Lab 0: Getting Started (Windows Setup)

Setup

Terminal

PowerShell:

The terminal is a program that allows you to interact with your computer by entering commands.

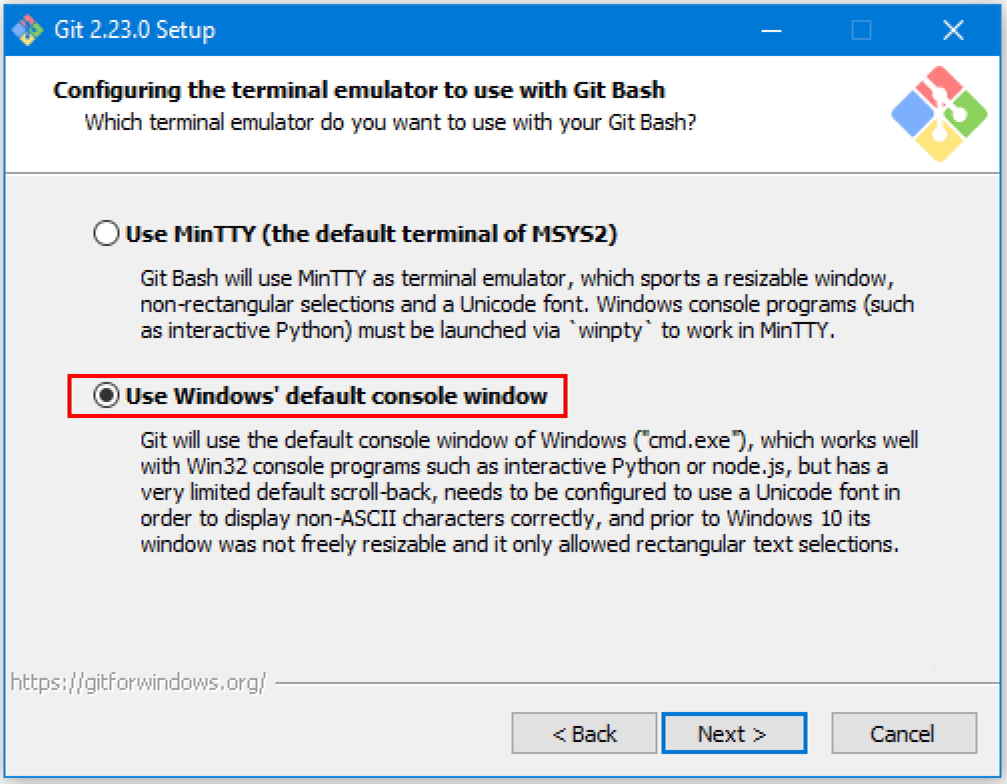
You already have a program called `Windows PowerShell`. Open that up and **you should be good to go!** Simple commands like `cd` and `ls` will work (`python` will work after the setup).

[OPTIONAL ALTERNATIVE] Git Bash:

You can also try and download a terminal called Git Bash (https://git-scm.com/downloads), which has a bit more utilities than `PowerShell` and may be required for future courses.

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 Git Bash terminal will not work!



Configuring the terminal emulator to use with Git Bash

Which terminal emulator do you want to use with your Git Bash?

○ **Use MinTTY (the default terminal of MSYS2)**

Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

● **Use Windows' default console window**

Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

Python

Python 3 is the primary programming language used in this course. If you have an older version of Python installed, please make sure to download and install Python 3.9. You can check your Python version with `python3 --version`.

If you'll be using PowerShell, open the Microsoft Store and search for "Python." Install Python 3.9 by the Python Software Foundation (this should be the first result).

Verify: We can use the terminal to check if your Python interpreter was installed correctly. Try the following command:

```
python3
```

If the installation worked, you should see some text printed out about the interpreter followed by `>>>` on its own line. This is where you can type in Python code. Try typing some expressions you saw in lecture, or just play around to see what happens! You can type `exit()` or `Ctrl-Z` and press enter to return to your command line.

> If the `python3` command doesn't work, try using `python` or `py`.

