CS5001 Project 4 Report

Shang Xiao

February 16, 2022

1 Problem Description

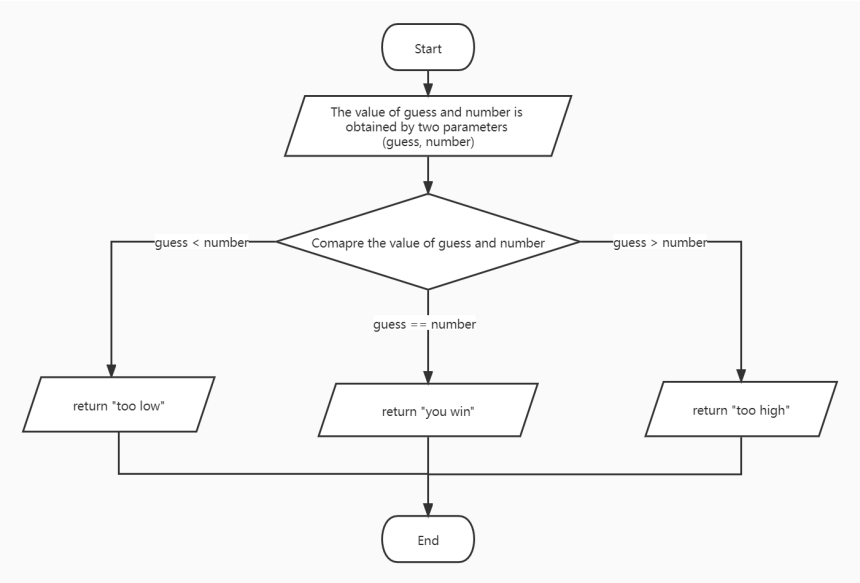
1.1 This week, we were introduced to a brand new condition: while, aka, the while loop. While loop works very similar to the if statement, meaning, when this happens, do stuff. What differentiates the while loop from the if statement is that while loop will repeat the staff we have written inside the loop forever until we tell it to stop; this means, while this is happening, do stuff. In project 4, we are asked to develop an algorithm that randomly generates a number (my initial approach to this request is to use the randint() function) from a given range (since I could not find a way to pick a number from 0 to infinity randomly, I asked the user to enter a range for me). Then prompt the user to enter their guesses to find the randomly generated number.

It is very unlikely that the user will get the number on their first attempt if the range is large. So we were also asked to develop the program further so that for each user attempt, we can tell the user whether they are closer (by using word getting colder) or further (by using word getting hotter) to the randomly chosen number. At the end, whether the guessing pattern was efficient.

The guessing pattern here means for any given range. We have an optimal number of attempts to figure out any random number in between. For example, if we were to guess a number from 1 to 100, the first guess should be 50. Then, if the program tells us 50 is too high or too low, we enter 25 or 75, accordingly; then 18 or 88... This means always cutting the remaining range in half. We can eliminate half of the possibilities to reduce our guess time. This is also known as binary search.

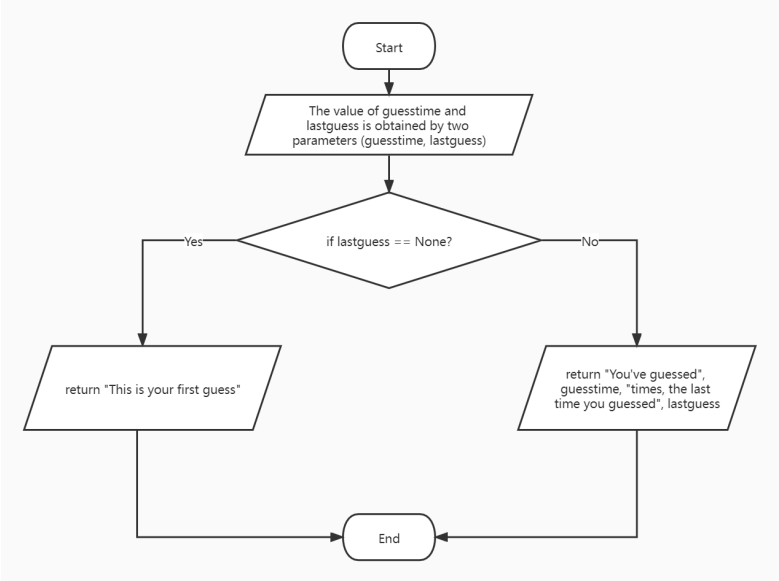
2 Activity Diagrams

2.1 The first function getPosition(guess, number) takes two parameters and compares the first parameter with the second parameter. If the first parameter is bigger than the second parameter, returns too high; if the first parameter is smaller than the second parameter, returns too low; else, prompt the user they had won.



