

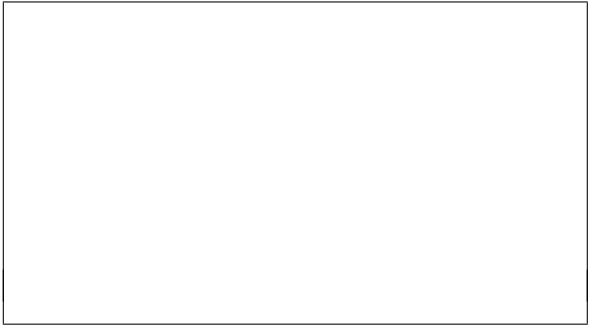
File Input-Output I

Model 1 Writing to a File

The following example creates a new file (in the current/default folder) named `out.txt` and writes several lines of output to it. The instructor will run the code with you. Examine the contents of the resulting `out.txt` file. In the space below, write the contents of `out.txt` to the right of the code.

```
1 def main() -> None:
2     outfile = open("out.txt", "w")
3     outfile.write("Example ")
4     outfile.write("output ")
5     outfile.write("text file\n")
6     outfile.write("xyz Coordinates\n")
7     outfile.write("MODEL\n")
8     outfile.write("ATOM\t1")
9     seq = "\n\t0\t1\t2"
10    outfile.write(seq)
11    outfile.write("\n")
12    outfile.close()
13 main()
```

out.txt



1. Based on the Python code:

- How many arguments are passed to `open`? What are their types?
- What variable stores the *file object* returned by the `open` function?
- Identify the names of all methods used on this file object in the code.
- What type of data does the `write` method require for its argument?

2. Based on the `out.txt` file:

- How many times was the `write` method called to create the first line of text?
- How many times was the `write` method called to create the second line of text?
- What do the `"\n"` and `"\t"` characters do?
- How is the `write` method different from the `print` function?

3. Write a program that creates a file named `lines.txt` and writes 100 lines like this:

```
Line #1
Line #2
Line #3
...
```

Model 2 Appending to a File

The second argument of `open` specifies the *mode* in which the file is opened. When writing output to a file, there are two basic modes:

- The write ("`w`") mode will overwrite/replace the file contents.
- The append ("`a`") mode will add new data to the end of the file.

Either mode will create the file automatically if it does not already exist.

4. Evaluate the following statements in the order that they are listed:

Python code	Output
<code>afile.write("new line\n")</code>	
<code>afile = open("out.txt", "a")</code>	
<code>print(afile.write("new line\n"))</code>	
<code>afile.write(2.0)</code>	
<code>afile.write("2.0")</code>	
<code>afile.close()</code>	
<code>afile.write("new line\n")</code>	

5. Explain what happens as a result of the line: `afile = open("out.txt", "a")`

6. How do the arguments passed to the `open` function differ for writing a new file in comparison to appending an existing file?

7. What does the write method return? Run `help(afile.write)` to check your answer.

8. Explain the reason for the error observed after entering:

- a) the first line of code: `afile.write("new line\n")`
- b) the last line of code: `afile.write("new line\n")`
- c) the statement: `afile.write(2.0)`

Model 3 Reading from a File

Programs often require input data from an external file source. Not surprisingly, there are methods for reading the contents of files.

9. Evaluate the following statements in the order that they are listed (assuming `out.txt` has the content from Model 1):

Python code	Output
<code>infile = open("out.txt", "r")</code>	
<code>infile.readline()</code>	
<code>infile.readline()</code>	
<code>infile.readlines()</code>	
<code>infile.readline()</code>	
<code>infile.close()</code>	
<code>infile = open("out.txt", "r")</code>	
<code>for line in infile:</code> <code> print(line)</code>	
<code>infile.close()</code>	
<code>infile = open("out.txt", "r")</code>	
<code>for i in range(3):</code> <code> infile.readline()</code>	
<code>line = infile.readline()</code>	
<code>line</code>	
<code>print(line[0])</code>	
<code>print(line[0:5])</code>	
<code>words = line.split()</code>	
<code>words</code>	
<code>print(words[0])</code>	
<code>infile.close()</code>	

10. Based on the output above:

- What type of data does the `readline` method return?
- What type of data does the `readlines` method return?

11. Why did the `readline` method return different values each time?
12. What happens if you try to read past the end of the file? Justify your answer.
13. What is the difference between the two `for` loops in the table?
14. Consider the output of the first `for` loop:
 - a) Why does the program display the file as if it were double spaced?
 - b) How would you change the code to avoid printing extra blank lines?
15. Based on the second half of the table:
 - a) Why was it necessary to open the file again?
 - b) Write code that would output 1.0 using `line`
 - c) Write code that would output 1.0 using `words`
16. Consider a file `names.txt` that contains first and last names of 100 people, with one name per line (e.g., "Anita Borg"). Write a program that prints all the last names (the second word of each line) in the file.