# File Input/Output

Most data is stored in files, not input by the user every time. In this activity, you'll learn the basics of reading and writing plain text files.

## Content Learning Objectives

*After completing this activity, students should be able to:*

- Create a new text file, and output several lines to it.
- Open an existing file, and append several lines to it.
- Read a text file line by line, and extract data from it.

## Process Skill Goals

*During the activity, students should make progress toward:*

- Justifying answers based on the results of an experiment. (Critical Thinking)

# 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. Run the code, and 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.

```python
1  outfile = open("out.txt", "w")
2  outfile.write("Example ")
3  outfile.write("output ")
4  outfile.write("text file\n")
5  outfile.write("xyz Coordinates\n")
6  outfile.write("MODEL\n")
7  outfile.write("ATOM " + str(1))
8  seq = "n " + str(0.0) + " " + str(1.0) + " " + str(2.0)
9  outfile.write(seq)
10 outfile.write("\n")
11 outfile.close()
```

out.txt

## Questions  (15 min)                                Start time: _____

1. Based on the Python code:

   a) How many arguments are passed to `open`? What are their types?

   b) What variable stores the *file object* returned by the `open` function?

   c) Identify the names of all methods used on this file object in the code.

   d) What type of data does the `write` method require for its argument?

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

   a) How many times was the `write` method called to create the first line of text?

   b) How many times was the write method called to create the second line of text?

   c) What does the `"\n"` character do?

   d) 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
   Line #4
   ...
   ```

# 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. Enter the following lines into a Python Shell, and record the output at each step.

| Python code | Shell output |
|---|---|
| `afile.write("new line\n")` | |
| `afile = open("out.txt", "a")` | |
| `afile.write("new line\n")` | |
| `afile.write(2.0)` | |
| `afile.write("2.0")` | |
| `afile.close()` | |
| `afile.write("new line\n")` | |

## Questions  (10 min)                              Start time: _____

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

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

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

**7.** 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. Enter the following lines into a Python Shell.

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

## Questions  (20 min)                               Start time: _____

8.  Based on the output above:

   a) What type of data does the `readline` method return?

   b) What type of data does the `readlines` method return?

**9**. Why did the `readline` method return different values each time?

**10**. What happens if you try to read past the end of the file? Justify your answer.

**11**. What is the difference between the two `for` loops in Model 3?

**12**. 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?

**13**. Based on the second half of Model 3:

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`

**14**. 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.