

more while loops.

Week 4 | Lecture 2 (4.2)

if nothing else, write **#cleancode**.

This Week's Content

- **Lecture 4.1**
 - function review, while loops
- **Lecture 4.2**
 - more loops
- **Lecture 4.3**
 - Midterm Review

While Loops

- The **while loop** keeps executing a piece of code as long as a particular condition is **True**.
- There must be a colon (:) at the end of the while statement.
- The action to be performed must be indented.

Must evaluate to
True or False

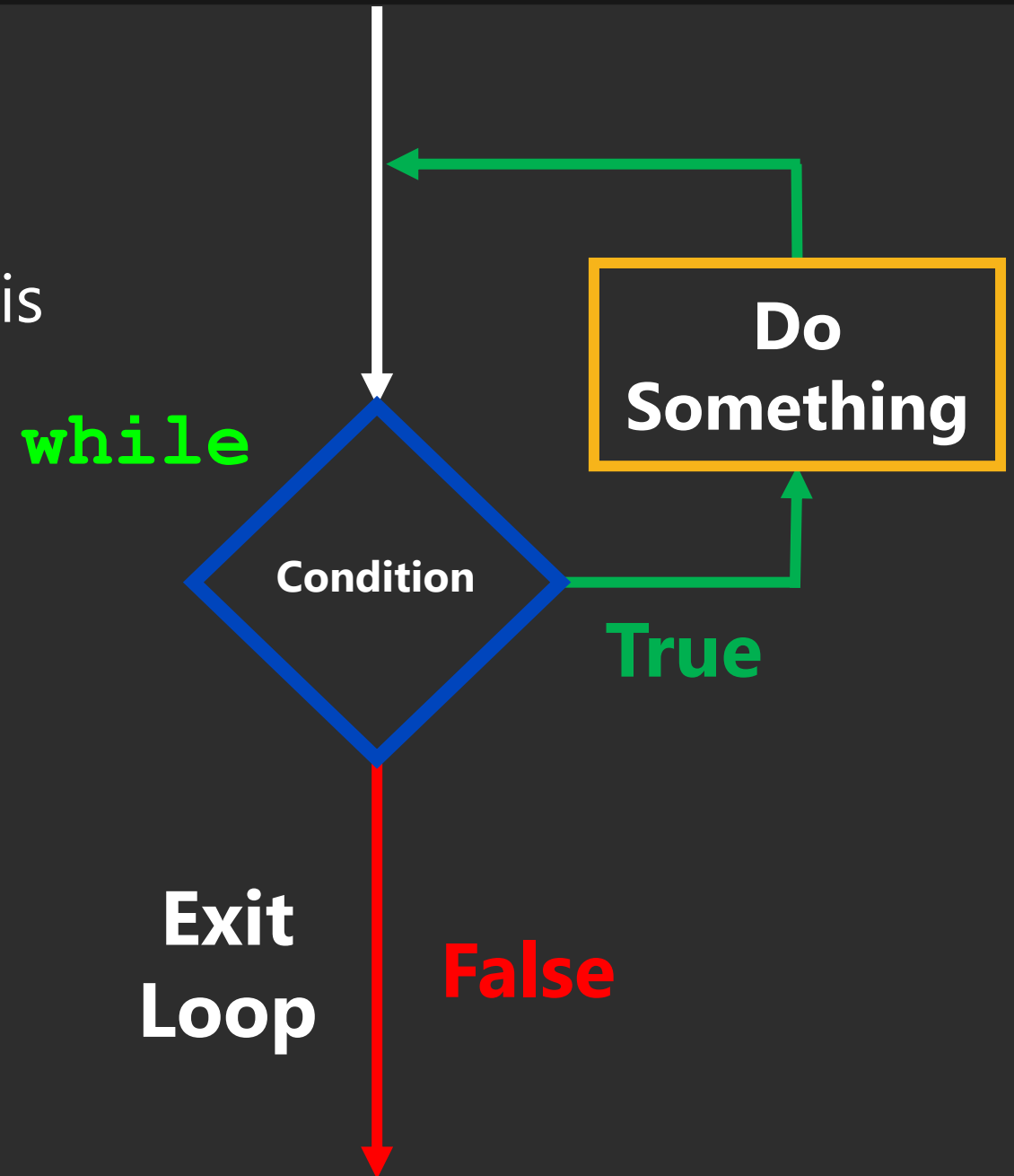
Colon

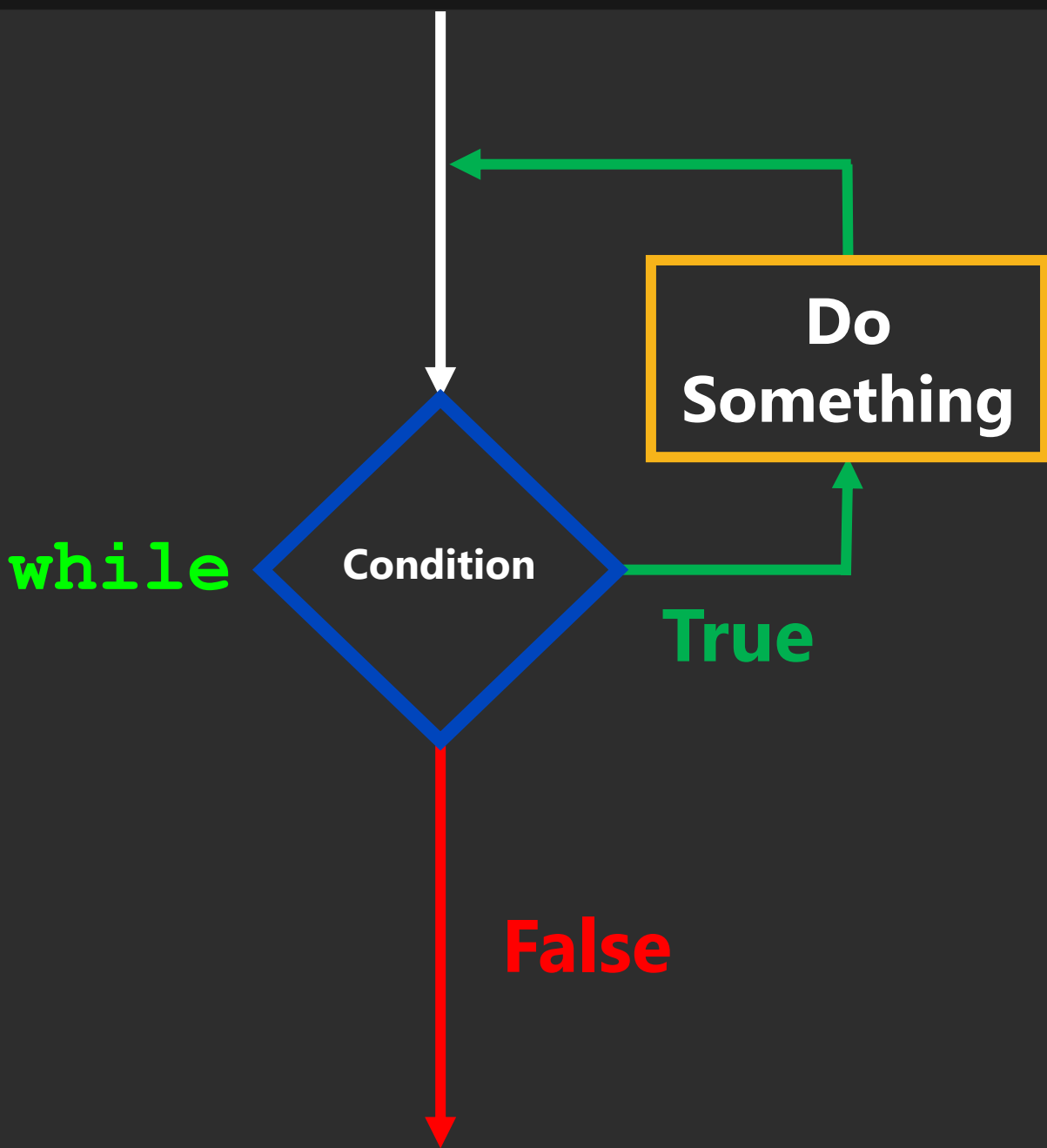
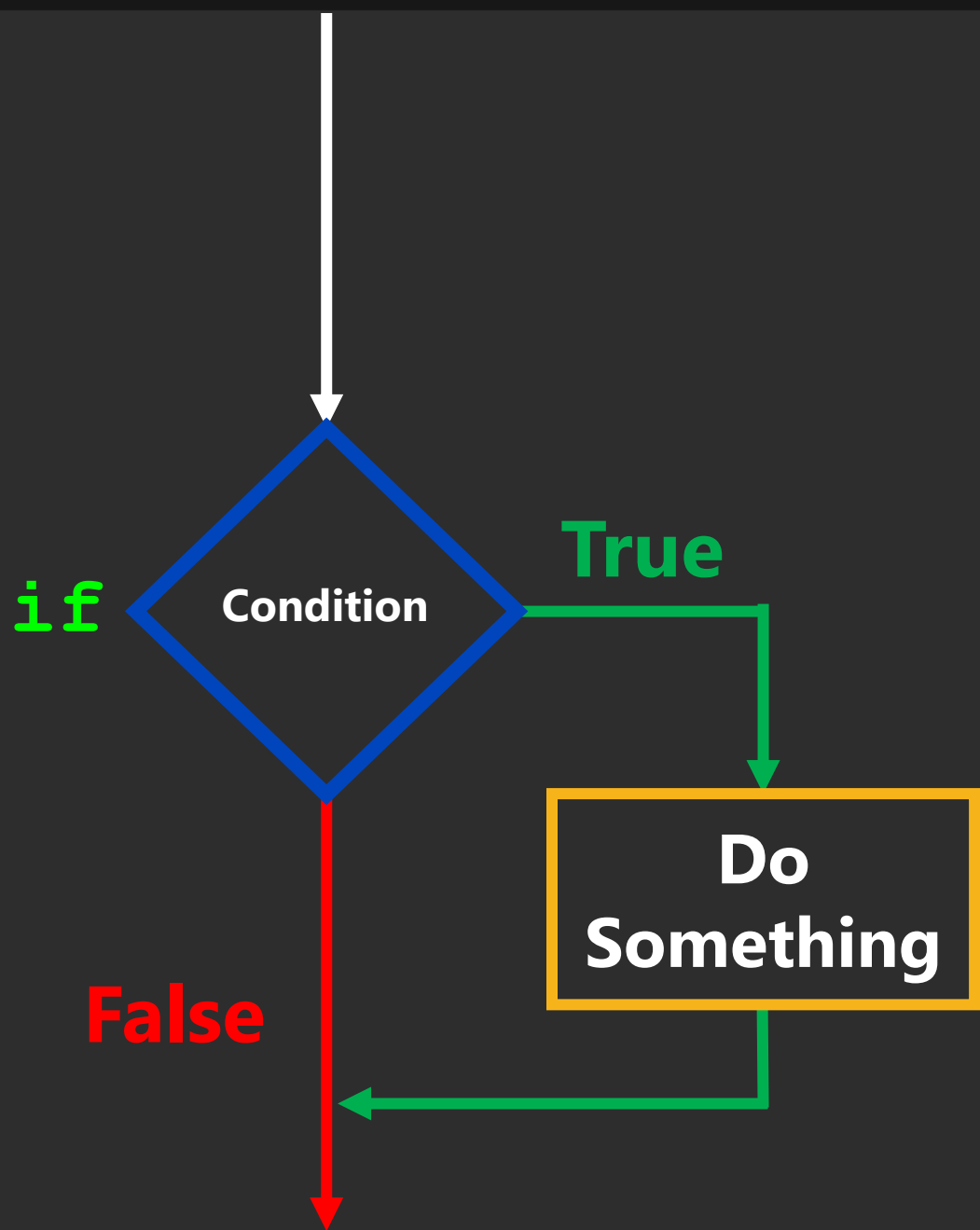
while expression:
do something.

