# This is a Guess the Number game from 'Invent Your Own Computer Game With Python' adapted by PMP.

# This version (01) is the original program.

import random # This imports a module to add randomizing functions to the basic functions available in Python.

print() # This prints a blank line.

print('\*' \* 26) # This prints a string of 26 asterisks.

print('\*' + ' ' \* 24 + '\*') # This prints an asterisk, then a string of 24 spaces, then an asterisk.

print('\* Number Guessing Game \*') # This prints the game's title with an asterisk and two spaces on each side.

print('\*' + ' ' \* 24 + '\*') # This prints an asterisk, then a string of 24 spaces, then an asterisk.

print('\*' \* 26) # This prints a string of 26 asterisks.

print() # This prints a blank line.

print('Hello! What is your name?')

playerName = input() # This allows the player to enter his/her name.

number = random.randint(1, 20) # This sets the variable 'number' to an integer between 1 and 20 inclusive.

guessesTaken = 0 # This sets the counter for the number of guess taken to zero.

print('Well, ' + playerName + ', I\'m thinking of a number between 1 and 20.')

# N.B. \' allows an apostrophe to be used within a string.

while guessesTaken < 6: # This 'while' loop checks that the number of guesses taken is less than 6.

print('Take a guess.') # N.B. There are 4 spaces in front of 'print' to show that this section of the program is part of the 'while' loop.

guess = input() # This allows the player to enter a guess.

guess = int(guess) # This converts the player's entry from a string to an integer.

guessesTaken = guessesTaken + 1 # This adds one to the counter for number of guesses taken.

if guess < number: # This 'if' statement gives an output if the player's guess is less than the computer's chosen number.

print('Your guess is too low.')

elif guess > number: # This 'elif' statement gives an output if the player's guess is greater than the computer's chosen number.

print('Your guess is too high.')

# N.B. 'elif' means 'else if'.

elif guess == number: # This 'elif' statement gives an output if the player's guess is equal to the computer's chosen number.

break # If the guess is equal to the number, this stops any more guesses being asked for.

if guess == number: # This 'if' statement gives an output if the player's guess equals the computer's number.

print('Well done, ' + playerName + '! You guessed my number in ' + str(guessesTaken) + ' guesses!')

# N.B. str() converts a numeric variable to a string.

else: # This 'else' statement gives an output if the player's guess is NOT equal to the computer's number.

print('Bad luck. The number I was thinking of was ' + str(number) + '.')

# This is a Guess the Number game from 'Invent Your Own Computer Game With Python' adapted by PMP.

# This version (02) uses variables for the limits for the numbers chosen by the computer and for the maximum number of guesses; it also shortens the coding of the title display.

import random # This imports a module to add randomizing functions to the basic functions available in Python.

minNumber = 1 # This sets the lower limit for the random number to be chosen by the computer.

maxNumber = 20 # This sets the upper limit for the random number to be chosen by the computer.

maxGuesses = 6 # This sets the maximum number of guesses allower to 6.

print('\n' + '\*' \* 26 + '\n\*' + ' ' \* 24 + '\*\n\* Number Guessing Game \*\n\*' + ' ' \* 24 + '\*\n' + '\*' \* 26 + '\n\n')

# At the start of the game, this gives a title surrounded by asterisks.

# Note that '\*' \* 26 means a string of 26 asterisks, ' ' \* 24 means a string of 24 spaces, and \n means 'start a new line'.

print('Hello! What is your name?')

playerName = input() # This allows the player to enter his/her name.

number = random.randint(minNumber, maxNumber) # This sets the variable 'number' to an integer between 'minNumber' and 'maxNumber' inclusive.

guessesTaken = 0 # This sets the counter for the number of guess taken to zero.

print('Well, ' + playerName + ', I\'m thinking of a number between ' + str(minNumber) + ' and ' + str(maxNumber) + '.')

# \' allows an apostrophe to be used within a string.

# str() converts a numeric variable to a string variable (so that it can be displayed correctly with 'print').

while guessesTaken < maxGuesses: # This 'while' loop runs as long as the number of guesses taken is less than the maximum number of guesses.

print('Take a guess.')

guess = input() # This allows the player to enter a guess.

guess = int(guess) # This converts the player's (string) entry to an integer.

guessesTaken = guessesTaken + 1 # This adds one to the counter for number of guesses taken.

if guess < number: # This 'if' statement gives an output if the player's guess is less than the computer's chosen number.

print('Your guess is too low.')

if guess < minNumber: # This 'if' statement gives an output if the player's guess is less than 'minNumber'.

print('You\'ve wasted this guess, as your number is below the minimum possible choice for the computer.')

elif guess > number: # This 'elif' statement gives an output if the player's guess is greater than the computer's chosen number.

print('Your guess is too high.')

if guess > maxNumber: # This 'if' statement gives an output if the player's guess is greater than 'maxNumber'.

print('You\'ve wasted this guess, as your number is above the maximum possible choice for the computer.')

elif guess == number: # This 'elif' statement gives an output if the player's guess is equal to the computer's chosen number.

break # If the guess is equal to the number, this 'break' stops any more guesses being asked for.

if guess == number: # This 'if' statement gives an output if the player's guess equals the computer's number.

print('Well done, ' + playerName + '! You guessed my number in ' + str(guessesTaken) + ' guesses!')

else: # This 'else' statement matches the 'if guess == number:' statement above.

print('Bad luck. The number I was thinking of was ' + str(number) +'.')

# This is a Guess the Number game from 'Invent Your Own Computer Game With Python' adapted by PMP.

# This version (03) adds error checking for the player's input and better output formatting.

import random # This imports a module to add randomizing functions to the basic functions available in Python.

minNumber = 1 # This sets the lower limit for the random number to be chosen by the computer.

maxNumber = 20 # This sets the upper limit for the random number to be chosen by the computer.

maxGuesses = 6 # This sets the maximum number of guesses allower to 6.

print('\n' + '\*' \* 26 + '\n\*' + ' ' \* 24 + '\*\n\* Number Guessing Game \*\n\*' + ' ' \* 24 + '\*\n' + '\*' \* 26 + '\n\n')

# At the start of the game, this gives a title surrounded by asterisks.

# Note that '\*' \* 26 means a string of 26 asterisks, ' ' \* 24 means a string of 24 spaces, and \n means 'start a new line'.

print('Hello! What is your name?')

playerName = input() # This allows the player to enter his/her name.

number = random.randint(minNumber, maxNumber) # This sets the variable 'number' to an integer between 'minNumber' and 'maxNumber' inclusive.

guessesTaken = 0 # This sets the counter for the number of guess taken to zero.

print('\nWell, ' + playerName + ', I\'m thinking of a number between ' + str(minNumber) + ' and ' + str(maxNumber) + '.')

