Lab 1: Running Programs

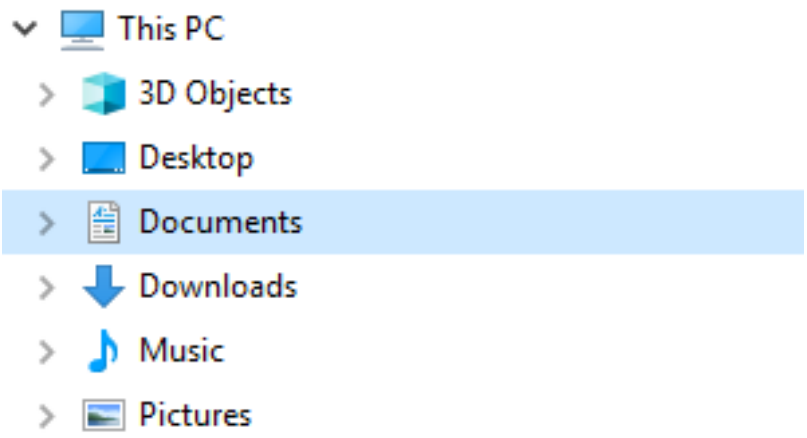
Welcome to your first lab! If you're using a lab computer, you're required to work in pairs (otherwise, we won't have enough machines). If you've already installed Python on your computer, you may still learn more by choosing to work with somebody. This document is meant to be self guiding, and you may leave when you're finished. Be sure to ask a neighbor or flag down a TA if you have any questions, though (don't be shy!).

This semester, you're going to learn how to write your own Python code. But for this lab, you're just going to practice running six Python programs we give you (including a game!).

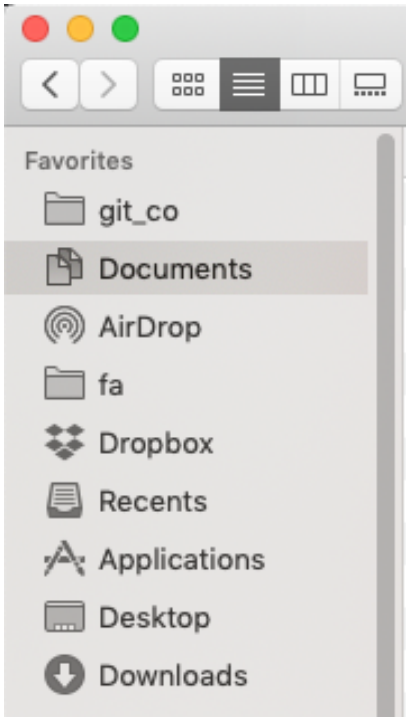
Before you begin, you'll need to activate your CS login here.

Download Your First Program

The first thing you're going to need to decide is where to keep your work this semester. If you don't have a preference, we recommend creating a folder named "cs301" under "Documents". How to find the Documents folder may vary from computer to computer. On a Windows machine, you might find it like this in File Explorer:

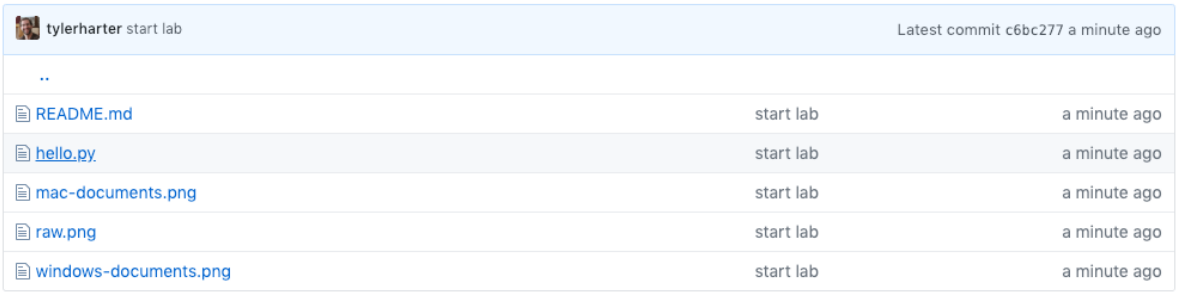


On a Mac, you might find it in Finder here:



Inside the new "cs301" folder you created under "Documents", we recommend you create a sub-folder called "lab1" and use it for all your files related to this lab.

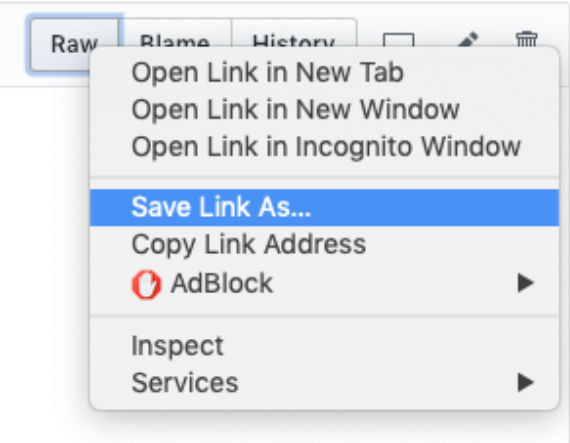
Next, you will need to download a file named "hello.py" to your "lab1" folder. At the top of this page, you'll see a list of files, something like this:



Downloading files from GitHub (the site hosting this document) is a little tricky for those new to it. Follow these steps careful:

1. left-click on "hello.py"
2. right-click on the "Raw" button
3. Choose "Save Link As..." (or similar)
4. Save the file in your "lab1" folder

We recommend you use the Chrome browser (other browsers will work too, but sometimes we've seen Safari automatically renaming files when downloaded, which is usually problematic). In Chrome, right-clicking the "Raw" button looks like this:



Run Your First Program

Now it gets a little tricky. You need to figure out the path of your "lab1" folder. You can think of a "path" as just a more complete name for a file or folder.