Indent

While Loops

- The condition that gets evaluated is just a boolean expression.
- In particular it can include:
 - Something that evaluates to **True** or **False**.
 - logical operators (**and**, **or**, **not**)
 - comparison operators
 - function calls
- ... really anything that evaluates to **True** or **False**.





Refresher

- How many printouts will the following **while** loop produce?

```
x = 1
while x < 4:
    print(x)
    x = x + 1
```

**Open your
notebook**

Click Link:
1. Refresher

Refresher

- Just like for **if**-statements, if you use **and** or **or** in a while-loop expression, it is subject to lazy evaluation.
- Only if **x < 4** is **True** will **y < 4** be evaluated. **#solazy**

```
while x < 4 and y < 4:  
    ...
```

**Open your
notebook**

Click Link:

2. Lazy Evaluation

Turtles and **while** loops

- I'm a little turtle and I want to take steps to the right until I get to the brick wall.
- However, I don't know how far away the brick wall I.



**Open your
notebook**

Click Link:

**3. Turtles and while
loops**

Breakout Session 1

- Write some code to allow someone to play Rock, Paper, Scissors, Lizard, Spock repeatedly until they beat the computer `3` times.



**Open your
notebook**

Click Link:

4. Breakout Session 1

Random Module

- This module implements pseudo-random number generators for various distributions.

```
import random
```

```
random.uniform()
```

```
random.random()
```

```
random.randint()
```

```
...
```

**Open your
notebook**

Click Link:

5. Random Module

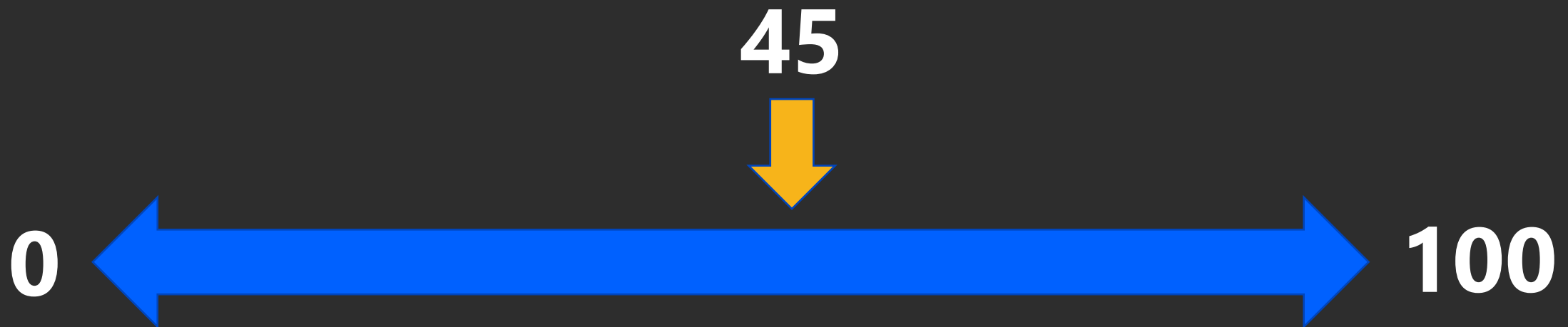
Guessing Game

- Let's build a simple guessing game.
 - Get the computer to choose a random integer from 0 to 100.
 - Ask the user for a guess and allow the user to input a guess or "q".
 - If the user inputs "q" print a nice message and end the program.
 - If the user enters a guess, tell them if they should guess higher, lower, or if they got it right.
 - If they got it right, print a nice message and quit.



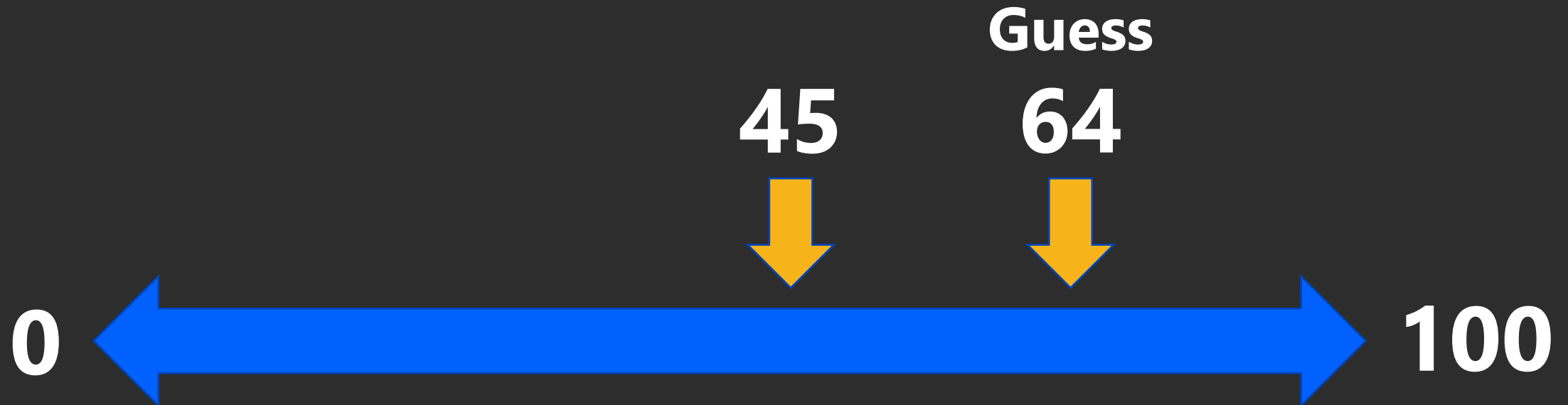
Guessing Game

- Get the computer to choose a random integer from 0 to 100.
 - The computer selects 45.

