Introduction to Python Computer Programming

North Star High School

Unit 4

Case Study – Interface Design

# Reading Material

The text for this unit is Think Python chapter 4 (pages 29 through 38).

It is encouraged to work on your reading with the Python interpreter open on your computer, so that you can type in the examples and experiment as you read.

You will gain your first taste of graphics in this unit, using the built-in “turtle” module of Python.

Full documentation of the “turtle” module can be found at the following location:

<https://docs.python.org/3.3/library/turtle.html?highlight=turtle>

# Guided Reading

Please complete the following questions using the assigned reading above.

Note: The following statement:

**>>> import turtle**

Causes the “turtle” module to be loaded into your Python environment, so that you can run the examples and complete the assignments in Chapter 4 of the text.

1. In section 4.1, printing the bob variable displays something like:

**<turtle.Turtle object at 0xb7bfbf4c>**

this means that bob refers to an \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (object) with type \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Turtle) as defined in the module \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (turtle).

1. A \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is similar to a function, but uses a slightly different syntax. (method)
2. Calling a method on an object is like \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. (making a request)
3. The argument of fd is a distance in \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ , so the actual size depends on your display (pixels).
4. Continue the example in section 4.1, and modify the program mypolygon.py to draw a square.
5. A \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ statement is one simple way to perform repetition. (for)
6. A for statement is also called a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ because the flow of execution runs through the body and then loops back to the top. (loop)
7. Work through the Exercises in section 4.3, saving your program as “ex\_4.3.py”. (ex\_4.3.py)
8. Wrapping up a piece of code in a function is called \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. This practice helps with both documentation and re-use of code. (encapsulation)
9. Adding a parameter to a function is called \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. (generalization)
10. The \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ of a function is a summary of how it is used: what are the parameters? What does the function do? What is the return value? (interface)
11. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is rearranging a program to improve interfaces and make code re-use easier. (refactoring)
12. A \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ is a process for writing programs. Programming takes much more thinking than typing! (development plan)
13. It is useful to include a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ at the beginning of a function to explain the interface. It is important to choose your words carefully! (docstring)
14. Requirements assumed to be true before executing a function are called \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. (preconditions)

**Be sure that you are familiar with all of the definitions in the glossary (section 4.11)!**

# Assignment

1. Exercise 4.1, part 1 (drawing a stack diagram). Note that the file ‘polygon.py’ has already been made available on your classroom page, so you will not need to download it from the textbooks website. Note that the call to ‘circle(bob, radius)’ is near the very end of the file. This is the where you will start your analysis! Also note that you only need to draw the stack diagram for functions that are defined in ‘polygon.py’ and not for library functions such as ‘abs’ (absolute value function) that are used in the code. (answer in ex4.1.png)

Stack diagrams were introduced in section 3.9. They are useful for visualizing the flow of program execution in small programs. These diagrams quickly get too complicated for large programs!

Here is a small example to help you get the hang of stack diagrams:

Consider the following example Python program:

|  |
| --- |
| def my\_add\_then\_print\_function( input1, input2):  result = input1 + input2  print(result)  a = 4  b = 5  my\_add\_then\_print\_function(a, b) |

Running this program will print the number 9.

The stack diagram for this program shows the assignment of the variables ‘a’ and ‘b’, then the function call of ‘my\_add\_then\_print\_function’, then the mapping of variables inside the function.

In a python program, variables defined at the “top-level”, that is to say outside of any function, belong to the special, built-in function named “\_\_main\_\_”. We will learn more about this later. For now, note that the variables ‘a’ and ‘b’ belong to “\_\_main\_\_”, as seen in the stack diagram.

|  |
| --- |
|  |

If the definition of the function ‘my\_add\_then\_print\_function’ is first in the program, why doesn’t it come first in the stack diagram? Think of the definition of the function like a blueprint or template. The function is not actually used with real data until it is called in the last line of the program, and the variables ‘a’ and ‘b’ are passed to it.

1. Exercise 4.4 in the textbook. (solution in typewriter.py and letters.py)
2. Use your solution to exercise 4.4 to create a Python program that writes your name! (solutions will vary)

If time allows, please complete exercise 4.2, 4.3, and 4.5. Create your own solutions before looking at the author’s work!

(solutions: 4.2 in flower.py, 4.3 in pie.py, and 4.5 in spiral.py)