Computer Programming
Spring 2020

CSI2100-01 Lab 8

Bernd Burgstaller
Yonsei University

Overview

- Questions
- Programming Problems
- Deliverables, Due Date and Submission
- Notes:
  1) Please consider using the OnlinePythonTutor when debugging your code.
  2) Some lab problems can be tested with Hyeongjae Python.
     - Please refer to the last page of this assignment for details.
  3) Because of our coronavirus precautions, you are provided with an excerpt of the relevant sections of the textbook together with this lab assignment.

Questions

Please submit the answers to the following questions in a file named **README.txt**

Question 1: **What** do the following programs print? For **all occurrences of all variables**, please name the Python frame to which the variable belongs. Please use the provided line numbers to refer to the occurrences of all variables. (For example: Line 2, variable n: frame ...).

Hints: please have a look at our discussion of scopes, local variables, and nested functions in the lecture. Stepping through the code in the PythonTutor (www.pythontutor.com) will be helpful, too.

```
1  def foo():
2    return n * 2
3
4  def foo1(n):
5    n = n + 1
6    return n
7
8  n = 10
9  x = foo1(n)
10 y = foo()
11 print(x, y)
```

(a)

```
1 def foo():
2   def bar():
3     y = x + 1
4     print(y)
5   x = 1
6   bar()
7
8 x = 0
9 foo()
```

(b)

```
1  def foo():
2    def bar():
3      y = x + 1
4      print(y)
5    #x = 1
6    bar()
7
8  x = 0
9  foo()
```

(c)

Programming Problems

Problems 1, 2, 3 and 6: you are asked to implement a function. To solve these problems, you must read the problem specification to understand:

1) What is the name of the asked function?
2) What input(s) does the asked function receive?
3) What should the function compute?
4) What should the function return?

- Unless requested in the problem specification, in your function you should **not read input from the user** and you should **not print the results of your function**!
- To **test** your function, you must write your own main program, which must call your function with appropriate input values. In the main program, you can then print the value(s) returned from your function to check for correctness.
  - **Do not hand in** such a main program that you used to test your function!
  - **Only the function itself must be handed in.**

Problems 4, 5 and 7: you'll be working on an entire program, so you must **hand in all code** (including the main function).

Problem 1: Write a function **resetValuesInPlace** which receives a list **L** and a threshold value **threshold**. All values in list **L** which are above (>) the **threshold** value should be set to zero (by mutating the list L itself).

Mutating a list is possible because the list parameter and the list argument are aliases of each other (see Lecture 5, p. 47).

Your function shall return the mutated list via a return statement.

Your function **must adhere to the following template** (otherwise automated grading will fail, resulting in 0 points):

```
def resetValuesInPlace(L, threshold):
    """ Pls. put your docstring here..."""
    # Your code here (provide your own comments, please):
    ...
    return L
```

You may wonder why your function should return list **L**, which has been already mutated. This is a **convenience feature** to allow the user to use this function inside an expression. For example:

```
print(resetValuesInPlace(L, 200)) # Does not work if
                                  # None is returned
```

instead of:

```
resetValuesInPlace(L, 200) # Less elegant, two lines
print(L)                   # instead of one.
```

Problem 2: Different from **Problem 1**, your function must **remove** all values from list **L**, which are above the **threshold**. Again, the list must be mutated as a side-effect of the function call, and then returned.

Hint 1: Please have a look at Lecture 4 on ``mutation during iteration'' to make sure you do not mutate list L while iterating over it.

Hint 2: Your function **must adhere to the following template** (otherwise automated grading will fail, resulting in 0 points):

```
def removeValuesInPlace(L, threshold):
    """ Pls. put your docstring here..."""
    # Your code here (provide your own comments, please):
    ...
    return L
```

Problem 3: Different from problems 1 and 2, you are not allowed to mutate list L. Instead, your function should compute a new list where all numbers from L above (>) the threshold are replaced by 0. This new list must be returned (see the below template).

Hint: Your function **must adhere to the below template** (otherwise automated grading will fail, resulting in 0 points):

```
def resetValues(L, threshold):
    """ Pls. put your docstring here..."""
    Result = []
    # Your code to compute ``Result'' here:
    return Result
```

Problem 4: Exercise P1 from page 244 of the textbook.

You may choose your own window size. Please use some code provided to you with Lecture 6 for setting up the window and the turtle. Please use **`exitonclick()`** to prevent the window from immediate closing.

- You should hand in a **complete program** that sets up a window, draws the X on the window, and waits for the user to click to close the window.

