# Initial Bootup

A screenshot of a computer

Description automatically generatedPython is most easily manipulated through the command line (the weird “hacker window” with only text. You’ll see!). To be able to read or write files to the drive from the command line, the USB drive has to be in “NTFS” format. Basically, this kind of formatting is more particular to windows. If you want to know more about the differences, [this page](https://www.howtogeek.com/235596/whats-the-difference-between-fat32-exfat-and-ntfs/) gives a nice overview. To format it this way, we need to finish the following steps:

1. Plug in the USB drive and find it on the “My PC” window. You can open the file explorer with the keyboard shortcut “Windows-E”.
2. Right click on the USB drive, click “Show More Options”, then select “Format…”
   1. The menu that opens will show the current state of the USB drive. Since I’ve already formatted the USB drive to the correct setting, it should read “NTFS” under “File system”. Click “close” to exit the menu.

It’s good to note that formatting a USB drive also means erasing every bit of data currently on the drive: you’re basically starting from scratch. This is nice if you find an old USB drive and want to make sure everything is clean before starting to use it again.

# Installing Python

“Python is a programming language that lets you work quickly and integrate systems more effectively.” (from <python.org>). Python is great because it reads way more like English than most any other language that’s seriously used. BYU’s Chemical Engineering requires everyone to take a coding class their sophomore year to learn the basics; after that, each class has assignments that are leagues easier if you’re somewhat fluent in the language. Let’s get it running on the flash drive:

1. Go to <python.org>
2. In the “Downloads” dropdown, find and download the most current version (3.12.1 at the time I’m writing this). The location you download the file doesn’t really matter, but you can download it to the USB drive for cleanliness’ sake.

A screenshot of a computer

Description automatically generated

1. Find the installer in the folder you downloaded it to. Because we’re installing a coding language, we should install it as an administrator to make sure we have all the permissions we need. To do this, right click the executable (named something like “python-3.12.1-amd64”) and select “Run as administrator”.
2. In the installer, select manual installation. We need to make sure we can select the exact place we want the files to be installed
   1. In the menu, make sure the following options are selected:
      1. Documentation (this is the “how-to guide” of a coding language)
      2. Pip (allows us to install DLC for python)
      3. Tcl/tk and IDLE (tk is short for tkinter and allows us to make our own windows)
      4. Python test suite
   2. On the next menu, make sure “Add Python to environment variables” is selected. If a program is added to the computer’s environment variables, it adds the folder where the program is stored to the computer’s list of places to look when executing a file.
   3. In the “Customize install location” bar, make sure the USB drive is selected. Each drive has a formal name in the computer: your main computer’s directory (the programmer’s name for a folder) starts with “C:”, so programmers might call it their “C Drive”. A standard flash drive’s directory often starts with “E:”, so it’s sometimes called the “E Drive”.
      1. You’ll notice that underneath the bar, it says “You will require write permissions for the selected location.” Reformatting the SD card to NTFS allows us to have these write permissions.
3. If nothing shows up in the folder, rerun the executable file (again, as an administrator). This time, click the “Repair” option on the menu that shows up. This tells the installer to go back and double-check that all the files are there. It will find that they aren’t and then reinstall them.

# “Hello World”

There’s a custom in the programming world that the first program written in a new language should be one that somehow prints “Hello World”. To do this, we first need a way to access python. Some coding languages require that you separately installed what’s called an IDE (“Integrated Development Environment”).[[1]](#footnote-1) Luckily for us, python comes out-of-the-box with an interpreter than can run code directly:

1. In the main directory of the USB drive, there should be a new application titled “python”. Double click it to run the program. It should open a window that looks something like this:

A screenshot of a computer

Description automatically generated

The “>>>” you see is the prompt where we can write commands. To create code files, we’ll want to use something a bit more robust, but we’ll start here because python can already do some pretty neat stuff with what we have. Let’s go:

1. Per custom, we’ll start with printing “Hello World”. Python makes this really easy for us: the command to print something is just >>> print(“Write here”). You can put whatever you want in the quotes. Python actually also doesn’t care if they are double quotes (“) or single quotes (‘) as long as the starting on and ending on match.
   1. TRY: >>> print(“write something here”)
2. Python can also naturally act as a calculator. TRY these:
   1. >>> 1+1
   2. >>> print(“That was easy!”)
   3. >>> 28 \* 65
3. In Python, a hashtag is a comment, meaning the code ignores it. If you see one in the examples here, you don’t have to type it. It’s just a way to include comments in-line. TRY:
   1. >>> 5\*\*2 #Using \*\* means “to the power of”.
4. Variables are part of what make coding so useful. Python is really flexible with creating and modifying variables, so there isn’t a whole lot to it!
   1. >>> x = 4
   2. >>> y = 5
   3. >>> x\*y

There’s a lot you can do with Python as it is now. Python.org has a great tutorial that walks you through all the basic functions: [click here](https://docs.python.org/3.12/tutorial/) to go to the first page. It’s very thorough, but a bit dry. There’s a neat website called “LearnXinYMinutes” that has a crash course for just about any coding language out there. [Click here](https://learnxinyminutes.com/docs/python/) to get to the one for python. This one has pretty much zero fluff, which also means that it’s really dense. If you’re looking to get the tools to play around with as quick as possible, this is the perfect resource.

