10.8. Writing Text Files

One of the most commonly performed data processing tasks is to read data from a file, manipulate it in some way, and then write the resulting data out to a new data file to be used for other purposes later. To accomplish this, the `open` function discussed above can also be used to create a new file prepared for writing. Note in Table 1 that the only difference between opening a file for writing and opening a file for reading is the use of the `'w'` flag instead of the `'r'` flag as the second parameter. When we open a file for writing, a new, empty file with that name is created and made ready to accept our data. If an existing file has the same name, its contents are overwritten. As before, the function returns a reference to the new file object.

Table 2 shows one additional method on file objects that we have not used thus far. The `write` method allows us to add data to a text file. Recall that text files contain sequences of characters. We usually think of these character sequences as being the lines of the file where each line ends with the newline `\n` character. Be very careful to notice that the `write` method takes one parameter, a string. When invoked, the characters of the string will be added to the end of the file. This means that it is the programmer's job to include the newline characters as part of the string if desired.

Assume that we have been asked to provide a file consisting of all the squared numbers from 1 to 12.

First, we will need to open the file. Afterwards, we will iterate through the numbers 1 through 12, and square each one of them. This new number will need to be converted to a string, and then it can be written into the file.

The program below solves part of the problem. We first want to make sure that we've written the correct code to calculate the square of each number.

Save & Run Original - 1 of 1 Show in CodeLens

```
for number in range(1, 13):
    square = number * number
    print(square)

```

```
1
4
9
16
25
36
49
64
81
100
121
144
```

Activity: 1 -- ActiveCode (ac9_7_1)

When we run this program, we see the lines of output on the screen. Once we are satisfied that it is creating the appropriate output, the next step is to add the necessary pieces to produce an output file and write the data lines to it. To start, we need to open a new output file by calling the `open` function, `outfile = open("squared_numbers.txt",'w')`, using the `'w'` flag. We can choose any file name we like. If the file does not exist, it will be created. However, if the file does exist, it will be reinitialized as empty and you will lose any previous contents.

Once the file has been created, we just need to call the `write` method passing the string that we wish to add to the file. In this case, the string is already being printed so we will just change the `print` into a call to the `write` method. However, there is an additional step to take, since the write method can only accept a string as input. We'll need to convert the number to a string. Then, we just need to add one extra character to the string. The newline character needs to be concatenated to the end of the line. The entire line now becomes `outfile.write(str(square)+ '\n')`. The print statement automatically outputs a newline character after whatever text it outputs, but the write method does not do that automatically. We also need to close the file when we are done.

The complete program is shown below.

Note

As with file reading, for security reasons the runestone interactive textbook environment does not write files to the file system on your local computer. In an activecode window, we simulate writing to a file. The contents of the written file are shown and you can do a subsequent read of the contents of that filename. If you try to overwrite a file that's built in to the page, it may not let you; don't try to get too fancy with our file system simulator!

Below, we have printed the first 10 characters to the output window.

Save & Run Original - 1 of 1

```
filename = "squared_numbers.txt"
outfile = open(filename, "w")

for number in range(1, 13):
    square = number * number
```

```
    outfile.write(str(square) + "\n")

outfile.close()

infile = open(filename, "r")
print(infile.read()[:10])
infile.close()

```

```
1
4
9
16
2
```

Activity: 2 -- ActiveCode (ac9_7_2)

Data file: `squared_numbers.txt`

```
1
4
9
16
25
36
49
64
81
100
```

You have attempted 3 of 2 activities on this page

10.7. Recipe for Reading and Processing a File">

ecipe for Reading and Processing a File">

✔ Completed. Well Done!

10.9. CSV Format">

10.9. CSV Format">Next Section - 10.9. CSV Format

| Back to top