You can write programs in the Unix shell scripting language. Often called scripts, they are typically used for tying together input and output from different programs.

Let's look at a sample script (from elf lord's tutorials on linux):

```
#!/bin/bash
echo "hello, $USER. I wish to list some files of yours"
echo "listing files in the current directory, $PWD"
ls  # list files
```

Looking at this script line-by-line:

```
#!/bin/bash
```

It's standard to include as the first line of your scripts that specifies the program that's running (this is often called the "shebang" line). There's different variants of shell scripts. We're using the default for Ubuntu (the type of Unix running on the lab laptops) called bash, so, we start our script by specifying that we want to use the bash shell to evaluate it.

The command echo is similar to print() in Python. It writes a message to the terminal:

```
echo "hello, $USER. I wish to list some files of yours"
echo "listing files in the current directory, $PWD"
```

$USER is a built-in variable that store the name of the current user. Similarly, the built-in variable, $PWD, stores the current directory (folder) that you are in.

Lastly, our scripts can include any of the Unix commands that you have already learned. The last line of this file lists the files in the current directory using the ls command:

```
ls  # list files
```

In the shell, the different types of quotes have similar, but different, meanings. We'll use the double quotes since strings in double quotes will have special characters (like \n for newline) interpreted as in Python and C++.

Use any text editor that will allow you to save a file with any or no extension such as gEdit (included with some Linux distributions), textEdit (included with MacOS) or notepad (included with Windows) to modify the above script to say "Hello, World". Note that we're leaving off the "!" since it is a bit confusing to print due to its special meaning in the shell (it lists the history, or previous commands, you have typed; for example: !! gives all commands in the history, !cd will repeat the most recent command you gave that starts with cd). Include in the

second and third lines your name and email for the grading scripts (the first line should be the #!/bin/bash. Save your file as helloScript

Next, we'll change the permissions on the file, so that we can run it directly, by just typing its name. In the terminal, move to the directory where you saved your script and type:

```
$ chmod +x helloScript
```

(changes the "mode" of the file helloScript to be executable-- if you name it something else, replace the name in chmod command above). To run your first shell script, you can now type its name at the terminal:

```
$ ./helloScript
```

Make sure to include a comment on the second line with your name (the first line has the #! line). Comments in shell are, like Python, preceded by a #.