# Using the Back Door: Installing Packages

Python can do a lot on its own. That said, the internet means that people can create and publish what are called “packages” that enable python to do WAY more. The other nice thing is that all the information for these packages is available online, so if you have a question, it’s reasonably easy to answer.

The other incredible resource available now is AI. ChatGPT is fantastic at answering coding questions, whether it’s an error or a “how do I do this” question. We’ll use that here in a bit to create our chicken game code. For now, we’ll practice installing and using packages with two packages in particular:

1. Numpy. This name is short for “Numerical Python” and basically allows python to do a whole lot more math. One particularly useful thing is array math. For example, if we wanted to find out what some number multiplied by every number on our keyboard was, we’d have to type each one out or write a loop to do it[[2]](#footnote-2). With Numpy, we can easily create an array of those numbers and then get our answer in one line of code.
2. A screenshot of a computer

   Description automatically generatedMatplotlib. Short for Mathematical Plotting Library. This package lets us create graphs super easily, which can be really nice for homework[[3]](#footnote-3).

To install a package, we need to interact with python through the native installation package on the command line (Dun Dun DUUUUUUN). Using the command line feels pretty awesome: you interact pretty directly with the computer and look like a hacker doing it[[4]](#footnote-4). The easiest way to access the command line is by clicking the Windows button and then typing “cmd”. To be able to install packages to the USB drive, we’ll need to run the command line as an administrator (see right). Be careful when using the command line: certain commands can do a lot of damage. Always be sure that you know what a certain command does. If you’re uncertain, try asking ChatGPT.

Similar to the python interpreter, you should see a prompt that reads “C:\Windows\System32>”. This is like the “>>>” and is where you type in commands.

1. For a first command, try typing “dir” and clicking enter (this runs the command). “Dir” is short for “directory; this command displays everything in the directory you’re currently in. Because the command prompt starts you in the main drive of your computer, this should be a very long list.
2. If you want to clear the screen, use the command “cls”.

We’re not looking to work with anything in the main drive of the computer, so we need to get to the USB Drive. You might remember that the flash drive’s formal directory typically starts with “E:”. To be sure:

1. A black and white sign with white text

   Description automatically generatedOpen your file explorer and go to “This PC” to show a list of every storage device on the computer. The directory name should be in parenthesis next to the device name (shown right).
2. If the drive’s directory is “E:”, then we’ll type “E:” in the command line and press enter. Then, use the “dir” command again to show all the files in this directory (shown below).

A computer screen shot of a computer program

Description automatically generatedThe file names that show up should look very similar: these are the files on the USB that we can see inside the normal file explorer! The command line is just a more direct way of talking to the computer. It can be a lot more complicated, so most people (including myself, most of the time) just use the file explorer. Your prompt should now be “E:\>”, shown in the bottom of the picture to the right. This is like the “>>>” prompt in python, just showing where to enter commands.

## Numpy

The whole reason we’re using the command line in the first place is to be able to install packages. Normally, installing something means getting online, finding a website, finding the right link, downloading the file, and then executing a setup file. Python has a built-in installer called pip (“Package Installer for Python”) that will do all of this for us. It can also check for and install updates for old programs we already have. We need to make sure that we have pip properly installed to help us:

1. Once we’re in our “E:” directory for the flash drive, we can check first if python is installed. We ask the computer if something is installed by asking for the version number. TRY:
   1. E:\> python --version
      1. This should print “Python <VERSION NUMBER>” if it’s installed correctly.
   2. E:\> python -m pip --version
      1. The “-m” here is called a “flag”. This flag means “module”: the command could read something like “In python, look at the module called ‘pip’. What version is it?”
      2. Assuming pip wasn’t installed with the default python package, this should return that there isn’t a module named pip installed. This tells us to go ahead and install the package, which we’ll do using this command:
   3. E:\> python -m ensurepip --default-pip
      1. Python has a module called “ensurepip” that does exactly what it’s name implies. The “--default-pip” flag specifies that we just want the standard installation.
      2. If the installation is successful, you should see a message that says “Successfully installed pip-<VersionNumber>”. We can now use pip to install numpy:
   4. E:\> python -m pip install numpy
      1. This command points the computer to use the newly installed pip module in python to install numpy. You should get a similar successful installation confirmation message when this is finished. Your terminal will look something like what’s shown below.