1. open your "Documents" in either File Explorer or Finder
2. copy the pathname of "lab1" using either these Windows directions or Mac directions
3. paste the pathname of "lab1" in your notes somewhere

Now, you'll need to open something called a "terminal emulator".

Windows:

1. hit the Windows logo key on your keyboard
2. type "powershell"
3. open "Windows PowerShell" (be careful, DO NOT choose the ones that say "ISE" or "x86")

Mac:

1. open Finder
2. click "Applications"
3. open "Utilities"
4. double-click Terminal.app

Ok, now the directions are the same for Mac and Windows again. Type this in the terminal (replace LAB1-PATH with the pathname of "lab1", as you determined above; keep the quotes around the pathname, though) and hit enter:

```
cd "LAB1-PATH"
```

Type `ls` and hit enter. If you've done everything correctly so far, you should see the "hello.py" file that you downloaded in step 1 listed.

Now, type `python hello.py` and hit ENTER (if you're not using a lab computer, you may need to instead type `python3 hello.py`, depending on your setup). If everything is working correctly, you should see the following message printed:

```
Hello, World!
```

Congrats! The above is the trickiest part of the lab. If there other students near you who are struggling to get this far, please take a minute and show them what you did (this helps if the TAs are swamped with questions).

Reading Code

Before we move onto the next program, type `cat hello.py` and hit ENTER. You should see the following:

```
print("Hello, World!")
```

What you're looking at is the code for the hello.py program. Just one line! The line is telling Python to print something to the screen, specifically whatever is inside the parentheses. In this case, we want to print some text, and text must be enclosed in quotes. Printing a different message would be as simple as putting that message between the quotes.

Feel free to use `cat` in the following steps to view the code of other programs. For example, after downloading the second program, you can view it by typing `cat double.py` and hitting ENTER. You aren't expected to understand what the code you view in this lab means, yet, but it's helpful to get some early exposure to what Python programs look like.

Program 2: Double

Repeat the steps above that you took to download "hello.py", but this time download the "double.py" file instead. Make sure that:

1. you download it to the same "lab1" folder as before
2. you still have the terminal window open where you typed `python hello.py` before

Type `ls` and hit enter. If you did everything correctly, you should see "double.py" listed along with "hello.py". If you don't, try again (carefully following the instructions) or ask a TA for help.

Type the following and hit ENTER:

```
python double.py
```

The program will say `please enter a number:`. This is known as a "prompt" (a fancy way to say a program is asking you a question).

Type `5` and hit ENTER. Make sure that the programs tells you the answer is `10.0`.

Find the up arrow key on your keyboard and hit it. What you last typed (i.e., the command to run double.py) should show up again. Hit ENTER again to run it a second time, and this time try typing a negative number.

Ok, let's run double.py one last time, but now when you're prompted for a number, type the word "five" and hit ENTER. Woah, a bunch of weird stuff is printed! Something like this:

```
Traceback (most recent call last):
  File "double.py", line 2, in <module>
    print("2 times your number is " + str(2*float(x)))
ValueError: could not convert string to float: 'five'
```

When you see the word "Traceback", it means the program crashed. The double.py program can only take numbers as digits (so "5" but not "five"), so it crashed when you typed something else. Eventually, we'll learn how to understand what gets printed when a program crashes to identify the root of the problem, but for now we won't worry about it any further.

Program 3: Double (Version 2)

Ok, now download the program named "double-v2.py" and run it like this:

```
python double-v2.py 5
```

Notice that `double-v2.py` does the same thing as the `double.py` program, but it gets the original number in a different way. Before, we typed "5" in answer to a prompt, but now we are specifying "5" at the same time we run our program.

Discussion: do you prefer using double.py or double-v2.py? Discuss with the person you are doing this lab with (or your neighbor).

Before continuing to the next program, try running it a couple more times to note what happens:

```
python double-v2.py 50
```

And this, noting the space between "5" and "0":

```
python double-v2.py 5 0
```

Finally, try running it without specifying any number:

```
python double-v2.py
```

Program 4: Fruit Chooser

Download "fruit.py" and run it:

`python fruit.py`

Keep running it, trying different numbers until it prints the following:

```
You chose mango
```

Other interesting things you should try: what happens if you enter a large number, such as 100? What if you enter a negative number, such as -1?

Program 5: Echo

Download "echo.py" and run it like this:

```
python echo.py 2 hello world
```

Can you find another way to run echo.py that makes it print the following?

```
ha
ha
ha
ha
ha
```

Here are three other interesting ways to run echo.py that you should try:

```
python echo.py    2    hello    world
```

```
python echo.py 5
```

```
python echo.py
```

Program 6: Take-Away Game

Download "take.py" and run it like this a few times:

```
python take.py
```

Here are a few things to ponder/try about this program:

1. can you beat the program?
2. what happens if you try to cheat?
3. can you exit before losing by typing "q"? What about a capital "Q"?

Congrats on finishing your first CS 301 lab!