# \' allows an apostrophe to be used within a string.

# str() converts a numeric variable to a string variable (so that it can be displayed correctly with 'print').

while guessesTaken < maxGuesses: # This 'while' loop runs as long as the number of guesses taken is less than the maximum number of guesses.

print() # This prints a blank line.

if maxGuesses - guessesTaken > 1: # This 'if' statement gives an output if the player has more than one guess left.

print('You have ' + str(maxGuesses - guessesTaken) + ' guesses left.')

else: # This 'else' statement matches the 'if maxGuesses - guessesTaken > 1:' statement above.

print('You have 1 guess left.')

print('What do you think my number is?')

while True: # This 'while' loop checks whether the player's guess is valid (i.e. a number, not a letter or word).

guess = input() # This allows the player to enter a guess.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

guess = int(guess) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

print('You must enter numbers, not letters - try again.')

# This point marks the end of the 'while True:' loop.

guess = int(guess) # This converts the player's (string) entry to an integer.

guessesTaken = guessesTaken + 1 # This adds one to the counter for number of guesses taken.

if guess < number: # This 'if' statement gives an output if the player's guess is less than the computer's chosen number.

print('Your guess is too low.')

if guess < minNumber: # This 'if' statement gives an output if the player's guess is less than 'minNumber'.

print('You\'ve wasted this guess, as your number is below the minimum possible choice for the computer.')

elif guess > number: # This 'elif' statement gives an output if the player's guess is greater than the computer's chosen number.

print('Your guess is too high.')

if guess > maxNumber: # This 'if' statement gives an output if the player's guess is greater than 'maxNumber'.

print('You\'ve wasted this guess, as your number is above the maximum possible choice for the computer.')

elif guess == number: # This 'elif' statement gives an output if the player's guess is equal to the computer's chosen number.

break # If the guess is equal to the number, this 'break' stops any more guesses being asked for.

# This point marks the end of the 'while guessesTaken < maxGuesses:' loop.

if guess == number: # This 'if' statement gives an output depending on whether the player got the computer's number right or not.

print('\nWell done, ' + playerName + '! You guessed my number in ', end='') # , end='' makes the next 'print' function continue on the same line.

if guessesTaken > 1: # This 'if' statement gives an output depending on how many guesses the player took.

print(str(guessesTaken) + ' guesses!')

else: # This 'else' statement matches the 'if guessesTaken > 1:' statement above.

print('one guess!')

else: # This 'else' statement matches the 'if guess == number:' statement above.

print('Bad luck. The number I was thinking of was ' + str(number) +'.')

# This is a Guess the Number game from 'Invent Your Own Computer Game With Python' adapted by PMP.

# This version (04) adds the ability for the player to choose whether or not to continue playing after one game.

import random # This imports a module to add randomizing functions to the basic functions available in Python.

minNumber = 1 # This sets the lower limit for the random number to be chosen by the computer.

maxNumber = 20 # This sets the upper limit for the random number to be chosen by the computer.

maxGuesses = 6 # This sets the maximum number of guesses allower to 6.

print('\n' + '\*' \* 26 + '\n\*' + ' ' \* 24 + '\*\n\* Number Guessing Game \*\n\*' + ' ' \* 24 + '\*\n' + '\*' \* 26 + '\n\n')

# At the start of the game, this gives a title surrounded by asterisks.

# Note that '\*' \* 26 means a string of 26 asterisks, ' ' \* 24 means a string of 24 spaces, and \n means 'start a new line'.

print('Hello! What is your name?')

playerName = input() # This allows the player to enter his/her name.

while True: # This 'while' loop runs as long as the player asks to continue playing after each game.

number = random.randint(minNumber, maxNumber) # This sets the variable 'number' to an integer between 'minNumber' and 'maxNumber' inclusive.

guessesTaken = 0 # This sets the counter for the number of guess taken to zero.

print('\nWell, ' + playerName + ', I\'m thinking of a number between ' + str(minNumber) + ' and ' + str(maxNumber) + '.')

# \' allows an apostrophe to be used within a string.

# str() converts a numeric variable to a string variable (so that it can be displayed correctly with 'print').

while guessesTaken < maxGuesses: # This 'while' loop runs as long as the number of guesses taken is less than the maximum number of guesses.

print() # This prints a blank line.

if maxGuesses - guessesTaken > 1: # This 'if' statement gives an output if the player has more than one guess left.

print('You have ' + str(maxGuesses - guessesTaken) + ' guesses left.')

else: # This 'else' statement matches the 'if maxGuesses - guessesTaken > 1:' statement above.

print('You have 1 guess left.')

print('What do you think my number is?')

while True: # This 'while' loop checks whether the player's guess is valid (i.e. a number, not a letter or word).

guess = input() # This allows the player to enter a guess.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

guess = int(guess) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

print('You must enter numbers, not letters - try again.')

# This point marks the end of the 'while True:' loop.

guess = int(guess) # This converts the player's (string) entry to an integer.

guessesTaken = guessesTaken + 1 # This adds one to the counter for number of guesses taken.

if guess < number: # This 'if' statement gives an output if the player's guess is less than the computer's chosen number.

print('Your guess is too low.')

if guess < minNumber: # This 'if' statement gives an output if the player's guess is less than 'minNumber'.

print('You\'ve wasted this guess, as your number is below the minimum possible choice for the computer.')

elif guess > number: # This 'elif' statement gives an output if the player's guess is greater than the computer's chosen number.

print('Your guess is too high.')

if guess > maxNumber: # This 'if' statement gives an output if the player's guess is greater than 'maxNumber'.

print('You\'ve wasted this guess, as your number is above the maximum possible choice for the computer.')

elif guess == number: # This 'elif' statement gives an output if the player's guess is equal to the computer's chosen number.

break # If the guess is equal to the number, this 'break' stops any more guesses being asked for.

