Looping

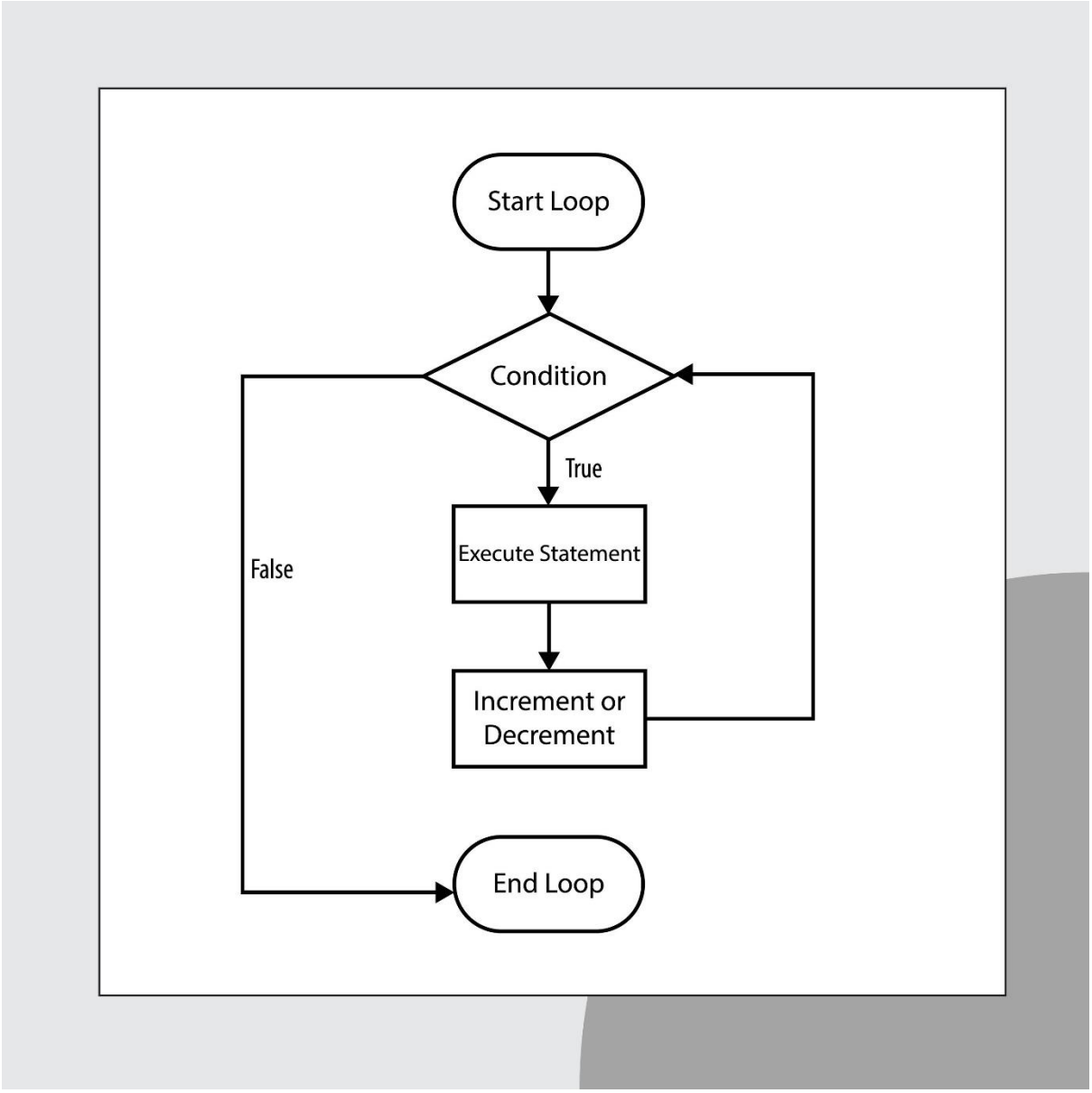
September 15, 2025

Looping: The Power of Repetition

Looping statements are used to execute a block of code repeatedly.

for Loop: Iteration over a Sequence

A **for loop** is used to iterate over the items of any sequence (a list, tuple, dictionary, string, or range). It's great when you know the number of times you need to loop.



- **Iterating over Different Data Types:**

```
# Looping through a list of items
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
print(fruit)


# Looping through characters in a string
for char in "Python":
print(char)
# Looping through a dictionary's keys and values
student = {"name": "Charlie", "age": 20, "major": "Computer Science"}
for key, value in student.items():
print(f"Key: {key}, Value: {value}")
```

- **Useful Functions with for Loops:**

```
range(start, stop, step): Generates a sequence of numbers.
for i in range(1, 10, 2):
   print(i, end=" ") # Output: 1 3 5 7 9


enumerate(iterable): Returns both the index and the value of each item.
for index, fruit in enumerate(fruits):
   print(f"Fruit at index {index} is {fruit}.")


zip(iterable1, iterable2, ...): Combines items from multiple iterables.
names = ["Alice", "Bob"]
ages = [25, 30]
for name, age in zip(names, ages):
   print(f"{name} is {age} years old.")
```

while Loop: Repetition Based on a Condition

A **while loop** repeats as long as its condition is True. This is useful when the number of iterations is unknown beforehand.

- **Example with a Counter:**

```
count = 0
while count < 5:
    print(f"Count is {count}")
    count += 1
```

- **Example with a Sentinel Value:**

```
user_input = ""
while user_input.lower() != "quit":
    user_input = input("Enter a word (or 'quit' to exit): ")
    print(f"You entered: {user_input}")
```

- **Loop Control Statements:**
    - **break:** Exits the loop immediately.

```
for number in range(10):
    if number == 5:
        break
    print(number) # Output: 0 1 2 3 4
```

    - **continue:** Skips the rest of the code in the current iteration and moves to the next one.

```
for number in range(5):
    if number == 2:
        continue
    print(number) # Output: 0 1 3 4
```

- **pass:** A null statement. It is used as a placeholder where a statement is syntactically required but you want no action to be performed.

```
for number in range(5):
    if number % 2 == 0:
        pass # This does nothing, just a placeholder
    else:
        print(number) # Output: 1 3
```