Do Now

bigd103.link/while-loop-penny

For Loops

Making Code Repeat with Boundaries


Friendship ended with
WHILE
Now
FOR
is my
best friend
imgflip.com

The `for` Loop - A Better Way to Count

Remember counting with `while`?

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

```
Count is 0
Count is 1
Count is 2
Count is 3
Count is 4
```

The `for` loop makes this much easier:

```
# With for loop
for count in range(5):
    print(f"Count is {count}")
```

```
Count is 0
Count is 1
Count is 2
Count is 3
Count is 4
```

Same result, less code!

Understanding `range()`

`range()` creates a sequence of numbers:

```
for num in range(5):
    print(num)

for num in range(1, 6):
    print(num)
```

```
0
1
2
3
4
1
2
3
4
5
```

Note: `range(n)` goes from 0 to n-1!

`range()` with Start and Stop

```
for i in range(3, 8):
    print(i)  # Prints: 3, 4, 5, 6, 7

for bill in range(10, 51, 10):
    total = bill * 1.20  # 20% tip
    print(f"Bill: ${bill}, Total: ${total:.2f}")
```

Loop Variable Names

The loop variable can be any name:

```python
# Common: i, j, k for simple counters
for i in range(3):
    print(f"Loop {i}")

# Descriptive names are better!
for student_number in range(1, 31):
    print(f"Welcome, Student {student_number}")

for day in range(1, 8):
    print(f"Day {day} of the week")
```

```
Loop 0
Loop 1
Loop 2
Welcome, Student 1
Welcome, Student 2
Welcome, Student 3
Welcome, Student 4
Welcome, Student 5
Welcome, Student 6
```

For Loops with Calculations

```
# Calculate squares
for num in range(1, 6):
    square = num ** 2
    print(f"{num} squared is {square}")

# Running total (accumulator pattern)
total = 0
for price in range(10, 31, 10):
    total = total + price
    print(f"Added ${price}, total is now ${total}")
```

```
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
Added $10, total is now $10
Added $20, total is now $30
Added $30, total is now $60
```

Loops with Conditionals

Combining loops with if statements:

```python
# Print only even numbers
for num in range(10):
    if num % 2 == 0:
        print(f"{num} is even")
    else:
        print(f"{num} is odd")

# Grade multiple students
for student in range(1, 6):
    score = 70 + student * 5  # Simulated scores
    if score >= 90:
        print(f"Student {student}: A")
    elif score >= 80:
        print(f"Student {student}: B")
    else:
        print(f"Student {student}: C")
```

```
0 is even
1 is odd
2 is even
3 is odd
```

Breaking Out of Loops

Use `break` to exit early:

```
# Password checker with limited attempts
for attempt in range(3):
    password = input("Enter password: ")
    if password == "secret":
        print("Access granted!")
        break
    else:
        print(f"Wrong! {2 - attempt} tries left")
```

```
# Find first number divisible by 7
for num in range(1, 100):
    if num % 7 == 0:
        print(f"Found it: {num}")
        break
```

```
Found it: 7
```

Common Loop Patterns

Counting

```
count = 0
for i in range(100):
    if i % 2 == 0:
        count = count + 1
print(f"Found {count} even numbers")
```

```
Found 50 even numbers
```

Accumulating

```
total = 0
for num in range(1, 11):
    total = total + num
print(f"Sum of 1-10: {total}")
```

```
Sum of 1-10: 55
```

Nested Loops

Just like conditions, we can nest loops inside loops:

```python
# Multiplication table
for i in range(1, 4):
    for j in range(1, 4):
        result = i * j
        print(f"{i} × {j} = {result}")
    print("")  # Empty line between sections
```

```
1 × 1 = 1
1 × 2 = 2
1 × 3 = 3

2 × 1 = 2
2 × 2 = 4
2 × 3 = 6

3 × 1 = 3
3 × 2 = 6
3 × 3 = 9
```

When to Use `while` vs `for`

Use `while` when:

- You don’t know how many times to loop
- You’re waiting for something to happen
- You need user input validation

Use `for` when:

- You know exactly how many times to loop
- You’re processing a sequence of items
- You’re counting up or down

Exercise: The Guessing Game

https://bigd103.link/guessing-game