# This point marks the end of the 'while guessesTaken < maxGuesses:' loop.

if guess == number: # This 'if' statement gives an output depending on whether the player got the computer's number right or not.

print('\nWell done, ' + playerName + '! You guessed my number in ', end='') # , end='' makes the next 'print' function continue on the same line.

if guessesTaken > 1: # This 'if' statement gives an output depending on how many guesses the player took.

print(str(guessesTaken) + ' guesses!')

else: # This 'else' statement matches the 'if guessesTaken > 1:' statement above.

print('one guess!')

else: # This 'else' statement matches the 'if guess == number:' statement above.

print('Bad luck. The number I was thinking of was ' + str(number) +'.')

print('\nDo you want another game, ' + playerName + '?')

newGameYesNo = str.lower(input()) # 'str.lower' converts a string to lower case.

if newGameYesNo != 'yes' and newGameYesNo != 'y': # This 'if' statement ends the program if the player enters anything other than 'yes' or 'y'.

break

# This point marks the end of the outermost 'while True:' loop.

# This is a Guess the Number game from 'Invent Your Own Computer Game With Python' adapted by PMP.

# This version (05) adds the ability for the player to choose the range of numbers to guess within.

import random # This imports a module to add randomizing functions to the basic functions available in Python.

import math # This imports a module to add mathematical functions to the basic functions available in Python.

minNumber = None # This variable sets the lower limit for the random number to be chosen by the computer.

maxNumber = None # This variable sets the upper limit for the random number to be chosen by the computer.

maxGuesses = None # This variable sets the maximum number of guesses allower to 6.

print('\n' + '\*' \* 26 + '\n\*' + ' ' \* 24 + '\*\n\* Number Guessing Game \*\n\*' + ' ' \* 24 + '\*\n' + '\*' \* 26 + '\n\n')

# At the start of the game, this gives a title surrounded by asterisks.

# Note that '\*' \* 26 means a string of 26 asterisks, ' ' \* 24 means a string of 24 spaces, and \n means 'start a new line'.

print('Hello! What is your name?')

playerName = input() # This allows the player to enter his/her name.

print('\nWhat is the the lowest number the computer may choose?\nIt must be an integer between 1 and 1,000,000.')

while True:

while True: # This 'while' loop checks whether the player's entry is valid (i.e. a number, not a letter or word).

minNumber = input() # This allows the player to enter a number.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

minNumber = int(minNumber) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

print('You must enter integers, not fractions or letters - try again.')

# This point marks the end of the inner 'while True:' loop.

minNumber = int(minNumber)

if minNumber < 1 or minNumber > 1000000:

print('This number is outside the acceptable range - try again.')

else:

break

# This point marks the end of the outer 'while True:' loop.

print('\nWhat is the the highest number the computer may choose?\nIt must be an integer between 1 and 1,000,000 and at least 9 greater than the lowest number.')

while True:

while True: # This 'while' loop checks whether the player's entry is valid (i.e. a number, not a letter or word).

maxNumber = input() # This allows the player to enter a number.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

maxNumber = int(maxNumber) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

print('You must enter integers, not fractions or letters - try again.')

# This point marks the end of the inner 'while True:' loop.

maxNumber = int(maxNumber)

if maxNumber < 1 or maxNumber > 1000000 or maxNumber < minNumber + 9:

print('This number is outside the acceptable range - try again.')

else:

break

# This point marks the end of the outer 'while True:' loop.

maxGuesses = int(math.log(maxNumber-minNumber, 2) + 1)

while True: # This 'while' loop runs as long as the player asks to continue playing after each game.

number = random.randint(minNumber, maxNumber) # This sets the variable 'number' to an integer between 'minNumber' and 'maxNumber' inclusive.

guessesTaken = 0 # This sets the counter for the number of guess taken to zero.

print('\nWell, ' + playerName + ', I\'m thinking of a number between ' + str(minNumber) + ' and ' + str(maxNumber) + '.')

print('You have ' + str(maxGuesses) + ' guesses in which to work it out.')

# \' allows an apostrophe to be used within a string.

# str() converts a numeric variable to a string variable (so that it can be displayed correctly with 'print').

while guessesTaken < maxGuesses: # This 'while' loop runs as long as the number of guesses taken is less than the maximum number of guesses.

print() # This prints a blank line.

if maxGuesses - guessesTaken > 1: # This 'if' statement gives an output if the player has more than one guess left.

print('You have ' + str(maxGuesses - guessesTaken) + ' guesses left.')

else: # This 'else' statement matches the 'if maxGuesses - guessesTaken > 1:' statement above.

print('You have 1 guess left.')

print('What do you think my number is?')

while True: # This 'while' loop checks whether the player's guess is valid (i.e. a number, not a letter or word).

guess = input() # This allows the player to enter a guess.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

guess = int(guess) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

print('You must enter numbers, not letters - try again.')

# This point marks the end of the 'while True:' loop.

guess = int(guess) # This converts the player's (string) entry to an integer.

guessesTaken = guessesTaken + 1 # This adds one to the counter for number of guesses taken.

if guess < number: # This 'if' statement gives an output if the player's guess is less than the computer's chosen number.

print('Your guess is too low.')

if guess < minNumber: # This 'if' statement gives an output if the player's guess is less than 'minNumber'.

print('You\'ve wasted this guess, as your number is below the minimum possible choice for the computer.')

elif guess > number: # This 'elif' statement gives an output if the player's guess is greater than the computer's chosen number.

print('Your guess is too high.')

if guess > maxNumber: # This 'if' statement gives an output if the player's guess is greater than 'maxNumber'.

print('You\'ve wasted this guess, as your number is above the maximum possible choice for the computer.')

elif guess == number: # This 'elif' statement gives an output if the player's guess is equal to the computer's chosen number.

break # If the guess is equal to the number, this 'break' stops any more guesses being asked for.

# This point marks the end of the 'while guessesTaken < maxGuesses:' loop.

if guess == number: # This 'if' statement gives an output depending on whether the player got the computer's number right or not.

print('\nWell done, ' + playerName + '! You guessed my number in ', end='') # , end='' makes the next 'print' function continue on the same line.

if guessesTaken > 1: # This 'if' statement gives an output depending on how many guesses the player took.

print(str(guessesTaken) + ' guesses!')

else: # This 'else' statement matches the 'if guessesTaken > 1:' statement above.

print('one guess!')

else: # This 'else' statement matches the 'if guess == number:' statement above.

print('Bad luck. The number I was thinking of was ' + str(number) +'.')

print('\nDo you want another game, ' + playerName + '?')

newGameYesNo = str.lower(input()) # 'str.lower' converts a string to lower case.

if newGameYesNo != 'yes' and newGameYesNo != 'y': # This 'if' statement ends the program if the player enters anything other than 'yes' or 'y'.

break

# This point marks the end of the outermost 'while True:' loop.

# This is a Guess the Number game from 'Invent Your Own Computer Game With Python' adapted by PMP.

# This version (06) converts most of the code to user-defined functions.

# Variables in a user-defined function (def) are local by default.

# If a variable in a def is declared as global, references to it in defs executed later usually treat it as global without specific redeclaration.