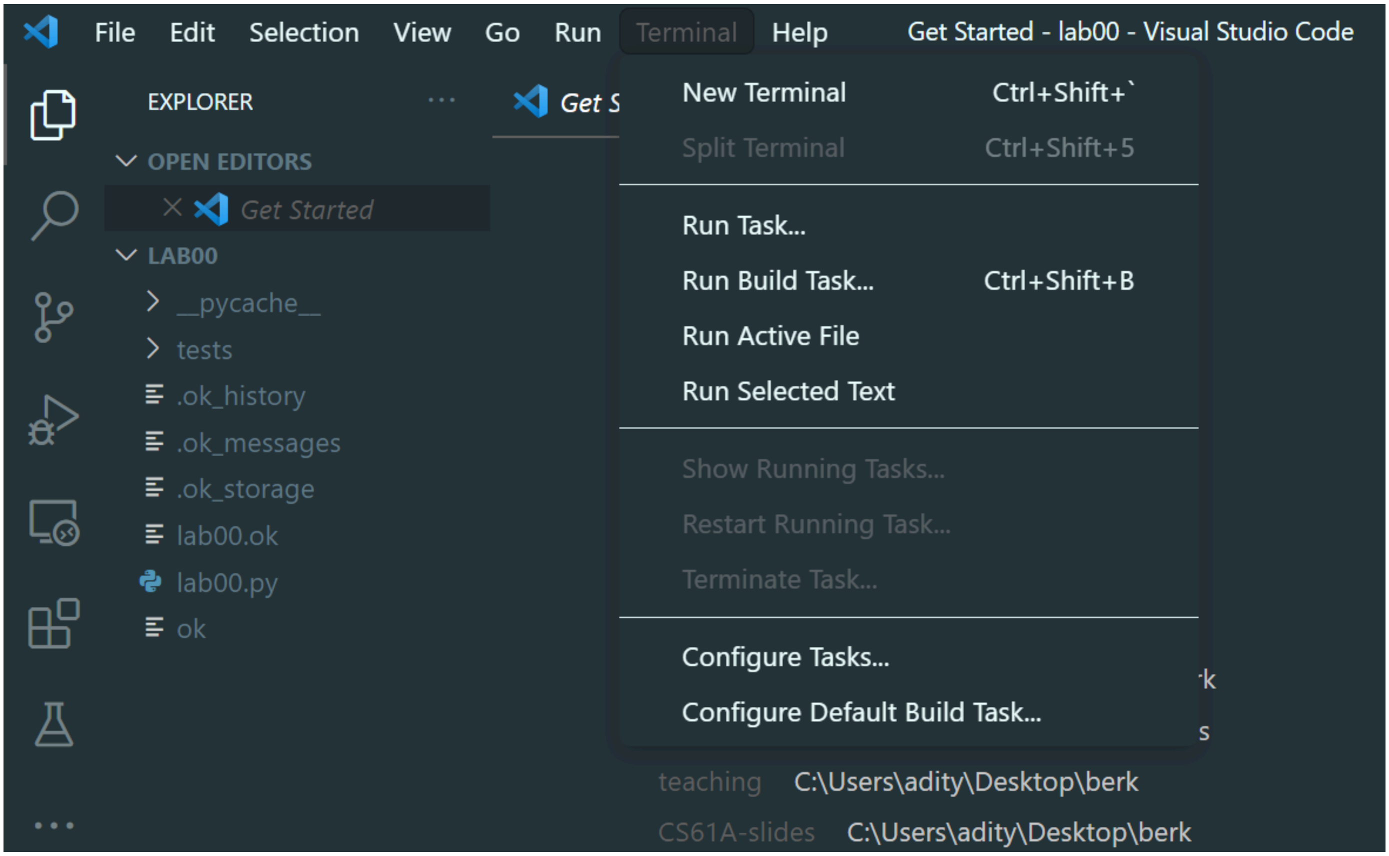
Text Editor

The **Python interpreter** that you just installed allows you to *run* Python code. You will also need a **text editor**, where you will *write* Python code.

Visual Studio Code (VS Code) (https://code.visualstudio.com/) is the most popular choice among the staff for this course for writing Python. After installing it, you may find our Visual Studio Code (vscode) helpful.

We highly recommend using VS Code for this class. This will help us support you best since most of staff uses VS Code. Please do not use word processors such as Microsoft Word to edit programs. Word processors can add extra content to documents that will confuse the interpreter.

You can open a terminal directly on VS Code. Thus, when running terminal commands, you can manage everything in VS Code rather than navigating back and forth between VS Code and a separate terminal application. You can open an embedded terminal by going to `Terminal > New Terminal` in VS Code's navigation bar.



