CS-101 CS for All | Project 1: Drawing

Due date: Due Monday, September 12 at 11pm

If you do not have a Gradescope account (thanks to using it in some other course), please create an account at `gradescope.com` using your Hamilton email address. Then join our course using the code `4VJW2X`. Please use your real name instead of a nickname.

You can submit your code to Gradescope more than once, and we will only see the latest version. However, if you submit after the due date, then your submission will appear to us to be late.

Please don't use the internet for anything more than looking up syntax and simple commands from reference websites. You should definitely not search online for anything resembling strategy. Keep such questions in-house by asking peers, teaching assistants, and professors.

---

1 Instructions

Use a turtle and the tools that you have learned so far in this course to draw something. If you would like to do something beyond a static drawing, go ahead, but check with us first if you have any doubts about what we would consider "beyond" a static drawing.

The tools that you should use are:

1. **Loops**: The drawing problem that you come up with should require at least one loop that helps you avoid a lengthy copy/paste chunk of code.
2. **Conditionals**: Your problem should also require at least one conditional statement that does not do something trivial, but rather helps you make one or more important choices.
3. **Functions**: Write at least one helpful function that does a job or returns one or more values that simplifies your overall task in a substantial way.

You are of course welcome to ask any of us, if need be, what we mean by words like "substantial" and "important". We will bear in mind that this is your first project in what may be your very first programming course. Feel free to use other tools, such as randomness, that you learned in class or learn about through some other source.

2 Collaboration and Citations

Please see the syllabus for details. You are welcome to work with another person on this project. Each of you should submit your code (it should be identical) individually, and make your own video.

3 Assessment of Code

We will grade each of the following categories on a scale from 0.0 to 4.0 (by 0.5s) that imitates how grade point averages are determined. We will think of a 3.0 as being very good with some flaws, while a 3.5 or 4.0 each denotes excellence. Grades lower than 3.0 simply indicate greater flaws.

- **(4.0 points) Focus**: *Your program is designed to do the overall job that we asked you to complete.* Some projects will have more complex instructions than others.
- **(4.0 points) Accuracy**: *Your code non-trivially satisfies our instructions.* Part of learning the content of this course is figuring out why we teach you the tools we do. You should use the tools appropriately for their intended purpose, which means that you should carefully consider what tasks you set for yourself.
- **(4.0 points) Ambition**: *You stretched yourself in what you chose to create, in its complexity and its details.* It is next to impossible to associate a time limit with ambition, as you have varying degrees of experience, so just challenge yourself and do your best. You should aim to accomplish at least one thing that you didn't directly learn in class or at lab.

4 Assessment of Video

Once your code is finished, you should use Zoom to create a video in which you share your screen and demo your program for us. If you worked with a partner, then each of you should make your own video. This is the rubric that we will use to assess your video:

- **(0.5 points)** *You explained (if need be) how to interact with your program.* You may not need to explain anything, of course, if all we would have to do is hit the Run button.
- **(0.5 points)** *Your video was an appropriate length.* Aim for at most five minutes. If you go past five minutes, your content had better be worth it!

- **(0.5 points)** *Your tour of your output was clear and helpful.* In some cases, the output will basically speak for itself, and if so, you do not need to elaborate much. But in other cases, your program will do something complex, so you should make sure that your narration tells us clearly what it can do.
- **(0.5 points)** *You demonstrated how your program met the demands of our instructions.* You can do this by pointing out certain features of your output and/or by showing us your code. If you do show us your code, don't scroll around like a squirrel. Stay calm and give us a chance to read.
- **(2.0 points)** *You sufficiently explained how the most complex part of your code worked.* Pretend that you are explaining your code to a friend who knows Python but has never seen your code, and tell them the overall purpose of the code as well as how each line of the code itself contributes toward that purpose.

Please do **not** walk us through all of your code. Follow the instructions above, and target what we asked for. If your video ends up being much less than five minutes, yet it successfully addresses our instructions, then we will be completely happy with that.

Put your video on Google Drive and, when you submit your code to Gradescope, you will have a chance to share that link with us. Be sure that you did this correctly by testing the link with someone else you know.

Anything you'd like to add to the above assessment is most welcome, including explanations of how you went above and beyond, or narrated tours of parts of the code that you're proud of.