Simon Fraser University
School of Computing Science

Cmpt 120 – Assignment 3

Nature Art

We are to do Assignment 3 on our own, not in a group.

Objective

In our Assignment 3, our objective is to practise solving a problem by creating a whole Python program using what we have learnt so far:

1. Python building blocks,
2. Iterative statements, especially the **`for` loops**,
3. Functions,
4. The 4 steps of the software development process, and
5. The Python Turtle Graphics module along with its built-in functions.

Before Doing This Assignment 3 …

… we must do **the Week 8 Lab**. Why? Because the Week 8 Lab contains information, clues, and exercises about the Python Turtle Graphics module that will greatly help us in this Assignment 3.

Nature Art

In this Assignment 3, we are to write a Python program, called **NatureArt.py**, which is to create a drawing representing an **outside nature** scene using the Python Turtle Graphics module and its built-in functions.

What do we mean by “nature scene”? We mean that our Python program can be a drawing of anything as long as all the elements can be found outside, in nature (e.g. humans, buildings, parks, plants, trees, sun, animals, grass, mountains, etc). The drawings cannot be indoor scenes.

This is to say that our nature scene cannot be a bunch of shapes drawn together with no theme like the drawing created by the program **Turtle_Lab_5.py** seen in our Week 8 Lab.

Our nature scene (program) is to also satisfy the following requirements.

Requirements:

1. We must use Python 3 (the Python IDLE version used in our CSIL lab) to write our program.

2. Our program is to execute as soon as one has pressed the F5 key or has selected the option *Run -> Run Module* from the Python IDLE menu. No user input is required in this assignment. This is to say that the user must not be prompted for any input data once our nature scene Python program has started executing. It executes all on its own.

3. Let us make sure that our drawing is not too large and can be seen in its entirety in the window without having the TA either resize the window or make use of the scroll bars to view our drawing.

   We may ask ourselves How large will the computer monitor the TA's will use to mark our assignment be? Answer: as large as the computer monitors in our CSIL lab.

   Download and execute the Python program Example_of_Turtle_Drawing_too_largeScreen_too_small.py posted below our Assignment 3 to see an example of what we mean by a drawing that is too large and that cannot be seen in its entirety in the window without having the viewer to scroll.

4. We cannot make use of Python modules that must first be downloaded on a computer before they can be used. We can only use Python modules that are already available on the computer (available by simply using the Python statement `import` in our Python program).

5. Our drawing must be composed of at least 30 shapes (Each shape must be labelled in the code using comments: e.g. # Shape 31. A red football in the air.).

   If we look at the drawing created by the program **Turtle_Lab_5.py** seen in our Week 8 Lab, we can count 9 shapes:

   - 2 rectangles,
   - 1 square,
   - 3 circles,
   - 2 semi circles (note that 2 semi circles drawn together count as 1 shape, but here, the 2 semi circles are not drawn together so they count as 2 shapes) and
   - 1 quarter of a circle.

6. Our drawing must include at least 10 colours.

   If we look at the drawing created by the program **Turtle_Lab_5.py** seen in our Week 8 Lab, we can count 8 colours.

   - Some of the shapes our program draws must be filled in.

     If we look at the drawing created by the program **Turtle_Lab_5.py** seen in our Week 8 Lab, we can count 1 filled in shape.

7. Our program must include at least 6 `for` loops.

   If we look at the program **Turtle_Lab_5.py** seen in our Week 8 Lab, we can count 2 `for` loops.

8. Our program must include at least 8 functions:

   - 4 of them must be **void** functions: functions that do not return any "value" (possibly doing drawings and/or pen movements, or creating more complex shapes),
     - Note that these 4 functions may or may not require parameters.
   - 6 of them must require parameters (the number of parameters is up to us).

- Note that these 6 functions may or may not be void.

If we look at the program **Turtle_Lab_5.py** seen in our Week 8 Lab, we can count 3 void functions that require parameters and 3 incomplete functions. Of course, we cannot submit a program with incomplete functions. ☺

9. We have applied the generalization guideline when we created all our functions. This is to say that our functions are as general as they can be. They are designed in such a way (with parameters and/or returned value) that they can solve several similar problems.
10. None of our code is repeated. This is to say that we encapsulated repeated code into functions and have called these functions whenever needed.
11. All our functions terminate with a `return` statement.

Since the drawing we create in this assignment is completely up to us, no two drawings will be similar!

Enjoy the creative process!

Incremental Development

I would strongly suggest we implement and test our Python program in an incremental fashion, i.e., we implement a few features (perhaps shapes or functions) of our drawing, test them, then we implement a few more, test them, etc., until the whole drawing is done.

This incremental software development strategy is in contrast with the "Big Bang" approach where the whole program in written in one go and only tested at the end. This approach often leads to lengthy debugging sessions.

Art Gallery

After the due date, we shall have Art Gallery moments at the beginning of our classes in which we shall present the nature scenes.

So, if we are proud of what we created in this assignment and wish to have it presented in class, please, let us email our nature scene (i.e., Assignment 3 Python program) to the instructor.

Submission

- Let's submit our file: **NatureArt.py** on CourSys by Friday, July 13, no later than 16:30.
- We can submit our work to CourSys as many times as we wish, but it is the last submission that counts. So, let's make sure that our last submission is done before the deadline.
- No late submission will be accepted: any submission made after the due date and time stated above will not be marked for grades. However, they will be marked to provide feedback to students.

- The only exception is for medical reasons. If we are not able to submit our assignment on time due to a medical reason, we must notify the instructor by email right away and provide her with a doctor's note as soon as possible.

Marking

When marking Assignment 3, we will look at some of the following criteria:

- All the requirements of this assignment are satisfied. These requirements are rules to which our Python program must abide and they are listed above in the section called **Nature Art**
- We have used the GPS guidelines described on our course web site. Let's make sure we have read these guidelines posted on the GPS web page and that we are familiar with them.
- All our functions are located at the appropriate place in our Python program.
- We call our functions either from the "# Main" part of our program and/or from our user-defined functions. All are our functions are called at least once.
- We have included a header comment block, which contains the name of the file, a description of the program, the name of the author and the creation date of the program.
- Our code easy to read. It is well commented
- We have selected the most appropriate/efficient conditional and iterative statements for each situation in our Python program.
- We are the only author of the code we have submitted to CourSys.
- Our Python program is a real program and no simply a screenshot of the Python Interpreter Shell window onto which we have entered some Python statements.
- Our Python program executes. It has no syntax and/or runtime errors.
- Our Python program solves the problem. It has no semantic errors.

Have fun!
