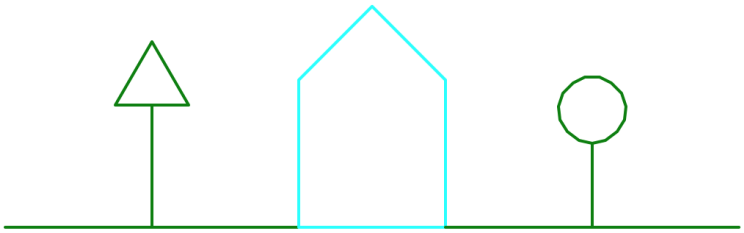
Computational Problem Solving CSCI-141
Forest Scenery Lab 1

August 2020

1 Problem

For this assignment, you will draw some pictures to make a scene in a forest. It will contain a couple of trees and possibly a house.



1.1 User Input

Users will specify the forest scene through a sequence of questions the program asks them.

- There are two trees, randomly chosen by the program to be pine or maple.
- The program asks the user if they want a house. If the user responds that they want a house, they must enter more information.
  1. The user specifies the house's position relative to the trees by entering a numerical code and pressing the *Enter* key. The codes are:
     - 1: to the left of both trees
     - 2: between the trees
     - 3: to the right of both trees
  2. The user also specifies the house's color.

A demonstration movie is available here.

1.2 Drawing Details

The program chooses the type of each tree at random each time the program runs. Here are the specifications for the drawing.

Trees and ground will be drawn in the color green.

- Pine trees have a trunk between 50 and 200 pixels tall and a triangle at the top.
- Maple trees have a trunk between 50 and 150 pixels tall and a circle at the top.

There should be 100 pixels between tree trunks as well as between the house walls and its neighboring tree(s).

The pine tree top is an equilateral triangle, with a horizontal bottom. The length of each side of the triangle is 60% of the height of the trunk.

The maple tree top is a circle whose radius is 40% of the height of the trunk.

The house will be a symmetric pentagonal shape with a 45 degree roof angle, as shown in the picture above. The house's vertical wall height will be 100 pixels, and the base of the houses (not including the roof) should be a square. It will appear in the position the user stated.

2 Problem Solving

For the problem-solving session, you will work in groups as directed by your instructor.

1. Write the code for drawing a pine tree — note where the turtle is assumed to start, and make sure the turtle ends in the same configuration.
2. How much ink (total drawing distance in pixels) is used to draw the house?
3. Write the formula for the amount of ink needed to draw a maple tree of trunk height h.
4. Write the formula for the amount of ink needed to draw a pine tree of trunk height h.
5. Your tree-drawing function should return the amount of "ink" used to draw the tree, in units of pixels. Modify the function you wrote originally to return this information.
6. Indicate which parts of the code you just wrote could also be used for drawing a maple tree.

3 Post Problem-Solving Work

This section contains guidelines for what to do. There is a separate dropbox in the mycourses Assignments area for it.

Using the Python development tool recommended by your instructor, create a code file named **scenery.py**.

Copy the code you wrote during the problem-solving session into it. Run the program just to make sure there are no syntax errors. There will be no output.

Write a `main` function that asks the questions as shown below. User input is what is shown after the question marks; everything else is program output. Note that if the user does not want a house, the other two questions should not be asked by your program. This means you will need some conditional (`if`) code. Save the answers in variables. However at this point you don't have to do anything with that stored information.

```
Is there a house in the forest (y/n)? y
At what position (1/2/3)? 2
What color is the house? cyan
```

Add this standard conditional execution code at the end of the file:

```
if __name__ == '__main__':
    main()
```

Again, you do not have to call the functions you entered earlier. Just make sure that the program now prompts the user appropriately.

Submit this partial solution[1] to the appropriate dropbox in the mycourses Assignments area.

4 Implementation

Your implementation should prompt the user as shown above.

Your output should perform the drawing and then print the amount of "ink" used to make the drawing.

We define amount of ink as the sum of the lengths of all the lines showing in the scene. This definition has one important consequence. If your algorithm for the drawing has the turtle retrace some steps with the pen down, that retracing does *not* add to the amount of ink *needed* to draw the scene.

1. the plain **scenery.py** file itself; not a zip file

For example, in this small piece of code

```
turtle.pendown()
turtle.forward( 200 )
turtle.penup() # <---
turtle.right( 180 )
turtle.forward( 200 )
```

the amount of ink reported should be 200, whether the third line is present or not.

Here is a complete sample of the interaction between the user and your program.

```
Is there a house in the forest (y/n)? y
At what position (1/2/3)? 2
What color is the house? cyan
We used 481.4213562373095 units of ink for the drawing.
Close the diagram to end the program.
```

The run above generated the graphic shown in this document. Because of certain random choices, it will not generate the exact same picture and output every time.

4.1 Restrictions

You may not use the `goto` and `heading` functions of the turtle. That is, all movement should be relative.

4.2 Design

You must have a main function that collects the user input and then calls other functions to do the work. The final output, the `print` statement(s), comes from `main`.

Your solution should contain a single function to draw all types of trees, with shared code as much as feasible between the different types.

4.3 Helpful Tools

Powers in Python are computed with the `**` operator. For example, `2**2` returns the value `4`. To compute square roots in Python, you must first import the math package. Enter this line at the top of your program.

```
import math
```

When you want to compute a square root, type

```
math.sqrt( n )
```

You prompt the user for input using the `input(message)` function, where message should be a quoted string that is printed to prompt the user for a response.

One way of generating random numbers in Python is to use the `randint` function from the random module.

The following code prints integers chosen randomly from the range $[10, 20]$. It was run directly in the Python interpreter ("Python Console", if using PyCharm).

```
>>> import random
>>> BIGGEST = 20
>>> SMALLEST = 10
>>> random.randint( SMALLEST, BIGGEST )
18
>>> random.randint( SMALLEST, BIGGEST )
19
>>> random.randint( SMALLEST, BIGGEST )
12
>>> random.randint( SMALLEST, BIGGEST )
17
```

You may wish to adjust the size of the canvas with the turtle function `setworldcoordinates`. For more information, type the following into the Python interpreter.

```
import turtle
help(turtle.setworldcoordinates)
```

Finally, to terminate your program, use the function `turtle.done`. It waits for user to close the window containing the turtle canvas.

5 Submission

Submit your documented code to the designated drop box in mycourses **Assignments**. This involves the following steps.

1. Create a zip file containing just your **scenery.py** file.
   Do not use another compresssion tool such as gzip, rar, or 7zip.
2. Locate the dropbox named "Lab 01" in the Assignments section of the CSCI 141 shell in mycourses.
3. Submit your zip file there.
4. Make sure mycourses confirms the file has been stored.
5. Double-check by ensuring that you got an email confirming the submission.

6 Grading

Your grade will be determined as follows.

- 15% : problem solving participation and solutions
- 10% : post-problem-solving setup work
- 10% : user input and basic setup
- 30% : correct tree drawing
- 20% : correct house drawing in random location
- 10% : appropriate code modularity and reuse
  - `main` function only sets things up and calls other functions to do the work, as described in Section 4.2.
  - Separate function to draw the house
  - Some code shared by both kinds of trees
- 5% : adherence to style guidelines posted on course website
  - Student name and program description at top of file in a docstring
  - Docstring for each function
  - Global constants defined in named variables
  - (See `http://www.cs.rit.edu/~csci141/Docs/PythonRecommendations.txt`.)