The basics of Bash -- Pedro Pessoa (https://pessoap.github.io/)

If you are new to using terminals, Bash scripting might seem intimidating at first. However, it is a powerful and versatile tool that can greatly simplify many tasks. Often, running software on supercomputers requires using the terminal exclusively, making Bash an essential skill. Bash, which stands for "Bourne Again SHell," allows you to automate and streamline complex workflows by writing scripts that execute a series of commands.

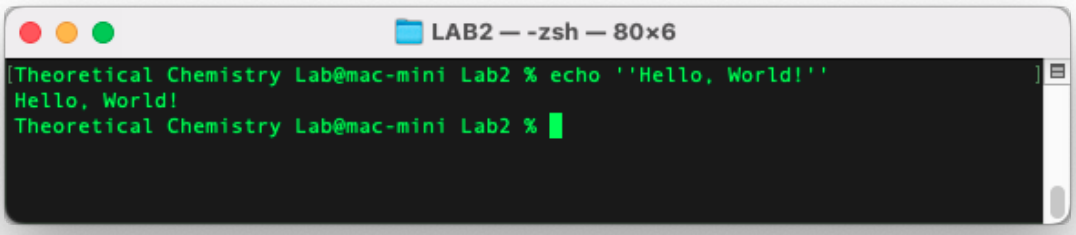
The purpose of this text is *not* to teach you basic Bash commands, there are many good resources for this online (https://devhints.io/bash). Instead, we aim to demonstrate how these commands can be used to accomplish specific tasks. We'll conclude with a simple yet interesting application where we use Bash to generate graphs for harmonic oscillators with different frequencies.

Using Bash on terminal

Open the terminal, in the same directory as the other files for this lab.

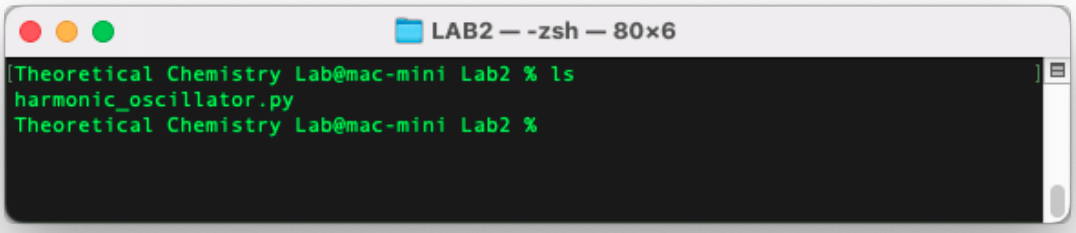
Command `echo`

Although it may not look like it sometimes, Bash is a programming language and, like many programming languages, the first example will be the classic "Hello, world!". Yes, everyone gives this example, but it's a tradition at this point! Type `echo ''Hello, world!''` on the terminal and observe the result.



Command `ls`

Now, for something a little more useful, let us list all the files within the directory of interest. This is done with the command `ls`