# If a def uses a statement like x = x + 1 and x is intended to be a global variable, it must be declared as global in the def, even if it has been declared already in an earlier executed def.

import random # This imports a module to add randomizing functions to the basic functions available in Python.

import math # This imports a module to add mathematical functions to the basic functions available in Python.

def displayTitle():

print('\n' + '\*' \* 26 + '\n\*' + ' ' \* 24 + '\*\n\* Number Guessing Game \*\n\*' + ' ' \* 24 + '\*\n' + '\*' \* 26 + '\n\n')

# At the start of the game, this gives a title surrounded by asterisks.

# Note that '\*' \* 26 means a string of 26 asterisks, ' ' \* 24 means a string of 24 spaces, and \n means 'start a new line'.

def inputName():

print('Hello! What is your name?')

playerName = input() # This allows the player to enter his/her name.

return playerName

def setNumberLimitLow():

print('\nWhat is the the lowest number the computer may choose?\nIt must be an integer between 1 and 1,000,000.')

while True:

while True: # This 'while' loop checks whether the player's entry is valid (i.e. a number, not a letter or word).

minNumber = input() # This allows the player to enter a number.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

minNumber = int(minNumber) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

print('You must enter integers, not fractions or letters - try again.')

minNumber = int(minNumber)

if minNumber < 1 or minNumber > 1000000:

print('This number is outside the acceptable range - try again.')

else:

break

return minNumber

def setNumberLimitHigh(minNumber):

print('\nWhat is the the highest number the computer may choose?\nIt must be an integer between 1 and 1,000,000 and at least 9 greater than the lowest number.')

while True:

while True: # This 'while' loop checks whether the player's entry is valid (i.e. a number, not a letter or word).

maxNumber = input() # This allows the player to enter a number.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

maxNumber = int(maxNumber) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

print('You must enter integers, not fractions or letters - try again.')

maxNumber = int(maxNumber)

if maxNumber < 1 or maxNumber > 1000000 or maxNumber < minNumber + 9:

print('This number is outside the acceptable range - try again.')

else:

break

return maxNumber

def setMaxGuesses(minNumber, maxNumber):

maxGuesses = int(math.log(maxNumber - minNumber, 2) + 1)

return maxGuesses

def startGame(playerName, minNumber, maxNumber, maxGuesses):

numberChosen = random.randint(minNumber, maxNumber) # This sets the variable 'number' to an integer between 'minNumber' and 'maxNumber' inclusive.

print('\nWell, ' + playerName + ', I\'m thinking of a number between ' + str(minNumber) + ' and ' + str(maxNumber) + '.')

print('You have ' + str(maxGuesses) + ' guesses in which to work it out.')

# \' allows an apostrophe to be used within a string.

# str() converts a numeric variable to a string variable (so that it can be displayed correctly with 'print').

return numberChosen

def askForGuess(maxGuesses):

print() # This prints a blank line.

if maxGuesses - guessesTaken > 1: # This 'if' statement gives an output if the player has more than one guess left.

print('You have ' + str(maxGuesses - guessesTaken) + ' guesses left.')

else:

print('You have 1 guess left.')

print('What do you think my number is?')

def makeGuess():

while True: # This 'while' loop checks whether the player's guess is valid (i.e. a number, not a letter or word).

guess = input() # This allows the player to enter a guess.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

guess = int(guess) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

print('You must enter numbers, not letters - try again.')

guess = int(guess) # This converts the player's (string) entry to an integer.

return guess

def checkGuess(minNumber, maxNumber, numberChosen, guess):

guessed = False

if guess < numberChosen: # This 'if' statement gives an output if the player's guess is less than the computer's chosen number.

print('Your guess is too low.')

if guess < minNumber: # This 'if' statement gives an output if the player's guess is less than 'minNumber'.

print('You\'ve wasted this guess, as your number is below the minimum possible choice for the computer.')

elif guess > numberChosen: # This 'elif' statement gives an output if the player's guess is greater than the computer's chosen number.

print('Your guess is too high.')

if guess > maxNumber: # This 'if' statement gives an output if the player's guess is greater than 'maxNumber'.

print('You\'ve wasted this guess, as your number is above the maximum possible choice for the computer.')

elif guess == numberChosen: # This 'elif' statement gives an output if the player's guess is equal to the computer's chosen number.

guessed = True

return guessed

def displayResult(playerName, numberChosen, guess):

if guess == numberChosen: # This 'if' statement gives an output depending on whether the player got the computer's number right or not.

print('\nWell done, ' + playerName + '! You guessed my number in ', end='') # , end='' makes the next 'print' function continue on the same line.

if guessesTaken > 1: # This 'if' statement gives an output depending on how many guesses the player took.

print(str(guessesTaken) + ' guesses!')

else:

print('one guess!')

else:

print('Bad luck. The number I was thinking of was ' + str(numberChosen) + '.')

def newGameRequest(playerName):

replay = True

print('\nDo you want another game, ' + playerName + '?')

newGameYesNo = str.lower(input()) # 'str.lower' converts a string to lower case.

if newGameYesNo != 'yes' and newGameYesNo != 'y': # This 'if' statement ends the program if the player enters anything other than 'yes' or 'y'.

replay = False

return replay

# Game

displayTitle()

playerN = inputName()

minNum = setNumberLimitLow()

maxNum = setNumberLimitHigh(minNum)

maxGuess = setMaxGuesses(minNum, maxNum)

while True: # This 'while' loop runs as long as the player asks to continue playing after each game.

numberC = startGame(playerN, minNum, maxNum, maxGuess)

guessesTaken = 0

while guessesTaken < maxGuess: # This 'while' loop runs as long as the number of guesses taken is less than the maximum number of guesses.

askForGuess(maxGuess)

mGuess = makeGuess()

guessesTaken = guessesTaken + 1

guessedN = checkGuess(minNum, maxNum, numberC, mGuess)

if guessedN == True:

break

displayResult(playerN, numberC, mGuess)

replay = newGameRequest(playerN)

if replay == False:

break

#-------------------------------------------------------------------------------

# Name: Guess the Number

# Purpose: Number guessing game

#

# Author: PMP

#

# Created: 12/01/2013

# Copyright: (c) PMP 2013

# Licence: n/a

#-------------------------------------------------------------------------------

# This is a Guess the Number game from 'Invent Your Own Computer Game With Python' adapted by PMP.

# This version (07) puts the main game loop into def main() .

# Variables in a user-defined function (def) are local by default.

# If a variable in a def is declared as global, references to it in defs executed later usually treat it as global without specific redeclaration.

