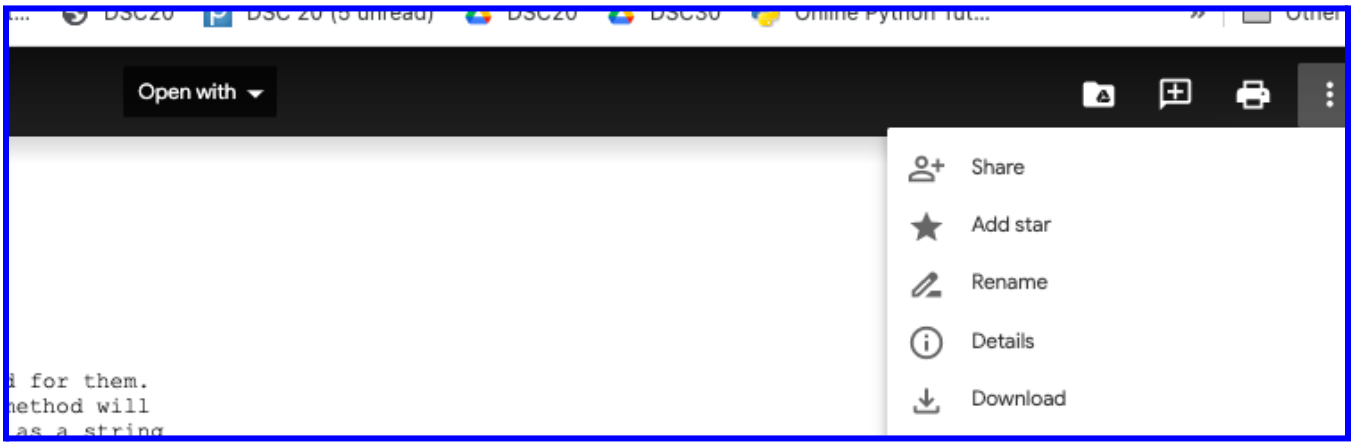
Lab 00: Setting up a productive workflow

Due: Tuesday, January 16th, 11:59pm

This lab is not graded. It just provides instructions on how to properly set up your environment. You must submit it for other assignments to be graded.

Starter File

Make a folder named `DSC20` and then subfolder `Labs` on your *Desktop*. Download lab00.py (right click on three dots, top right corner)



and save it to the newly created folder `Labs`. **Note**: You can change the names and the location of your folders later but I will use these names in the writeup.

Submission

By the end of this lab, you should have submitted the lab via Gradescope. You may submit more than once before the deadline; only the final submission will be graded.

Setup

Install a terminal

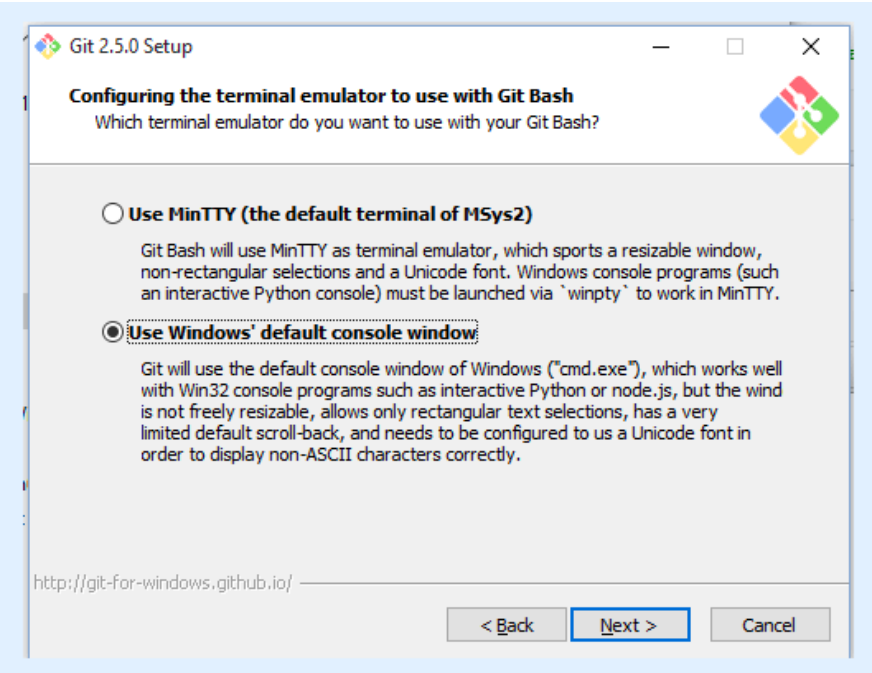
The terminal is a program that allows you to interact with your computer by entering commands. No matter what operating system you use (Windows, macOS, Linux), the terminal will be an essential tool for DSC 20.

