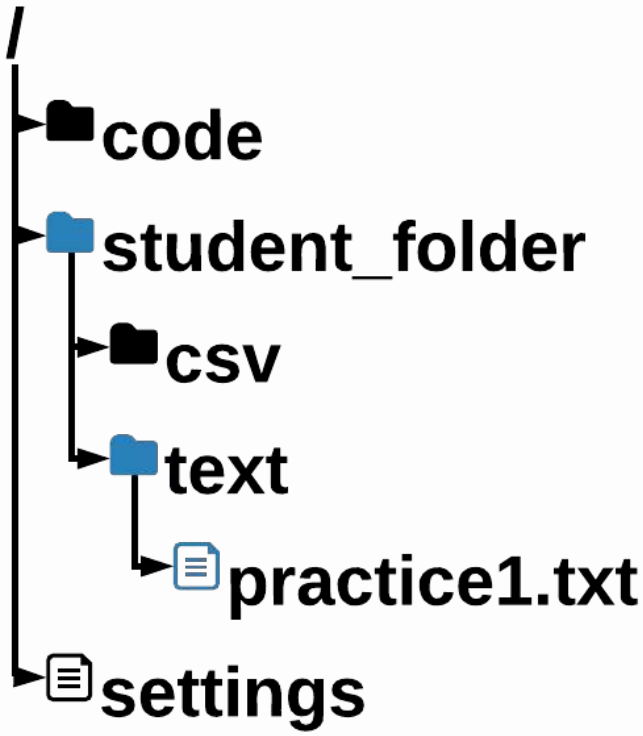
Learning Objectives - Writing

- **Navigate the file system to the appropriate folder**
- **Demonstrate how to open a file in write mode**
- **Explain what happens when you write a file that does not exist**
- **Demonstrate how to use the `writelines` method**
- **Explain what happens when you write to a file that already exists**
- **Differentiate between write and append modes**
- **Differentiate between `open` and `with open`**

Navigating the File System

Locating Student Files

This unit is all about working with files on a computer. The first step is to open the desired file. That means navigating the file system to find the file in question. The open command requires the location (path) of the file and the filename. The file is called `practice1.txt`. It is located in the `text` folder, which is inside the folder called `student_folder`. **So the path to the file is `student_folder/text/practice1.txt`.**



File Path

Importing the `os` Python module allows Python to interact with the operating system. `os` can also join the file path and filename together, allowing you to open the file.

Open File

There are three different modes when opening a file: read (`"r"`), write (`"w"`), and append (`"a"`). You will focus on the write mode for now. Be sure to close the file when you are done with it.

```
output_file = open("student_folder/text/practice1.txt", "w")
output_file.close()
```

▼ **Where is the output?**

You should see a green check mark after running your program. This means the code ran without any errors. But what about the output? The code above only opens and then closes a file.

challenge

What happens if you:

- Open `student_folder` in the sidebar on the left. Open the `text` folder and right-click on `practice1.txt`. Select “Delete…” and run the program again.

Writing to a File

Writing to a File

Once the file is opened, the `writelines()` method is used to write text to the file. Any string of text passed to `writelines()` will appear in the file. Once you are done writing to the file, close the file.

```
output_file = open("student_folder/text/practice1.txt", "w")
output_file.writelines("Hello there")
output_file.close()
```

▼ **Raw Text**

When writing to text files, Python outputs raw text. Raw text is the text that appears in a text editor. There is no special formatting or extra information attached to this text. Text in MS Word is not raw text. Raw text files have the extension `.txt`.

challenge

What happens if you:

- Change the string in `writelines()` to `"Goodbye"`?
- Change the string in `writelines()` to `""`?
- Change the mode to `open("student_folder/text/practice1.txt", "r")`?

Multiline Strings

Multiline Strings

Imagine that you want to write the words `Hello` and `there` on separate lines of a file called `practice2.txt`. If the `print` statement writes each string on its own line, then `writelines` should too.

```
output_file = open("student_folder/text/practice2.txt", "w")
output_file.writelines("Hello")
output_file.writelines("there")
output_file.close()
```

▼ **Closing a File**

Closing the file is an important step in working with files. If you forget to close a file, some unpredictable actions may take place. For example, if you open a file with newly written text before closing the file, that text may not be in the file. Be sure that you close all of the files that you open.

If you want to have text appear on a new line, then you need to use the newline character (`\n`).

challenge

What happens if you:

- Change the `writelines()` code to:

```
output_file.writelines("Hello\n")
output_file.writelines("there")
```

- Change the `writelines()` code to:

```
output_file.writelines("Hello\nthere")
```

A List of Strings

It is possible to use a list of strings with the `writelines()` method. However, these strings will be written one after another with no space between. If you want spaces, be sure to add them. If you want text to appear on a newline, use `\n`.

```
lines_to_write = ["First sentence.", "Second sentence.", "Third
        sentence."]
output_file = open("student_folder/text/practice2.txt", "w")
output_file.writelines(lines_to_write)
output_file.close()
```

challenge

What happens if you:

- Change `lines_to_write` to:
  `["First sentence. ", "Second sentence. ", "Third sentence."]`
- Change `lines_to_write` to:
  `["First sentence.\n", "Second sentence.\n", "Third sentence.\n"]`

Append Mode

Append Mode

Write mode has some unusual behavior. Writing to a non-existent file creates the file. Writing to an already existing file erases the previous contents of the file. The append mode will also create a file if one does not already exist. However, append mode will not erase content already saved to the file. It will add the text to the end of the file. Run this code and look at the output.

```
output_file = open("student_folder/text/practice3.txt", "w")
output_file.writelines("First sentence")
output_file.close()
```

Now append the following text to the file.

```
output_file = open("student_folder/text/practice3.txt", "a")
output_file.writelines("Second sentence")
output_file.close()
```

challenge

What happens if you:

- Change the append text to `writelines("\nSecond sentence")`?
- Change the program to:

```
new_text = ["How many boards\n", "Could the Mongols
        hoard\n", "If the Mongols hordes got bored?"]

output_file = open("student_folder/text/practice3.txt", "a")
output_file.writelines(new_text)
output_file.close()
```

With Open

Opening and closing files is an important part of working with files, but it can be tedious to perform both steps. The `with open` command combines the two lines of code. Once the end of the indented code is reached, the file is automatically closed.

With Open

```
with open("student_folder/text/practice3.txt", "a") as
        output_file:
    output_file.writelines("Some new text!")
```

challenge

What happens if you:

- Change the text in `writelines()` to `"\nSome new text!"`?
- Change the program to:

```
with open("student_folder/text/practice3.txt", "a") as
        output_file:
    output_file.writelines("\nSome new text!")
    output_file.writelines("\nAnd some more text!")
    output_file.writelines("\nYet even more text!")
```

Formative Assessment 1

Formative Assessment 2