# If a def uses a statement like x = x + 1 and x is intended to be a global variable, it must be declared as global in the def, even if it has been declared already in an earlier executed def.

import random as r # This imports a module to add randomizing functions to the basic functions available in Python.

import math as m # This imports a module to add mathematical functions to the basic functions available in Python.

def main(): # This is the game main function.

displayTitle()

playerN = inputName()

minNum = setNumberLimitLow()

maxNum = setNumberLimitHigh(minNum)

maxGuess = setMaxGuesses(minNum, maxNum)

while True: # This 'while' loop runs as long as the player asks to continue playing after each game.

numberC = startGame(playerN, minNum, maxNum, maxGuess)

guessTaken = 0

while guessTaken < maxGuess: # This 'while' loop runs as long as the number of guesses taken is less than the maximum number of guesses.

askForGuess(maxGuess, guessTaken)

mGuess = makeGuess()

guessTaken = guessTaken + 1

guessedN = checkGuess(minNum, maxNum, numberC, mGuess)

if guessedN == True:

break

displayResult(playerN, numberC, mGuess, guessTaken)

replay = newGameRequest(playerN)

if replay == False:

break

def displayTitle():

print('\n' + '\*' \* 26 + '\n\*' + ' ' \* 24 + '\*\n\* Number Guessing Game \*\n\*' + ' ' \* 24 + '\*\n' + '\*' \* 26 + '\n\n')

# At the start of the game, this gives a title surrounded by asterisks.

# Note that '\*' \* 26 means a string of 26 asterisks, ' ' \* 24 means a string of 24 spaces, and \n means 'start a new line'.

def inputName():

print('Hello! What is your name?')

playerName = input() # This allows the player to enter his/her name.

return playerName

def setNumberLimitLow():

print('\nWhat is the lowest number the computer may choose?\nIt must be an integer between 1 and 1,000,000.')

while True:

while True: # This 'while' loop checks whether the player's entry is valid (i.e. a number, not a letter or word).

minNumber = input() # This allows the player to enter a number.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

minNumber = int(minNumber) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

print('You must enter integers, not fractions or letters - try again.')

minNumber = int(minNumber)

if minNumber < 1 or minNumber > 1000000:

print('This number is outside the acceptable range - try again.')

else:

break

return minNumber

def setNumberLimitHigh(minNumber):

print('\nWhat is the highest number the computer may choose?\nIt must be an integer between 1 and 1,000,000 and at least 9 greater than the lowest number.')

while True:

while True: # This 'while' loop checks whether the player's entry is valid (i.e. a number, not a letter or word).

maxNumber = input() # This allows the player to enter a number.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

maxNumber = int(maxNumber) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

print('You must enter integers, not fractions or letters - try again.')

maxNumber = int(maxNumber)

if maxNumber < 1 or maxNumber > 1000000 or maxNumber < minNumber + 9:

print('This number is outside the acceptable range - try again.')

else:

break

return maxNumber

def setMaxGuesses(minNumber, maxNumber):

maxGuesses = int(m.log(maxNumber + 1 - minNumber, 2) + 1)

return maxGuesses

def startGame(playerName, minNumber, maxNumber, maxGuesses):

numberChosen = r.randint(minNumber, maxNumber) # This sets the variable 'number' to an integer between 'minNumber' and 'maxNumber' inclusive.

print('\nWell, ' + playerName + ', I\'m thinking of a number between ' + str(minNumber) + ' and ' + str(maxNumber) + '.')

print('You have ' + str(maxGuesses) + ' guesses in which to work it out.')

# \' allows an apostrophe to be used within a string.

# str() converts a numeric variable to a string variable (so that it can be displayed correctly with 'print').

return numberChosen

def askForGuess(maxGuesses, guessesTaken):

print() # This prints a blank line.

if maxGuesses - guessesTaken > 1: # This 'if' statement gives an output if the player has more than one guess left.

print('You have ' + str(maxGuesses - guessesTaken) + ' guesses left.')

else:

print('You have 1 guess left.')

print('What do you think my number is?')

def makeGuess():

while True: # This 'while' loop checks whether the player's guess is valid (i.e. a number, not a letter or word).

guess = input() # This allows the player to enter a guess.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

guess = int(guess) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

print('You must enter numbers, not letters - try again.')

guess = int(guess) # This converts the player's (string) entry to an integer.

return guess

def checkGuess(minNumber, maxNumber, numberChosen, guess):

guessed = False

if guess < numberChosen: # This 'if' statement gives an output if the player's guess is less than the computer's chosen number.

print('Your guess is too low.')

if guess < minNumber: # This 'if' statement gives an output if the player's guess is less than 'minNumber'.

print('You\'ve wasted this guess, as your number is below the minimum possible choice for the computer.')

elif guess > numberChosen: # This 'elif' statement gives an output if the player's guess is greater than the computer's chosen number.

print('Your guess is too high.')

if guess > maxNumber: # This 'if' statement gives an output if the player's guess is greater than 'maxNumber'.

print('You\'ve wasted this guess, as your number is above the maximum possible choice for the computer.')

elif guess == numberChosen: # This 'elif' statement gives an output if the player's guess is equal to the computer's chosen number.

guessed = True

return guessed

def displayResult(playerName, numberChosen, guess, guessesTaken):

if guess == numberChosen: # This 'if' statement gives an output depending on whether the player got the computer's number right or not.

print('\nWell done, ' + playerName + '! You guessed my number in ', end='') # , end='' makes the next 'print' function continue on the same line.

if guessesTaken > 1: # This 'if' statement gives an output depending on how many guesses the player took.

print(str(guessesTaken) + ' guesses!')

else:

print('one guess!')

else:

print('\nBad luck. The number I was thinking of was ' + str(numberChosen) + '.')

def newGameRequest(playerName):

replay = True

print('\nDo you want another game, ' + playerName + '?')

newGameYesNo = str.lower(input()) # 'str.lower' converts a string to lower case.

if newGameYesNo != 'yes' and newGameYesNo != 'y': # This 'if' statement ends the program if the player enters anything other than 'yes' or 'y'.

print('\nThank you for playing \'Guess The Number\', ' + playerName + ".")

replay = False

return replay

if \_\_name\_\_ == '\_\_main\_\_': # This runs main() if this module has not been imported into another module.

main()

#-------------------------------------------------------------------------------

# Name: Guess the Number

# Purpose: Number guessing game

#

# Author: PMP

#

# Created: 19/01/2013

# Copyright: (c) PMP 2013

# Licence: n/a

#-------------------------------------------------------------------------------

# This is a Guess the Number game from 'Invent Your Own Computer Game With Python' adapted by PMP.

# This version (08) puts the game in a window using PyGame and PygCurse.

# Variables in a user-defined function (def) are local by default.

# If a variable in a def is declared as global, references to it in defs executed later usually treat it as global without specific redeclaration.