If you're on a Mac or are using a form of Linux (such as Ubuntu), you already have a program called Terminal or something similar on your computer. Open that up and you should be good to go. Another popular option for Mac is Iterm2, an open-source modernized alternative to terminal.

For Windows users, we recommend downloading a terminal called Git Bash.

You should be able to install Git Bash with most of the default configuration options, with one exception. In the *Configuring the terminal emulator to use with Git Bash* step, select the second option: *Use Windows' default console window*.

This is very important! If you do not select this option, your terminal will not work!

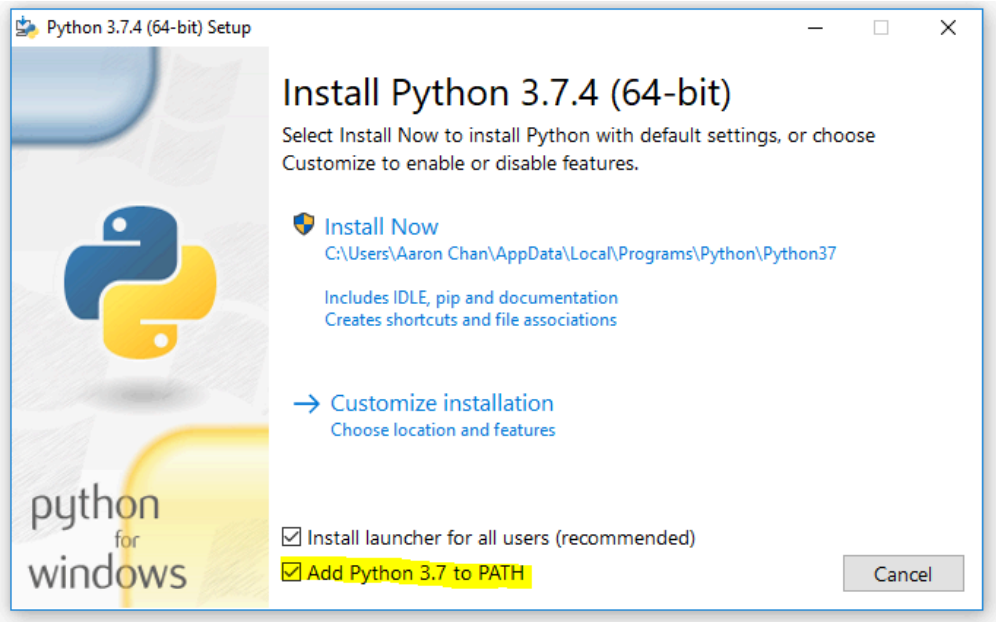


Install Python 3

Python 3 (3.12 specifically) is our primary programming language.

- **macOS users: Installer**
  - You may need to right-click the download icon and select "Open".
- **Windows users: Installer**

  Important: make sure to check the "Add Python to PATH" box, which will allow you to execute the python command from your terminal. (Look below for the example, although it is for 3.7)



These methods are assuming you have the latest compatible OS. You can also look at https://www.python.org/downloads/ for other installers.

Install a text editor

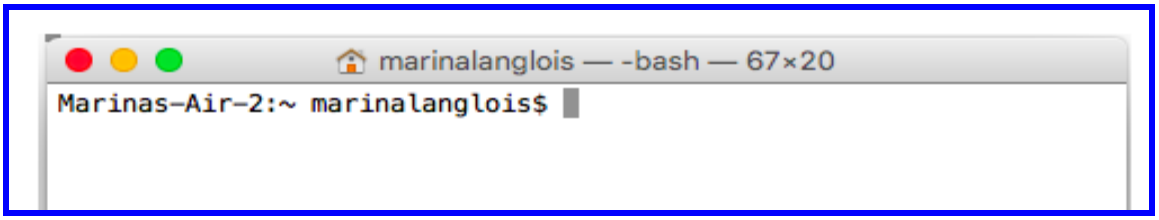
The **Python interpreter** that you installed earlier allows you to *run* Python code. You will also need a **text editor**, which will help you *write* Python code.
There are many editors out there, each with its own set of features. We find that Sublime Text 4 is a popular choice among students. You are free to use other text editors but you will be responsible for understanding how it works. **Make sure that you compile your files using a terminal since it is a useful practice for the future.**

