CSE 20
Beginning Programming in Python
Programming Assignment 3

In this assignment you will write a Python program that plays an interactive guessing game with the user. Your program will generate a random integer in the range 1 to n, where n is obtained from user input. It will then allow the user at most n guesses to determine the integer. After each guess your program will inform the user whether the guess was correct, too high, or too low. Your source file for this project will be called `Guess.py`. Here is a transcript of several plays of the game. As usual, `$` stands for the command line prompt.

```
$ python3 Guess.py

Enter a positive integer: 100

Try to guess my random integer in the range 1 to 100

Guess 1: 50
   50 is too low
Guess 2: 75
   75 is too low
Guess 3: 87
   87 is too low
Guess 4: 93
   93 is too high
Guess 5: 90
   90 is correct, you found my number in 5 guesses :)

$ python3 Guess.py

Enter a positive integer: 5

Try to guess my random integer in the range 1 to 5

Guess 1: 4
   4 is correct, you found my number in 1 guess :)

$ python3 Guess.py

Enter a positive integer: 3

Try to guess my random integer in the range 1 to 3

Guess 1: 1
   1 is too low
Guess 2: 1
   1 is too low
Guess 3: 1
   1 is too low
   my number was 2, you're out of guesses :(

$
```

Observe that there are blank lines at the beginning and end of program output, and blank lines separating certain lines of output. The program prompts the user for each guess by printing "`Guess k: `", where *k* is the number of the guess. The user responds by typing a guess, then enter. There is no separation between

guesses and responses, but the program responses are indented 3 spaces. Also notice that if the user finds the number within n guesses, the program states the number of guesses used, and that the word "`guess`" is correctly pluralized. Your program should match the above format exactly. Your program is not required to correct the user for wrong input of any kind, such as non-numeric strings and non-positive integers.

Use the `random` module, and its `randint()` method to generate the random number. You can read about this module and its methods at

https://docs.python.org/3/library/random.html?highlight=random#module-random

Use iterative (`for`, `while` and `break`) and conditional (`if`, `if-else`, and `if-elif-else`) statements to control your program's behavior, and adapt to user input. Conditional statements are discussed in Chapter 5 of the text. Loops are discussed in Chapter 7. Both topics will be covered at length in class.

In order to facilitate testing of your program by the graders, you will place all instructions within a function called `main()`, as illustrated in the template below. This will allow us to evaluate your program by calling function `main()` from within another module.

```
import random

# function main() ------------------------------------------------------------
def main():

   # program instructions: your code goes here




# end main() ----------------------------------------------------------------

# closing conditional -------------------------------------------------------
if __name__=='__main__':

   main()

# end -----------------------------------------------------------------------
```

See also the example GeneralTemplate.py in the Examples section of the class webpage. Submit your program `Guess.py`, to the assignment name `pa3` before the due date. As always start early and ask questions in office hours and on Ed Discussion.

Further Discussion

Although this game is very simple, it is interesting to consider what strategy the user might adopt. Try to verify for yourself that a strategy exists that would require at most $\lfloor \lg(n) \rfloor$ guesses in order to determine the mystery number, where lg () is the logarithm base 2. With a little more effort, it is possible to show that this strategy is in fact optimal, i.e. there is none better. As an exercise, write a program that plays the other side of the game (i.e. the role of guesser) by implementing the optimal strategy. This exercise will yield no credit for the current assignment, but it may form the basis of some future project.