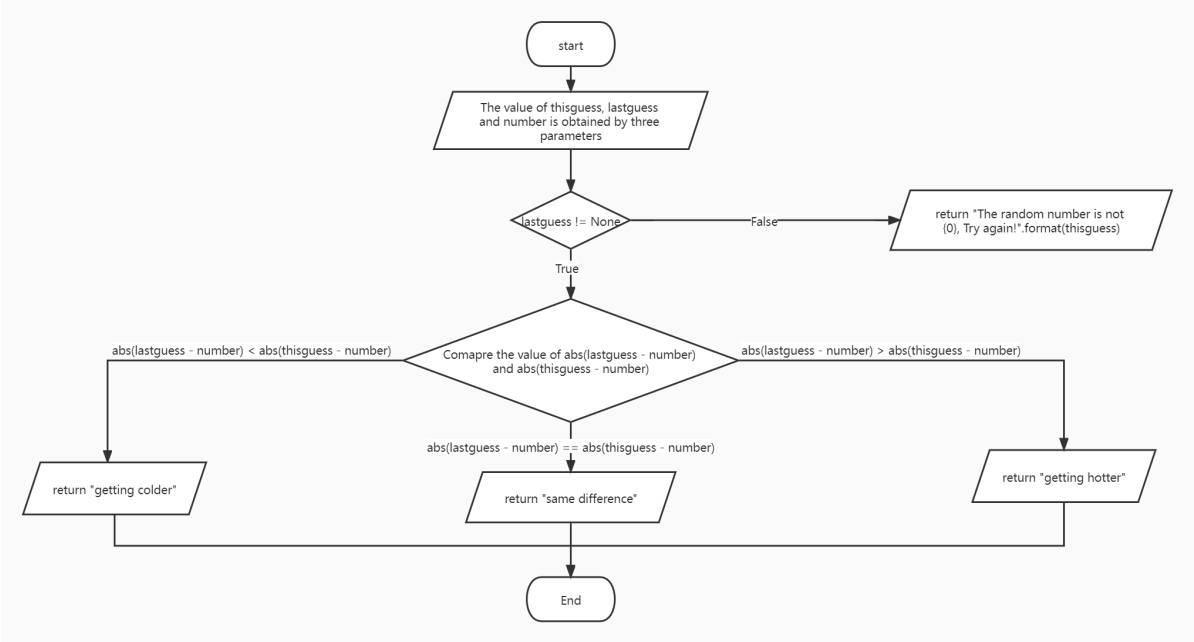
Flowchart for getPosition()

2.2 The second function printGuessTime(guesstime, lastguess) return the guess time and the last guess from the user. If lastguess != None, then return the guess time and the last guess from the user. If lastguess == None, then return "This is your first guess".



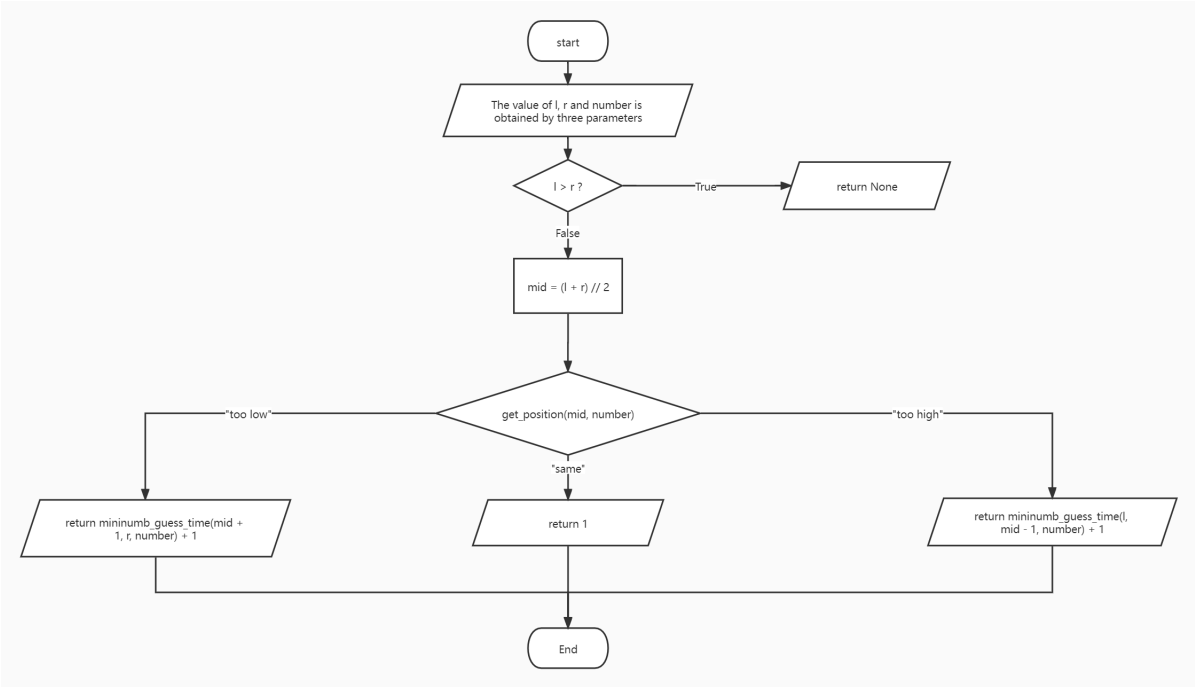
Flowchart for printGuessTime()

2.3 The third function printPrompt(lastguess, thisguess, number) prompts the user if they are getting closer to the random value. The function returns getting colder if the absolute difference of this guess is more than the last guess. The function returns same difference if the absolute difference of this guess is equal to the last guess. Else, the function returns getting hotter.



