Exercise: Hello World

Summary

This exercise goes over the basics of writing text from Python scripts. It also goes over concatenating strings and how scripts can create Markdown files, which can be convenient when preparing nicely-formatted output.

Deliverables

Your finished repository should have three new files: scripts called **hello1.py** and **hello2_file.py** and an output file called **hello2_file.md**.

Instructions

1. Please create a new script called `hello1.py` that uses the `print` function to write the string "Hello, World!" Once the script is working properly, commit it to the repository.
2. Create a new script called `hello2_file.py` that does the following.
    1. Creates a variable called `author_name` that is equal to your name as a string;
    2. Uses string concatenation to create a new variable called `author_msg` that is equal to the concatenation of three parts: the string `"Author: *"`, the variable `author_name`, and the string `"*\n"`;
    3. Opens a new file called `hello2_file.md` for writing;
    4. Writes four lines to the file using the `write()` method: the string `"# Hello, World!\n"`, a blank line (the string `"\n"`), a line consisting of the variable `author_msg`, and another blank line;
    5. Creates a numeric variable called `days` equal to the number 365;
    6. Creates a numeric variable called `hours` equal to the number 24;
    7. Writes one line to the file using the `print()` function with the arguments `"Hours per year:"` and `days*hours` plus the `file=` argument.
    8. Closes the file.
3. If you aren't sure how to do one or more of these steps, have a look at the accompanying `demo.py` file. It has some example code that may be useful.
4. When `hello2_file.py` is working properly, it should produce output similar to `example.md`. Once you're happy with it, commit the finished versions of `hello2_file.py` and `hello2_file.md` to the repository.

Submitting

Once you're happy with everything and have committed all of the changes to your local repository, please push the changes to GitHub. At that point, you're done: you have submitted your answer.

Notes

- Basing the name of a script's output file on the name of the script itself can be very helpful in projects that involve a lot of files. It ensures that the two are close to each other when someone looks through the directory.
- Having to include the `\n` explicitly at the end of lines in write statements may seem annoying at first but actually it's very useful in some circumstances. It allows you to build up a long line with several write calls before adding the `\n`.
- The print function's automatic formatting is very handy but there are times when it can be undesirable. A prime example is when you don't actually want spaces inserted between variables. For example, `print("Author:`

`*",author_name,"*")` adds spaces between the asterisks and the name so the result looks like "Author: * Otto Orange *". That keeps the name from being shown in italics in a Markdown viewer.