# If a def uses a statement like x = x + 1 and x is intended to be a global variable, it must be declared as global in the def, even if it has been declared already in an earlier executed def.

from sys import exit # This imports just the exit function from Sys.

from pygame import quit # This imports just the quit function from PyGame.

import pygcurse as pc # This imports the PygCurse text console add-on for PyGame.

import random as r # This imports randomizing functions.

import math as m # This imports mathematical functions.

win = pc.PygcurseWindow(27, 30, 'Guess the Number') # Creates a window 27 characters wide x 30 lines high.

print = win.pygprint # Makes the PygCurse pygprint function replace standard print().

input = win.input # Makes the PygCurse input function replace standard input().

win.setscreencolors('white', 'black', clear=True) # Sets the colours for the window and clears it (necessary to allow proper scrolling).

def main(): # This is the game main function.

displayTitle()

playerN = inputName()

minNum = setNumberLimitLow()

maxNum = setNumberLimitHigh(minNum)

maxGuess = setMaxGuesses(minNum, maxNum)

while True: # This 'while' loop runs as long as the player asks to continue playing after each game.

numberC = startGame(playerN, minNum, maxNum, maxGuess)

guessTaken = 0

while guessTaken < maxGuess: # This 'while' loop runs as long as the number of guesses taken is less than the maximum number of guesses.

askForGuess(maxGuess, guessTaken)

mGuess = makeGuess()

guessTaken = guessTaken + 1

guessedN = checkGuess(minNum, maxNum, numberC, mGuess)

if guessedN == True:

break

displayResult(playerN, numberC, mGuess, guessTaken)

replay = newGameRequest(playerN)

if replay == False:

break

quit() # This closes the PyGame / PygCurse window.

exit() # This exits the program.

def displayTitle():

win.colors = ('blue', 'yellow')

print('\n' + '\*' \* 26 + '\n\*' + ' ' \* 24 + '\*\n\* Number Guessing Game \*\n\*' + ' ' \* 24 + '\*\n' + '\*' \* 26 + '\n\n')

# At the start of the game, this gives a title surrounded by asterisks.

# Note that '\*' \* 26 means a string of 26 asterisks, ' ' \* 24 means a string of 24 spaces, and \n means 'start a new line'.

def inputName():

win.colors = ('white', 'black')

print('Hello! What is your name?')

win.fgcolor = 'aqua'

playerName = input() # This allows the player to enter his/her name.

return playerName

def setNumberLimitLow():

win.colors = ('white', 'black')

print('\nWhat is the lowest number\nthe computer may choose?\nIt must be an integer\nbetween 1 and 1,000,000.')

while True:

while True: # This 'while' loop checks whether the player's entry is valid (i.e. a number, not a letter or word).

win.fgcolor = 'aqua'

minNumber = input() # This allows the player to enter a number.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

minNumber = int(minNumber) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

win.fgcolor = 'red'

print('You must enter integers,\nnot fractions or letters -\ntry again.')

minNumber = int(minNumber)

if minNumber < 1 or minNumber > 1000000:

win.fgcolor = 'red'

print('This number is outside\nthe acceptable range -\ntry again.')

else:

break

return minNumber

def setNumberLimitHigh(minNumber):

win.colors = ('white', 'black')

print('\nWhat is the highest number\nthe computer may choose?\nIt must be an integer\nbetween 1 and 1,000,000\nand at least 9 greater\nthan the lowest number.')

while True:

while True: # This 'while' loop checks whether the player's entry is valid (i.e. a number, not a letter or word).

win.fgcolor = 'aqua'

maxNumber = input() # This allows the player to enter a number.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

maxNumber = int(maxNumber) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

win.fgcolor = 'red'

print('You must enter integers,\nnot fractions or letters -\ntry again.')

maxNumber = int(maxNumber)

if maxNumber < 1 or maxNumber > 1000000 or maxNumber < minNumber + 9:

win.fgcolor = 'red'

print('This number is outside\nthe acceptable range -\ntry again.')

else:

break

return maxNumber

def setMaxGuesses(minNumber, maxNumber):

maxGuesses = int(m.log(maxNumber + 1 - minNumber, 2) + 1)

return maxGuesses

def startGame(playerName, minNumber, maxNumber, maxGuesses):

win.colors = ('lime', 'black')

numberChosen = r.randint(minNumber, maxNumber) # This sets the variable 'number' to an integer between 'minNumber' and 'maxNumber' inclusive.

print('\nWell, ' + playerName + ',\n I\'m thinking of a number\n between ' + str(minNumber) + ' and ' + str(maxNumber) + '.')

print('You have ' + str(maxGuesses) + ' guesses\n in which to work it out.')

# \' allows an apostrophe to be used within a string.

# str() converts a numeric variable to a string variable (so that it can be displayed correctly with 'print').

return numberChosen

def askForGuess(maxGuesses, guessesTaken):

win.colors = ('lime', 'black')

print() # This prints a blank line.

if maxGuesses - guessesTaken > 1: # This 'if' statement gives an output if the player has more than one guess left.

print('You have ' + str(maxGuesses - guessesTaken) + ' guesses left.')

else:

print('You have 1 guess left.')

print('What do you think\n my number is?')

def makeGuess():

win.colors = ('aqua', 'black')

while True: # This 'while' loop checks whether the player's guess is valid (i.e. a number, not a letter or word).

guess = input() # This allows the player to enter a guess.

try: # The 'try' statement allows the program to check whether the player's entry is valid.

guess = int(guess) # This tries to treat the player's entry as an integer number.

break # If the player's entry is something which can be converted to an integer, then 'break' out of this 'while' loop.

except ValueError: # If the player's entry cannot be converted to an integer, then print an error message.

win.fgcolor = 'red'

print('You must enter numbers,\nnot letters -\ntry again.')

guess = int(guess) # This converts the player's (string) entry to an integer.

return guess

def checkGuess(minNumber, maxNumber, numberChosen, guess):

win.colors = ('yellow', 'black')

guessed = False

if guess < numberChosen: # This 'if' statement gives an output if the player's guess is less than the computer's chosen number.

print('Your guess is too low.')

if guess < minNumber: # This 'if' statement gives an output if the player's guess is less than 'minNumber'.