***Note**: Please, please, please do not use word processors such as Microsoft Word to edit programs. **Also, we do not recommend using a Jupyter notebook** in this course. Please practice using text editors, terminals, and doctests.*

For your reference, we've also written some guides on using popular text editors. After you're done with the lab, you can find them on the course website (under Links and Resources section).

Open a terminal

First, open a terminal, if you haven't already.
When you first open your terminal, you will start in the home directory. The **home directory** is represented by the ~ symbol.



Don't worry if your terminal window doesn't look exactly the same; the important part is that the text on the left-hand side of the $ has a ~ (tilde). That text might also have the name of your computer.

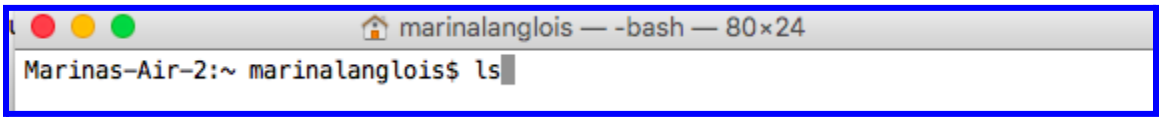
Organize your files

Now you will learn how to find downloaded files, run Python and test your code using terminal commands.

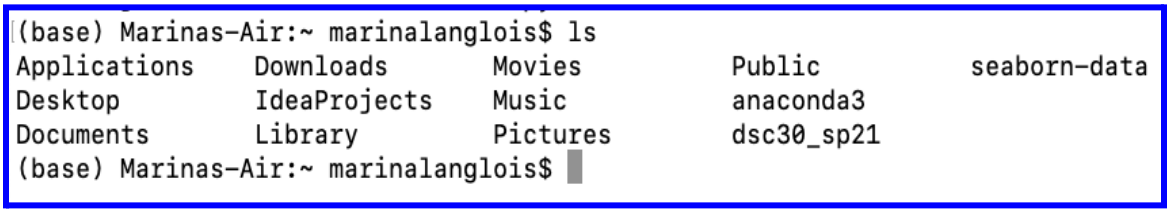
At this point we assume that your file is at `Desktop->DSC20->Labs` location. Let's find this file using a few command lines.

The first command you'll use is ***ls***. The *ls* command **l**i**s**ts all the files and folders in the current directory. A **directory** is another name for a folder (such as the *Documents* folder). Since you're in the home directory right now, you should see the contents of your home directory.

Try typing it in the terminal:



This is the content of my home directory:



As you can see, one of the folders is `Desktop`, this is where I saved my lab file, therefore we need to go inside that directory.

Changing directories

To move into another directory, use the ***cd*** command. The `cd` command will **c**hange **d**irectories, moving you into the specified folder.

Try typing the following command into your terminal and hit Enter:

```
(base) Marinas-Air:~ marinalanglois$ cd Desktop/
```

You will see that that you location has changed to `Desktop`:

```
(base) Marinas-Air:Desktop marinalanglois$
```

There's a few ways to return to the *home* directory in case you want to go back:

- *cd ..* (two dots). The .. means "the parent directory". In this case, the parent directory of Desktop is your home directory, so you can use *cd ..* to go up one directory.
- *cd ~* (the tilde). Remember that ~ means home directory, so this command will always change to your home directory.
- *cd* (cd on its own). Typing just cd is a shortcut for typing *cd ~*

You can type `ls` again just see what files and folders you have on your `Desktop`. One of the folders should be DSC20. We need to go there to get to lab00.py How? Using `cd` command again!

```
(base) Marinas-Air:Desktop marinalanglois$ cd DSC20
(base) Marinas-Air:DSC20 marinalanglois$
```

Almost there! Can check where you are with `ls` again and then move to the `Labs` folder.

```
(base) Marinas-Air:DSC20 marinalanglois$ ls
Labs
(base) Marinas-Air:DSC20 marinalanglois$ cd Labs
(base) Marinas-Air:Labs marinalanglois$ ls
lab00.py
(base) Marinas-Air:Labs marinalanglois$
```

Finally, you're ready to start editing the lab files! Don't worry if this seems complicated -- it will get much easier over time. Just keep practicing!

