Bi 410/510: Introduction to Programming for Biologists

Spring 2018

Project 6: Scripts

To complete this project you need to answer one written question and write three small Python programs that interact with the host operating system. Write the answer to Project 1 in a plain text file, put that file and the Python programs in in a single folder named `scripts`, and compress the folder:

```
$ zip -r scripts.zip scripts
```

Upload `scripts.zip` to Canvas as your submission for Project 6.

Note: The examples for each program below show messages printed by the programs as they run. You do not have to print your results in the same format – we'll be looking for correct results and good style but not the same exact output format.

Data

Most of the programs for this project work with the `sim` folder you used with Unit 2 (Shell Commands). Copy the zip file (or download a new copy from Canvas) to the folder where you work on this project.

Unzip the folder each time you test a script. Some of your programs will change the contents of the `sim` folder. To restore the folder to its original state before you run your script again, or before you start working on the next project, **delete the current folder and replace it with a new copy**.

You can do both of these operations by typing two shell commands on a single line:

```
$ rm -rf sim; unzip sim.zip
```

Program 1: `mystery`

A Python program named `mystery.py` is shown at the top of the next page. Explain what the program will do if we run it in the `sim` folder with this command:

```
$ python mystery.py sim.cfg
```

Write your answer in a plain text file named `mystery.txt` and include it in the zip file with your programs.

Note: You can answer the question by typing in the program and running it, but we suggest you try to answer it by reading and understanding the code.

```
from sys import argv
import os.path
import shutil

fn = argv[1]

if not os.path.isfile(fn):
    print(’no such file:’, fn)
    exit()

head, tail = os.path.split(fn)
base, ext = os.path.splitext(tail)

newfn = base + ’.orig’ + ext
if os.path.isfile(newfn):
    print(’copy exists’)
    exit()

shutil.copyfile(fn,newfn)
```

Figure 1: The Mystery Program

Program 2: `temps`

Write a program named `temps.py` that will convert temperature values from Fahrenheit to Celsius or vice versa.

Use `argparse` to get the temperature value and to figure out which scale to convert from. The user should specify either `celsius` or `fahrenheit` using `--scale`, and specify the temperature value with `--temp`.

Examples:

```
$ python temps.py --scale fahrenheit --temp 32
0.0

$ python temps.py --scale fahrenheit --temp 212
100.0

$ python temps.py --scale celsius --temp 0
32.0

$ python temps.py --scale celsius --temp 100
212.0
```

Grading: Any program that uses `argparse` to get the command line values and produces the correct output will earn 3 points. For full credit set up the argument parser so that (a) Fahrenheit is the default scale, (b) the only possible values for the `--scale` option are `fahrenheit` and `celsius`, and (c) the temperature is a required input.

Here are some examples of how a full-credit program would work:

```
$ python temps.py --temp 100
37.8

$ python temps.py --scale kelvin --temp 212
usage: temps.py [-h] [--scale {fahrenheit,celsius}] --temp TEMP
temps.py: error: argument --scale: invalid choice: ’kelvin’ (choose from ’fahrenheit’, ’ce

$ python temps.py --scale fahrenheit
usage: temps.py [-h] [--scale {fahrenheit,celsius}] --temp TEMP
temps.py: error: the following arguments are required: --temp
```

Program 3: `delete_empty`

Write a program named `delete_empty.py` that will delete any empty files (files with size = 0) from a folder.

The program should print a message with the name of a file that will be deleted. Three files in the `uniq` folder in `sim` are empty, so this is what you should see when you pass that file name to the program:

```
$ python delete_empty.py sim/uniq
removing sim/uniq/unique.1.fasta2.clstr
removing sim/uniq/unique.3.fasta2.clstr
removing sim/uniq/unique.2.fasta2.clstr
```

Note: Since there is only one command line argument you do not need to use `argparse` for this project (but you can if you want).

Hint: We strongly suggest you test your program and make sure it’s printing all the correct file names. When you see the names you expect add the call to `os.remove` that actually deletes the files.

Program 4: `move_logs`

Write a program that searches a directory in the `sim` folder to find log files (files with names that start with “log”) and move them to a folder named `log` (this program automates two of the operations you did for the project in Unit 2).

You can assume your program will always be run in the same directory that contains `sim`. The program should first see if there is a folder named `sim/log`, and if not, create it. Then it should look in the folder specified on the command line, and any files with names that start “log” should be moved to `sim/log`. **Hint:** use the `glob` function to get a list of all files with names that match `log.*`.

If your program prints a message as each file is moved this is what you will see if you tell it to move all the log files in the `merged` folder:

```
$ python move_logs.py merged
moving sim/merged/log.1.txt
moving sim/merged/log.2.txt
moving sim/merged/log.3.txt
```

Using `argparse` is optional for this project, and we again strongly recommend testing your program thoroughly before adding the call to the function that actually moves the files.