win.fgcolor = 'red'

print('You\'ve wasted this guess,\nas your number is below\nthe minimum possible\nchoice for the computer.')

elif guess > numberChosen: # This 'elif' statement gives an output if the player's guess is greater than the computer's chosen number.

print('Your guess is too high.')

if guess > maxNumber: # This 'if' statement gives an output if the player's guess is greater than 'maxNumber'.

win.fgcolor = 'red'

print('You\'ve wasted this guess,\nas your number is above\nthe maximum possible\nchoice for the computer.')

elif guess == numberChosen: # This 'elif' statement gives an output if the player's guess is equal to the computer's chosen number.

guessed = True

return guessed

def displayResult(playerName, numberChosen, guess, guessesTaken):

win.colors = ('lime', 'black')

if guess == numberChosen: # This 'if' statement gives an output depending on whether the player got the computer's number right or not.

print('\nWell done, ' + playerName + '!\nYou guessed my number\nin ', end='') # , end='' makes the next 'print' function continue on the same line.

if guessesTaken > 1: # This 'if' statement gives an output depending on how many guesses the player took.

print(str(guessesTaken) + ' guesses!')

else:

print('one guess!')

else:

win.fgcolor = 'red'

print('\nBad luck.\nThe number I was thinking\nof was ' + str(numberChosen) + '.')

def newGameRequest(playerName):

win.colors = ('lime', 'black')

replay = True

print('\nDo you want another game,\n' + playerName + '?')

win.fgcolor = 'aqua'

newGameYesNo = str.lower(input()) # 'str.lower' converts a string to lower case.

win.fgcolor = 'lime'

if newGameYesNo != 'yes' and newGameYesNo != 'y': # This 'if' statement ends the program if the player enters anything other than 'yes' or 'y'.

print('\nThank you for playing\n\'Guess The Number\',\n' + playerName + ".")

win.fgcolor = 'yellow'

print('\nPress a key\n to close the window.')

pc.waitforkeypress()

replay = False

return replay

if \_\_name\_\_ == '\_\_main\_\_': # This runs main() if this module has not been imported into another module.

main()

#-------------------------------------------------------------------------------

# Name: Guess the Number

# Purpose: Number guessing game

#

# Author: PMP

#

# Created: 20/01/2013

# Copyright: (c) PMP 2013

# Licence: n/a

#-------------------------------------------------------------------------------

# This is a Guess the Number game from 'Invent Your Own Computer Game With Python' adapted by PMP.

# This version (09) converts the game to a GUI base using Tkinter.

import tkinter as tk # Adds a module to Python to allow GUI development.

#import tkinter.ttk as ttk # Adds new-style widgets to tkinter (not used in this program).

import tkinter.messagebox as tm # Adds a module to Python to allow message boxes to be used.

import random as r # Adds a module to Python to allow random numbers to be generated.

import math as m # Adds a module to Python containing various mathematical functions.

class App:

def \_\_init\_\_(self, master):

self.fontMain = ('MS Sans', 12) # Sets the main font to use for widgets.

self.fontSpace = ('MS Sans', 4) # Sets the font for labels used as spacers in a frame.

master.title('Guess the Number') # Puts a title on the program window.

master.geometry('360x200+400+200') # Sets the window size (360x200) and initial screen position (400 across, 200 down from top left).

self.frameSetupGame = tk.Frame(master, width=360, height=90, padx=4, pady=8, borderwidth=2, relief='flat') # Sets up a frame to hold widgets.

self.framePlayGame = tk.Frame(master, width=360, height=200, padx=4, pady=4, borderwidth=2, relief='flat') # Sets up a frame to hold widgets.

self.frameSetupGame.grid\_propagate(0) # .grid\_propagate(0) ensures that a frame stays at its set dimensions when displayed.

self.framePlayGame.grid\_propagate(0)

self.frameSetupGame.grid(column=0, row=0) # .grid(...) displays a widget on screen at a column and row position within a user-defined grid.

self.framePlayGame.grid(column=0, row=1)

self.frameSetup(master) # Calls the function 'frameSetup' (passing a reference to 'master', the main program window).

def frameSetup(self, master): # This section sets up the contents (labels, buttons, etc.) of the two frames defined in \_\_init\_\_.

# This section defines the widgets for the first (SetupGame) frame.

self.labelNumFrom = tk.Label(self.frameSetupGame, font=self.fontMain, text='Number to guess is from : ')

self.labelNumTo = tk.Label(self.frameSetupGame, font=self.fontMain, text=' to : ')

self.numhidefault = tk.StringVar() # This defines a variable to use in Spinbox 'spinboxHi' so that a starting value can be set for it.

self.numhidefault.set('100') # This sets the starting value for the 'spinboxHi' variable defined above to '100'.

self.spinboxLo = tk.Spinbox(self.frameSetupGame, font=self.fontMain, from\_=1, to=9991, increment=1, width=4)

self.spinboxHi = tk.Spinbox(self.frameSetupGame, font=self.fontMain, from\_=8, to=9999, increment=1, width=4, textvariable=self.numhidefault)

# In a Spinbox: 'from\_' and 'to' set the range of possible values; 'increment' sets the amount of change when the Spinbox arrows are clicked;

# 'width' is how many characters the Spinbox can show.

self.buttonPlay = tk.Button(self.frameSetupGame, font=self.fontMain, text='Play', background='yellow', padx=2, pady=2,

command=lambda:self.startPlay(master, self.spinboxLo.get(), self.spinboxHi.get()))

# Buttons contain a 'command' which activates a function when the button is pressed;

# 'lambda:' is necessary to prevent the function from running when the button is first placed in a window.

# This section places the widgets for the first (SetupGame) frame and displays them on screen.

self.labelNumFrom.grid(column=0, row=0)

self.labelNumTo.grid(column=2, row=0)

self.spinboxLo.grid(column=1, row=0)

self.spinboxHi.grid(column=3, row=0)

self.buttonPlay.grid(column=0, row=1, columnspan=4, pady=8)

