MTH 3300
Fall 2022

Programming Problem 2: binary.py

Directions: Download the template files I have provided on Blackboard. Then open Spyder, load these template files, and write the following programs. Submit your source code via Gradescope, in `.py` format. READ THE INSTRUCTIONS on how to submit your work in the Course Documents section of Blackboard.

Specify collaborators/resources used or explicitly specify that none were used in the comments at the top of your `.py` file. Failure to include this will result in a zero on the assignment.

Be sure to read the SPECIFICATIONS carefully! And write comments!

Recall that we briefly discussed the *binary system* (or *base two*) in class. A number is represented in binary by a sequence of `1`'s and `0`'s (also known as *bits*), which have place values given by powers of 2 instead of powers of 10. In a four-digit binary number, the left digit would be the 8's digit; the next digit would be the 4's digit, the third would be the 2's digit, followed by the 1's digit.

For example, `1101` in binary would represent the (base-10) number 13, because this number has, reading from the left, one 8, one 4, no 2, and one 1, and $8 + 4 + 1 = 13$. And `0111` would represent the (base-10) number 7, since this number has no 8, one 4, one 2, and one 1.

Write a program that takes input from the user the 8's digit, 4's digit, 2's digit, and 1's digit of a 4-digit binary number and prints its base-10 equivalent. The program should then display those four digits in a row, together with the equivalent decimal number.

Next, your program should generate a random integer between 0 and 15 (inclusive) and print the boolean value (`True` or `False`) that results from checking if the binary number is less than the randomly generated number. Specifically, your program should use `randrange` from the `random` module to generate a single random number (let's call it `x`) between 0 and 15 inclusive. Then print the result of the comparison `num<x`, where `num` is the base-10 version of the binary number.

When you run your program, a sample run should look like this:

```
Enter 8's digit: 1
Enter 4's digit: 1
Enter 2's digit: 0
Enter 1's digit: 0
The binary number 1100 is 12 in base ten
Less than the randomly generated number 5? False
```

where `5` is randomly generated, or this:

```
Enter the 8's digit: 0
Enter the 4's digit: 0
Enter the 2's digit: 1
Enter the 1's digit: 1
The binary number 0011 is 3 in base ten
Less than the randomly generated number 9? True
```

where `9` is randomly generated.

Note that every time I run the program, I get a different number (between 0 and 15). (I have some code at

the start of the template that you should comment out to test this, but make sure to leave it in when uploading to Gradescope! If you run the code in Spyder without commenting these lines out, you should see nothing.)

You should complete this program only using things we have discussed already in the class: **don't** use lists or tuples (these are sequences of variables enclosed in either `[]` or `()`), **don't** use loops, and definitely **don't** use any of Python's functions for binary (like `bin`).

Specifications: your program must

- Get the four digits of the binary number as input from the user. You may assume the user complies and enters only a `0` or a `1` for each of the binary digits. (You do *not* have to worry about the user entering something like `3` or `2.2` or `1.0` for the binary digits.)
- use `random.randrange` to generate a number between 0 and 15 inclusive. Display this randomly generated number as specified above in the sample runs.
- display the four digits, as well as the equivalent decimal number, in the manner shown above. In particular, I want to see binary number listed like `1100`, *not* like `(1, 1, 0, 0)` or `1 1 0 0`.
- display the comparison result between the binary and randomly generated numbers as specified and in the format above.
- only uses techniques that we have discussed in class thus far – in particular, **no use of the `bin` function, no use of loops, and no use of lists or tuples**.

Note: This problem has five visible test cases on Gradescope. You will see if your code passes these five test cases. In addition, it has three invisible test cases, which you will not be able to see at all until after grades are released (you will not see if your code passed or did not pass these). So make sure you run additional tests on your own to cover all the possible inputs described!

Your input prompts and output must match the format given above *exactly* (including spaces) to pass the test cases on Gradescope.

Failure to follow specifications will result in deduction of points, even if test cases pass.