Shell Scripts (.sh files)
Software Engineering for Scientist

A **script file** is a file that contains a series of commands to execute a program or scripting engine. These commands get to run without being compiled.

You can think of this as an easy way to run a series of multiple commands that you would have to type in Terminal. Writing a script file allows you to easily call the series of commands every time you want to run your program, without having to retype it all.

You can create this file in any text editor you want, the scripting language which we will use is called bash. We will save these files as .sh files, however, the extension is not necessary due to how we initialize the file.

To initialize the file:

You can use any text editor to begin your file.
Every file begins the same way, with a “shebang”, shown below. This makes the bash script executable.

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

The meat of the file:

The remainder of your file you can type commands just like you would into the terminal. For example, if you would like to run the python function math.py with input arguments 3, 4 you can write:

```
python3 math.py 3 4
```

You can also create “print” statements which will output to your terminal by using “echo”

```
echo This function runs math.py
echo ..running...
```

For additional information and commands to use in your bash file, please see this guide:
http://tldp.org/LDP/Bash-Beginners-Guide/html/

Example file:

A bash script to add with multiple arguments.

run.sh

```
#!/bin/bash

a=$1
b=$2
c=$3
d=$4
echo "Running calc.py add $a $b"
python calc.py add $a $b
echo "Running calc.py add $c $d"
python calc.py add $c $d
```

Making the file executable:

Next, we will add a line to tell the computer this is an executable file, where <file_name> should be the name of your file. For convention, **we will always use run.sh** for the file name.

```
chmod -x run.sh
```

Maybe you want to activate a specific conda environment, you can do that just like in terminal:

```
conda activate swe4s
```

Running your shell file:

We will run this file from the terminal. You can do any of the following to run the file

```
$ sh run.sh
```

OR since we made it an executable

```
$ ./run.sh
```

For the script above, we get the following output:

We see the output from both our shell script echos as well as any output the functions add and calculate will output

```
$ ./run.sh 1 2 3 4
Running calc.py add 1 2
3
Running calc.py add 3 4
5
```

WAIT

Bash scripts do not stop on error and don't really care about unset variables.

```
$ ./run.sh 1 2
Running calc.py add 1 2
3
Running calc.py add
Traceback (most recent call last):
  File "/Users/rl/Downloads/calc.py", line 5, in <module>
    b = int(sys.argv[3])
IndexError: list index out of range
```

To make your scripts more robust, add the following lines to your scripts

`run.sh`

```
#!/bin/bash

set -e # stop on error
set -u # raise error if variable is unset
set -o pipefail # fail if any prior step failed

a=$1
b=$2
c=$3
d=$4
echo "Running calc.py add $a $b"
python calc.py add $a $b
echo "Running calc.py add $c $d"
python calc.py add $c $d
```

With these safeguards in place the errors will be easier to identify and fix

```
$ ./run.sh 1 2
./run.sh: line 9: $3: unbound variable
```