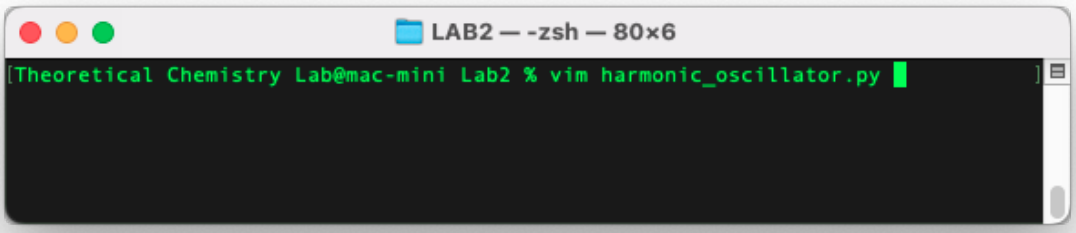


where we can see the single file in the folder - `harmonic_oscillator.py`

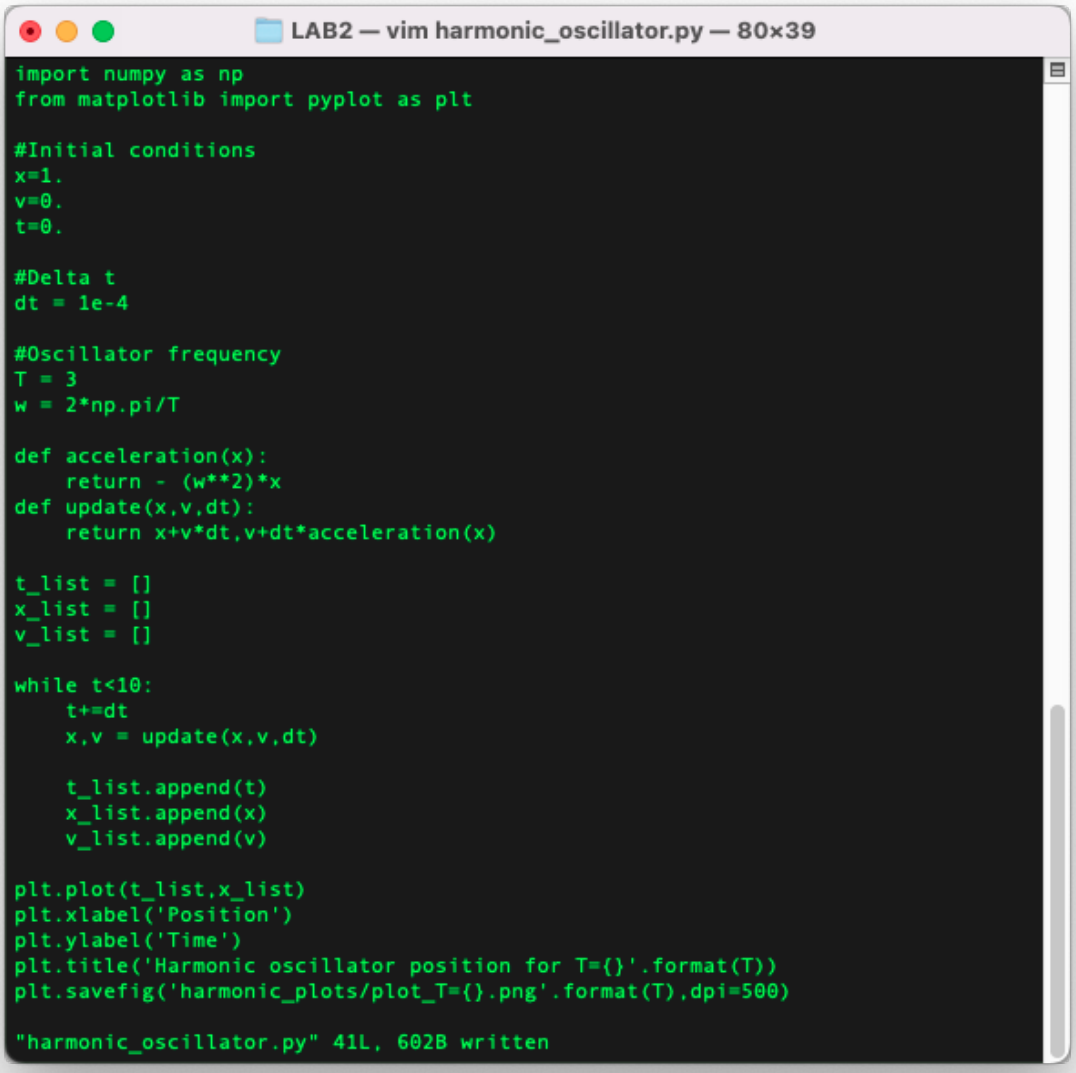
Using software `vim`

Running software from the terminal is an important task. However, software that relies on a graphical interface often cannot run directly in the terminal. Thus, when it comes to reading or changing a file, only a handful of text editors can be used.

One popular choice among programers is the text editor called `vim`. Let us use `vim` to open our file `harmonic_oscillator.py` typing the following command:



Your terminal should now look something like this:

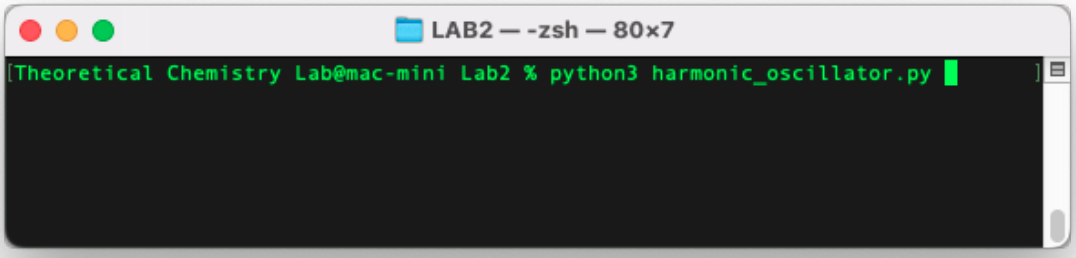


If you try to make any changes to this Python code, you will notice how `vim` may not look that user-friendly at first. Later in

this lab we will eventually to learn how to use it. effective. For now simply type `:q` to close `vim` .

