PA04: Guessing Game: Loops and Conditionals in Python

University of Virginia, Department of Computer Science
CS 1112 - Intro to Programming
Spring 2024

Due by: 11:00pm on Wednesday, March 20, 2024

Objective: The objective of this assignment for students is to learn how to integrate **conditional statements**, **loops**, the **break** statement, and the **continue** statement in Python while building a simple mini-game. By the end of this assignment, students should be able to understand how using loops can more effectively complete tasks that are repetitive and integrate conditional statements within loops to do specific tasks under specific conditions.

1 Task

For this assignment, you will design a program and complete a function that is designed to play a game where the user (you) will think of a number and give the computer a limited number of tries to guess what the number that the user is thinking of.

In a Python file named `guessing_game.py`, write a function called `guess_number_game()` that takes in no parameters. (As always, be sure to **name your file** and define your **function name** *exactly* as mentioned here.)

The program prints the following at the start of the game:

```
PA04: Guessing Game
Think of a number between 1 and 100, and I will try to guess it!
```

Steps for your program:

1. You, the user, begin by thinking of a number between 1 - 100 (inclusive) that the computer will try to guess. Keep this number in mind, you do not input this number anywhere in the program.

2. In your program, create three variables:

   (a) the lower bound of the numbers to be guessed

   (b) the upper bound of the numbers to be guessed

   (c) the current number of attempts, which is initially `0`

   In the beginning, these variables would be set as follows:

   ```
   lower_bound = 1
   upper_bound = 100
   attempts = 0
   ```

3. The computer will "ask" the user (by printing), "How many guesses do I get?", and wait for the user to respond. The user will input (respond with) the number of guesses that the computer gets (e.g., `3`).

   (a) In your program, create a variable that keeps track of the number of attempts the computer gets (don't forget to type cast!) Perhaps, something like this:

   ```
   max_attempts = int(input("How many guesses do I get? "))
   ```

4. The computer will then pick a number (you decide how this number is chosen - randomly, or in the middle of the boundary, etc.) and ask (print), "My guess is X. Is it higher, lower, or the same?" Replacing X with the actual number.

   (a) **Note for clarity:** the question is asking if *the number to be guessed* is higher, lower, or the same as *the number the computer guessed.* For example, if the computer guessed 47, and the number to be guessed is 55, the correct response is "higher" (because $55 > 47$).

5. The user will be able to respond with "higher", "lower", "same", or "Sorry, I gave the wrong instructions" For example, you can use the following `input()` function to guide the user to input the appropriate response:

   ```
   user_input = input("Enter 'higher', 'lower', 'same', or \
           'Sorry, I gave the wrong instructions': ").strip().lower()
   ```

   (Feel free to use this code in your program.)

   Note: the function `.strip()` removes any leading or trailing whitespace (spaces, tabs, and/or newlines) that may have been entered. The function `.lower()` converts what was typed into all lower case for easier comparison.

   (a) If the user answers "same", the game will terminate (end) and the computer will respond with (print) "Too easy!"

(b) Else, if the user answers "higher" (the number is higher than the number guessed by the computer) then the **lower bound** should be equal to the computer's guess plus one:

```
lower_bound = computer_guess + 1
```

(c) Else, if the user answers "lower" (the number is lower than the number guessed by the computer) then the **higher bound** should be equal to the computer's guess minus one:

```
upper_bound = computer_guess - 1
```

(d) Else, if the user answers "Sorry, I gave the wrong instructions", the game will **terminate (end)** and the computer will respond with (print) "It's okay! Let's play again sometime!".

(e) Otherwise, (an invalid input is received that is not one of these four options), the computer will print "It's alright, to err is to a human and not a computer!" repeat the question asked in step 3: "How many guesses do I get?"

The ***boundaries*** of the numbers that the computer can guess needs to be ***reset*** back to 1 and 100, and the current number of attempts needs to be ***reset*** back to 0:

```
lower_bound = 1
upper_bound = 100
attempts = 0
```

6. The user is kept in a loop as long as the computer has not exhausted the number of guesses it is allowed (as long as the game is being played). *In every iteration*, check if `lower_bound` is lower than the `upper_bound`. If that *is not the case* (`lower_bound` is unexpectedly *greater than* `upper_bound`, print "Oops, it seems like you made a mistake" and exit.

7. Once all guesses have been used and the computer still has yet to guess the right number, the computer will admit defeat and ask (print), "I give up. What was the answer?" It will then ask what the answer was (although it doesn't do anything with it.):

```
input("Enter your number: ")
```

