


A (debatably) Brief Introduction to Python



CHAPTER 0: THE PYTHON FILE

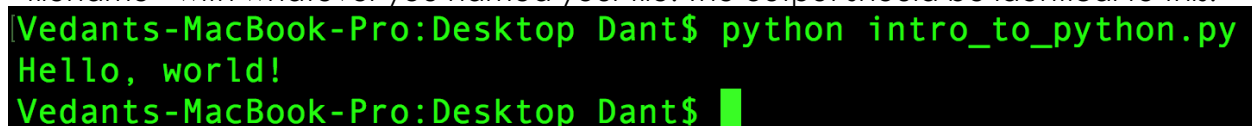
Let's start this introduction to python in the most generic way possible, shall we? Using what you've learned from the command line, navigate to your development folder (if you don't have one, make one) and create a python file. Name it whatever you want, but remember that **all python files end with the extension ".py"**, so your filename should look something like `<filename>.py`. Open it up with whatever text editor you like (I use sublime text), but keep your command line window open. Great. Now, you have a blank python file.

The best part of all python files is that (at least in python 3), there is no need to create a main function (more on functions later). Whatever code you write in the python file, regardless of where it is located, will be run by the compiler. Of course, bespoke functions will not be run unless called in the file. Speaking of which, let's call python's built-in `print()` function on line 1 of our blank python file. It should look a little something like this:

A screenshot of a text editor window titled 'intro_to_python.py'. The editor shows three lines of code: line 1 contains `print("Hello, world!")`, line 2 is empty, and line 3 is empty. The line numbers 1, 2, and 3 are visible on the left side of the editor.

As is the case with all functions, the `print()` function processes whatever is inside the parentheses. More on `print()`: this function will pretty much output whatever is passed through the parentheses, which makes it very useful for debugging.

Now that we've got something in our python file, let's do something with it! Go back to your command line window and type in `python <filename>.py`, of course replacing `<filename>` with whatever you named your file. The output should be identical to this:

A screenshot of a terminal window. The prompt is 'Vedants-MacBook-Pro:Desktop Dant\$'. The command 'python intro_to_python.py' has been entered and executed, resulting in the output 'Hello, world!'. The prompt is now 'Vedants-MacBook-Pro:Desktop Dant\$' followed by a cursor.

If all goes to plan, congratulations! You have successfully created the aptly-named "hello world" file, as made famous by Brian Kernighan and Dennis Ritchie – the creators of C. Otherwise, please see me.

There is more that can be done while running python files, however! You can include command line arguments, which is basically a series of words that follow `python <filename>.py`. The cool thing about command line arguments is that you can do

things with them! We're not far enough in this introduction to be able to do things with them, however we can simply print them for now. There are many ways to handle arguments, such as importing `click` or `argparse`; however, for now, we'll stick to `sys` for now. Hold on. Pause. What do I mean by importing? Let me explain. Certain benevolent python developers have packaged a ton of pre-made code for us to use into libraries. We can gain access to all of that code by simply adding `import library_name` at the beginning of our python programs. If we were to use any of the library's functions after importing it, we would call that function/variable like so: `library_name.function_name(*args)`, or `library_name.variable_name`. For reference, `*args` refers to the necessary parameters for a function.

Back to arguments. In the case of `sys`, in order to access all of the arguments typed in while running the program, we would use its `argv` variable. If you complete assignment 0, you will realize that the arguments are organized in an odd manner. Let me explain. What you see is a list: a python data structure that stores multiple instances of a certain data type. I'll discuss lists in greater detail in chapter 3. Until then, stay patient. Also notice that the first element of `sys.argv` is the filename. Interesting, right? Not really. Just keep that in mind going forward.