Running python code

Let us move to a more interesting part of this lab: running a Python program from the terminal. As some of you may have noticed, the file `harmonic_oscillator.py` is a python script that calculated the position of a harmonic oscillator as a function of time, as you were taught earlier today. More specifically, it generates a graph showing the position of the harmonic oscillator over time. We will see it in action later. For now, let's run the script using the following command:



However, the way it is setup here you will probably see an error message. Specific instructions on how to fix it will be given in class.

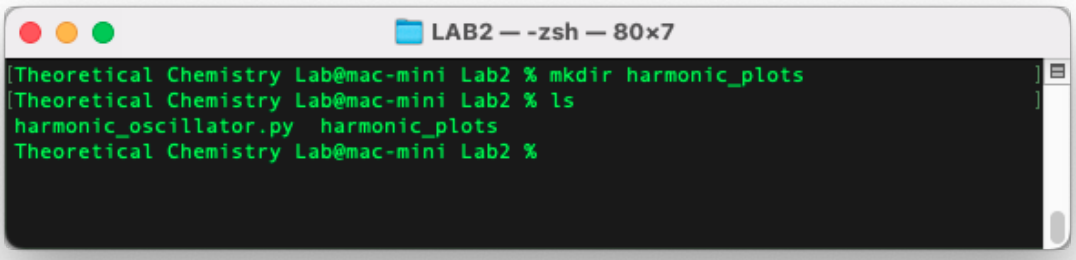
Although I do not want to sound negative, confusing error messages like this are (unfortunately) an intrinsic part of programming.

Here we are going to give, on the board, some instructions on how to setup libraries for python.

Once this is done, what is not working is that the script is trying to save a figure in the directory `harmonic_plots` but such directory does not exist, which is what we are going to fix with our next command.

Command `mkdir`

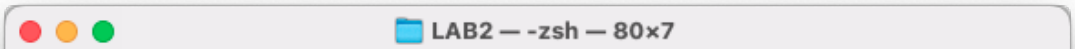
To create a directory named harmonic_plots, use the `mkdir` (make directory) command as shown below:

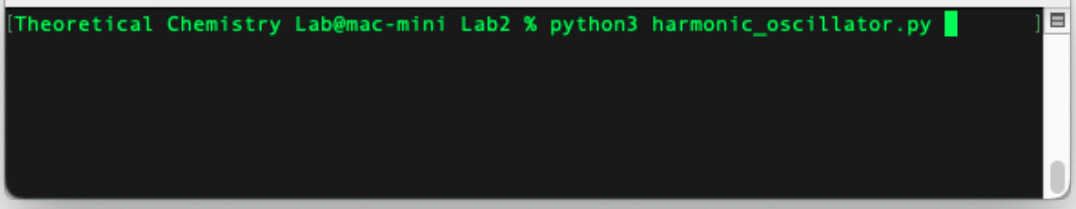


Note that we also verified that the directory was indeed created using the `ls` command.

Running python code (again)