Other Text Editors

For your reference, we've written some guides on other popular text editors:

- Vim (vim): A command-line editor.
- Emacs (emacs): A command-line editor.

Some students also use:

- PyCharm (https://www.jetbrains.com/pycharm/): A desktop editor designed for Python.

Pair Programming

Throughout this course, you'll have many chances to collaboratively code in labs and projects. We recommend you download these pair programming extensions now to use in the future.

For sharing code, you can follow the instructions for your editor:

- VS Code (vscode#pair-programming)

Walkthroughs & Reviews

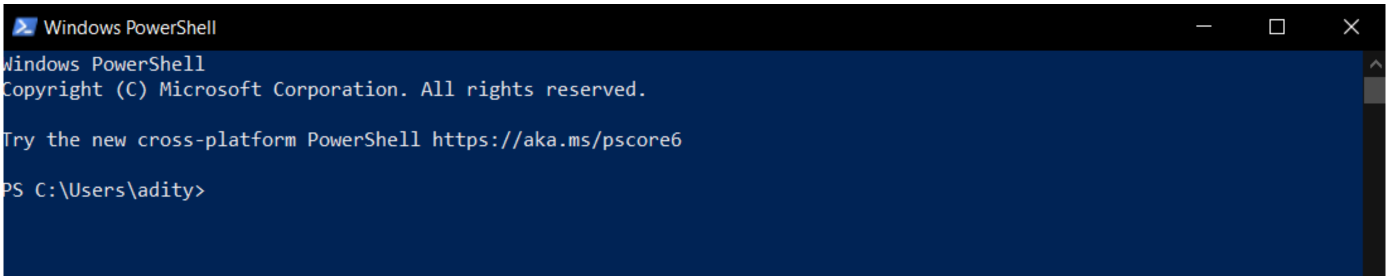
Walkthrough: Using the Terminal

First, open a Powershell/Bash window.

Path

A path is like an address: it tells both you and the computer where to find a particular file or folder (also called a "directory"). For example, `C:\Users\adity\Desktop\berk\courses\cs` is a path that references a particular folder on your computer: the folder `cs`, which is contained in the folder `courses`, which is contained in the folder `berk`, etc.

Your terminal is always open in some folder, which is called the "working directory." The path to the working directory is visible next to `PS` in PowerShell. In the below screenshot of PowerShell, the working directory is `C:\Users\adity`.

