

2 Exercises

Using with

If you manually close the file by calling `.close()`, the file may not be closed in exceptional circumstances. **Always use the `with` statement when opening files** in Python. See prelab9 for more detail.

Exercise 2.1 (`word_count.py`).

Write a program that takes in a filename and string as input. Then print how many times that string appears inside the chosen file. If the file does not exist, continue asking for a filename until one is given that exists. Use your source code file as test input.

Make sure to test files with that contain the same word multiple times.

```
1 $ python3 word_count.py
2 Please enter a filename: word_count.py
3 Please enter a string to search for: print
4 The string 'print' appears 102 times in the file 'word_count.py'
```

Exercise 2.2 (`design`, `simplifiediff.py`).

Before attempting to code this problem, create a file design that contains some analysis of how you think the problem will be solved. Examples include but are not limited to: a flowchart of events in the program, pseudocode, or a step-by-step process written in plain English. If you choose to scan your design, please make sure that it is legible.

Write a “diff” program that prints out the differences, line by line, of two files. Your program should ask the user for the names of two files, then print the differences between them. Follow the format output as shown below. Make sure to use proper error handling techniques for file I/O.

Assume all files have the same number of lines. The following output shows the output of the files `file1.txt` and `file2.txt`.

```
1 $ cat file1.txt
2 John goes to work.
3 Keith and Kyle went to the Ensiferum concert.
4 Alice ate an apple pie.
5 Joe cut down a tree.
6 The dog jumped over the wall.
7 $ cat file2.txt
8 John goes to work.
9 Coral went to a Kesha concert.
10 Alice ate an apple pie.
11 Joe planted a tree.
12 The dog jumped over the wall.
```

This is the result of running the script `simplifiediff.py` on the two files:

```

1 $ python simplifiediff.py
2 Enter file name 1 >>> file1.txt
3 Enter file name 2 >>> file2.txt
4
5 2c0
6 < Keith and Kyle went to the Ensiferum concert.
7 ---
8 > Coral went to a Kesha concert.
9 4c4
10 < Joe cut down a tree.
11 ---
12 > Joe planted a tree.
```

The 2c0 tag refers to where the difference occurred and can be read as line 2 character 0 (where lines are 1 indexed and characters are zero indexed). Compare the output of your script to that of the `diff` program by typing `diff file1.txt file2.txt` in your shell.

Exercise 2.3 (`readscores.py`).

Download the file `actsat.txt` provided on Canvas. It contains the following columns of whitespace-separated data:

Column 1 2-letter state/territory code (includes DC)

Column 2 % of graduates in that state taking the ACT

Column 3 Average composite ACT score

Column 4 % of graduates in that state taking the SAT

Column 5 Average SAT Math score

Column 6 Average SAT Reading score

Column 7 Average SAT Writing score

You must open this file and generate a list of dictionaries containing each row of data. Please use these keys for the dictionaries:

- "state"
- "act_percent_taking"
- "act_average_score"
- "sat_percent_taking"
- "sat_average_math"
- "sat_average_reading"
- "sat_average_writing"

For example, your code should process this two line file to form the following list of dictionaries.

1	AK	27	21.2	48	517	519	491
2	AL	81	20.3	9	556	563	554

```

1 [{ "state": "AK",
2   "act_percent_taking": 27
3   "act_average_score": 21.2
4   "sat_percent_taking": 48
5   "sat_average_math": 517
6   "sat_average_reading": 519
7   "sat_average_writing": 491
8 },
9 {"state": "AL",
10  "act_percent_taking": 81
11  "act_average_score": 20.3
12  "sat_percent_taking": 9
13  "sat_average_math": 556
14  "sat_average_reading": 563
15  "sat_average_writing": 554
16 }]

```

Exercise 2.4 (README).

For the following labs you will be required to create and submit a README file.

A README file is a simple text file that contains basic information regarding the purpose and use of the software program, utility, or game. README files often contain instructions or additional help regarding the software it accompanies.

For this lab you must have the following in your README file::

- **Purpose:** Describes what each program does and what problem it solves. You can keep this brief.
- **Conclusion:**
 - What you learned during the lab? What new aspect of programming did you learn from the lab? Be analytical about what you learned.
 - Did pair programming help in solving the problems and completing the prelab? Did you have problems with your buddy?
 - Did you work with your buddy on the lab? What sections did you discuss? Did you and your buddy carry out a review session with each others' code?
 - Did you encounter any problems? How did you fix those problems?
 - What improvements could you make?

The conclusion does not have to be lengthy, but it should be thorough. You will include the README file with your submissions.