Mini-Project: The Number-Guessing Game

Python for Absolute Beginners · Lesson 16

What you'll learn

- How a whole program is assembled from the ideas in Lessons 1–15
- How to pick a secret number with random
- How to read the player's guess with input
- How a loop, an if chain, and a counter work together
- How to test your game and make it yours

The game, in plain words

- Before writing any code, say what the program must do, in plain words
- 1. The computer secretly picks a number between 1 and 10
- 2. The player guesses
- 3. After each guess, the computer says “too high”, “too low”, or “you got it!”
- 4. The game counts the guesses and reports the total when the player wins
- Every line of code below exists to do one of those four steps

A new tool: random

- random is a built-in tool for chance: random.randint(1, 10) picks a whole number from 1 to 10
- Like Turtle, it must be loaded first: import random
- You already know input() from Lesson 7 — it asks the player for a guess.

Reading a guess

```
    guess_text = input("Guess a number from 1 to 10: ")
guess = int(guess_text)
print(guess)

7
```

If the player types 7, this prints 7 — as a number now, ready for comparison.

The whole game — part 1: setup and loop

```
import random

secret = random.randint(1, 10)
tries = 0

print("I am thinking of a number from 1 to 10.")
print("Can you guess it?")

while True:
    guess_text = input("Your guess: ")
    guess = int(guess_text)
    tries = tries + 1
```

The secret is picked once, the counter starts at zero, and the loop repeats until the player wins.

The whole game — part 2: hints and win

```
                    if guess == secret:
        print("You got it!")
        print("It took you " + str(tries) + " tries.")
        break
    elif guess < secret:
        print("Too low — try higher.")
    else:
        print("Too high — try lower.")
```

Exactly one response per guess: equal, too low, or too high. Break exits the loop the moment the player wins.

A sample run

```
          I am thinking of a number from 1 to 10.
Can you guess it?
Your guess: 3
Too low — try higher.
Your guess: 8
Too high — try lower.
Your guess: 5
You got it!
It took you 3 tries.
```

The secret is different every time, so your run will differ — that is why the game stays fun.

How the pieces fit together

- while True means “repeat forever, until something stops the loop” — that something is break
- Without break, the game would never end: the infinite-loop mistake from Lesson 10, used deliberately here
- The if / elif / else chain gives exactly one response per guess, and == is checked first
- tries = tries + 1 counts guesses; str(tries) converts the count to text for the final message
- random.randint(1, 10) picks a fresh secret every run, so the game is different each time

Make it yours

- Change the range to 1–100 for a harder game
- Count down from 5 guesses instead of counting up, and end with a “you ran out of guesses” message
- Track the player's best score across rounds, with a variable kept outside the loop
- Handle a non-number guess gracefully instead of letting int() complain
- Try one change at a time, and run the game after each change

Hands-on exercise

- Type the full game into guessing_game.py and play it until you win at least once
- Change the range to 1–20 and play again — what changed about the difficulty, and why?
- Add a line at the end, next to the win message, so it reads “You got it in 3 tries!” as one sentence
- Pick one “make it yours” idea above and build it

Key takeaways

- Design in plain words first: four sentences became the whole game.
- random.randint() picks a secret; input() reads the player; int() converts text to a number.
- while True: plus break repeats until the win condition ends it.
- The if/elif/else chain gives exactly one response per guess.
- Change a working program one thing at a time — that is how programmers grow.

Take the quiz

Take the Lesson 16 quiz — 5 quick questions to check what you learned.

Next lesson

What's Next: Stepping into Python 3 Programming