A screenshot of a computer program

Description automatically generated

Perfect! We now have numpy installed and ready to use. We can try it out right here in the terminal by running the python command to open the terminal. TRY:

1. E:\> python
   1. You should see the prompt change to the default python prompt, “>>>”.
   2. Python doesn’t include every package you have installed in every program by default (thank goodness), so we have to tell python to include it:
2. >>> import numpy
   1. Now we can use python in our code. Let’s try a few commands:
3. >>> numpy.linspace(0,10,6)
   1. Can you figure out what this command does? Try changing each of the numbers in the command to figure it out. If you don’t have any ideas, ask ChatGPT! Here’s another one:
4. >>> my\_array = numpy.array( [ [1, 2, 3] , [4, 5, 6] , [7, 8, 9] ] )
5. >>> print(my\_array)
   1. This creates a 2-D array, basically a table. Numpy allows us to do operations on an entire array. For example:
6. >>> print(my\_array - 5)
7. >>> print(my\_array\*2)
8. >>> print(my\_array\*\*2)
9. >>> print(my\_array)
   1. As that last command showed, none of our calls actually modified my\_array. If we wanted to do this, we would re-assign the variable:
10. >>> my\_array = my\_array – 1

The last thing we’ll try is using a nickname for numpy. When we import a package, we can use a nickname so we don’t have to type out “numpy” every time we want to import the package:

1. >>> import numpy as np
2. >>> np.linspace(0,10,6)
   1. You’ll notice this does the same thing as it did before. The computer now treats “np” exactly as it would “numpy”.

The extra bit of array function ends up being super useful when working with lots of data. There’s a lot more you can do with Numpy: if you’re curious, ask ChatGPT or check out the [documentation](https://numpy.org/doc/) online.

## Matplotlib

We’ll install matplotlib the same way we did with Numpy:

1. First, exit python using >>> exit()
2. E:\> python -m pip install matplotlib

Now, we can re-enter python to try it out! There’s one specific module inside matplotlib called “pyplot” that’s the most useful: we’ll import only this module and give it the nickname “plt” (see below).

1. E:\> python
2. >>> import matplotlib.pyplot as plt
3. >>> import numpy as np
   1. Now that we have our modules imported, we’ll create some data to plot:
4. >>> xValues = np.linspace(-10,10,100)
   1. Make sure you understand what this line does. If you don’t, ask ChatGPT! Try asking “What does this line of code do in python?” with the line of code as well.
5. >>> yValues = xValues\*\*2
   1. I didn’t explicitly explain this earlier: using two asterisks (\*\*) means “to the power of”. Now that we have x and y values, we’ll plot them!
6. >>> plt.plot(xValues, yValues)
   1. Awesome we made a graph! Wait what, you don’t see it? Oh yeah, you have to tell matplotlib to show it to you. Try this:
7. >>> plt.show()
   1. Much better. That said, I have no idea what this graph is. Let’s add some labels. Each time you plot the graph, it also erases the data. To avoid this, we can turn on interactive mode for the graph. This allows us to modify the graph and see updates in real-time. To do this, use this command:
8. >>> plt.ion()
   1. This is short for “Interactive ON”. You might’ve then guessed the command to turn off interactive mode: “ioff()”. Now that interactive mode is on, let’s try plotting again:
9. >>> plt.plot(xValues, yValues)
10. >>> plt.xlabel(“Time”)
    1. Note the quotes. This function only accepts a string as an argument, so we have to include the quotes.
11. >>> plt.ylabel(“Knowledge of Python”)
12. >>> plt.title(“YOUR NAME HERE”)

Nice! You should recognize the shape of the graph: the parabola matches that of y = x2. This makes sense: we plotted an array of x values against those same values squared. Out of curiosity, let’s plot y = x5:

1. >>> plt.plot(xValues, xValues\*\*5)
   1. Your plot probably looks a bit silly: the original parabola should appear flat. This is because at the endpoints, the new plot is WAY bigger: 10 to the fifth power is 100,000, but 10 squared is only 100. To fix this, let’s only look at the values between -1 and 1:
2. >>> plt.xlim( [ -5, 5 ] )
   1. Looks like the y-axis limit is also too large. Let’s fix that too:
3. >>> plt.ylim( [ -10, 10 ] )
   1. Nice! Now we can see a lot better what these two functions look like. Let’s add a legend so that future you can remember what we plotted here:
4. >>> plt.legend( [ " y = x\*\*2" , "y = x\*\*5" ] )
   1. This tells matplotlib what to call each of the lines, then automatically includes the legend in the plot.