If you ever forget what a terminal command does, add `--help` to the end of the command:

```
ls --help
```

This will bring up info on the command, such as usage and arguments.

Unix Commands Summary

- *ls*: Lists all the files and folders in the current directory.
- *cd*: Move into another directory.

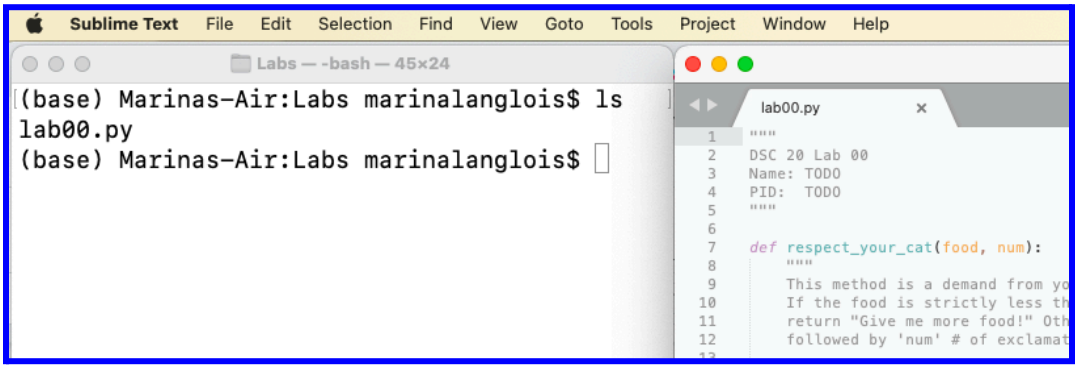
Note: You do not need to use ls each time. Once you get comfortable, you can put all folders in one line, like this:

```
(base) Marinas-Air:~ marinalanglois$ cd Desktop/DSC20/Labs
```

Important (Windows Users): Desktop might not be a folder in your home directory. If that is the case, try finding it under the *OneDrive/* folder. Use *ls* and *cd* to navigate there accordingly.

Opening, Editing starter file

Now open the editor and find your saved file at the SAME location: `Desktop->DSC20->Labs`. You may open `lab00.py` in your text editor by right clicking on the file and selecting "Open With" > your text editor. You may set your default application of .py files to your text editors to save time in the future as well.



Understand the question

Problem to solve:

Write a Python program to generate a demand for food from your cat. If the `demand` message (a string) is strictly less than 5 characters long, return the string `"Give me more food!"`. Otherwise add `num` number of exclamation marks at the end of the demand and return it; You may assume `num` is a nonnegative integer.

Hint: In Python you could multiply a string by an integer to repeat the string multiple times, and use "+" to concatenate strings.

After opening the file, you should see something like this:

```
def respect_your_cat(food, num):
    """
    This method is a demand from your cats to provide food for them.
    If the food is strictly less than 5 characters, this method will
    return "Give me more food!" Otherwise return the food as a string
    followed by 'num' # of exclamation marks.

    >>> respect_your_cat("mud", 5)
    'Give me more food!'
    >>> respect_your_cat("lots of fish and lots of fish", 4)
    'lots of fish and lots of fish!!!!'
    >>> respect_your_cat("Fruits", 0)
    'Fruits'
    """
    # YOUR CODE GOES HERE #
    return
```

The lines in the triple-quotes `"""` are called a **docstring**, which is a description of what the function is supposed to do. When writing code in DSC20, you should always read/provide the docstring!

The lines that begin with `>>>` are called **doctests**. Doctests explain what the function does by showing actual Python code: "if we input this Python code (the lines that say `>>>`), what should the expected output be (the lines underneath the `>>>`)."

In *respect_your_cat*,

- The docstring summarizes the method "generate a demand for food from your cat by adding exclamation marks or returning …"
- The doctest for *respect_your_cat("Fruits", 0)* checks that *respect_your_cat()* returns the expected string, which is provided on the next line.

Write code

Once you understand what the question is asking, you're ready to start writing code! You should replace the
`# YOUR CODE GOES HERE #` and `return` with your code. Do not forget to save it.

