Nicholas Bowman, Sonja Johnson-Yu, Kylie Jue
CS 106AP

Handout #11
July 14, 2019

Using the Command Line

This handout explains how to use the command line to run your Python scripts (files). For Assignments 1 and 2, we provided run configurations with the starter code so that all you needed to do was hit the green arrow  in PyCharm to run your code. Run configurations tell PyCharm what file your program is located in and what command line arguments your code requires (if any).

> **Fun Fact**:
> When "configuring the Python Interpreter," as you had to do when you first set up PyCharm, you were telling PyCharm what version of Python to use behind-the-scenes (i.e. saying `python3` vs. `python2` when running scripts and using python3 when executing commands in the Python Interpreter/Console).

But now that you've learned about lists and the command line, we want to encourage you to run your programs using the command line itself! In fact, this is how most everyday Python scripts are run, and learning how to do this means that if you wanted to, you could run your programs without using PyCharm at all.

Note that depending on your device, there are multiple different ways to open a command line application:

- All computers have a built-in command line application that allows you to communicate directly with your computer's operating system. On Macs (and usually on Linux machines), the application is called **Terminal**, and on Windows computers, the application is called **Command Prompt**.
- PyCharm also has a command line application that has the ability to talk to your computer's operating system. This is built in as a "**Terminal**" tab at the bottom of the PyCharm window (see Figure 1).

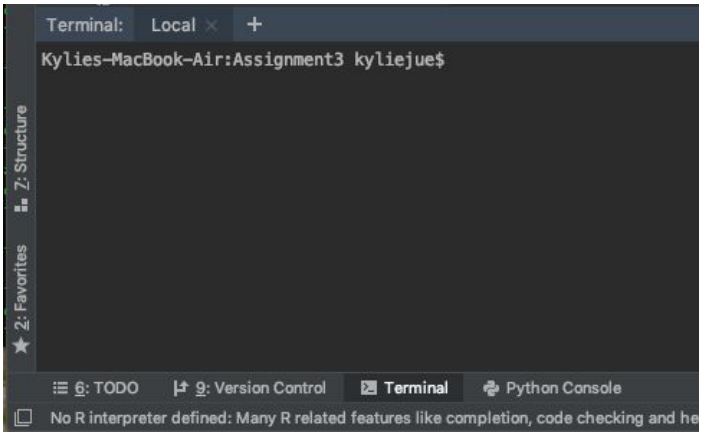

Figure 1: The PyCharm command line application is a tab labeled "Terminal" at the bottom of the application.

To understand how the command line works, we must first break down the syntax and structure of a Python command line command:

Figure 2: The general command line structure for running Python programs

If you are on Windows, you should use `py` instead of `python3` in all these commands.

This is the command that gets typed directly into the command line application (Terminal/Command Prompt) and tells the computer to run your program. Depending on the program you wrote, the command may include zero or more arguments. In the example above, the program file `script_name.py` takes a filename (`filename.txt`) as its only argument. But you could also have a script that takes in no arguments, such as all of your programs for Assignment 2:

Figure 3: The command line command you would use to run your `liftoff.py` program.

Note that in order to run this command, **you'll need to navigate into the folder where your program file is located.** Otherwise, the "run this script (file)" portion of the command would need to include the full file path to your script. To navigate directories within your command line application, use the `cd` (change directory) command. **If you use the terminal application *within PyCharm*, you won't need to do this because you'll already be inside your project folder.** However, if you're using your computer's built-in command line application, your default directory may not be where your code files are stored.

For example, Figure 4 shows the path to the Assignment 2 folder where the `liftoff.py` script is located on Kylie's Macbook. **But again, this file path will be different for each person depending on where you store your code files on your computer.**