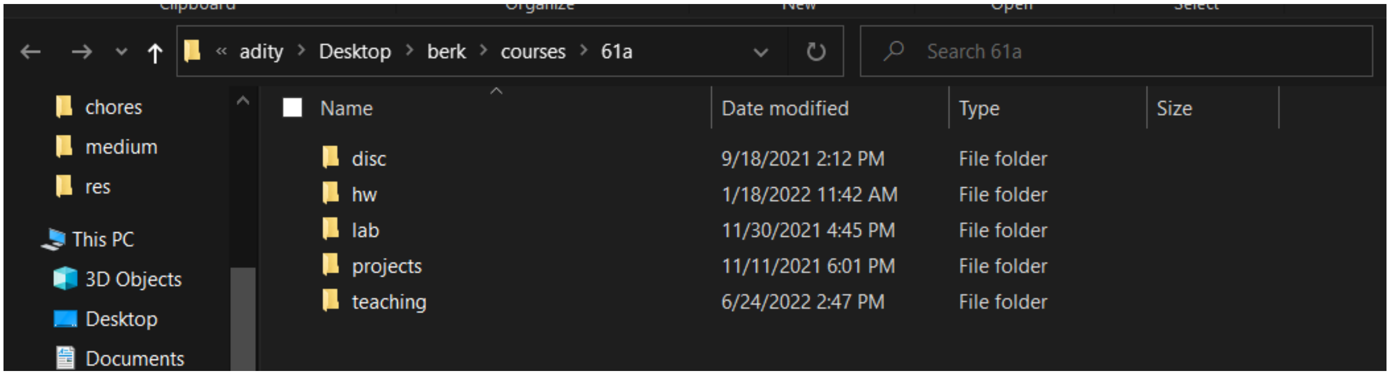


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\adity>
```

Home Directory

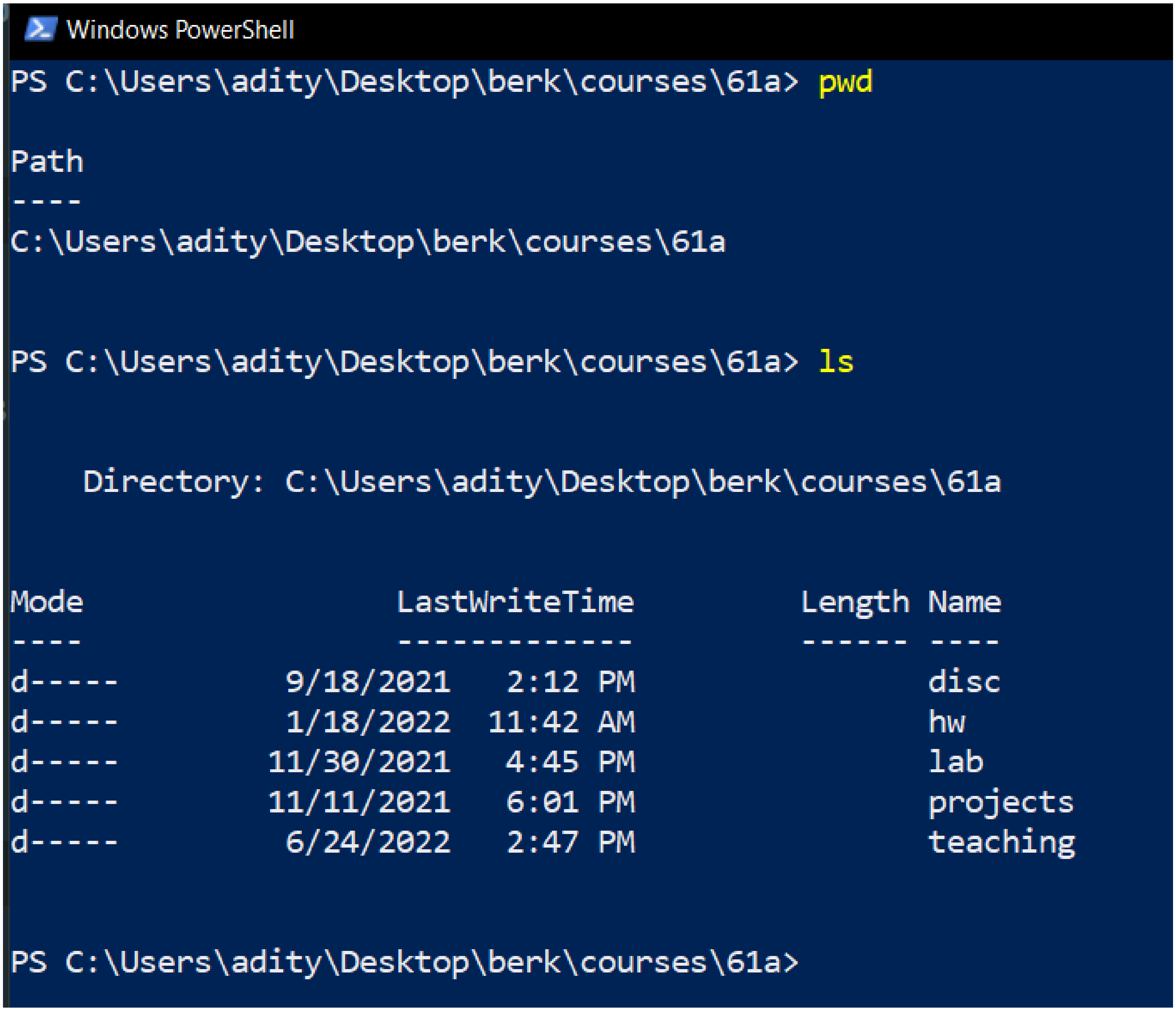
When you first open your terminal, you will start in the "home directory." This is the default working directory for your terminal.

CLI vs GUI Remember that you can access the files and directories (folders) on your computer in two different ways. You can either use the terminal (which is a **c**ommand **l**ine **i**nterface. or CLI) or you can use File Explorer. File