduction

Goals for this lab:

- Working with files
- Reading files, iterating through data contained in files

Introduction to files and folders

A file is a collection of related information stored as a unit on your computer. All files have names (the file name) that are used to identify the contents and type of file. Files are useful for storing information on your computer long-term.

In general, Python can open files using the following syntax:

```
x = open("myfilename.txt")
```

However, you often have to provide more information to Python about where your files are stored.

Folders and paths

Files are stored on the permanent storage device of your computer (e.g., solid state drive or hard disk drive). All such devices are organized in a hierarchical manner using *folders*. A folder on your disk is a named storage area that can contain files and even other folders. All storage devices have a *root* folder. The root folder holds all other files and folders on the device.

A *path* specifies the location of a file (or folder) on your disk. Using this path, programs (including the programs you write in Python) can open files to read contents from or write contents to these files. Paths come in two flavors: *absolute* and *relative*. Typically, we don't have to think much about folder paths. However, when programming with files, we will need to understand how to specify the location of the files using paths.

The following sections show examples of file paths. The examples will not work for your computer, since the path is unlikely to exist.

Absolute paths

Absolute paths specify the location of a file starting from the root folder of your device. For example, the following is a possible absolute path specification on a PC for a file with the name *dna.txt*.

```
C:\Users\jason\Documents\Programming\alice.txt
```

This path begins at the root folder of the storage device named *C* and gives a list of folders separated by backslashes (\). The path terminates with the filename of the intended file.

If you are on a Mac, your paths may look like the following.

```
/Users/jason/Documents/Programming/alice.txt
```

Both paths above can be read roughly as: start at the root folder of your storage device, go into the folder called "Users", next go into the folder called "jason", next go into the folder called "Documents", next go into the folder called "Programming", then look for the file called "alice.txt". Again, paths on your computer may look different from the above examples!

In Python, if the above path were valid for your computer, you could run following statement to open the file `alice.txt`

```
x = open("C:\Users\jason\Documents\Programming\alice.txt")
```

Absolute paths have some limitations. Absolute paths tend to be quite long, especially for files that are deeply embedded on your storage device. Moreover, programs that use absolute paths won't work well on different computers, since every computer may have a different folder structure. The other flavor of folder paths will solve these issues!

Relative paths

When programming, our Python interpreter can be told to run from a specific path on our computer (often called a working directory). File names can then be specified *relative* to that path.

For example, assuming the above absolute folder paths were valid, you could set the working directory of Python to `/Users/jason/Documents/Programming/` using the function `chdir()` from the `os` module. After, you can open the file without specifying the path.

```
    os.chdir("/Users/jason/Documents/Programming/")
    x = open("alice.txt")
```

IDLE sets the working folder when Running Module

Many IDEs, including IDLE, will automatically set the working folder to the folder of the running program.

Exercise 0: Discovering the Python working directory

The following steps are not submitted. However, they will help you understand how to deal with files in Python. **To discover the working directory of Python follow the steps below:**

1. Open up IDLE shell.
2. Enter the statement: `import os #import module for interfacing with the operating system`
3. Enter the statement: `os.getcwd() #get the current working directory`
   What path is your IDLE shell working in?

Now, try to change the Python working directory to the location of the Lab 14 files:

1. Unzip the contents of the ZIP file provided for Lab 14 into a folder on your computer. The choice of location is yours!
2. Try to identify the absolute path of the resulting folder. You can see this information in File Explorer (Windows) in an area similar to the address bar on a web browser (where you would type in a website).

If on Mac, open your finder go to the View menu > Show Path Bar, or press the Option key to show the path bar momentarily. The location and nested folders that contain your file or folder are displayed near the bottom of the Finder window.

3. Use an `os.chdir()` statement to change the Python working directory to the path you identified.
4. Enter the statement: `os.getcwd()`
   Did your working directory change to the desired path?

Using Python, try to read the contents of the text file: `my_first_open.txt`:

1. After following the above steps. Enter the statement: `FILE = open("my_first_open.txt")`
   Ideally, it should run without error.
2. Enter the statement: `text = FILE.read()`
3. Print the contents of `text`

If it worked, you should see the contents of the file printed by Python. If something went wrong, try to verify that Python is working in the correct folder. If you are having difficulty progressing, try inspecting the file `help_me.py`. Running the file should set your interpreter to the correct location of the lab files.

1. What *type* is the object `FILE` (you can check it out by using the function `type()`).
2. What *type* is the object `contents`?
3. What did the `FILE.read()` method do?

Exercise 1: The case of the missing grant money

You are part of a special digital forensic team tasked with solving a mystery involving missing grant money. A research team has depleted its entire budget after two months, and all members have disappeared without a trace. The researchers left important research notes hidden across several files in a mystery folder. Your team has provided you with the forensic copy of the folder `mystery_folder`. Your team has determined that a specific keyword ("formula") is crucial to unlocking the next step in your investigation.

Objective

Your mission is to write a Python program `forensic_analysis.py` that will:

a. Search through all text files in the folder `mystery_folder`, including its subfolders `clues`, `notes`, and `experiments`.

b. Count how many times the keyword `"formula"` appears across all files.

c. Identify the location of the file with the most occurrences of the keyword and keep track of the number of occurrences in the file.

d. Generate a report: the total occurrences of the keyword across all files, the filename with the most occurrences, the number of occurrences in the file, the text of the file with the most keyword occurrences.

What most likely happened to the missing grant money?

Requirements

- Determine the paths of all the files you must search. These paths can be stored in a Python `list`.
- Use Python's file handling capabilities to read text data from the files.
- Ensure your program is efficient in handling the various file paths. In other words, you should avoid repetitious code.
- Making use of pre-defined string methods may help you comb through the files.
- Make sure to print your results in a clear and organized format.

Exercise 2: Counting letter frequencies

Enclosed in this lab is a file, `alice.txt`, that contains the entire text of the book "Alice's Adventures in Wonderland"[1]. Write a program named `letters.py` that computes the frequencies of the letters of the alphabet, based on the text in the book. Represent the letter frequencies as a `dict`, where each key will be a lower case letter, and the value will be the number of times that letter was found in the text. Look at Lab 13, Exercise 1, # 10 for a hint about creating a dictionary with lowercase letters as the keys.

The program should do the following:

1. Use `open()` to access the file `alice.txt`
2. Read the contents of the file.
3. Count all of the letters 'a' through 'z' in the file contents. Be sure to convert all letters to lower case, so all letters are counted together. It is unnecessary, but harmless, to count non-letters, although that might complicate printing your results.
4. Close the file.
5. Print all of the letters and their frequencies in alphabetical order (e.g. from 'a' to 'z').
6. Using `matplotlib`, display a bar chart showing the frequencies of the letters.
7. **Optional** - Display the bar chart in increasing order of the letter frequency.

What to hand in

Combine your two programs in a ZIP file and upload them to Omnivox.

[1]Courtesy of Project Gutenberg, www.gutenberg.org