# This section defines the widgets for the second (PlayGame) frame.

self.labelGuessLimits = tk.Label(self.framePlayGame, font=self.fontMain, text='')

self.stringMaxGuesses = tk.StringVar()

self.labelMaxGuesses = tk.Label(self.framePlayGame, font=self.fontMain, textvariable=self.stringMaxGuesses)

self.labelYourGuess = tk.Label(self.framePlayGame, font=self.fontMain, text='Your guess : ')

self.stringPlayerGuess = tk.StringVar()

self.entryGuess = tk.Entry(self.framePlayGame, font=self.fontMain, width=5, textvariable=self.stringPlayerGuess)

self.buttonGuess = tk.Button(self.framePlayGame, font=self.fontMain, text='Make guess', background='green', padx=2, pady=2,

command=lambda:self.readGuess(self.numtoguess))

self.labelComment = tk.Label(self.framePlayGame, font=self.fontMain, text='')

self.labelSpacer = tk.Label(self.framePlayGame, font=self.fontSpace, text=' ')

self.buttonNew = tk.Button(self.framePlayGame, font=self.fontMain, text='New game', background='cyan', padx=2, pady=2,

command=lambda:self.newGame(master))

self.buttonQuit = tk.Button(self.framePlayGame, font=self.fontMain, text='Quit', background='red', padx=2, pady=2,

command=lambda:self.endGame(master))

# This section places the widgets for the second (PlayGame) frame and displays them on screen.

self.labelGuessLimits.grid(column=0, row=0, columnspan=4, pady=4, sticky='w')

self.labelMaxGuesses.grid(column=0, row=1, columnspan=4, pady=4, sticky='w')

self.labelYourGuess.grid(column=0, row=2, sticky='w')

self.entryGuess.grid(column=1, row=2, pady=4, sticky='w')

self.buttonGuess.grid(column=3, row=2, pady=4, sticky='e')

self.labelComment.grid(column=0, row=3, columnspan=4, pady=4, sticky='w')

self.labelSpacer.grid(column=0, row=4, columnspan=4)

self.buttonNew.grid(column=0, row=5, columnspan=3, padx=4, pady=4)

self.buttonQuit.grid(column=3, row=5, padx=4, pady=4)

self.framePlayGame.grid\_remove() # Hides the second (PlayGame) frame.

def startPlay(self, master, userNumlo, userNumhi): # This function closes the SetupGame frame and opens the PlayGame frame when 'Play' is pressed.

try: # Checks if player's choice for the lower number of the range to guess is an integer and gives an error message if not.

numlo = int(userNumlo)

except ValueError:

tm.showwarning('Warning', 'The lower number in the range to guess\nmust be an integer.')

return

try: # Checks if player's choice for the higher number of the range to guess is an integer and gives an error message if not.

numhi = int(userNumhi)

except ValueError:

tm.showwarning('Warning', 'The higher number in the range to guess\nmust be an integer.')

return

if numhi < numlo+7: # Checks if the range of numbers to guess is suitable.

tm.showwarning('Warning', 'The higher number in the range to guess\nmust be at least 7 greater than the lower number.')

# Opens a text message box to give a warning message.

else:

self.frameSetupGame.grid\_remove() # Hides the first (SetupGame) frame.

self.framePlayGame.grid() # Displays the second (PlayGame) frame.

self.labelGuessLimits.config(text='The number to guess is between ' + str(numlo) + ' and ' + str(numhi) + '.')

self.numofguesses = int(m.log(numhi + 1 - numlo, 2) + 1) # Calculates an appropriate number of guesses for the range of numbers chosen.

self.numtoguess = r.randint(numlo, numhi) # Gives a random number within the range of numbers chosen.

self.guessesTaken = 0 # Resets the counter for the number of guesses so far taken by the player to 0.

self.showGuessesLeft() # Calls a function to display the number of guesses left in a label.

self.labelComment.config(text='') # Clears the label which displays comments based on the player's guess.

self.entryGuess.config(state=tk.NORMAL) # Enables the entry box which allows the player to enter a guess.

self.buttonGuess.config(state=tk.NORMAL) # Enables the 'Make guess' button.

self.buttonNew.config(state=tk.DISABLED) # Disables the 'New game' button.

self.buttonQuit.config(state=tk.DISABLED) # Disables the 'Quit' button.

self.stringPlayerGuess.set('') # Clears the contents of the entry box in which the player enters a guess.

self.entryGuess.focus\_set() # Sets the focus of the cursor so that the player can enter a guess directly.

def readGuess(self, ntg): # This function checks the player's guess and reacts accordingly.

readg = 0 # Initializes the player's guess as 0 - necessary for 'try' to work correctly.

try: # Checks player's guess is an integer and gives an error message if not.

readg = int(self.entryGuess.get()) # Reads the player's guess into an integer variable 'readg'.

except ValueError:

self.labelComment.config(text='This isn\'t an integer - try again.')

return # The function ends if the guess is not an integer.

self.guessesTaken +=1 # Adds one to 'guessesTaken', the counter for the number of guesses taken so far by the player.

self.showGuessesLeft() # Calls a function to display a label showing the number of guesses taken so far.

if self.guessesTaken <= self.numofguesses: # This 'if' block gives comments on the player's guess as long as there are guesses left.

if readg < ntg:

self.labelComment.config(text='Your guess is too low.')

elif readg > ntg:

self.labelComment.config(text='Your guess is too high.')

elif readg == ntg:

self.labelComment.config(text='Well done - your guess is correct!')

if (self.guessesTaken == self.numofguesses) and (readg != ntg): # This 'if' block gives a comment if the player has run out of guesses without getting the correct answer.

self.labelComment.config(text='Bad luck - the answer is ' + str(ntg) + '.')

if (self.guessesTaken >= self.numofguesses) or (readg == ntg): # This 'if' block prevents the player from making more guesses.

self.entryGuess.config(state=tk.DISABLED) # Disables the entry box for the player's guesses.

self.buttonGuess.config(state=tk.DISABLED) # Disables the 'Make guess' button.

self.buttonNew.config(state=tk.NORMAL) # Enables the 'New game' button.

self.buttonQuit.config(state=tk.NORMAL) # Enables the 'Quit' button.

def showGuessesLeft(self): # This function changes the content of a variable for use in 'labelMaxGuesses'.

self.stringMaxGuesses.set('You have ' + str(self.numofguesses-self.guessesTaken) + ' out of ' + str(self.numofguesses) + ' guesses left.')

def newGame(self, master): # This functions hides the game-play frame and displays the set-up frame.

self.frameSetupGame.grid()

self.framePlayGame.grid\_remove()

def endGame(self, master): # This function ends the game - both 'destroy' and 'quit' are necessary.

master.destroy() # Closes the program window.

master.quit() # Ends the program.

def main():

root = tk.Tk() # Sets up an empty program window.

app = App(root) # Runs the application as an instance of Class App in the window set up above (i.e. starts the program).

root.mainloop() # Makes the program continue until ended by the user.

if \_\_name\_\_ == '\_\_main\_\_': # This runs main() if this module has not been imported into another module.

main()