Explorer is an example of a **g**raphical **u**ser **i**nterface (GUI). The techniques for navigating are different, but the files are the same. For example, here's how my folder for this course looks in my GUI:



Name	Date modified	Type	Size
disc	9/18/2021 2:12 PM	File folder	
hw	1/18/2022 11:42 AM	File folder	
lab	11/30/2021 4:45 PM	File folder	
projects	11/11/2021 6:01 PM	File folder	
teaching	6/24/2022 2:47 PM	File folder	

And here's how the exact same folder looks in terminal:

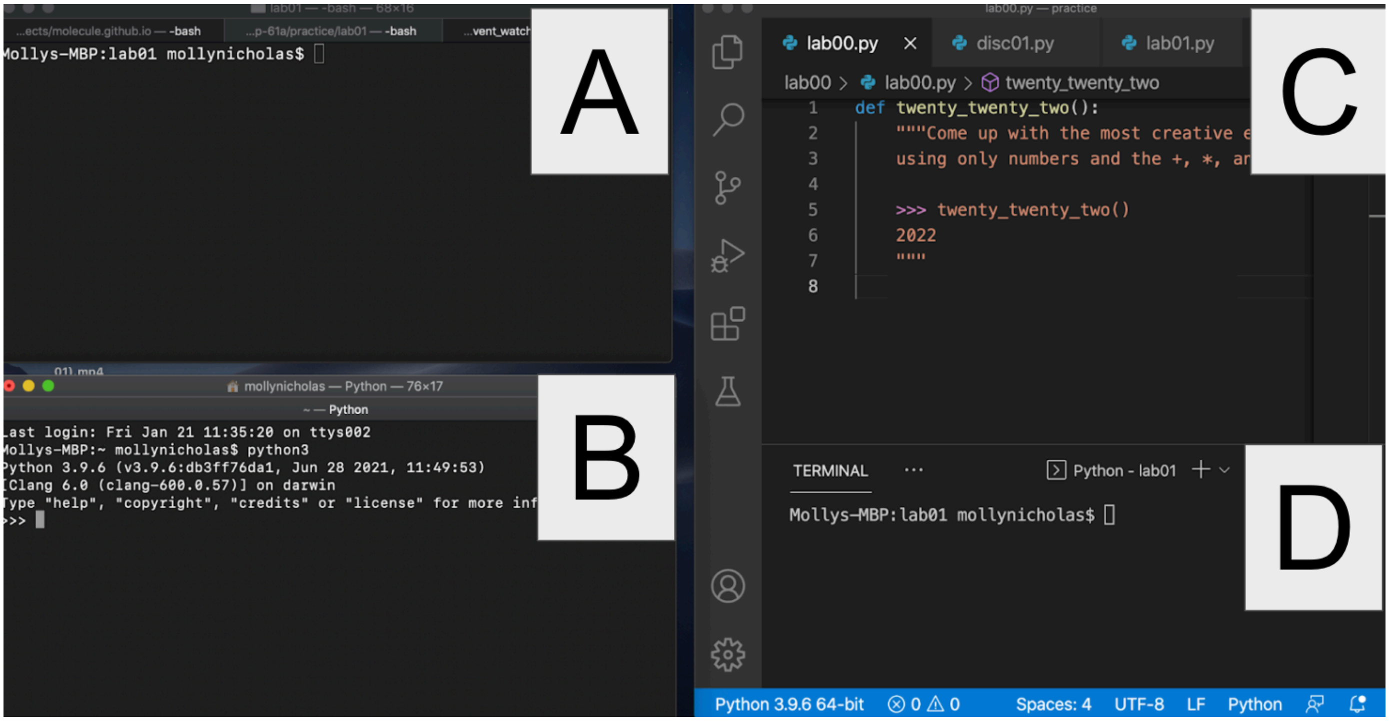