With the directory ready, let us run the Python script again

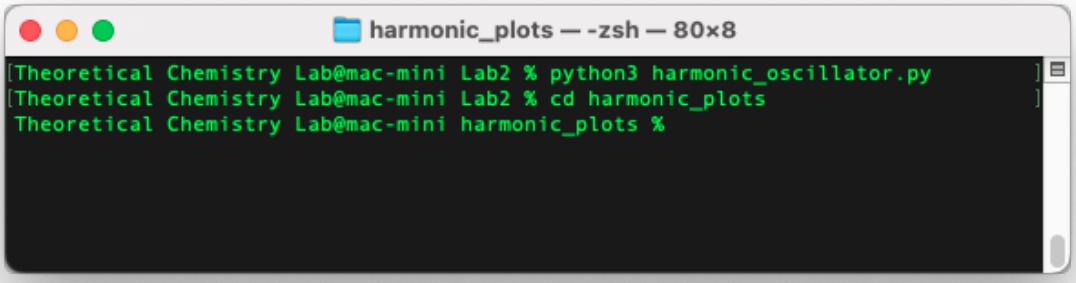




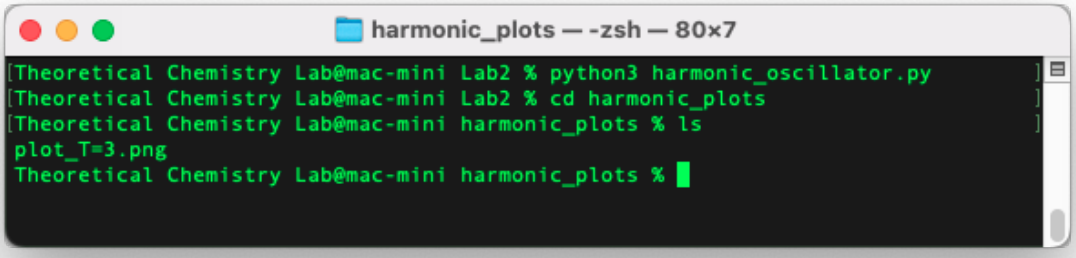
This time, it should execute correctly. However, as previously mentioned, it saved the figure in another directory. During this summer school, most of the code you run will do something similar. The next command will help you see the resulting figure.

Checking for output --- Command `cd`

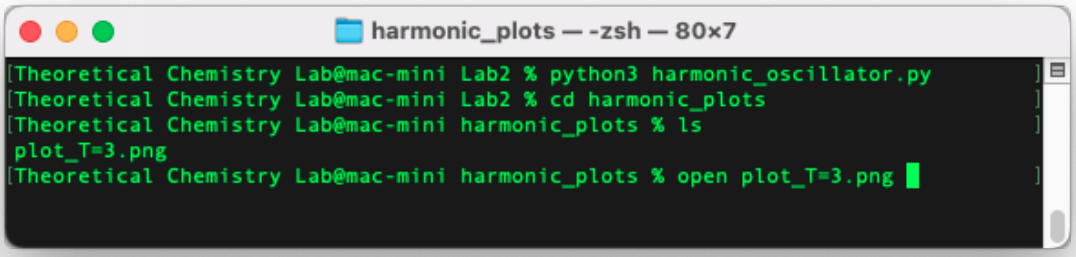
Entering the directory `harmonic_plots` where the figure was saved is straightforward:



Here it is also interesting to obseve the files within that directory using the command `ls`



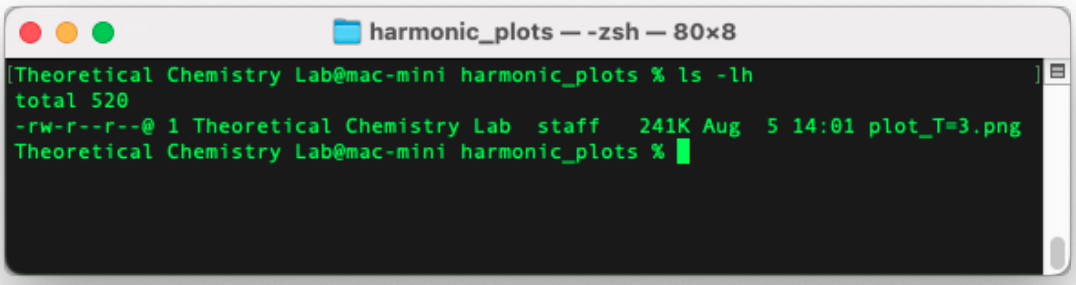
Now to, finally, observe the results by opening the file:



When you execute the command above, a new window should pop up with the harmonic oscillator results. Before moving on with this tutorial, ask yourself, does this result makes sense?

