**COMP130 HW2: Functions and loops  
instructor: John MacCormick**

Please remember to write any Python code in a **fixed width font** (e.g. Consolas or Courier New) and to ensure that it is **indented correctly** and that the lines are **single-spaced**. The easiest way to achieve correct formatting is to copy the code from IDLE, then change it to a fixed width font. In MS Word, you can use the “code” style which is available in the toolbar in the homework assignments for this course. You can also choose “Remove Space After Paragraph” to manually ensure that lines are spaced correctly.

Consider the following snippet of Python code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | def upper\_and\_lower(first\_string, second\_string):  both = first\_string + second\_string  print('Both strings concatenated:', both)  upper = first\_string.upper()  print('First string in upper case:', upper)  lower = second\_string.lower()  print('Second string in lower case:', lower)  upper\_and\_lower('abcDEF', 'pqrSTU') |

Question 1. (2 points) What is the name of the function defined here?

Question 2. (2 points) How many parameters does the function have?

Question 3. (2 points) What is the name of the second parameter?

Question 4. (2 points) What is the intended data type of the first parameter?

Question 5. (2 points) At what line number in the code is the function executed?

Question 6. (2 points) What is the value of the second argument in the function call?

Question 7. (5 points) What is the output of the function defined above when it is invoked with the first argument 'Dickinson' and second argument 'College'?

Question 8. (10 points) Write some code that defines a function called concat3. The function should have three parameters, which are all expected to be strings. The function should print the three parameters concatenated together, with the first parameter converted to uppercase and the last parameter converted to lowercase. Add a line of code that invokes your function with the arguments "Apple", "Banana", and "Cherry". Make sure all indentation is correct.

Question 9. (2 points) What is the output of the program you wrote for the previous question?

Question 10. (2 points) Write some code that prints a random integer between 1 and 10 inclusive.

Question 11. (2 points) Write some code that prints a random integer between -20 and 50 inclusive.

Question 12. (4 points) Write some code that prints a random *even* integer between 2 and 30 inclusive. (That is, it prints one of the numbers 2, 4, 6, …, 28, 30.)

Question 13. (10 points) Consider the following code:

import turtle

def draw\_square(t):

t.forward(50)

t.left(90)

# More to be added here.

This code defines a function to draw a square, but it is not complete. Give the complete code for this function.

Question 14. (15 points) Write a function whose signature is staircase(t, size, num\_stairs). The function should use a turtle t to draw a staircase with the given number of stairs, num\_stairs. The width and height of each stair is given by the parameter size. For example, the output of the invocation staircase(t, 10, 5) should be:



Question 15. (10 points) Consider the following Python code.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import graphics  import turtle  import random  win = graphics.GraphWin("homework", 100, 100)  p = graphics.Point(50, 75)  p.draw(win)  win.getMouse()  win.close()  t = turtle.Turtle()  t.forward(250)  turtle.mainloop()  x = random.randint(1, 5)  print(x) |

(a) How many examples of dot notation are there in the above code?

(b) At which line numbers is a constructor invoked?

(c) At which line numbers is dot notation used to invoke a function in a module? Do not include constructors.

(d) At which line numbers is dot notation used to invoke a method on an object?

(e) At which line numbers is there a function call without any dot notation?

Question 16. (optional) Write a function whose signature is widening\_segments(t, num\_segments). The function should use a turtle t to draw a sequence of line segments. Each segment is of length 30 pixels. The width of the first segment is 2 pixels. The width of the second segment is 4 pixels. The width of the segments continues to increase in this way, with the width of the final segment being 2\*num\_segments pixels. For example, the output of the invocation widening\_segments(t, 12) should be:



Total points on assignment: 72