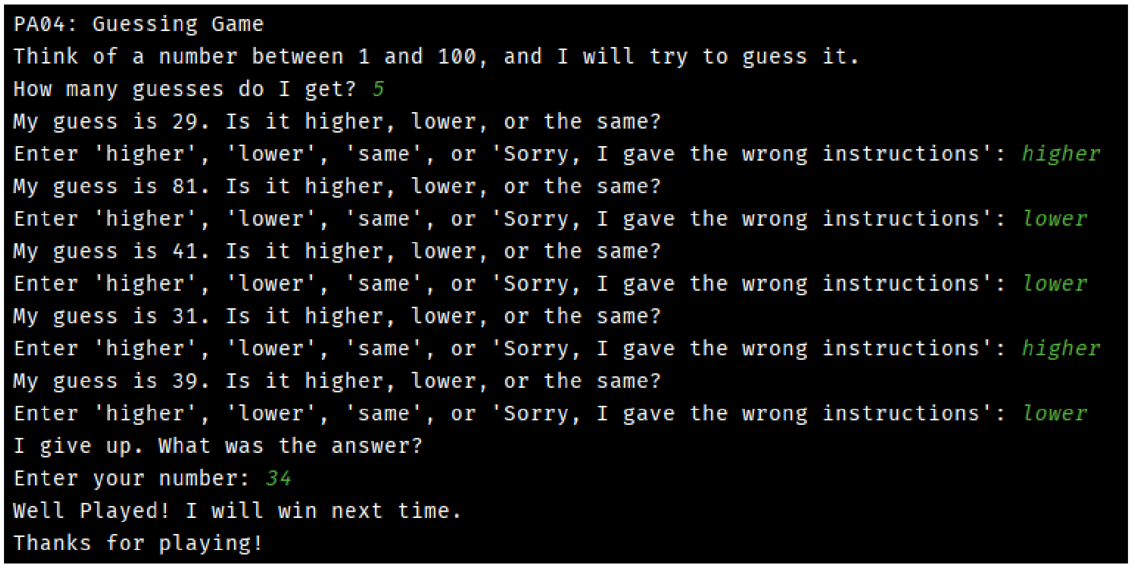
After the user **gives** the answer, the computer will say (print), "Well Played! I will win next time" and the game will **finish**.

8. At the very end the computer will print one time, "Thanks for playing!"

9. **Test** your program with different amounts of guesses, numbers, and inputs to verify that it produces the right response for each scenario described above.
(Before submitting to Gradescope, **REMOVE** the call to your function in your file.)

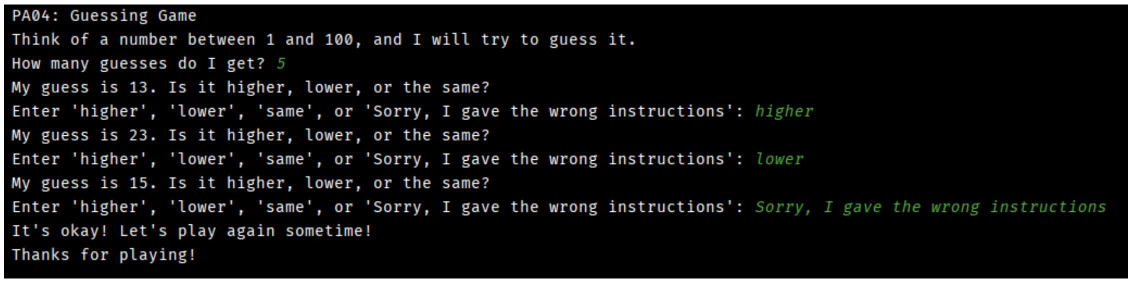
10. In a separate document, write a short **reflection** (about 150-200 words) discussing your experience working with conditional statements, loops, and the break and continue statements in this guessing game scenario. What challenges did you encounter? How do you think conditional statements, loops, and the break and continue statements can be useful in programming?

1.1 Sample Runs and Expected Output



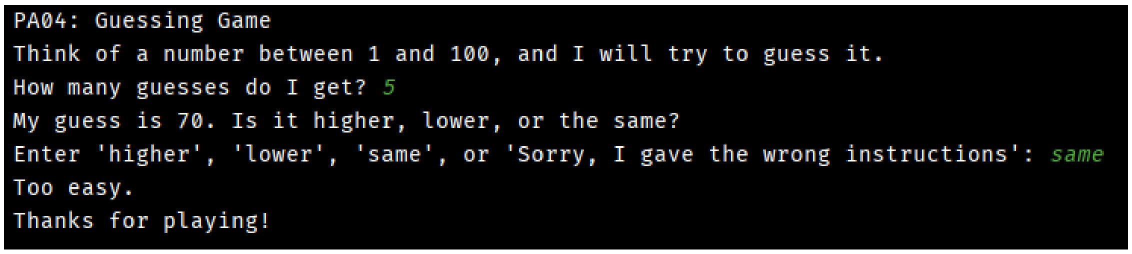
```
PA04: Guessing Game
Think of a number between 1 and 100, and I will try to guess it.
How many guesses do I get? 5
My guess is 29. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': higher
My guess is 81. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': lower
My guess is 41. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': lower
My guess is 31. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': higher
My guess is 39. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': lower
I give up. What was the answer?
Enter your number: 34
Well Played! I will win next time.
Thanks for playing!
```

Figure 1: My number was 34, computer could not guess correctly in allowed number of guesses (5).



