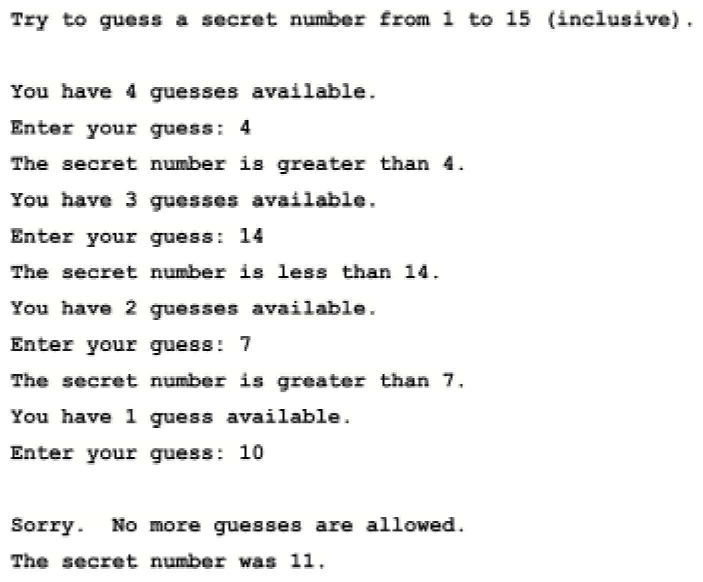
Guessing Game

This guessing game is based on an algorithm called a “binary search”. The goal is to guess a number within a certain range in the fewest number of guesses. The game has two participants, one (the *holder*) is holding a secret number selected at random. The other (the *guesser*) will make guesses at what the number is. When the guesser makes a guess, the holder tells the guesser if the guess is correct or if the secret number is higher or lower than the guess. The guesser has a limited number of guesses, so someone guessing at random is likely to lose. To guess a number from 1 through 255, the holder will only allow 8 guesses. Let’s look at some small examples guessing a number from 1 through 15. The guesser gets 4 opportunities to guess correctly.

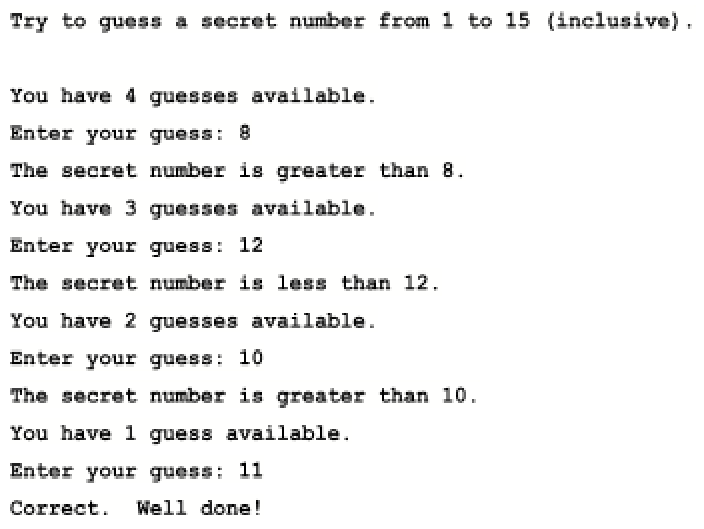
An example of guessing at random. Random guesses do not guarantee success with four tries (unless the guesser has superpowers!).



```
Try to guess a secret number from 1 to 15 (inclusive).

You have 4 guesses available.
Enter your guess: 4
The secret number is greater than 4.
You have 3 guesses available.
Enter your guess: 14
The secret number is less than 14.
You have 2 guesses available.
Enter your guess: 7
The secret number is greater than 7.
You have 1 guess available.
Enter your guess: 10

Sorry.  No more guesses are allowed.
The secret number was 11.
```

A strategy is necessary. With each successive guess, the user can divide the range of possible guesses in half to win the game. So, the first guess will be 8 and the next guess will be either 8 plus 4 (if the secret number is greater than 8) or 8 minus 4 (if the secret number is less than 8).

Here’s an example of guessing with this strategy. It’s called a “binary search”. A binary search depends on knowing two pieces of information: the search range and if each guess is too high or too low.



```
Try to guess a secret number from 1 to 15 (inclusive).

You have 4 guesses available.
Enter your guess: 8
The secret number is greater than 8.
You have 3 guesses available.
Enter your guess: 12
The secret number is less than 12.
You have 2 guesses available.
Enter your guess: 10
The secret number is greater than 10.
You have 1 guess available.
Enter your guess: 11
Correct.  Well done!
```

The amazing thing about a binary search is that every time the search range doubles, only one extra guess is required. To guess from 1 through 4095 only requires 12 guesses!

