In Python, the **modulus operator** (%) returns the **remainder** after dividing one number by another.

Basic Syntax

```
remainder = a % b
```

- a → dividend (number to be divided)
- b → divisor (number you divide by)
- Result → remainder after division

How It Works

Mathematically: a%b = a − b × floor (ba)

$$a\%b = a - b \times \text{floor}\left(\frac{a}{b}\right)$$

This means:

- Python divides a by b.
- It takes the **floor** (largest integer ≤ result).
- Multiplies that by b.
- Subtracts from a to get the remainder.

Examples

```
print(10 % 3)    # 1  → 10 = 3*3 + 1
print(14 % 5)    # 4  → 14 = 5*2 + 4
print(7 % 7)     # 0  → 7 = 7*1 + 0
```

With Negative Numbers

Python's % always returns a remainder **with the same sign as the divisor** (b):

```
print(10 % -3)   # -2 → 10 = (-3)*(-4) + (-2)
print(-10 % 3)   # 2  → -10 = 3*(-4) + 2
print(-10 % -3)  # -1 → -10 = (-3)*3 + (-1)
```

Common Uses

- **Check even/odd**:

```python
if num % 2 == 0:
    print("Even")
else:
    print("Odd")
```

- **Cycle through values**:

```python
colors = ["red", "green", "blue"]
for i in range(7):
    print(colors[i % len(colors)])  # repeats colors
```

- **Wrap-around indexing** (e.g., circular buffers, games).

✅ **Key takeaway**:

- % gives the remainder, **sign matches the divisor**.
- Works with integers and floats.
- Useful for periodic patterns, checks, and modular arithmetic.

If you want, I can also show you a **visual diagram** of how Python's modulus works with positive and negative numbers so it's easier to remember. Would you like me to prepare that?