Assignment 1: Conditionals and Loops

The University of British Columbia
Science One CS 2015-2016
Michael Gelbart

Parts 1 and 2 due Thursday, October 1st at 11:59pm
Parts 3 and 4 due Thursday, October 8th at 11:59pm

Last updated Friday, September 25th around noon.

Learning Goals

In this assignment you will write several short Python programs. The skills/concepts you will practice in this assignment are:

- Boolean expressions
- conditionals (`if`, `else`, `elif`)
- `while` loops
- nested loops
- reading command-line arguments
- importing libraries with `import`
- printing to the screen (`print`, `sys.stdout.write`)
- integer division

1 Are these numbers ordered?

Write a program `Ordered.py` that reads in three integer command-line arguments, x, y, and z. Define a Boolean variable `isOrdered` whose value is `True` if the three values are either in strictly ascending order ($x < y < z$) or in strictly descending order ($x > y > z$), and `False` otherwise. Print out the variable `isOrdered` using `print isOrdered`.

```
% python Ordered.py 10 17 49
True

% python Ordered.py 49 17 10
True

% python Ordered.py 10 49 17
False
```

Note: the `%` symbol is used to indicate the prompt of a terminal (sometimes `$` or `>>` is also used). So, the first example above should be interpreted as follows: if your program is working correctly and you type `python Ordered.py 10 17 49` into the terminal, then it should produce the output `True`. Keep in mind that, generally speaking, producing the correct output for the provided examples is a necessary but not sufficient condition for having a correct program; come up with your own test cases!

Useful syntax. In order to read command-line arguments, you need to add `import sys` on its own line at the top of your file. Then, access the *n*th command-line argument with `sys.argv[n]`. For example, If the user types in `python Ordered.py 10 17 49` then `sys.argv[1]` will give '10', `sys.argv[2]` will give '17', and `sys.argv[3]` will give '49'. Note that the entries of `sys.argv` are strings, so you will need to manually convert them to the proper type. For example, `int(sys.argv[1])` would yield the integer 10, which is what you want for the first command-line argument.

2 Colour formats

There are a number of different formats for representing colour. RGB format specifies the level of red (R), green (G), and blue (B) on an integer scale from 0 to 255: It is the primary format for LCD displays, digital cameras, and web pages. CMYK format specifies the level of cyan (C), magenta (M), yellow (Y), and black (K) on a real scale of 0.0 to 1.0: It is the primary format for publishing books and magazines. Write a program `RGBtoCMYK.py` that reads in three integers `red`, `green`, and `blue` from the command line and prints out equivalent CMYK parameters. The mathematical formulas for converting from RGB to an equivalent CMYK colour are:

$$\begin{aligned}
W &= \max\{R/255, G/255, B/255\} \quad \text{(white)} \\
C &= \frac{1}{W}\left(W - \frac{R}{255}\right) \\
M &= \frac{1}{W}\left(W - \frac{G}{255}\right) \\
Y &= \frac{1}{W}\left(W - \frac{B}{255}\right) \\
K &= 1 - W
\end{aligned}$$

Note: if all three red, green, and blue values are 0, the resulting colour is black (0 0 0 1).

```
% python RGBtoCMYK.py 75 0 130
cyan    = 0.423077
magenta = 1.000000
yellow  = 0.000000
black   = 0.490196

% python RGBtoCMYK.py 0 0 0
cyan    = 0.000000
magenta = 0.000000
yellow  = 0.000000
black   = 1.000000
```

Useful syntax. The function `max(x, y)` returns the maximum of x and y. `max` can also take more than two arguments.

3 Checkerboard

Write a program `Checkerboard.py` that takes an integer command-line argument N, and uses two nested `while` loops to print an N-by-N "checkerboard" pattern like the one below: a total of N^2 asterisks, where each row has $2N$ characters (alternating between asterisks and spaces).

```
% python Checkerboard.py 4
* * * *
 * * * *
* * * *
 * * * *

% python Checkerboard.py 5
* * * * *
 * * * * *
* * * * *
 * * * * *
* * * * *
```

Useful syntax. The `print` command in Python automatically adds a newline character to the end of whatever string you are trying to print. To print only the string without starting a new line, use `sys.stdout.write(string)` instead of `print string`. To print just a new line, use `print ''` or `sys.stdout.write('\n')`.

