Today we are going to learn how to write to text files using Python. Copy or type the following into a new Python program:

```
myFile=open("day36_output.txt","w")
myFile.write("hello world")
myFile.close()
```

Run the program. Switch to the folder where you saved your program and find the new text file called "day36_output.txt". Double click it, you should see "hello world" in the file. You just told Python to create a text file and write something to it. Close the text file.

Let's look at the commands we used. The first command tells Python to open a file and to refer to the file as "myFile". You can use any legal variable name for the name of the file. The first argument in quotes is the file name, and the second item is a "w" for "write mode". This opens a text file, creating it if it wasn't already on disk, and gets ready to write to it. Then, the line "myFile.write…" writes text to the file. Finally, the line with the .close() on it closes the file. You have to close a file after writing to it or nothing will actually get written to the disk.

Change your program to look like this:

```
myFile=open("day36_output.txt","w")
myFile.write("hello world")
myFile.write("whassup?")
myFile.close()
```

We just added a second myFile.write() call with more text. Open the text file again and you'll see you have the following:

```
hello worldwhassup?
```

Notice the contents of our two write lines is jammed together with no space in between. Each time you write to a text file it just adds your new stuff to the end of the old stuff without putting blank links in between. In other words, Python doesn't break up your writes into lines like we get when we print to the console. Close the text file.

You can write different things to a file, one thing per line, by adding a "\n" to your write calls like this:

```
myFile=open("day36_output.txt","w")
myFile.write("hello world")
myFile.write("\n")
myFile.write("whassup?")
myFile.close()
```

Now you'd have the following in the text file:

```
hello world
whassup?
```

Close the text file. Let's say you want to write two different variables to a text file in one call. You can do this by making sure each variable is a string, and then by using the "+" sign between items.

(Continued on next page)

Test out this code to see how it works:

```
myFile=open("day36_output.txt","w")
a=2020
b="Go Seniors!"
myFile.write(str(a)+" "+b)
myFile.close()
```

This time the text file should contain “2020 Go Seniors!” Notice the “str” before the number variable. We need that because you can only write strings to a text file. Also notice the plus sign with the space in quotes. If you want to write two strings to a text file in a single .write() call you need to use a plus sign. The " " between the plus signs is to put a space between the 2020 and the Go Seniors! If we didn’t do that we’d get “2020Go Seniors!” with the number and the text jammed together.

When we print multiple items in Python to the screen you can separate items using commas. When we write multiple items to a text file in Python we have to actually combine them into a single item using + signs. See the sample below which writes three strings in a single .write() call.

```
myFile.write("Hello, "+username+", nice to meet you.")
```

You could alternatively do this with three write calls:

```
myFile.write("Hello, ")
myFile.write(username)
myFile.write(", nice to meet you.")
```

Either way is fine, it's your choice.

Today's assignment:

Create a program called day36_fileWrite as follows:

- Your name and the date are in a comment at the top
- The program asks the user for their name
- The program opens a text file called day36_output.txt.
- The program writes your name to the top of the file, followed by a blank line. Write "\n" to make Python go to a new line.
- The program writes to the file a greeting to the user by name in a welcoming manner (feel free to be creative). See last page for an example.
- The program writes another blank line to the output file.
- The program incorporates your day13_sentences program and adds the user name to the list of subjects. Remove the part that asked you if you want to run again; we don't want that today.
- The program writes 10 sentences to the text file. Make sure each sentence is on its own line. You will need to adapt your day13 code to use "+" (plus signs) between parts that you are trying to write to the text file (or simply write one word at a time to the text file, each on its own line as shown above).
- The program closes the file and prints "Done." to the screen.
- All that gets printed to the Shell window are the prompt for the user to enter a name and the word "Done.", they key parts of this program are all written to the text file. To see what happened, open the text file and look at it. Close it before you run the program again.

Make your program ask the user for their name and write the greeting to the text file before you try to incorporate your sentences program. Make sure it's working. How can you tell? Double click the file

day36_output.txt to make sure it has what you want in it. Only then should you open sentences and work on incorporating that program into this one.

If you never turned in day13_sentences let me know and I'll give you a skeleton version to use for today's assignment.

Sample run:

When I run my program, here is what the Python Shell shows:

```
RESTART: C:\Users\Walt Hays\Desktop\36_fileWriteHays.py
Please enter your name: Frankie
Done.
>>>
```

The file day36_output.txt is created in the same folder as my program. The contents of the file are here:

```
Coded by Mr. Hays

Hello, Frankie, nice to meet you.

Here are some cool sentences:

Frankie squawked quietly.
A chicken ate cans loudly.
Frankie squawked quietly.
The goat squawked quietly.
Frankie ate cans loudly.
Frankie ate cans loudly.
Frankie ate cans quietly.
A chicken squawked quietly.
A chicken ate cans quietly.
A chicken ate cans quietly.
```

Notice that the name the user entered, Frankie, was added as one of the possible random subjects for the sentences. That is one of the requirements.