Tasks – Arithmetic Operators and Order of Operations

Practice Python arithmetic operators (`+`, `-`, `*`, `/`, `//`, `%`) and order of operations (סדר פעולות חשבון). Create each file, run it, and check the output.

Run scripts with: `python3 script_name.py`

---

1. Addition and subtraction

Task 1.1 – Add and subtract (`ops_add_sub.py`)

- Create `ops_add_sub.py`.
- Print the result of `20 + 7`.
- Print the result of `20 - 7`.
- Print the result of `-5 + 3`.

Expected output:

```
27
13
-2
```

Task 1.2 – Multiplication (`ops_mul.py`)

- Create `ops_mul.py`.
- Print the result of `6 * 7`.
- Print the result of `-4 * 5`.

Expected output:

```
42
-20
```

Task 1.3 – All three: +, -, * (`ops_plus_minus_mul.py`)

- Create `ops_plus_minus_mul.py`.
- In one script, print one result for addition, one for subtraction, and one for multiplication (use any numbers you like).

Expected output (example):

```
15
5
50
```

---

2. Division: / and //

Task 2.1 – Regular division and integer division (`ops_div.py`)

- Create `ops_div.py`.
- Print `17 / 5` (regular division – gives a float).
- Print `17 // 5` (integer division – quotient, no remainder).

Expected output:

```
3.4
3
```

Task 2.2 – More / and // (`ops_div_more.py`)

- Create `ops_div_more.py`.
- Print `10 / 3` and `10 // 3`.
- Print `-10 // 3` (note: result is rounded toward minus infinity in Python).

Expected output:

```
3.3333333333333335
3
-4
```

Task 2.3 – Remainder with % (`ops_modulo.py`)

- Create `ops_modulo.py`.
- Print `17 % 5` (remainder of 17 ÷ 5).
- Print `10 % 3` and `10 % 2` (even/odd idea: 10%2 is 0).

Expected output:

```
2
1
0
```

Task 2.4 – Use % for “remainder” (`ops_modulo_use.py`)

- Create `ops_modulo_use.py`.
- Imagine 73 seconds: how many full minutes and how many leftover seconds? Use integer division for minutes and `%` for the remainder. Print both (e.g. “1 minute, 13 seconds” or just two numbers).

Expected output (example):

```
1
13
```

---

3. Order of operations (סדר פעולות חשבון)

Python follows the usual math order: parentheses first, then `*` `/` `//` `%`, then `+` `-`. Same priority goes left to right.

Task 3.1 – Parentheses change the result (`order_parens.py`)

- Create `order_parens.py`.
- Print `2 + 3 * 4` (multiplication first: 3*4=12, then 2+12=14).
- Print `(2 + 3) * 4` (parentheses first: 2+3=5, then 5*4=20).

Expected output:

```
14
20
```

Task 3.2 – Without vs with parentheses (`order_parens2.py`)

- Create `order_parens2.py`.
- Print `10 - 2 * 3` (multiplication first).
- Print `(10 - 2) * 3`.

Expected output:

```
4
24
```

Task 3.3 – Same priority, left to right (`order_left_right.py`)

- Create `order_left_right.py`.
- For `*` and `/`, same priority means left to right. Print `24 / 4 * 2` (first 24/4=6, then 6*2=12).
- Print `24 / (4 * 2)` to see how parentheses change it.

Expected output:

```
12.0
3.0
```

Task 3.4 – One expression using order of operations (`order_mixed.py`)

- Create `order_mixed.py`.
- Write one expression that uses `+`, `-`, and `*` (e.g. `5 + 3 * 2 - 1`). Print the result. Then print the same calculation using parentheses to force a different order (e.g. `(5 + 3) * (2 - 1)`). Show that the two results differ.

Expected output (example):

```
10
8
```

Done

You've used: `+`, `-`, `*`, `/`, `//`, `%` and practiced order of operations (parentheses first, then `*` / `/` / `//` / `%`, then `+` / `-`, left to right for same priority).