```
PS C:\Users\adity\Desktop\berk\courses\61a> pwd

Path
----
C:\Users\adity\Desktop\berk\courses\61a


PS C:\Users\adity\Desktop\berk\courses\61a> ls


    Directory: C:\Users\adity\Desktop\berk\courses\61a


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        9/18/2021   2:12 PM                disc
d-----        1/18/2022  11:42 AM                hw
d-----       11/30/2021   4:45 PM                lab
d-----       11/11/2021   6:01 PM                projects
d-----        6/24/2022   2:47 PM                teaching


PS C:\Users\adity\Desktop\berk\courses\61a>
```

The contents of the "61a" folder can be found listed in the GUI, and underneath the yellow `ls` command in PowerShell

Terminal vs Python Interpreter

Let's pause and think about the difference between the terminal and the Python interpreter.



```
Mollys-MBP:lab01 mollynicholas$
```

```
Last login: Fri Jan 21 11:35:20 on ttys002
Mollys-MBP:~ mollynicholas$ python3
Python 3.9.6 (v3.9.6:db3ff76da1, Jun 28 2021, 11:49:53)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more inf
>>>
```

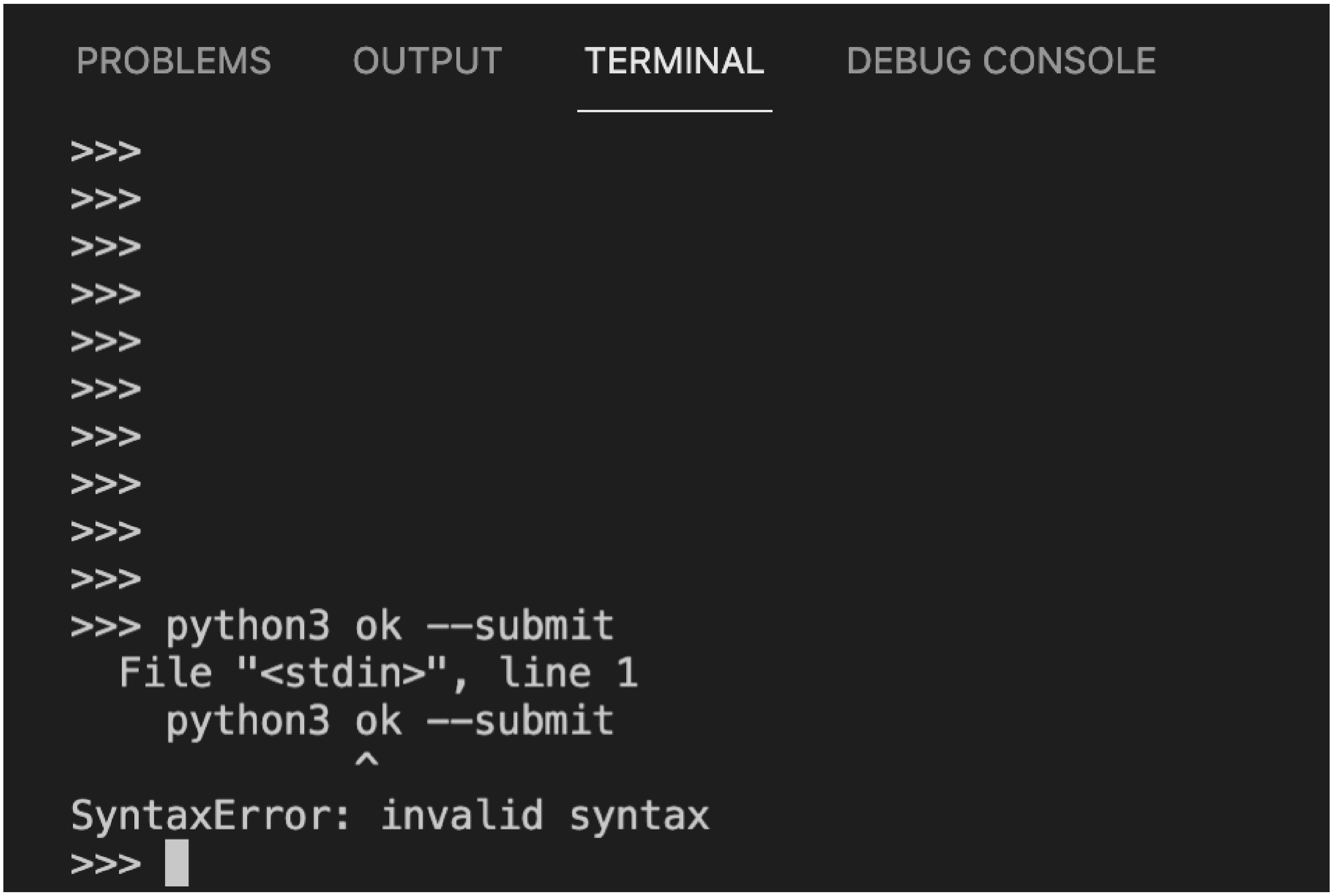
```
def twenty_twenty_two():
    """Come up with the most creative e
    using only numbers and the +, *, ar