Checking files sizes

Before moving into the lab exercises, let us look into another useful tool. The command `ls` can be extended as `ls -lh`, when done so it will not only show the files in the directory, as with the regular `ls`, but also show more detail about each file



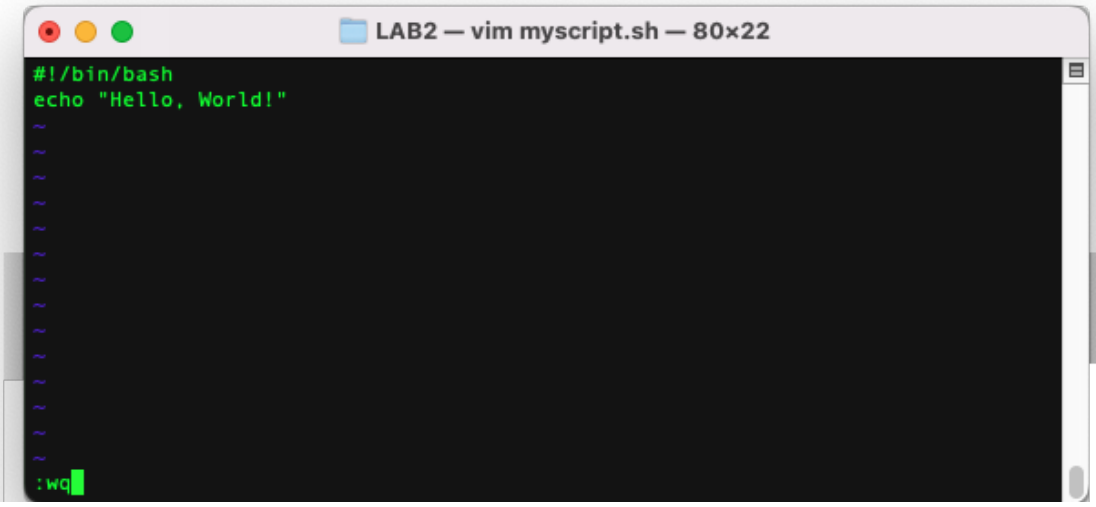
For example, one information we can gather from above is that the figure generated has a file size of 241Kb

Bash scripts

While above we were executing every command one by one. In some cases it might be preferable to give the computer a "to-do list". This is when Bash scripts become necessary. In short, instead of typing each command one by one, scripts allow you write them down in a file, that can be runned independently.

Write your first script

- Open the terminal.
- Crate a new script file using the command `vim myscript.sh`
- To be able to write on vim type `i` (command for insert) and type a textfile like the one below.

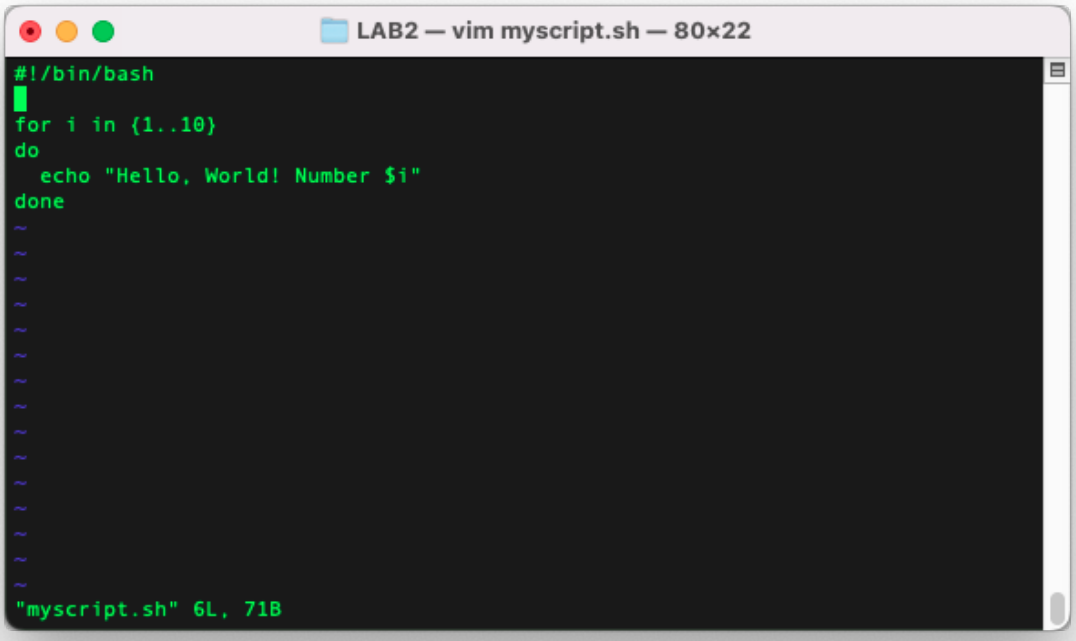


- Use the key `esc` followed by `:wq` to exit `vim` with the file saved.
- Type `chmod +x myscript.sh` in the terminal. This gives your script permission to run.
- Type `./myscript.sh` and press Enter. You should see "Hello, World!" printed on the screen