This is where we’ll stop here. There are tons of resources online for making these plots prettier, making them more interactive, for including them in documents, etc. For the time being, we’ll end by saving the image as a file on the flash drive. Try asking ChatGPT to do this with “How can I save a matplotlib graph as a file?”. You’ll know you did it right when you see an image file show up in the main file directory of the flash drive (you can check in your normal file explorer).

# Chicken Simulator

A few times now, we’ve used ChatGPT to help us understand code. ChatGPT (and other AI engines) are fantastic at writing code, and they’ll only continue to improve as more features and training are added. A while ago, we used ChatGPT to write the base for a chicken game. Since we did this, I’ve worked to add some features and uploaded the current version to GitHub.

GitHub is a massive online library for code. It’s kind of like Google Drive, but specifically to help with collaborating on code. Image you have a massive project with hundreds of people working on it: it gets really hard to have everyone working on the code at the same time because so many things are changing all the time. If you’re using GitHub, each person can “pull” a version of the current files from GitHub, modify it, then “push” it back to the main directory. Each project is called a repository, or “repo” for short.

There’s a repo linked to my GitHub account with the files for the Chicken Simulator. Before we get started with the command line interface for GitHub, go check out the [main website](https://github.com) just to get familiar with the site. There’s a lot to explore there; you actually don’t even need an account to get the code, though.

1. Go to the main GitHub repo for the Chicken Simulator at [github.com/jwp91/ChickenSimulator](https://github.com/jwp91/ChickenSimulator)
2. Find the big green “Code” button and click it to open the pop-up menu
3. In this menu, select “Download ZIP File”. This will allow you to simply download the files directly to your computer. Find the USB drive folder and download the files here.

The files are downloaded as a zipped file by default. There are ways to unzip files from the command line, but it’s easier this time to just unzip it from the file explorer.

1. Open File Explorer (Windows + E), find the flash drive, right click on the new zip file, then select “Extract All”. This will place all files in a new folder in the main directory of the flash drive.

I personally like to delete the original zipped file after extracting it to get rid of extra fluff, but that’s totally up to you. We can now run the chicken game through the terminal:

1. Open the command line and navigate to the E: drive like we did before.
2. E:\> dir
   1. Use this to see the current folder contents. You should see the new unzipped folder displayed near the top.
3. E:\> cd ChickenSimulator-main
   1. The “cd” command steps into the specified folder. Make sure the name after the command matches the unzipped file. After you’ve typed the first few characters, you can use the tab key to have it autofill the rest of the name.
4. E:\ChickenSimulator-main> dir
   1. Show the contents again. If there’s only one folder inside, use “cd <FOLDER\_NAME>” to step inside that one, then “dir” to check the contents.
   2. The program file will have the extension “.py”. Locate it and note the name.

To edit the man code block, we need to open the code in an editor. Python has a built-in IDE (“Integrated Development Environment”) for doing just that. It’s called IDLE (Integrated Development and Learning Environment) and works fine to things like this. To access it and take a look at the code, try this:

1. E:\ChickenSimulator-main\ChickenSimulator-main> python -m idlelib ChickenSim.py
   1. This will open the editor. You can take a look around if you’re curious, but for now let’s just run the code! Exit the interpreter, then try this:
2. E:\ChickenSimulator-main\ChickenSimulator-main> python ChickenSim.py
   1. The program will ask for the initial number of chickens in the terminal. Once you type in a number and hit enter, the code will run!

1. I personally enjoy using one called Jupyter Lab, but using it requires installing the program to your computer (i.e. it can’t be run from the flash drive). You’re more than welcome to install it: it’s not difficult, I just wanted to keep this limited to what we can do with just the flash drive. [↑](#footnote-ref-1)
2. if you’re not sure what a loop is, try asking ChatGPT! To start, ask it “How does a loop work in python? Can you give some examples of each one?” You can ask clarifying questions or for more examples if you’re curious. [↑](#footnote-ref-2)
3. If you’re just looking to create a graph, desmos.com has a fantastic graphing calculator. It’s super easy to use and is worth taking a look at: [click here](https://www.desmos.com/calculator) [↑](#footnote-ref-3)
4. If you want people to think you’re *really* cool, just throw the word “mainframe” when you explain what you’re doing. It might be worth learning about what a mainframe is in the meantime, or just hoping they don’t either! [↑](#footnote-ref-4)