Python Input & Output Using Files

(plus some stuff on multi-level lists)

CMSC 201 Section 60
October 21, 2024

Administrative Notes

Homework 5 is due tonight

Project 1 is out; due NEXT Friday - November 1

Administrative Notes (part 2)

For reference:

Project 1 is due November 1

Project 2 - file I/O, lists of dictionaries, building a database & writing an interface for it - released November 4; due November 18

Project 3 - using Jupyter Notebooks and your Python skills to analyze data - release November 18; due December 9 (the last class)

Homework 6 - either:

- Released November 18; due November 25; or
- Released December 2; due December 9

Midterm 2 is still Wednesday, November 6

Final Exam is Monday, December 16

Input and Output in Python

Up until now, all input has been from the keyboard and all output has been to the screen

X = input(“kindly hand over the data I need.”)

- Everything comes in as a string; you have to cast it to another type if you need integers, floats, etc.

print (“the output is”, output)

That has limited usefulness

It’s a pain in the neck to ask the user to type in data, over and over and over…

You can’t easily preserve your program results for later use, unless you do a whole bunch of screenshots

It would be far better if you could have all the data stored in a file that was saved on a computer, so it didn’t have to be typed in

And it would help if you could save your output in a file that you could reference later on, or even use for further processing

Fortunately, you can!!!

File Input in Python

You can read from any file that can be found on the computer - that is in the “Python path”

- Exactly how this works varies by operating system, but there is an environment variable that can be set that tells the Python interpreter where to look for any files you reference

The easiest thing to do is just make sure that your data files are in the directory in which you’re working

- Or you can specify the full path in the **open** statement

Opening a file

Before you can read from or write to a file in Python, you have to open it

There are several different open statements; we'll use

```
with open("filename", "mode") as f:
        #rest of your code indented one tab
```

"

"Mode" is one of:

- "r" if you just want to read from the file
- "r+" if you want to both read from and write to the file - modify existing contents
- "w" if you want to write to the file - ERASE EXISTING CONTENTS!!
- "a" if you just want to append to the file - add new content without modifying existing content

Modes

- If you leave off the mode value, it defaults to “r” - you can read from the file but you can’t modify it
- Using “w” gives you a new, empty file!! If the file previously existed, any content that was there before will be erased.
- Using “a” does not erase existing content; it just moves the “stream pointer” to the end of the existing file and adds from there

Text mode

For our purposes, all files are just text. They consist of one or more strings and lines of characters. All read commands return strings.

There are “binary” files in Python that are different, but we’re not going to talk about them right now.

Reading from a file

File name in the current directory or in the path

```
with open(“test_data.txt”, “r”) as f:
    data = f.read()  #reads in all data from the file as a single string
    data_lines = f.readlines() # reads in all data from the file
                               # returns a list of strings - each element in the
                               #list is one line from the file
    next_line = f.readline() #reads in one line from the file; returns it as a string
```

Use one of these three

If you want to give a path name for the file:

On a Mac:

with open (“/Users/alfredarsenault/PycharmProjects/Fall_2021/next_test.py”, “r”) as f:

This will work fine because MacOS uses the forward slash / which is not a reserved character in Python.

On Windows:

```
with open("C:\Users\Dad\PycharmProjects\next_test.py", "r") as f:
    s = f.read()
```

Results in:

```
C:\Users\Dad\PycharmProjects\exam_qs\venv\Scripts\python.exe C:/Users/Dad/PycharmProjects/exam_qs/new_dem.py
  File "C:/Users/Dad/PycharmProjects/exam_qs/new_dem.py", line 1
    with open("C:\Users\Dad\PycharmProjects\next_test.py", "r") as f:
              ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape

Process finished with exit code 1
```

You have to escape the ‘\’ character

```
with open("C:\\Users\\Dad\\PycharmProjects\\next_test.py",
"r") as f:
    s = f.read()
```

Linux is like Mac; it uses / which doesn’t have to be escaped. But make sure your code runs on gl!!

How it works

There is a ***file pointer*** for any open file, that points to the next character to be read or the place where the next item will be written.

When you open a file for reading, the file pointer starts at the beginning of the file.

Each time you read a line, the file pointer is advanced to the start of the next line.

When you open a file for writing, the file pointer starts at the beginning of a new file.

When you open a file for appending, the file pointer starts at the end of the current content

Reading a file - an example

```
# read in a file of integers, and print the largest integer

with open(“integers.txt”, “r”) as f:
        #read the whole file in as a single string
        data = f.read()
        #split the string into components at whitespace
        numbers = data.split()
        #we have a list of strings, each of which must be converted to integers
        for i in range(len(numbers)):
                numbers[i] = int(numbers[i])
        #now find the largest integer. Start by
        #setting the “largest” to the first value
        largest = numbers[0]
        #now see what’s bigger
        for i in range(len(numbers)):
                if numbers[i] > largest:
                        largest = numbers[i]
        # now print out the largest value
        print(“The largest integer in the file was”, largest)
```

What if we wanted to write the result to a file?

“print” puts the output on the screen. What if we wanted it to go to a file?

The file.write() method will work

```
with open(“results.txt”, “w”) as out:
        out.write(“The largest integer in the file is” + largest)
```

Next example: read from & write to the same file

Read data from a file

Manipulate the data - replace every occurrence of the word “dog” with the word “cat”

Write the modified data back out to the same file we started with

Be careful how you do this! You don’t want to erase all the data in the file before you’ve finished reading it!!!

IF YOU’RE GOING TO READ AND WRITE THE SAME FILE; OPEN FOR READING; CLOSE; THEN OPEN FOR WRITING - less dangerous than opening for r+

If we have time..

Using .csv (comma-separated value) files

Using .json (JavaScript Object Notation) files

(if no time today, we'll briefly cover these Wednesday)

Multilevel Lists

A two-dimensional, or 2D, list, is just a list whose elements are themselves lists

E.g.,

```
#2D lists
results = [
   [1, "USA", 8, 3, 11],
   [2, "GB", 7, 2, 9]
   [3, "Kenya", 5, 3, 8]
   ]
```

You can have lists as many levels deep as you want

A 3-D list

```
#3D lists
heat_results = [
   [
   [1, "USA", 8, 3, 11],
   [2, "GB", 7, 2, 9],
   [3, "Kenya", 5, 3, 8]
   ],
    [
   [4, "Nigeria", 1,2, 3],
   [5, "China", 4, 0, 4]
      ]
   ]
But in this class we won't really use things beyond 2D lists - they're
Just not practical
```

So what do you need to know about 2D lists

1. Syntax:
   a. An element is identified by two subscripts:
      i. Row comes first
      ii. Column comes second
      iii. Each subscript is contained in its own square brackets
   b. If there is only one subscript, it refers to a row in the 2D list
   c. You create a new empty 2-D list with one subscript
2. 2D lists in Python are row-oriented
   a. One subscript => row. There is no easy way to refer to a column; you have to use a loop
   b. You can add a row by appending it or inserting it the same way you do with a one-dimensional list
   c. The same with remove, pop or del
   d. You can’t do that with a column. You have to use a loop to add or remove (or do anything to) all the elements in a column

What you need to know about 2D lists (cont.)

3. Do all rows in a 2D- list have to be the same? No

E.g.,

```
results = [
   [1, "USA", 8, 3, 11],
   “vacated”,
   [3, "Kenya", 5, 3, 8]
   ]
```

If you do this with your data structures, you have to be very careful with your code.

If you tried to do something with a row or column and all that was there was one string, your program could fail in interesting ways.

2D lists

If you know those three big things about multi-dimensional lists, you’re in good shape.