Note:
The `return` (sometimes it comes with `pass`) here is just a placeholder. When the code only comes with a `#` comment instead of the `"""` docstring, Python will also complain about some indentation error. So we just provided the return line to make sure the starter code can run without syntax errors. Feel free to replace it with anything, as you don't need it in your submission.

Run tests

Back to the terminal! Make sure you are in the Labs directory we created earlier (remember, the `cd` command lets you change directories).

In that directory, you can type `ls` to verify that correct file is there:

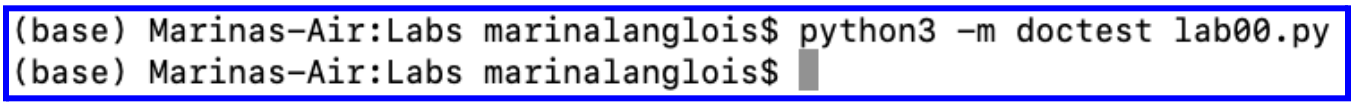
- lab00.py: the starter file you just edited

Now, let's test our code to make sure it works:

Run the following command:

`python3 -m doctest lab00.py`

If you wrote your code correctly, you should see that nothing is showing. This indicates a successful test.



Note: If the `python3` command doesn't work, try using just `python` or `py` (`py -m doctest lab00.py`). If neither of those work, take another look at the install Python 3 section to make sure you are setting up your PATH correctly. Ask for help if you get stuck!

Important (Windows Users): You will most likely find `python` or `py` as your working command, please try these 2 first. If you see `py -m doctest lab00.py` gives you nothing, please try again using `py -m doctest lab00.py -v` (let the terminal show all doctests run, both failed and passed). The following is expected:

```
$ py -m doctest lab00.py -v
2 items had no tests:
    lab00
    lab00.respect_your_cat
0 tests in 2 items.
0 passed and 0 failed.
Test passed.
```

If this command also gives you nothing, this means `py` is NOT the correct command to use, and please try again using `python`

If you didn't pass the test, you will see something like this:

```
(base) Marinas-Air:Labs marinalanglois$ python3 -m doctest lab00.py
**********************************************************************
File "/Users/marinalanglois/Desktop/DSC20/Labs/lab00.py", line 14, in lab00.respect_your_cat
Failed example:
    respect_your_cat("dog", 5)
Expected:
    'Give me more food!'
Got nothing
**********************************************************************
File "/Users/marinalanglois/Desktop/DSC20/Labs/lab00.py", line 16, in lab00.respect_your_cat
Failed example:
    respect_your_cat("lots of fish and lots of fish", 4)
Expected:
    'lots of fish and lots of fish!!!!'
Got nothing
**********************************************************************
File "/Users/marinalanglois/Desktop/DSC20/Labs/lab00.py", line 18, in lab00.respect_your_cat
Failed example:
    respect_your_cat("Fruits", 0)
Expected:
    'Fruits'
Got nothing
**********************************************************************
1 items had failures:
   3 of   3 in lab00.respect_your_cat
***Test Failed*** 3 failures.
(base) Marinas-Air:Labs marinalanglois$
```

Fix your code in your text editor, save it and try again until the test passes.

You will find it useful to write some of your own tests in the form of doctests. Then, you can try them out using the command above. Remember, you should thoroughly test your implementation, as our autograder tests will be comprehensive.

Submit the assignment

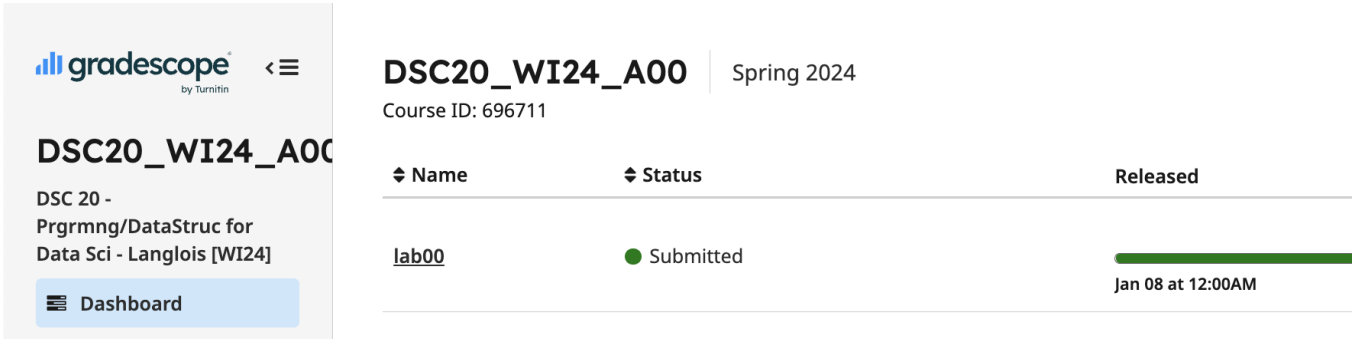
Now that you have completed your first DSC20 assignment, it's time to turn it in.