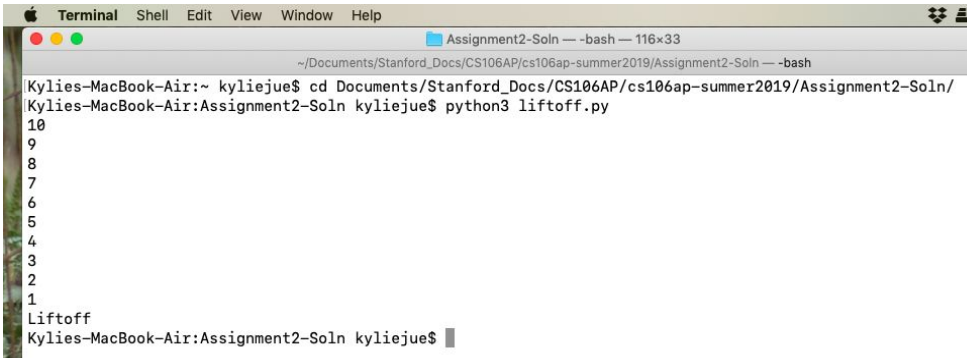

Figure 4: You can use the `cd` command in a command line application like Terminal to navigate to the project folder where your Python files are stored.

For Assignment 3, we've already written the code in **`main()`** to help you parse the command line arguments. For example, **`ghost.py`** contains the following code inside **`main()`** (we've renumbered the lines below for ease of reference):

```
1      def main():
2          # (provided, DO NOT MODIFY)
3          args = sys.argv[1:]
4          # We just take 1 argument, the folder containing all images.
5          # The load_images() capability is provided above.
6          images = load_images(args[0])
7          solve(images)
```

Figure 5: The provided code inside **`main()`** for **`ghost.py`**

Line 3 uses the Python sys library to get the arguments provided via the command line. Since we learned in Lecture 11 that these arguments are **passed into the code as a list**, we can also see that we use list slicing to get only the arguments after the first list item. This is because the first item in the list **`sys.argv`** is actually the name of the Python script itself (**`ghost.py`**)! Therefore, our new **`args`** list will only contain the arguments that appear after the script name. In the case of **`ghost.py`**, this list will only contain one argument, the name of the folder containing all of the images we want to process. In summary, the command to run **`ghost.py`** looks like this:

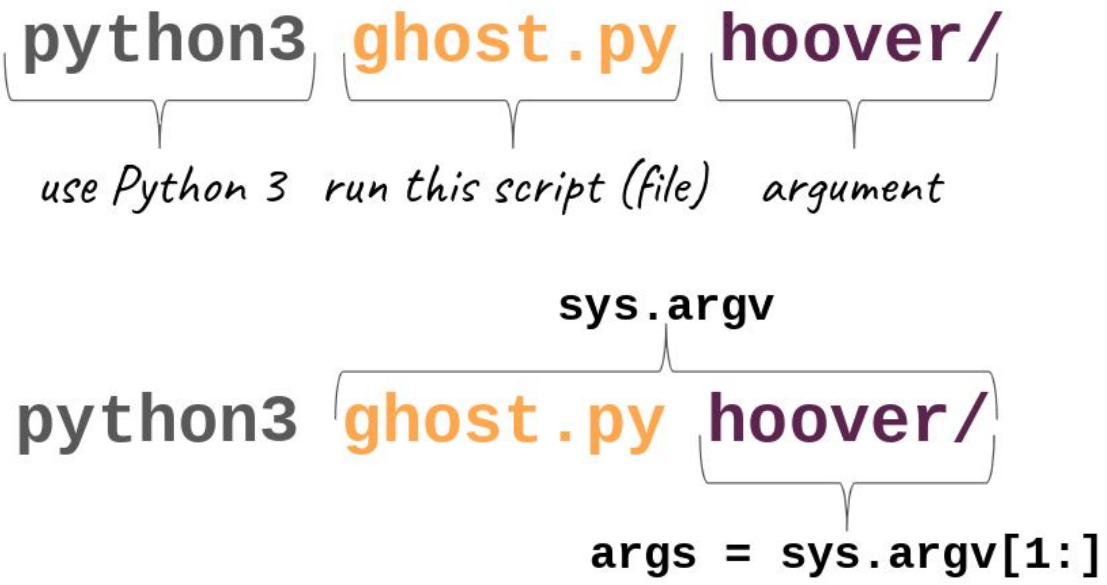

Figure 6: The command to run **`ghost.py`** on all of the photos in the **`hoover/`** folder via the command line

That's it! Now you know how to run your Python programs using the command line. Although we'll continue to use PyCharm as a code editor throughout the quarter (especially since it makes running doctests easier), you now also have the knowledge to write and run Python scripts without using PyCharm at all!