Turtle Party!

Discovering Python using the turtle library

Brayan Quevedo Ramos

Porterville College

Engineering: Introduction to Programming, 30214

Lab Report

Date: 1, March 2022

Contents

1 Assignment Overview

1.1 Purpose of Program

1.2 Requirements

Listed below are the formal requirements given:

- Must Have Nested Loop
- Must iterate through a list of [pen] colors
- Must include one Parametric System
- Must include at least two different shapes (square and circle are forbidden from counting)

1.3 Theory

Within this project, there were a few extra targets to accomplish that were not specifically stated. For example, the code imported used the Threading library. Threading is the process of splitting up multiple processes within different CPU cores. In this program, instead of taking up CPU time by waiting one core to parse, process, and manipulate code, it was assigned to different CPU threads for parallel execution. Furthermore, the Random library was pushed to it's extreme by utilizing it to make dynamic decisions based on sensitive initial conditions.

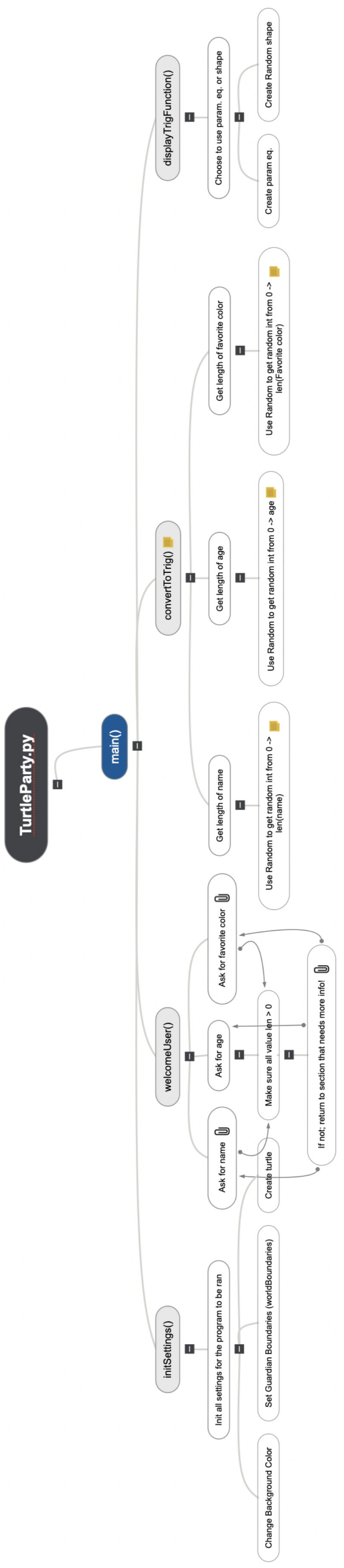
2 Solution Overview

2.1 Program Overview

The solution presented included all of the requirements but implemented in an extremely unique way. To begin, once the program begins running, two windows should appear: terminal and the turtle window. The terminal will ask the user for a few personal questions (e.g: name and age). Assuming validation checks passed, it'll use these responses as input conditions to create a parametric system, using sin and cos. The input conditions will create the general shape of the image, while Random elements of shapes will appear throughout the process. Throughout this process, the color of the line being drawn will change to a random color from a set list.

2.2 Flowchart

Located on the next page is the flowchart that dictates how this program is supposed to run:


TurtleParty.py
main()
displayTrigFunction()
Choose to use param. eq. or shape
Create param eq.
Create Random shape
convertToTrig()
Get length of favorite color
Use Random to get random int from 0 -> len(Favorite color)
Get length of age
Use Random to get random int from 0 -> age
Get length of name
Use Random to get random int from 0 -> len(name)
welcomeUser()
Ask for favorite color
Ask for age
Ask for name
Make sure all value len > 0
If not; return to section that needs more info!
Create turtle
initSettings()
Init all settings for the program to be ran
Set Guardian Boundaries (worldBoundaries)
Change Background Color

2.3 Implementation Issues

There were two extremely difficult issues when creating this program: recursion and authentication.