1) Login to Gradescope

You should already be enrolled into Gradescope (check your email). If you're not enrolled, try to click on the Gradescope tab on the left side of your Canvas course page. If that does not work, please email course staff or make an Edstem post. Login to gradescope.com and select our course, DSC20.

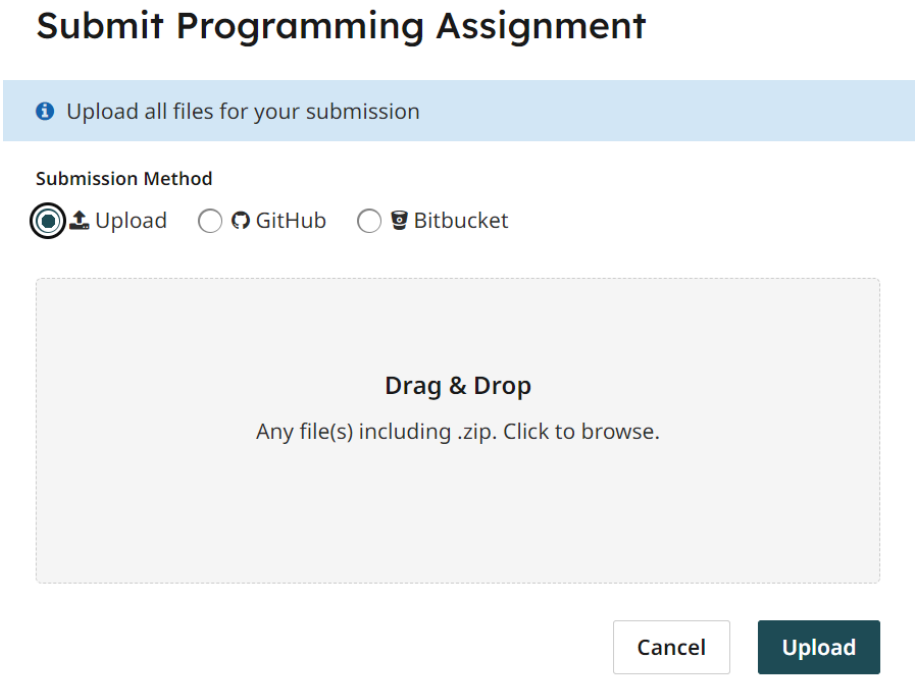
2) Select Correct Assignment

On the class dashboard, select the current assignment. For this assignment, select lab00.