Problem 5: Problems M2 and M3 from page 245 of the textbook, together (as one problem):

- Change color randomly each time a ball hits a wall. Please make sure that the new random color is **different** from the previous random color. For example: changing from 'red' to 'red' (same color!) is **not allowed**. (If you use random colors, you must check each new color against the previous color to make sure that they are not identical.)
- Leave a trail on each ball's path (put the turtle's pen down). You do not have to change the color of the trail.

Please refer to the description of the bouncing balls program in the provided **textbook excerpt** for Lab 8.

Problem 6: Implement the following function:

```
def drawSquare(myturtle, x, y, a):
```

Function `drawSquare()` should draw a square on the screen:

- `myturtle` is the turtle drawing object to be used.
- `x` and `y` specify the `x` and `y` coordinates of the lower left corner of the square.
- `a` is the length of each side of the square.
- x, y and a are given in units of pixels.
- You should assume that the turtle `myturtle` is initially hidden.

(continued on the next page...)

Please use the following code template for **submitting** function drawSquare():

```
import turtle
import ...   # import any other module that you use in drawSquare( )
             # (if any)

def drawSquare(myturtle, x, y, a):
    """your docstring here: """
    # Your code to draw the square here:
    ...
# Nothing else!
```

lab8_p6.py

This is a main program that you can use to test your function (**do not submit any such testcode with your function**):

```
...
turtle.setup(800, 600)
window = turtle.Screen( )
the_turtle = turtle.getturtle( )
the_turtle.hideturtle( )
drawSquare(the_turtle, 0, 0, 20)
drawSquare(the_turtle, -30, -30, 40)
window.exitonclick( )
```

Problem 7: Textbook page 245, Problem M5.

Please download and unzip the zip-archive with the source-code from the Lab8 folder on YSCEC. Please follow the description of the horse-racing program in the **provided slides** and the **textbook excerpt for Lab 8**. You can find the different versions of the program on YSCEC (please consider and run them to see how the program evolves.)

Note: you must import the provided source-file (**HorseRace_Final.py**) and the folder with the image files (**img**) into your PyCharm project.

Your program should ask the user whether to run another race. After the user presses 'q', your program shall display the cumulative wins of all horses, close the graphics window and terminate. Use ``**`turtle.bye()`**'' to close the graphics window.

```
<race simulation 1>
Press 'q' to quit: a
<race simulation 2>
Press 'q' to quit: q
Horse 1 won 0 times
Horse 2 won 1 times
...
Horse 10 won 1 times
```

- Please do not close the Turtle window between simulations.
- Instead, you must clear the contents of the Turtle window before starting the next simulation.
- For an illustration, please refer to our instruction video titled ``lab_p7example.mp4'' provided with the video material of week 9.

Continued on the next page...

- Please hand in only the modified Python code
  - File lab8_p7.py
- Do not hand in any gif files, etc.
- Do not change the names of gif files in your program, otherwise your program will fail automated grading.

Marking Criteria and Plagiarism

- Marking Criteria
  - Score is only given to programs that compile and produce the correct output with Python version 3.
  - Points are deducted for programs that are named wrongly. See the list of deliverables for the required file names.
  - Points are deducted for programs that produce warnings.
  - Points deductions on programming style: please provide comments in your code.
    - Please provide docstrings with **all your** functions.
    - It is not necessary to provide docstrings for functions **in code provided to you**.
  - Please pay particular attention to the **requested output format** of your programs. Deviating from the requested output format results in points deductions.

- Plagiarism (Cheating)
  - This is an individual assignment. All submissions are checked for plagiarism.
  - Once detected, measures will be taken for **all** students involved in the plagiarism incident (including the ``source'' of the plagiarized code).

Deliverables, Due Date and Submission

- Please prepare the files for the programming problems and questions. The names of the files, their due dates and the archive file-name is given in the below table.
  - Please upload your archive file by the stated due date on YSCEC.
- Lab problems marked as '✓' can be tested on our Hyeongjae Python site http://hyeongjaepython.elc.cs.yonsei.ac.kr/

Problem	File name	Due	Archive name	Hyeongjae Python
Questions	README.txt	Wednesday May 20, 2020 23:00	lab8_<student id>.zip	—
1	lab8_p1.py			✓
2	lab8_p2.py			—
3	lab8_p3.py			✓
4	lab8_p4.py			—
5	lab8_p5.py			—
6	lab8_p6.py			—
7	lab8_p7.py			—