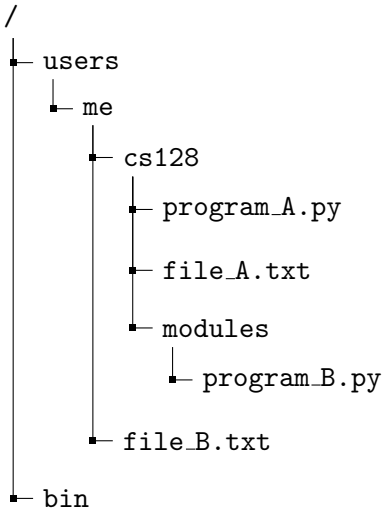
1. Up until this week, our programs have run exclusively on input typed into the text console by the user during runtime. This is not how most people interact with software day to day.

   (a) List some other kinds of runtime data, or data formats, that could be provided to a software program. We are only considering software, so your answers should be limited to digital data.

   (b) What benefits does allowing files as program input provide us as programmers?

   (c) What benefits does allowing files as program *output* provide us as programmers?

2. Consider the following folder and file structure:



   (a) What is the absolute file path to file_A.txt?

   /users/me/cs128/file_A.txt/

   (b) What is the relative file path to file_A.txt from program_A.py?

   /program_A.py/cs128/file_A.txt

(c) What is the relative file path to program_B.py from program_A.py?

/program_A.py/modules/program_B.py

(d) What is the relative file path to file_B.txt from program_B.py? (Remember that '..' refers to the parent folder of the current folder)

/program_B.py/modules/CSI28/me/file_B.txt/

3. Consider the code below:

```
1  with open('file.txt', 'r') as file_1:
2    for line in file_1:
3      print(line)
4
5  file_2 = open('file.txt', 'r')
6  lines = file_2.readlines()
7  print(lines)
8
9  with open('file.txt', 'r') as file_3:
10   all = file_3.read()
11   all = all.replace("\n", "")
12   all = all.replace("two", "one")
13   print(all)
```

Assume file.txt contains:

```
This is a text file
It has two lines
```

(a) What is the ouptut of this code?

You can assume file.txt is stored in the same folder as the program. You may want to copy the code into VSCode to run it and see. (Unfortunately, File I/O does not work with Python Tutor)

This is a text file
[
It has two lines
[
This is a tex t file
[
This is a text file
It has two line

(b) Why is the same file opened three times in this example? What would happen if our code looked like this?

```
1 with open('file.txt', 'r') as file_1:
2   for line in file_1:
3     print(line)
4 lines = file_1.readlines()
5 print(lines)
6
```

It's opened 3 times because they perform 3 different operations.

This code is more efficient because it opens the file once and performs 2 different operations with it.

(c) What important line of code (related to responsible file use) is missing?

proper error handling

4. In your own words, explain the difference between the 'a' and 'w' modes when opening a file. Give one example of when you would want to use each mode.

A is append mode and W is write mode.

Use A when adding a data entry.

Use W when you want to modify data in a file.

5. Read through the function code below

```
1  def piglatin_ify(filename):
2    with open(filename, 'r') as in_file:
3      lines = in_file.readlines()
4      lines_out = []
5      for line in lines:  # loop through all lines
6        words = line.split()
7        line_out = []
8        for word in words:  # loop through words in each line
9          new_word = word[1:] + word[0] + "ay"
10         line_out.append(new_word)
11       line_out = " ".join(line_out)
12       lines_out.append(line_out)
13
14   new_filename = filename.replace(".", "_pl.")
15   with open(new_filename, 'w') as new_file:
16     for line in lines_out:
17       new_file.write(line + "\n")  # have to add the line break
18
19   return new_filename
```

(a) In your own words, describe what this function does.

(b) This code works for some cases, but not all. Give one example of input that would cause a *runtime* error in this function.

(c) Even if this code doesn't throw an error, its output may not be what the code writer intended. Give one example of file contents that reveal a *semantic* error in the code. Refer to your answer to part a.