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

```
Example output text file
xyz Coordinates
MODEL
ATOM    1n    0    1    2
```

1. Based on the Python code:

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

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

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

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

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

a) How many times was the `write` method called to create the first line of text? 3

b) How many times was the write method called to create the second line of text? 1

c) What do the `"\n"` and `"\t"` characters do?

newline ends the current line; tab adds a fixed amount of spaces

d) How is the `write` method different from the `print` function? doesn't append a newline

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

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

```
outfile = open("lines.txt", "w")
for i in range(1, 101):
    outfile.write(f"Line #{i}\n")
outfile.close()
```

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
`afile.write("new line\n")`	NameError: name 'afile' is not defined
`afile = open("out.txt", "a")`	
`print(afile.write("new line\n"))`	9
`afile.write(2.0)`	TypeError: write() argument must be str, not float
`afile.write("2.0")`	3
`afile.close()`	
`afile.write("new line\n")`	ValueError: I/O operation on closed file.

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

The existing out.txt file is open for appending.

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

The second argument is `"a"` instead of a `"w"`.

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

The number of characters written (which is always equal to the length of the string).

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

a) the first line of code: `afile.write("new line\n")` the file wasn't open

b) the last line of code: `afile.write("new line\n")` the file was closed

c) the statement: `afile.write(2.0)` you can only write strings to files

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
`infile = open("out.txt", "r")`	
`infile.readline()`	'Example output text file\n'
`infile.readline()`	'xyz Coordinates\n'
`infile.readlines()`	list of the remaining lines
`infile.readline()`	`''`
`infile.close()`	
`infile = open("out.txt", "r")`	
`for line in infile:` `  print(line)`	prints each line double spaced
`infile.close()`	
`infile = open("out.txt", "r")`	
`for i in range(3):` `  infile.readline()`	
`line = infile.readline()`	
`line`	`'ATOM\t1n\t0\t1\t2\n'`
`print(line[0])`	A
`print(line[0:5])`	ATOM
`words = line.split()`	
`words`	['ATOM', '1n', '0', '1', '2']
`print(words[0])`	ATOM
`infile.close()`	

10. Based on the output above:

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

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

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

Each time `readline` is called, it returns the next line of the file.

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

From then on, `readline` returns `''`, and `readlines` returns an empty list.

13. What is the difference between the two `for` loops in the table?

The first loop iterates and prints every line of the file. The second for loop reads only the first three lines, making it possible to assign the fourth line in the next statement.

14. Consider the output of the first `for` loop:

a) Why does the program display the file as if it were double spaced?

Each line in the file ends with a newline character `\n`, and `print` adds another one.

b) How would you change the code to avoid printing extra blank lines?

Change the function: `print(line, end='')` Or change the string: `print(line[:-1])`

15. Based on the second half of the table:

a) Why was it necessary to open the file again? The data had already been read.

b) Write code that would output 1.0 using `line` print(line[17:20])

c) Write code that would output 1.0 using `words` print(words[3])

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.

```
infile = open("names.txt", "r")
for line in infile:
    words = line.split()
    print(words[1])
infile.close()
```