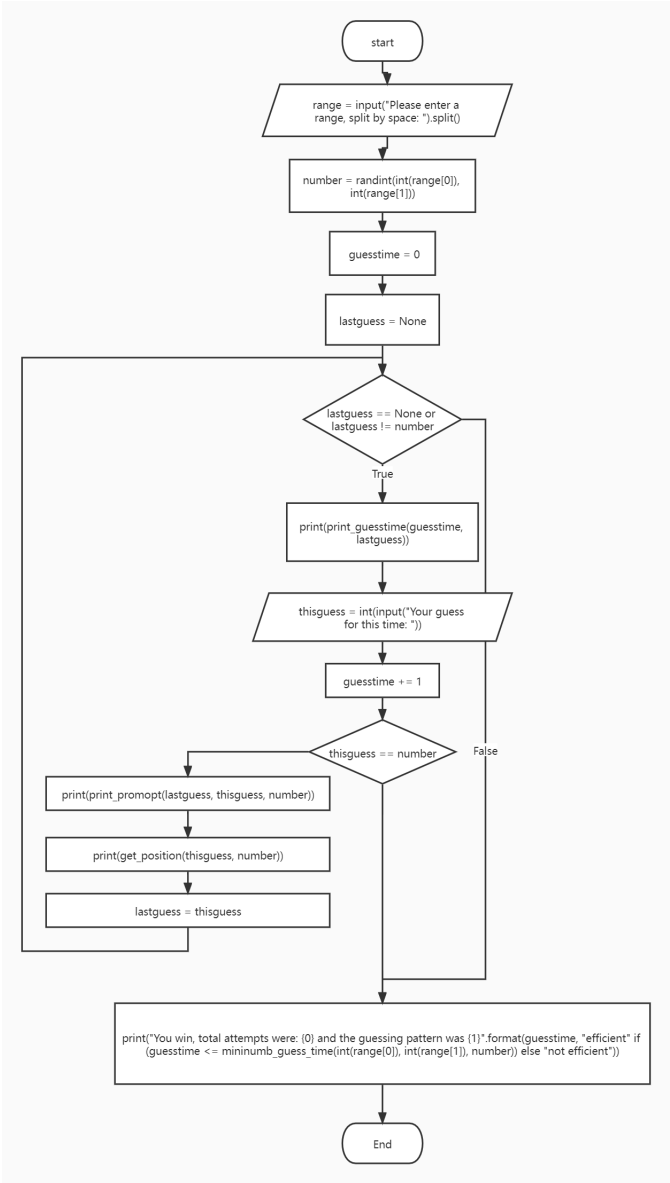
Flowchart for printPrompt()

2.4 The fourth function mininumbGuessTime(l, r, number) applies binary search algorithm to decide which range does the random number falls into. For any given range, l to r, the midpoint is (l+r)//2. We call getPosition(), use mid as the first parameter, and then compare it with the second parameter, the random number. If getPosition() returns too high, then the random number is within range [l, mid - 1]; if getPosition() returns too low, then the random number is within range [mid + 1, r]; else, the random number is the midpoint number.



Flowchart for mininumbGuessTime()

2.5 The main() function integrates all four functions. We begin by asking the user to enter a range, split by space. Then we start building our while loop, which breaks when the user gets the correct answer; repeat otherwise. For each new guess, guess time +=1. We want to prompt the user how many times they have guessed and which number they guessed for their last guess. Then ask the user to enter a new number for this guess. For each guess, we want to prompt the user if this guess is "getting colder" or "getting hotter" to the actual random number and if this guess is "too low" or "too high" compared to the actual random number. In the end, we prompt the user how many guesses they have made and if their guessing pattern was efficient compared with the binary search algorithm.



Flowchart for main()

3 Reflection

3.1 When working on this project, I did some extra research on binary search and its code implementations. I was able to apply the binary search algorithm into my mininumbGuessTime() function. In the main function, I called

mininumbGuessTime() to determine whether the user's guessing pattern was efficient. In addition, I also get more used to write more complicated while loop functions, and debug those loops by combining my previous knowledge in conditionals with topics related to debugging a loop taught in this module. I was able to figure our the problems in my loop functions a lot quicker than before.

4 Acknowledgements

4.1 Website consulted (lecture notes and documentations can be found in these links):

https://www.geeksforgeeks.org/
https://northeastern.instructure.com/courses/102943
https://greenteapress.com/thinkpython2/html/index.html

4.2 Website used for flowchart: https://www.processon.com/
Website used for debugging codes and loops for each line execution:
https://pythontutor.com/visualize.htmlmode=edit