Lab Assignment

1 - Write your second bash script

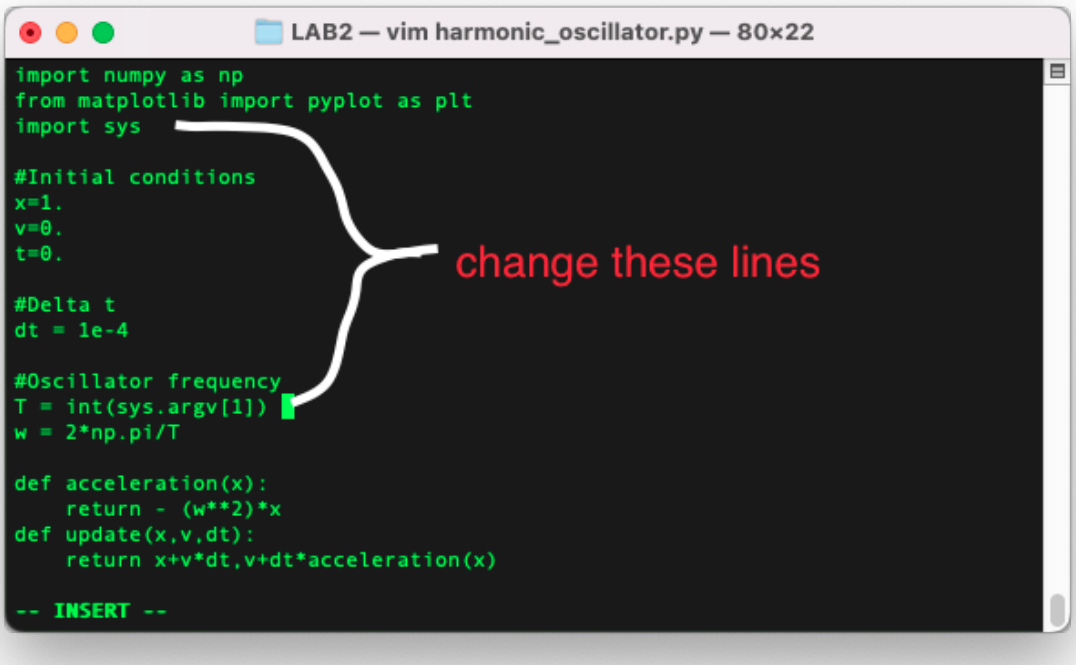
Analogous to how you did your first bash script, create and run another bash script



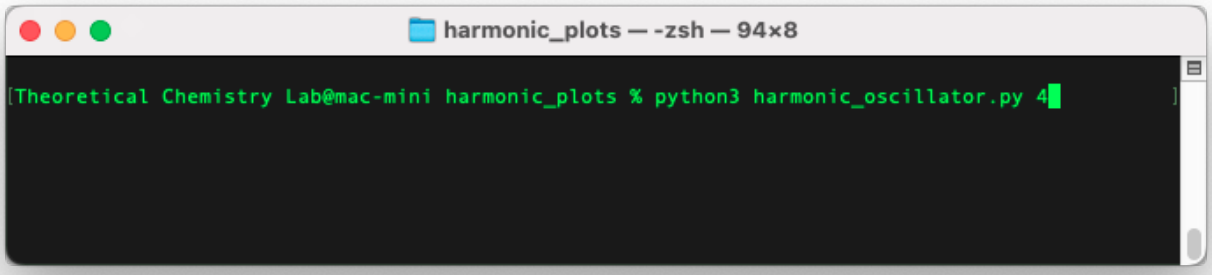
Once you run it, did it do what you expected?

2 - Change you python file on the terminal

Using vim make the following changes to the file `harmonic_plots.py`



Then, run the file as follows



What did this do?

3 - Run many harmonic oscillators

Based on the results of 1 and 2, create a bash script to run `harmonic_plots.py` with different harmonic oscillator periods (from 1 to 10) automatically. Open the images generated and tell