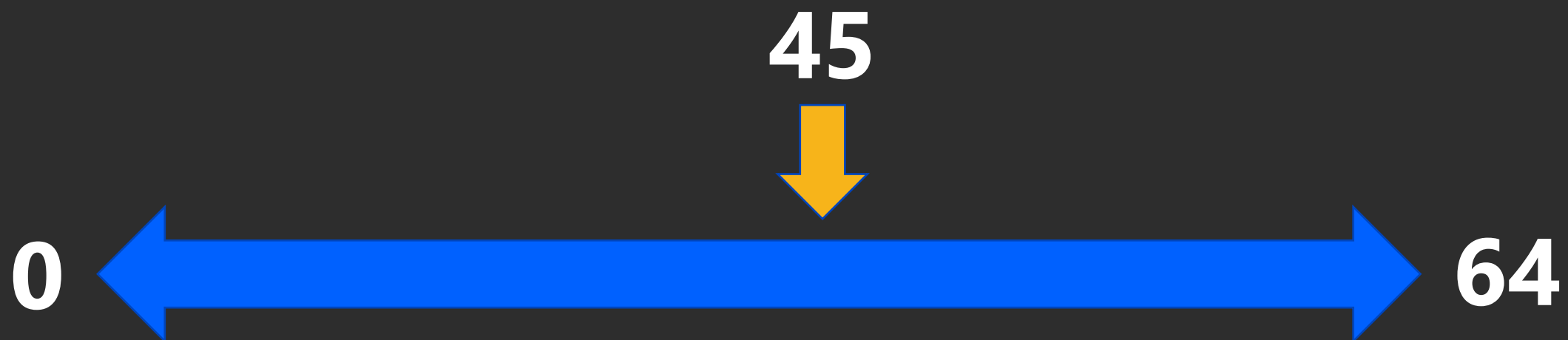


Guessing Game

- The user guesses 64.
 - The computer says **LOWER**.

