CS417 Lab 2

In this lab, you will process command-line arguments.

Please read the attached introduction to using the terminal.

Getting Started

To begin the lab:

- Create a folder, in your `Documents` folder; call it `417_lab02` .
- Go to `mycourses.unh.edu`, find CS417, find lab #2, and locate the program `argv.py`. Download it into the folder that you just created.
- Run the terminal program. It should start in your home directory. Change to the directory you just created, by typing:

```
          cd Documents
          cd 417_lab02
```

Now, verify that `argv.py` is there:

```
          ls
```

Finally, run that program:

```
          python argv.py
```

The Goal

In this lab, you will create a program that prints a block of text, using command-line arguments. The purpose is to learn a simple way to parse *named* and *positional* arguments. Here are several ways to invoke the program:

```
        python argv.py X -width 5 -height 3
        python argv.py X -height 3 -width 5
        python argv.py -width 5 X
```

All of these should print the symbol `X`, repeated into a block 5 wide and 3 high:

```
        XXXXX
        XXXXX
        XXXXX
```

There two kinds of arguments:

- Named arguments. They come in pairs: a name, followed by a value.
- Positional arguments. They occur alone.

In the example above, the width and height are named arguments. The symbol being printed is a positional argument.

Names are marked with a dash (`-`). Thus the pair `-width 5` means that "set width to 5". The pair `-height 3` means "set height to 3".

Notice the third example: there is no mention of the width. In that case, we will use a *default* value for that variable. The default width will be 3.

Your Tasks

Goal: handle named and positional command-line arguments. We will work gradually towards this goal.

Begin by using a text editor, or IDLE, to edit the file `argv.py`.

Notice that you will *NOT* use IDLE to *RUN* the program. You will always run the program using the *command line*.

1. Implement the function `print_args()` . Just write a `for` loop that travels through the list slice `sys.argv[1:]`, and prints each entry on a separate line. We are just looking at the arguments, so we skip `sys.argv[0]`, which is the program name `"argv.py"`.

   Try it: go to the terminal, and type:

   ```
       python argv.py X -width 5 -height 3
   ```

   CAVEAT: if you have a Mac, it comes with python *2* installed. This course uses python *3*. In order to get python 3, type `python3` instead of `python`.

   Shortcut: on the terminal, you often want to repeat the last command. To do this, press the UP arrow, and press ENTER. Try it!

---

2. Implement the function `print_names()` . You probably wrote something like this:

```
              for arg in sys.argv[1:] :
                  print (arg)
```

```
Instead of printing _every_ argument, just print the names.  First
check if `arg` starts with a minus sign: `if arg[0] == '-':`

Then, instead of `print (arg)`, discard the dash: `print (arg[1:])` .

Go to the terminal and type the same thing again:
```

```
        python argv.py X -width 5 -height 3
```

---

3. Now you are ready to implement `parse_arguments()`. If you find a name, you have to access the argument *after* it. A `for` loop makes this awkward: a `while` loop is easier. Begin with this code:

```
        index = 1
        while index < len(sys.argv):
            arg = sys.argv[index]
            ...
            index += 1
```

If `arg` is a name, you should

```
- figure out if the name is `"width"` or `"height"`.

- increment `index`, and retrieve the *next* argument (which is a
value).

- remember to convert the value into an `int`!

- change either the `width` or `height` variable.

If `arg` is not a name, then it's a positional argument.  In this
case, you should just store it into `symbol`.
```

---

4. If a user omits a parameter, it gets a default value. So, in `parse_arguments()`, give the three parameters `width`, `height`, and `symbol` the values `5`, `3`, and `"X"` initially (*before* you start the `while` loop).

---

5. At the end of `parse_arguments()`, return the three parameters, as a *tuple*:

```
        return (width, height, symbol)
```

---

6. Last step: implement `print_block()`. It gets passed the three parameters, and prints a block of symbols, with no spaces, and a `"\n"` after each line.

Turning in your work

When you are done, go to `mycourses.unh.edu`, find the cs417 course, and find Lab 2. Click the Submit button, and upload `argv.py`. You should turn in your work at the end of the lab session, even if you haven't completed it. You will have until midnight to turn in your work again without a lateness penalty.

SAVE YOUR WORK! Before you leave, save your work on a USB key, on UNH Box, or email yourself the files.

Testing

I am supplying a program `test_argv.py` that tests your work. You should use it; we will use a similar program to grade your work. If, on your computer, typing `python` brings up python 2, you should edit this file, so that the calls to `os.system()` say `"python3 argv.py ..."` instead.