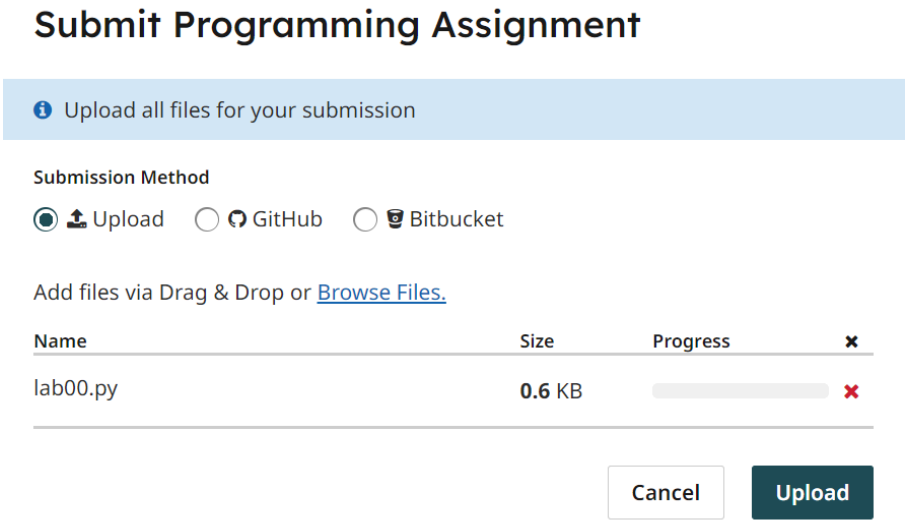


3) Upload your files

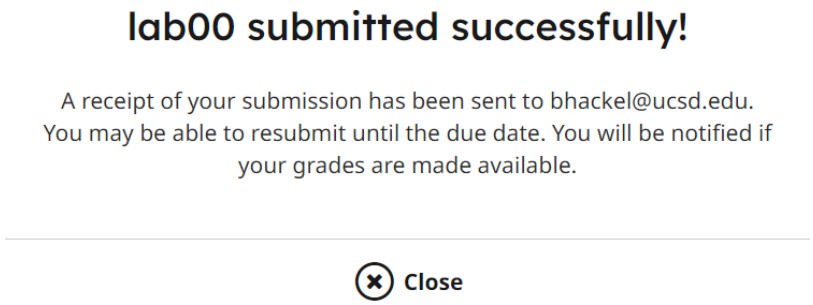
Click on **DRAG & DROP** and choose your file(s).



Click on the **Upload** button.



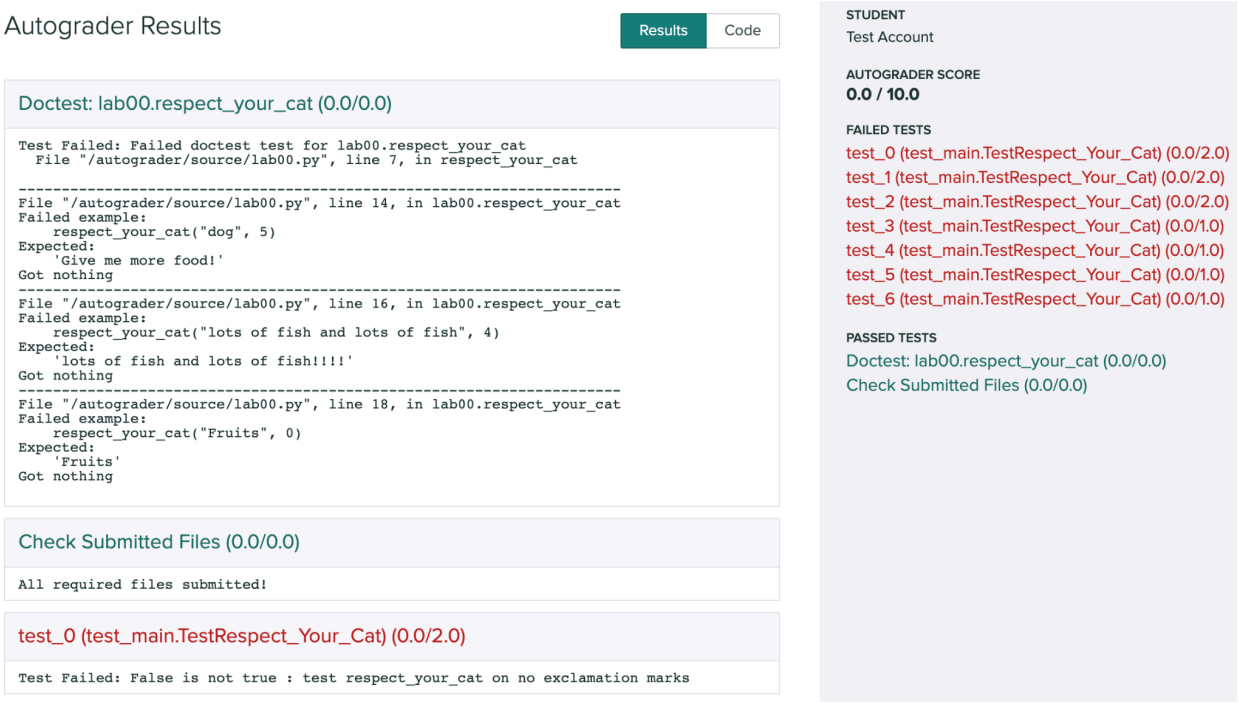
4) Wait for the process to complete.