To calculate how many guesses are required to guess any range from 1 to n, calculate the exponent of 2 that results in n. You may recall from secondary school math that such a calculation is called a *logarithm* and, in this case, we need to calculate the *radix 2* logarithm ($\log_2$). Fortunately, Python’s **`math`** module can do this for us. If we import the math library, then

Guessing Game

we can use the **`math.log2()`** function to compute the number of guesses required for any range from 1 through nMax like this:

```
nGuesses = int(math.log2(nMax)+1) # number of guesses
```

You are encouraged to try it out in the Python shell to verify it’s validity.

The Guessing Game Project. In this project, you will write a Python program to act as the holder. The program will ask the user to provide the maximum number in the range of guesses (called nMax, above). The program uses nMax to determine: (1) the number of allowable guesses (called nGuesses, above) and (2) the secret number. To calculate nGuesses, import the math module. To determine the secret number, import the random module and use the **`randrange()`** function. Experiment with these functions in the Python shell until you are comfortable with them. Once your program has established the range of numbers to guess, the number of allowable guesses and the secret number, you can use a for loop and the range function to count down the number of available guesses and implement a user interface to implement the role of the holder. Here’s an example of a successful run of the program …

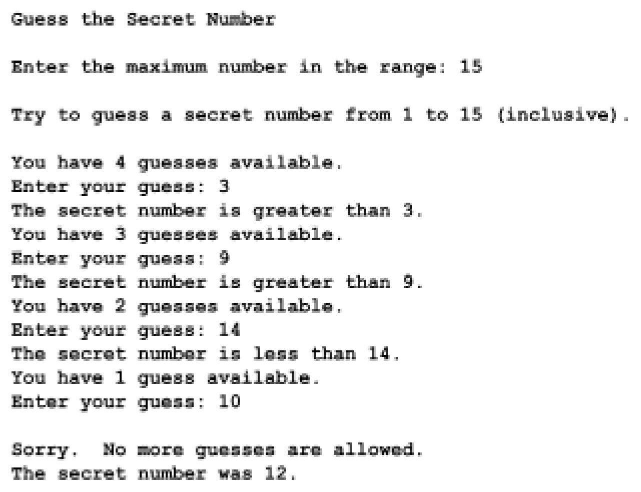
```
Guess the Secret Number

Enter the maximum number in the range: 31

Try to guess a secret number from 1 to 31 (inclusive).

You have 5 guesses available.
Enter your guess: 16
The secret number is less than 16.
You have 4 guesses available.
Enter your guess: 8
The secret number is less than 8.
You have 3 guesses available.
Enter your guess: 4
The secret number is greater than 4.
You have 2 guesses available.
Enter your guess: 6
The secret number is less than 6.
You have 1 guess available.
Enter your guess: 5
Correct.  Well done!
```

Not all interactions will require all of the available guesses as in the example to the right. If the secret number were even, it would take fewer tries to guess the number as shown below ...

```
You have 5 guesses available.
Enter your guess: 16
The secret number is less than 16.
You have 4 guesses available.
Enter your guess: 8
The secret number is greater than 8.
You have 3 guesses available.
Enter your guess: 12
Correct.  Well done!
```

An unsuccessful attempt at the game is shown below …

Guessing Game



```
Guess the Secret Number

Enter the maximum number in the range: 15

Try to guess a secret number from 1 to 15 (inclusive).

You have 4 guesses available.
Enter your guess: 3
The secret number is greater than 3.
You have 3 guesses available.
Enter your guess: 9
The secret number is greater than 9.
You have 2 guesses available.
Enter your guess: 14
The secret number is less than 14.
You have 1 guess available.
Enter your guess: 10

Sorry.  No more guesses are allowed.
The secret number was 12.
```

HINTS:

1. Pay attention to detail. Note how the word “guess” changes from plural (guesses) to singular (guess) when only one opportunity remains.
    a. “2 guesses available.”
    b. “1 guess available.”
2. When the guesser loses. This is one opportunity for you to use the **`else`** clause associated with the **`for`** loop. Statements in the else clause will be executed when all available guesses have been used up.
3. When the guesser wins. This is an opportunity to use the break statement.
4. Some simple f-strings may save you a few lines of code here. My complete program (minus comments) is 25 lines long.