4 Simulating a random walk

A drunkard begins walking aimlessly, starting at a lamp post. At each time step, the drunkard forgets where he or she is, and takes one step at random, either north, east, south, or west, each with probability 25%. How far will the drunkard be from the lamp post after N steps? Write a program `RandomWalker.py` that takes an integer command-line argument N and simulates the motion of a random walker for N steps. After each step, print the location of the random walker, treating the lamp post as the origin (0, 0). Also, print the square of the final distance from the origin. An example run is shown below, and illustrated in Figure 1. (You do NOT need to produce a figure or plot as part of this assignment, this is just for illustration purposes.) Keep in mind that, due to randomness (and unlike the other programs you wrote for this assignment), your program will not produce the same output every time you run it!

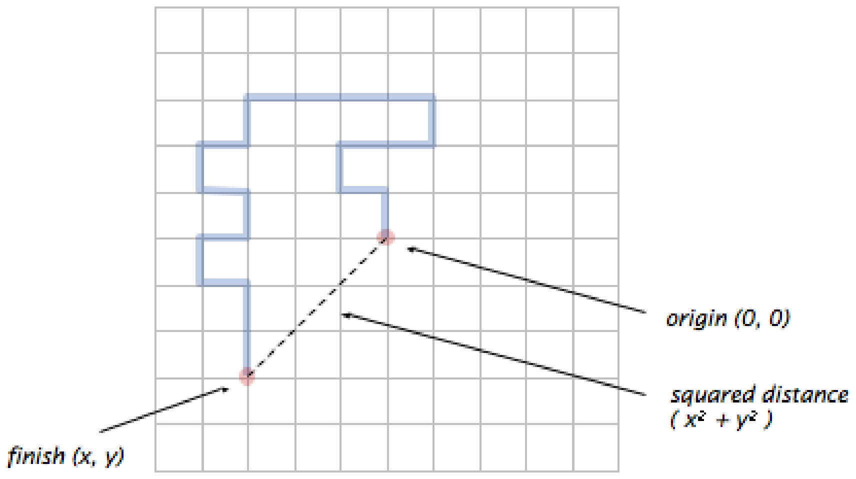

Figure 1: A drunkard's walk after 20 steps. In this particular case, the squared distance from the origin to the final position is $3^2 + 3^2 = 18$.

```
% python RandomWalker.py 20
(0, 1)
(-1, 1)
(-1, 2)
(0, 2)
(1, 2)
(1, 3)
(0, 3)
(-1, 3)
(-2, 3)
(-3, 3)
(-3, 2)
(-4, 2)
(-4, 1)
(-3, 1)
(-3, 0)
(-4, 0)
(-4, -1)
(-3, -1)
(-3, -2)
(-3, -3)
squared distance = 18
```

Remark: this process is a discrete version of a natural phenomenon known as Brownian motion. It serves as a scientific model for an astonishing range of physical processes from the dispersion of ink flowing in water, to the formation of polymer chains in chemistry, to cascades of neurons firing in the brain.

Useful syntax. To take random steps, you should `import random` and then use `random.random()`, which generates a (pseudo-)random `float` uniformly distributed between 0 and 1. You can then use this number to choose one of the four directions with equal probability.

Submission instructions

Submit the files `Ordered.py`, `RGBtoCMYK.py`, `Checkerboard.py`, and `RandomWalker.py` on Connect using the instructions posted on the course web page.

Attribution

Adapted with permission from Princeton COS 126, "Conditionals and Loops" at `http://www.cs.princeton.edu/courses/archive/fall10/cos126/assignments/loops.html`.