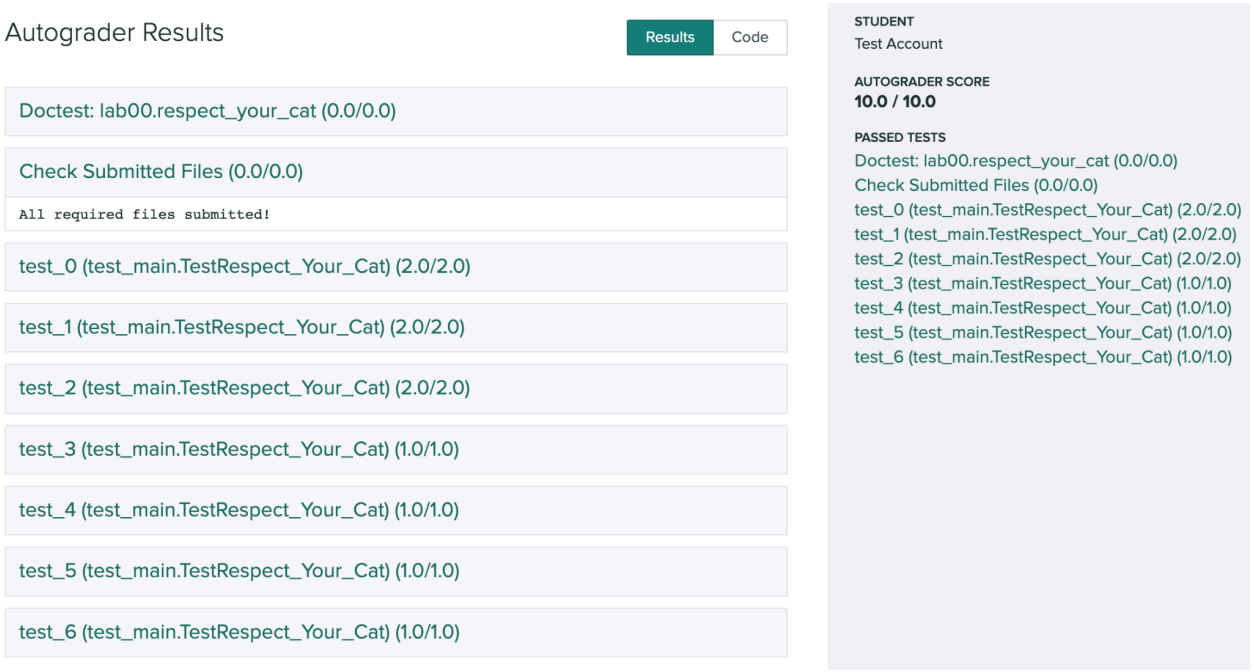
5) Make sure all given doctests passed

If you see that at least one test is in RED, it means that at least one of the doctests is broken. In this case, fix your code and resubmit.

Example of a submission with failed tests:



Example of a successful submission:



You **should** see a successful submission for lab00.

Note: For future assignments, do not worry about "-/10.0" or "-/100.0", missing grade. You will see your score when the grades are released. You should pay attention to what the Autograder states and ensure that the tests properly run, but for other assignments, you will typically not be able to see the scores of the Autograder tests immediately after submission.

Congratulations, you just submitted your first DSC 20 assignment!

Appendix: Useful Python Command Line Options

When running a Python file, you can use **options** on the command line to inspect your code further. Here are a few that will come in handy. If you want to learn more about other Python command-line options, take a look at the documentation.

- Using no command-line options will run the code in the file you provide and return you to the command line.
    - `python3 lab00.py`

- **-i**: The -i option runs your Python script, then opens an interactive session. In an interactive session, you run Python code line by line and get immediate feedback instead of running an entire file all at once. To exit, type `exit()` into the interpreter prompt. You can also use the keyboard shortcut `Ctrl-D` on Linux/Mac machines or `Ctrl-Z Enter` on Windows.
    - If you edit the Python file while running it interactively, you will need to exit and restart the interpreter in order for those changes to take effect.
    - `python3 -i lab00.py`

- ***-m doctest***: Runs doctests in a particular file. Doctests are surrounded by triple quotes (""") within functions. Each test consists of >>> followed by some Python code and the expected output.
    - `python3 -m doctest lab00.py`