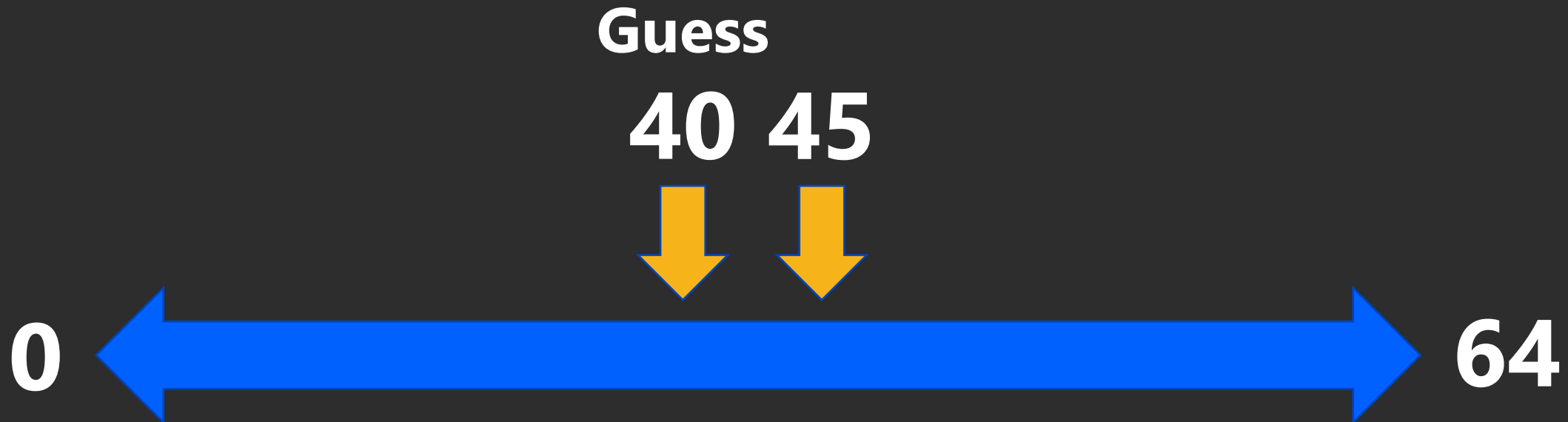


Guessing Game

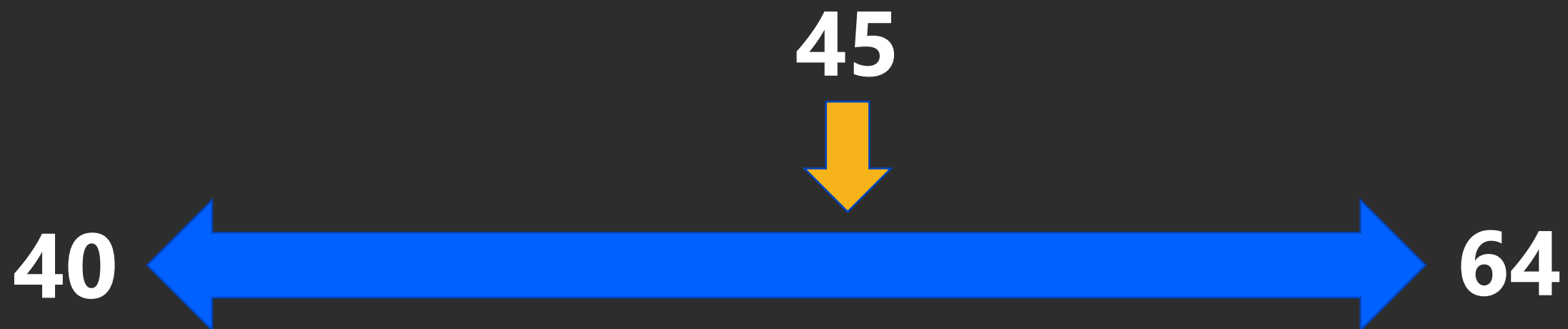


Guessing Game

- The user guesses 40.
 - The computer says **HIGHER**.

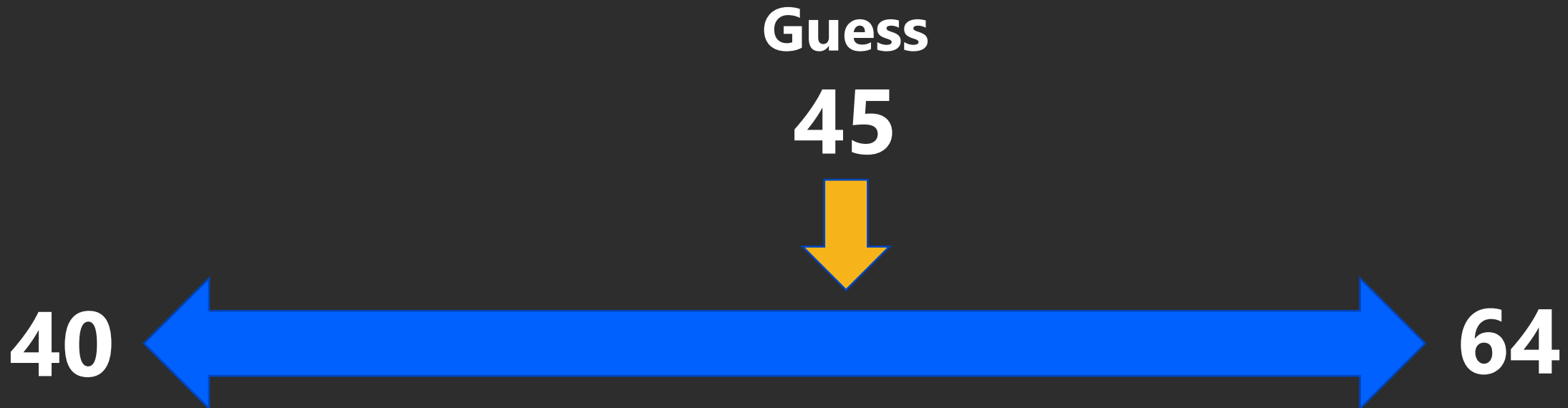


Guessing Game



Guessing Game

- The user guesses 45.
 - The computer says **YOU WIN.**



Guessing Game

- Let's build a simple guessing game.
 1. Get the computer to choose a random integer from 0 to 100.
 2. Ask the user for a guess and allow the user to input a guess or "q".
 3. If the user inputs "q" print a nice message and end the program.
 4. If the user enters a guess, tell them if they should guess higher, lower, or if they got it right.
 5. If they got it right, print a nice message and quit.

**Open your
notebook**

Click Link:

**6. A Simple Guessing
Game**

Lecture Recap

- Looping (aka iteration) is the second key control structure in programming (if-statements/branching was the first).
- The basic idea of loops is to repeatedly execute the same block code.
- Looping is a very powerful idea.
- While loops.

more while loops.

Week 4 | Lecture 2 (4.2)

if nothing else, write **#cleancode**.