2.3.1 Recursion Issues

Recursion is the process of using one input condition to perform an output, and use the output as the input for the next iteration. This was extremely difficult to implement as I had to consistently change the method my program worked to create curved lines.
The curved lines worked by finding the X & Y distance between two points, and using that to create the hypotenuse for the triangle. Then, find the midpoint between the X line and the hypotenuse and add a dot (turtle) there. Repeat for the entirety of the triangle base.
The issue that this method arose was that while recursion worked great, (as opposed to my previous iterations of this algorithm), it's horribly inefficient as the inputs MUST be whole numbers, and the curve may only travel in 4 directions. Even if these conditions were met, there was immense difficulty attempting to find a solution that didn't "over-engineer" the drawn curved line. For every coordinate in the X distance, a small part of the line would be drawn. This make the curve extremely smooth, but more CPU time is utilized for negligible return in quality. Ultimately a loop was implemented to divide the X distance into "chunks" depending on the quality of the curve desired.

2.3.2 Authentication Issues

One of the most difficult problems with high-level programming languages is that for very niche edge-cases, the lack of language control reeked havoc with the vision for the program as ValueError in Try/Except statements only states that an error occurred, but no further details were elaborated. This caused major issues in this particulate scenario as there was a function attempting to check whether the inputs were either 'str' or 'int', and WITHIN those variables, there needed to be a check to see if len() could be performed. To solve this, instead of looking at memory addresses for authentication, the same try/catch statement was implemented as planned, the program assumed the input was "True", and break if ValueError was checked as false. See below for more details:

```
while True:
    try:
        localIntUserSettings = int(input("What’s your age?: "))
        break
    except ValueError:
        print("Sorry! You chose an invalid option. Please try again \n")
        continue
```

2.4 Known Bugs

Besides the implementation issues (listed above), at this time there seems to be no bugs. There were no bugs throughout the process as only integers were selected to be the base of all computations (for efficiency and ease-of-use).

2.5 Edge-Case Testing

As there are only three inputs that the user has control over, it's difficult to have edge-cases. If the user has invalid input, the program will alert the user and ask to re-input the specific field.

3 Conclusion

3.1 Lessons Learned

Even though I could consider myself an intermediate programmer, I do believe that I did learn some valuable lessons. To begin, I learned the basics of how multi-threading works. It's fairly simple in Python as one doesn't need to know the hardware level, simply assign and run! Secondly, I gained a very strong understanding on how Python handles data types. I used to look at the first few bits of the address to determine what data-type it was. Instead, now all that must happen is assume it's the data type you want it to be, and if it's not ask user to repeat input – less intuitive but easier to read. Finally, I learned to actually think and develop a program that randomly generates an image each time ran regardless of input parameters. Also, Random isn't actually random, as there was extremely few instances where the center curve deviated to the outer-edges of the program. It tended to always be more on the extremes rather than in one quadrant.

