Python Basic Arithmetic

We have learned variable types

`int` – whole numbers

```
a1 = 3
a2 = -100
```

`float` – numbers with a decimal point

```
print(3.1415)
print(-0.25)
```

Addition + and Subtraction −

Variable addition and subtraction:

```
a = 6
b = 7

print(a + b)   # 13
print(a - b)   # -1
```

Addition + and Subtraction −

Variable addition and subtraction:

```
a = 6
b = 7

print(a + b)   # 13
print(a - b)   # -1
```

You can also add or subtract directly:

```
print(6 + 7)   # 13
print(6 - 7)   # -1
```

Question + − Simple Math

What is the output?

```
x = 8
y = 5

print(x - y)
print(x + y - 3)
```

Question + − Simple Math

What is the output?

```
x = 8
y = 5

print(x - y)
print(x + y - 3)
```

Output:

```
3
10
```

Multiplication *

What is the output?

```
apples_per_bag = 4
bags = 3

total_apples = apples_per_bag * bags
print(total_apples)
```

Multiplication *

What is the output?

```
apples_per_bag = 4
bags = 3

total_apples = apples_per_bag * bags
print(total_apples)
```

Output:

```
12
```

Division `/` – True Division

In Python, `/` always gives a **float** (even if it looks like a whole number).

```
print(10 / 2)
print(7 / 2)
print(9 / 4)
```

Division `/` – True Division

In Python, `/` always gives a **float** (even if it looks like a whole number).

```
print(10 / 2)
print(7 / 2)
print(9 / 4)
```

Output:

```
5.0
3.5
2.25
```

Notice the **decimal point** in results.

Floor Division `//`

Divide, then keep the **whole number** part.
It drops the decimal part.

```
print(7 // 2)    # 3   (because 7 / 2 = 3.5)
print(11 // 3)   # 3   (because 11 / 3 ≈ 3.66666)
print(12 // 4)   # 3
```

Floor Division `//`

Divide, then keep the **whole number** part.
It drops the decimal part.

```
print(7 // 2)    # 3   (because 7 / 2 = 3.5)
print(11 // 3)   # 3   (because 11 / 3 ≈ 3.66666)
print(12 // 4)   # 3
```

Output:

```
3
3
3
```

Question ➗ True Division and Floor Division

What is the output?

```
print(8 / 2)
print(9 / 2)
print(9 // 2)
```

Question ➗ True Division and Floor Division

What is the output?

```
print(8 / 2)
print(9 / 2)
print(9 // 2)
```

Output:

```
4.0
4.5
4
```

- True division `/` always gives a float, even if the decimal part is zero.
- Floor division `//` keeps only the whole number part and drops the decimal part.

Remainder `%` (Modulo)

`%` gives the **remainder** after division.

```
print(7 % 2)   # 1   ( 7 ÷ 2 = 3 remainder 1)
print(10 % 3)  # 1   (10 ÷ 3 = 3 remainder 1)
print(12 % 4)  # 0   (12 ÷ 4 = 3 remainder 0)
```

Remainder `%` (Modulo)

`%` gives the **remainder** after division.

```
print(7 % 2)   # 1   ( 7 ÷ 2 = 3 remainder 1)
print(10 % 3)  # 1   (10 ÷ 3 = 3 remainder 1)
print(12 % 4)  # 0   (12 ÷ 4 = 3 remainder 0)
```

Useful to check if a number is **even** or **odd**:

```
print(8 % 2)   # 0 → even
print(9 % 2)   # 1 → odd
```

Question 🍬 Sharing Candy

Question:

- You have `17` candies and `5` friends.
- Give each friend the **same number** of candies.

Write a Python program to find:

1. How many candies does each friend get?
2. How many candies are left over?

Question 🍬 Sharing Candy

Quotient and remainder of `17 ÷ 5` are the answers!

Python code:

```
candies = 17
friends = 5

print(candies // friends) # quotient is 3
print(candies % friends)  # remainder is 2
```

Question 🍬 Sharing Candy

Quotient and remainder of `17 ÷ 5` are the answers!

Python code:

```
candies = 17
friends = 5

print(candies // friends) # quotient is 3
print(candies % friends)  # remainder is 2
```

Output:

```
3
2
```

Operator Precedence (Order)

When an expression has many operators, Python uses a **priority order**:

1. Parentheses `( )`
2. Multiplication and division: `*`, `/`, `//`, `%`
3. Addition and subtraction: `+`, `-`

Operator Precedence (Order)

When an expression has many operators, Python uses a **priority order**:

1. Parentheses `( )`
2. Multiplication and division: `*`, `/`, `//`, `%`
3. Addition and subtraction: `+`, `-`

Example:

```
print(2 + 3 * 4)        # = 2 + 12 = 14
print((2 + 3) * 4)      # = 5 * 4 = 20
```

- First line: `3 * 4` happens before `+`
- Second line: parentheses force `2 + 3` to go first.

Question 🧠 Precedence Practice

What is the output?

```
print(2 + 6 // 3)
print((2 + 6) // 3)
```

Use the order:

1. Parentheses `( )`
2. Floor division `//`
3. Addition `+`

Question 🧠 Precedence Practice

What is the output?

```
print(2 + 6 // 3)
print((2 + 6) // 3)
```

Answer:

- 2 + 6 // 3 = 2 + 2 = 4
- (2 + 6) // 3 = 8 // 3 = 2 (`//` keeps the whole number and drops the decimal)

Mix `int` and `float`

When you mix `int` and `float`, Python gives a `float`.

```
print(3 + 2.5)      # 5.5
print(6 * 0.5)      # 3.0
print(7 / 2)        # 3.5
```

Mix `int` and `float`

When you mix `int` and `float`, Python gives a `float`.

```
print(3 + 2.5)      # 5.5
print(6 * 0.5)      # 3.0
print(7 / 2)        # 3.5
```

You can check the type using `type()`:

```
x = 3 + 2.5
print(type(x))      # <class 'float'>
```

Type Conversion: `int()` and `float()`

You can **convert** between `int` and `float`.

`int(x)` → keeps only the whole number part

```
a = 3.9
b = int(a)      # 3  (decimal part is dropped)
```

Type Conversion: `int()` and `float()`

You can **convert** between `int` and `float`.

`int(x)` → keeps only the whole number part

```
a = 3.9
b = int(a)      # 3  (decimal part is dropped)
```

`float(x)` → adds a decimal point

```
c = 5
d = float(c)    # 5.0
```

Question 📊 Average Score

We have 3 test scores: `7`, `8`, and `9`.

```
total = 7 + 8 + 9
average1 = total / 3
average2 = int(average1)

print(average1)
print(average2)
```

What is the output?

Question 📊 Average Score

We have 3 test scores: `7`, `8`, and `9`.

```
total = 7 + 8 + 9
average1 = total / 3
average2 = int(average1)

print(average1)
print(average2)
```

Output:

```
8.0
8
```

Summary

- Arithmetic operations:
  - `+` addition, `-` subtraction, `*` multiplication
  - `/` true division (always gives `float`)
  - `//` floor division (whole number part only)
  - `%` remainder after division
- **Precedence**: first `()`, then `*`, `/`, `//`, `%`, then `+`, `-`
- **Type conversion**:
  - `int(x)` → to whole number
  - `float(x)` → add a decimal point