    >>> twenty_twenty_two()
    2022
    """
```

```
Mollys-MBP:lab01 mollynicholas$
```

1. Which is the terminal?
2. Which one is the Python interpreter?
3. Which one is my code editor?
4. And how can you tell?

Both A and D are my terminal. This is where you can run bash commands like `cd` and `ls`. D is the terminal that is built-in to VS Code.

B is the Python interpreter. You can tell because of the `>>>` prompt that means you've started a Python interpreter. You can also tell because the command that started it is visible: `python3`. The `python3` command launches a Python interpreter. If you type an ordinary command into the Python interpreter, you'll probably get a syntax error! Here's an example:



```
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>> python3 ok --submit
  File "<stdin>", line 1
    python3 ok --submit
            ^
SyntaxError: invalid syntax
>>> 
```

C is my code editor. This is where I can write Python code to be executed via my terminal.

Walkthrough: Organizing your Files

In this section, you will learn how to manage files using terminal commands.

> Make sure your prompt does not begin with `>>>`. If it begins with `>>>` you are still in a Python shell, and you need to exit. See above for how.

Directories

The first command you'll use is `ls`. Try typing it in your terminal:

```
ls
```

The `ls` command (which stands for **lis**t) lists 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, after you type `ls` you should see the contents of your home directory.

Changing Directories

To move into another directory, use the `cd` command (which stands for **c**hange **d**irectory).

Let's try moving into your `Desktop` directory. First, make sure you're in your home directory by typing:

```
cd ~
```

Then, use the `ls` command to see if the `Desktop` directory is present.

If you see the `Desktop` directory, you can move into it with the following command:

```
cd Desktop
```

However, if the `Desktop` directory is not listed, it might be located elsewhere on your system. This can happen due to system configurations or custom setups. In that case, you'll need to find the correct location of your `Desktop` folder. Most of the time, the `Desktop` folder will be located in the `OneDrive` directory. So, you can try the following command:

```
cd OneDrive\Desktop
```

Making New Directories

Once you're in the `Desktop` directory, the next command you'll use is `mkdir`, which stands for **m**a**k**e a new **dir**ectory. Let's make a directory called `cs61a` in your `Desktop` directory to store all of the assignments for this class:

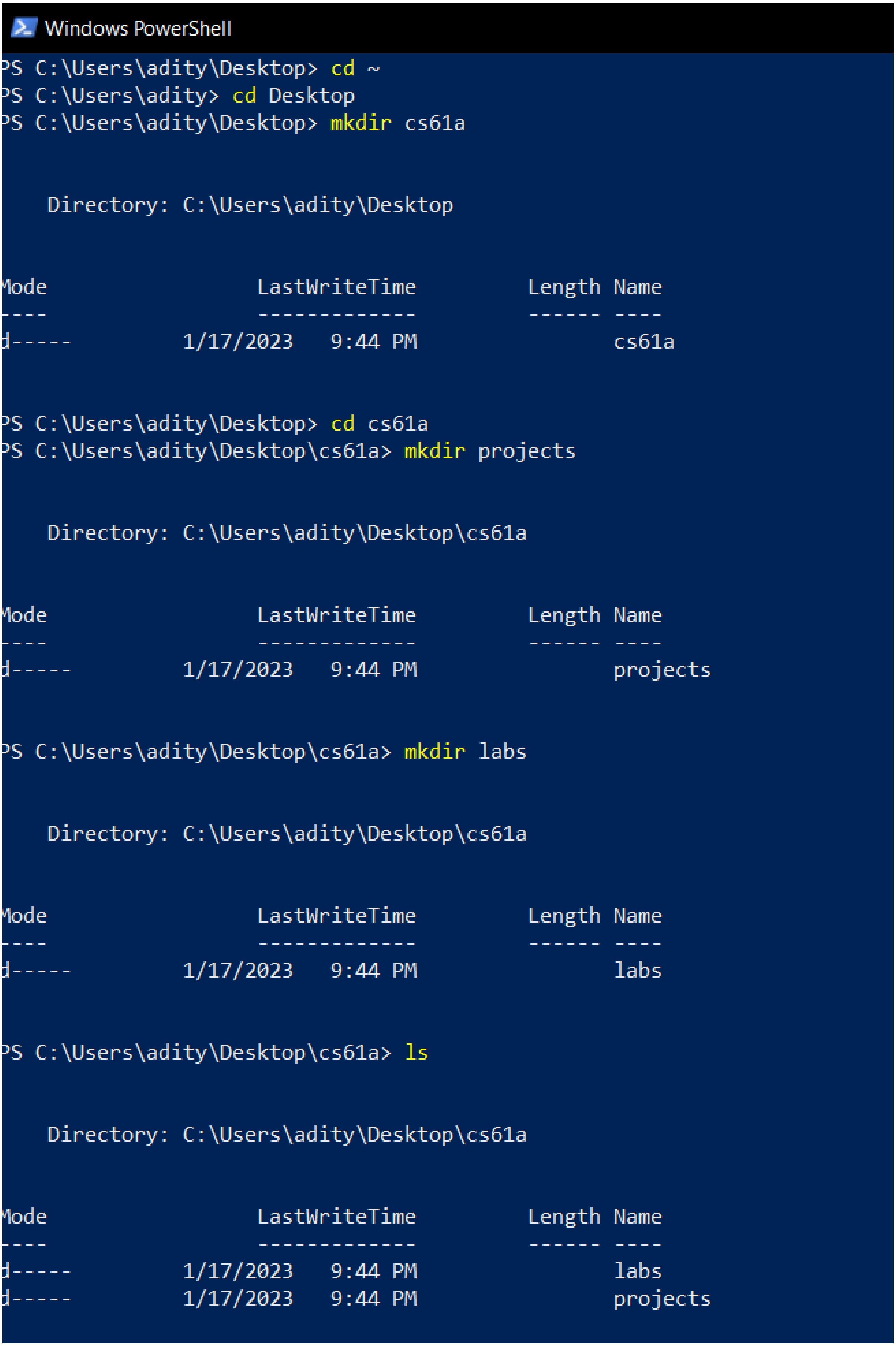
```
mkdir cs61a
```

A folder named `cs61a` will appear on your Desktop. You can verify this by using the `ls` command again or by checking your Desktop using File Explorer.

Now, let's create some more directories. First, make sure you are in the `cs61a` directory, which should be in a path like `C:\Users\<USER>\Desktop\cs61a` where `<USER>` is your actual username. Then, create two new folders, one called `projects` and the other called `lab`. Both should be inside your `cs61a` folder:

```
cd cs61a
mkdir projects
mkdir lab
```

Now if you list the contents of the directory (using `ls`), you'll see two folders, `projects` and `lab`.



```
PS C:\Users\adity\Desktop> cd ~
PS C:\Users\adity> cd Desktop
PS C:\Users\adity\Desktop> mkdir cs61a


    Directory: C:\Users\adity\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/17/2023   9:44 PM                cs61a


PS C:\Users\adity\Desktop> cd cs61a
PS C:\Users\adity\Desktop\cs61a> mkdir projects


    Directory: C:\Users\adity\Desktop\cs61a


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/17/2023   9:44 PM                projects


PS C:\Users\adity\Desktop\cs61a> mkdir labs


    Directory: C:\Users\adity\Desktop\cs61a


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/17/2023   9:44 PM                labs


PS C:\Users\adity\Desktop\cs61a> ls


    Directory: C:\Users\adity\Desktop\cs61a


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/17/2023   9:44 PM                labs
d-----         1/17/2023   9:44 PM                projects
```

More Directory Changing

- `cd ..` (two dots). The `..` means "the parent directory", or one directory above your current directory.

> You do not have to keep your files on your Desktop if you prefer otherwise. Where you keep your files locally will not affect your grade. Do whatever is easiest and most convenient for you!

Downloading the Assignment

If you haven't already, download the zip archive, lab00.zip (/lab/lab00/lab00.zip), which contains all the files that you'll need for this lab. On most computers, `lab00.zip` is probably located in a directory called `Downloads` in your home directory.

You must expand the zip archive before you can work on the lab files. You need to first click on the .zip file, then choose "Extract all". If you run into trouble, you can search online for how to unzip a file. Different operating systems and different browsers have different ways of unzipping.

You only need to unzip the files once.

Once you unzip `lab00.zip`, you'll have a new folder called `lab00` which contains the following files (check it out with `cd lab00` and `ls`):

- `lab00.py` : The template file you'll be adding your code to
- `ok` : A program used to test and submit assignments
- `lab00.ok` : A configuration file for `ok`

Summary

Here is a summary of the commands we just went over for your reference:

- `ls` : **lis**ts all files in the current directory
- `cd <path to directory>` : **c**hange into the specified **d**irectory

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! You can also take a look at our UNIX tutorial (/articles/unix) for a more detailed explanation of terminal commands.