3.2 Redoing The Program

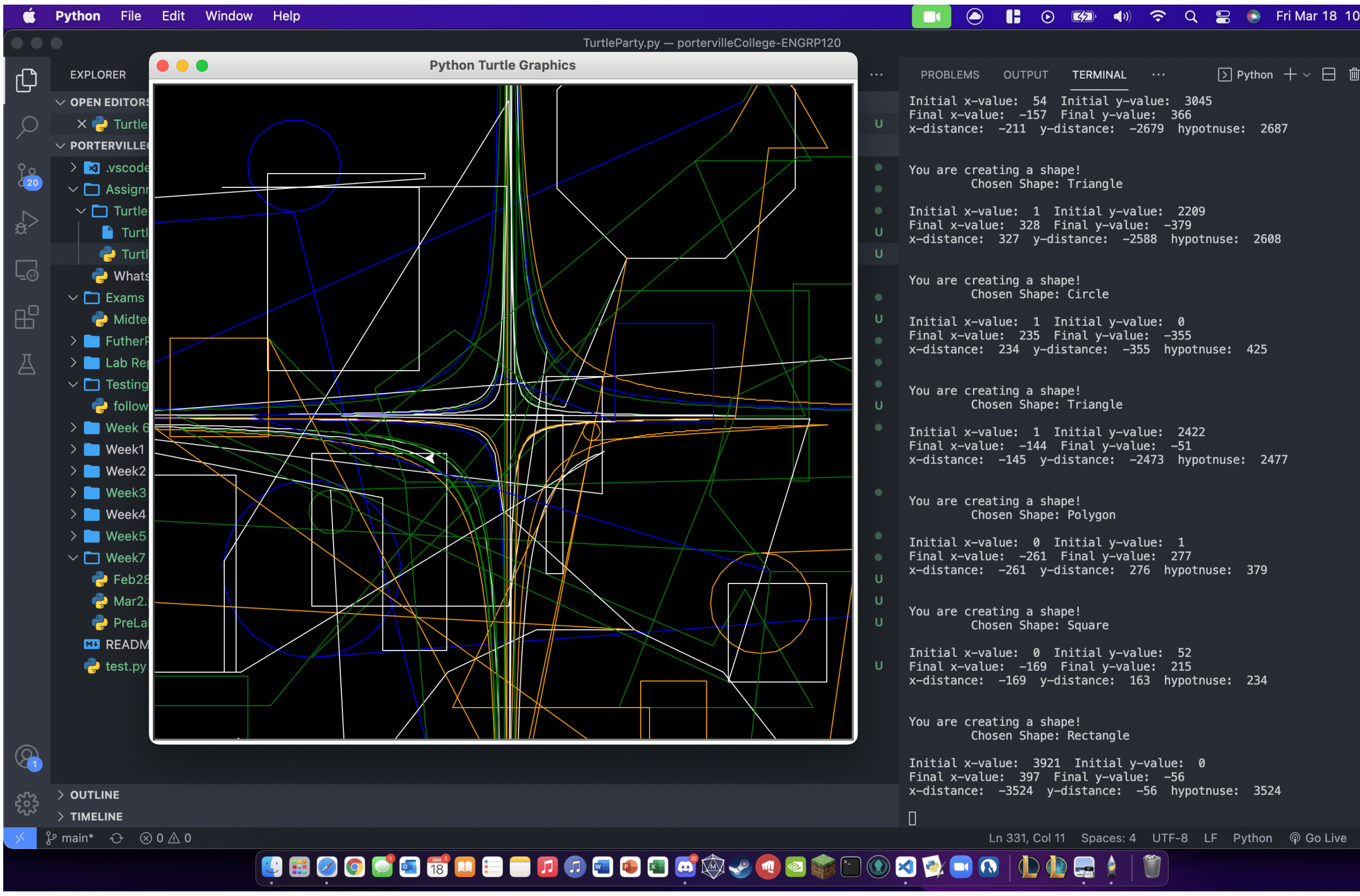
If I was given the exercise again, I don't really think I would've done much different. An easy area to improve on is to not make Random calls constantly, but even then it defeats the purpose of having totally unique images. Personally, I think the implementation could've been better, but due to my skills, not sure if there were more efficient ways to improve. For example, all my algorithms for drawing had a "precision" indicator to either improve performance or make better curves. If there was more time, I could've tried alternative methods to make curves instead of the "triangle" method that would've saved processing power. Furthermore, there were opportunities to use threading much more, but since I am still a bit basic on it, didn't want to mess with that library just yet.

3.3 Revision of Exercise

Personally, I really liked this exercise as it pushed me to develop better skills of how to create never-before-seen algorithms. Instead of knowing what to Google, I had to sit down and truly think about what my desires for this program were, and what structure would allow me to quickly make this.

4 Results

4.1 Program Running



```
Initial x-value:  54  Initial y-value:  3045
Final x-value:  -157  Final y-value:  366
x-distance:  -211  y-distance:  -2679  hypotnuse:  2687


You are creating a shape!
         Chosen Shape: Triangle

Initial x-value:  1  Initial y-value:  2209
Final x-value:  328  Final y-value:  -379
x-distance:  327  y-distance:  -2588  hypotnuse:  2608


You are creating a shape!
         Chosen Shape: Circle

Initial x-value:  1  Initial y-value:  0
Final x-value:  235  Final y-value:  -355
x-distance:  234  y-distance:  -355  hypotnuse:  425


You are creating a shape!
         Chosen Shape: Triangle

Initial x-value:  1  Initial y-value:  2422
Final x-value:  -144  Final y-value:  -51
x-distance:  -145  y-distance:  -2473  hypotnuse:  2477


You are creating a shape!
         Chosen Shape: Polygon

Initial x-value:  0  Initial y-value:  1
Final x-value:  -261  Final y-value:  277
x-distance:  -261  y-distance:  276  hypotnuse:  379


You are creating a shape!
         Chosen Shape: Square

Initial x-value:  0  Initial y-value:  52
Final x-value:  -169  Final y-value:  215
x-distance:  -169  y-distance:  163  hypotnuse:  234


You are creating a shape!
         Chosen Shape: Rectangle

Initial x-value:  3921  Initial y-value:  0
Final x-value:  397  Final y-value:  -56
x-distance:  -3524  y-distance:  -56  hypotnuse:  3524

▯
```