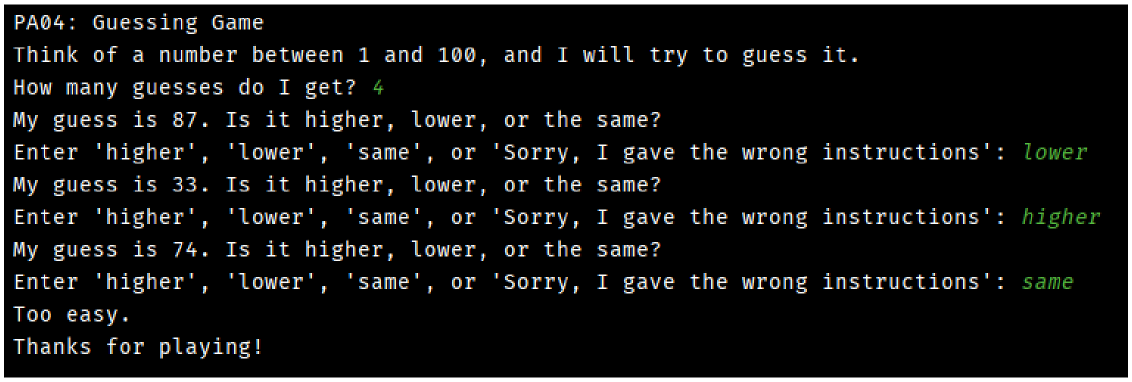
```
PA04: Guessing Game
Think of a number between 1 and 100, and I will try to guess it.
How many guesses do I get? 5
My guess is 13. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': higher
My guess is 23. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': lower
My guess is 15. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': Sorry, I gave the wrong instructions
It's okay! Let's play again sometime!
Thanks for playing!
```

Figure 2: My number was 31, but I gave the wrong instructions, so the program ended at that point (5 guesses initially allowed).



```
PA04: Guessing Game
Think of a number between 1 and 100, and I will try to guess it.
How many guesses do I get? 5
My guess is 70. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': same
Too easy.
Thanks for playing!
```

Figure 3: My number was 70, program guessed on the first try (5 guesses initially allowed).



```
PA04: Guessing Game
Think of a number between 1 and 100, and I will try to guess it.
How many guesses do I get? 4
My guess is 87. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': lower
My guess is 33. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': higher
My guess is 74. Is it higher, lower, or the same?
Enter 'higher', 'lower', 'same', or 'Sorry, I gave the wrong instructions': same
Too easy.
Thanks for playing!
```

Figure 4: My number was 74, program guessed it before maximum allowable guesses (4 guesses allowed).

Keep the following in mind as you write your code, as you will be graded on these aspects:

- Don't forget your header at the top of your `.py` file (see section 2.1)
- Cite your resources clearly or state you did not use any
- Use appropriate variables names throughout your program
- Include in-line comments throughout your code to explain what you are doing

2 Submission and Collaboration Policy

2.1 Your Submission–Comment Header

Your `.py` file should contain the following information (header) at the **top** of your file:

```
# NAME: e.g I. Lv. Sneks
# COMPUTING ID: e.g. sneks_r_ssssuper@python.edu
# PA NUMBER and NAME: e.g. PA## - Name of the Assignment
# Resources used (if applicable):
```

Note about submitting: When submitting to Gradescope make sure you remove your call to your function in your file! Gradescope will not be able to grade your assignment if you do!

For **Resources**, include URLs to any online resources where you studied or reviewed code that is specific to these problems, including any physical textbooks you referenced. Include any other resources you used here. *Note on resources:* Please feel free to refer to the lecture slides, demos, and in-class labs to complete this assignment, which can all be located on Canvas.

Absolutely no collaboration with any fellow students (who are not current CS 1112 course TAs) is allowed. Your work must represent individual effort. You must write your own code: not *just* type it, but also compose it yourself as your own original work.

You must cite any and every source you consult, other than those explicitly provided by the course itself. If you work with, obtain or receive help from another source (Internet website, textbook, TA, tutor, online video, etc.), nothing should be copied or retyped into the submitted solution. References must be documented in a comment in the code on the assignment. Any copied work is an Honor Code violation.

2.2 Gradescope

2.2.1 Submitting on Gradescope

1. Go to Gradescope (linked in Canvas course website)
2. Click on **PA04 – Guessing Game** in the assignment list on Gradescope
3. Upload your Python program as `guessing_game.py`.
4. Include the output of your program for a few scenarios mentioned. Do this by including a multi-line comment (or use single-line comments) at the bottom of your Python file and paste the output from your program.
5. Write your reflection in a separate document and save/download it as a PDF (please ask us if you are unsure how to save your file in PDF format. We are happy to help!)
6. Submit to the same Gradescope assignment PA04 - Guessing Game
7. Ensure you have uploaded **two (2) files** before you submit!

Note: While testing your code, you will need to call your function in your file. **Before** submitting your .py file to Gradescope, **REMOVE** the call to your function in your file.
Helpful guidance: Do not wait until the last minute to start this assignment! Remember you can **submit multiple times** to Gradescope. Look at the feedback you receive and then go back and fix your code, then you can resubmit. There is NO penalty for submitting multiple times to Gradescope. If you face any difficulties or have questions, post on Piazza or reach out to your professor or teaching assistants (TAs) for guidance. We are here to help!