Computer Science 217
Assignment #2

John Aycock

Due: 4pm Friday, 5 March 2021 (Calgary time)

Purpose

To write code to specifications; to understand and implement simple methods for error checking; to apply loops of different kinds and Boolean expressions.

Important Notes

- This is an individual assignment. What you submit must be your own work, and you **must** write the code yourself, although you may discuss the problem in general terms with other people. You should definitely **not** be showing other people your code, and generally speaking, it is not a good idea to talk about the assignment when you're sitting in front of the computer.
- Any references that you use for algorithms and code references, if any, must be properly cited. Remember that plagiarism regulations apply to code too. There's no required citation format but the citation must uniquely identify the source. You can put citations into comments in your Python code. Note that a reference would be used to answer a minor question like "how does this Python construct work?" If you are copying and pasting code into your assignment, this is **not** writing the code yourself, and completely defeats the purpose of what you're supposed to learn in this course. Using the 217 lecture/tutorial videos and slides as references can be safely implied from this context, I think, and you need not cite those unless you're drawing a nontrivial amount of code from them verbatim.
- If you have any questions about what you can and can't safely do, feel free to email me.
- Per course policy, late assignments are **not** accepted. Check the due date and time carefully.

Assignment

Turtle graphics featured in a programming language called Logo. One thing computer scientists do is implement programming languages, and you will be implementing a (drastically) scaled-down version of Logo called Picologo that can be used to draw pictures.

Your program will read Picologo programs from the standard input, either the terminal (i.e., keyboard) or from files containing Picologo commands that are redirected into your program. For marking, your program will be run by typing, on the CPSC Linux command line:

```
spy3 as2.py < testfile.pico
```

Important: if you use a greater-than sign by mistake, your data file will be erased! You have seen this method of running Python programs in lectures and tutorials.

Sample Picologo files are available. To copy them into your current directory, run the Linux command

```
fetch as2
```

Your program must correctly implement the Picologo specification below, and your code must contain comments to identify what the different parts of it are doing. *Note that you know everything about Python that you need for this assignment already.*

Output from a sample solution to this assignment may be seen in the Assignment 2 video.

The Picologo Specification

A Picologo program consists of zero or more commands ending with the `x` command. Any input following an `x` command is ignored. The Picologo commands are processed in the order in which they appear.

Picologo programs may also contain blank, zero-length lines that should be skipped and do not otherwise affect processing of the Picologo program.

A nonblank line will contain exactly one Picologo command. A Picologo command may be one of the following:

- `b` – moves the turtle backward 1 unit
- `d` – places the turtle's pen down
- `f` – moves the turtle forward 1 unit
- `l` – turns the turtle to the left by 1 degree
- `r` – turns the turtle to the right by 1 degree
- `u` – raises the turtle's pen up
- `x` – exits the Picologo program

All movements are relative to the turtle's current location and heading.

Additionally, a Picologo command may be preceded by one or more lines containing single-digit *repeat counts*. If present, a repeat count indicates that the Picologo command immediately following it should be repeated that many times. For example, the input sequence

```
4
f
```

indicates that the `f` command should be repeated 4 times.

Repeat counts may also be applied to repeat counts themselves. For example, the input sequence

```
4
2
f
```

indicates that the `f` command should be repeated 8 times. There is no limit as to how many consecutive repeat counts may appear.

The initial Picologo state consists of a 640x480 window, with the turtle located in the center of it pointing to the right with its pen down.

Error diagnostics must be reported if any of the following conditions are true:

- the command on a line is longer than 1 character;
- an unknown command is encountered.

In both these cases, the error diagnostic message must report the error type and the input line number on which the error occurred. Erroneous lines are skipped and Picologo processing resumes with the next line. If the repeat count is nonzero when an error is encountered, the repeat count is reset to 1.

If errors have been encountered during Picologo processing, a summary message should be printed afterwards indicating how many errors were seen in the Picologo program. This summary message should not be printed if no errors were encountered.

Tips for Getting Started

Click to view (spoiler warning!)

You are free to ignore this advice; it's only here to give guidance and is not the only way to go about this assignment.

1. Watch the behavior of the sample solution on the test files.
2. Read the Picologo spec.
3. Fetch the test files.
4. Look at some of the test files. (In general, *always* look at your data!) I suggest starting with the ones with the filenames containing "basic" and "error".
5. Re-read the Picologo spec and use the contents of the shorter test files to work through what each line means and why, in the case of the "error" files, errors are found.
6. Review the notes on Boolean logic, particularly examples that read input until a sentinel is reached. *The structure of your program will mimic these.*
7. You might start writing your solution at this point, focusing on reading through non-error test files without performing any output. Test your program to ensure it doesn't crash on the non-error files.
8. Add turtle output to your program for all but the repeat count commands. Test your program using the non-error test files that don't contain repeat count commands.
9. Add error checking to your program. Test it on the "error" test files.
10. Add support for the repeat count commands. This will probably necessitate a number of changes across the program, and you may want to make a backup copy before performing drastic surgery. Test your program.
11. Test your program carefully using all the test files and ensure that it behaves correctly.

Again, you know everything you need to know about Python to do this assignment. If you find yourself about to search for something in Python we haven't covered in class: stop and review your notes again.

Assignment Submission

To hand the assignment in:

1. Place your name, student ID number, and tutorial number into comments at the top of your Python program.

2. On the CPSC Linux machines, assuming your Python program is named `as2.py`, run:

   ```
   submit as2.py
   ```

   You can resubmit your file as many times as you like prior to the deadline. Note that a new file submission overwrites the previous file submission, so be sure to only submit your Python program and not any other files!

   Afterwards, if you want to see what you submitted, run:

   ```
   seesub
   ```

Evaluation

You cannot be given a grade above zero if:

- You do not submit a Python program with the required information.
- Your program will *only* work for the supplied test files, and not any others meeting the Picologo specification.
- Image data is not rendered using the `spy3` Python turtle module.

Marking: there are 12 test files supplied, 1 point each. You get the point for a test file if:

- It runs on the CPSC Linux machines using `spy3` without crashing.
- It runs on the Linux command line as described (i.e., by redirecting input using the less-than sign).
- It produces the correct turtle output when run, if any.
- It produces the correct error diagnostics, if any. An error message's phrasing may differ from the sample solution as long as it contains the same information.
- It does not produce any extraneous output (e.g., printing an error summary when there are no errors).

There is also one point for your code being commented appropriately. In other words, there are 13 points in total.

John Aycock, 07 February 2021