Tutorial Objectives

- Practice writing code using **functions**, as well as the other control structures we have used previously
- Practice organizing large programs using a **procedural design**

In this tutorial you will be creating a single large program organized into several separate procedures. Start by creating a file called tutorial5.py, and skim over all four parts before beginning in order to get an idea of where you're going.

Problem 1 – Powers of two without going over

Write and test a function called `pow2()` that takes one (positive) number as input and returns the largest power of 2 that's less than or equal to that number. Some examples include:

- \( \text{pow2}(2) \to 2 \)
- \( \text{pow2}(12) \to 8 \)
- \( \text{pow2}(20) \to 16 \)
- \( \text{pow2}(256) \to 256 \)

Problem 2 – Decimal to Binary

Write and test a function called `decToBin()` that takes in a decimal number as an argument and returns a string representing that number in binary. Your solution should **not** use the built-in `bin()` function (though it may be helpful when testing your function to verify your answers). The design for this algorithm should work as follows:

Let \( x \) be the input value,
Let \( p \) be the largest power of 2 less than or equal to \( x \),
and Let output be "".

Repeat until \( p \) is 0:
- If \( p \leq x \),
  Then subtract \( p \) from \( x \), and add a "1" to the output.
- Otherwise add a "0" to the output

Divide \( p \) by 2
Return output
Some examples:

\[
\begin{align*}
\text{decToBin}(2) & \rightarrow "10" \\
\text{decToBin}(10) & \rightarrow "1010" \\
\text{decToBin}(16) & \rightarrow "10000"
\end{align*}
\]

Problem 3 – User inputs
Write a function called userBin() that takes an integer as an argument. This function should display the integer to the user and prompt them to enter the corresponding binary value. Have your function simply return the user's input (it will be used in the next problem).

Problem 4 – Making a game
Finally, create a main() function for your program. The purpose of this function is to control the game-logic for a Binary Quiz game. The steps of this game are as follows:

- Generate a random integer in the range \([0, 256]\)
- Ask the user to enter the corresponding binary number.
- Check if their answer is correct or not
  - If so, add 1 to their score

The above steps should repeat until the user types "stop".

Some tips:

- Your program should not use the built-in bin() function, use the decToBin function you created.
- The user's input, and the result of decToBin are both strings, so don't convert the binary numbers to integers.
- The user's score should be displayed each round.