# Tasks – input() and Using What You’ve Learned

Start with basic `input()`, then combine it with variables, types, operators, and print formatting. Create each file and run it; for tasks that use input, type something when the script waits.

Run scripts with: `python3 script_name.py`

---

## Part 1 – Basics

### Task 1.1 – Read and print (`input_basic.py`)

- Create `input_basic.py`.
- Use `name = input()` (no prompt). Run the script; it will wait. Type your name and press Enter.
- Print `name`.

**Expected output (example – if you type Alice):**

```
Alice
```

---

### Task 1.2 – input() with a prompt (`input_prompt.py`)

- Create `input_prompt.py`.
- Use `city = input("Enter your city: ")`. The string inside `input()` is shown before the program waits.
- Print the value of `city` (e.g. with a short message like `"City:"` and the value, using `sep` if you like).

**Expected output (example – if you type Tel Aviv):**

```
Enter your city: Tel Aviv
City: Tel Aviv
```

---

### Task 1.3 – Two inputs ( `input_two.py` )

- Create `input_two.py` .
- Ask for first name and last name (two `input()` calls, with prompts). Store them in two variables.
- Print both in one line (e.g. "Hello, First Last" or use `sep` ).

**Expected output (example):**

```
First name: John
Last name: Doe
Hello, John Doe
```

---

## Part 2 – input() and types

### Task 2.1 – input() returns a string ( `input_string.py` )

- Create `input_string.py` .
- Use `x = input("Enter a number: ")` and then `print(x + x)` . Run it and enter e.g. `5` . You should see `55` (string concatenation), not `10` .

**Expected output (if you type 5):**

```
Enter a number: 5
55
```

---

### Task 2.2 – Convert to int ( `input_int.py` )

- Create `input_int.py` .

- Use `s = input("Enter a number: ")` then convert: `n = int(s)`. Print `n + 10`. If the user types `7`, output should be `17`.

**Expected output (if you type 7):**

```
Enter a number: 7
17
```

---

### Task 2.3 – Convert to float ( `input_float.py` )

- Create `input_float.py`.
- Read one number with `input()`, convert it with `float()`, store in a variable. Print that variable multiplied by 2.

**Expected output (example – if you type 3.5):**

```
Enter a number: 3.5
7.0
```

---

## Part 3 – input() with variables and operators

### Task 3.1 – Two numbers, then add ( `input_add.py` )

- Create `input_add.py`.
- Ask for two numbers (two `input()` calls). Convert both to `int` (or `float`). Print their sum.

**Expected output (example – if you type 10 and 3):**

```
First number: 10
Second number: 3
13
```

---

### Task 3.2 – One number, use with operators ( `input_ops.py` )

- Create `input_ops.py`.
- Read one integer. Print that number, then the same number plus 5, then the same number times 2. Use one variable and reuse it in expressions (or use shortcuts if you prefer).

**Expected output (example – if you type 4):**

```
4
9
8
```

---

### Task 3.3 – Simple “calculator” (two numbers) (`input_calc.py`)

- Create `input_calc.py`.
- Read two numbers (int or float). Print their sum, difference, product, and (if you want) quotient. Use variables for the two numbers and for the results, then print.

**Expected output (example – 12 and 4):**

```
First: 12
Second: 4
16
8
48
3.0
```

---

## Part 4 – input() with print formatting and comments

### Task 4.1 – Label and value with sep (`input_label.py`)

- Create `input_label.py`.
- Ask for a value (e.g. "Enter your age: "). Store it. Print a label and the value on one line using `print(label, value, sep=": ")`.

**Expected output (example):**

```
Enter your age: 25
Age: 25
```

---

### Task 4.2 – Multiple values on one line with sep ( `input_multi_print.py` )

- Create `input_multi_print.py`.
- Ask for name and age. Print one line like "Name: Alice, Age: 20" using one or two `print()` calls and `sep` (and maybe `end=""` if you use two prints for one line).

**Expected output (example):**

```
Name: Alice
Age: 20
Name: Alice, Age: 20
```

---

### Task 4.3 – Comments and input ( `input_commented.py` )

- Create `input_commented.py`.
- Write a short script that: has a comment at the top explaining what it does, asks for one number with a prompt, converts to int, and prints the number plus 1. Use a variable for the converted number.

**Expected output (example – input 6):**

```
Enter a number: 6
7
```

---

## Part 5 – Short practical tasks

### Task 5.1 – “Total” from price and quantity ( `input_total.py` )

- Create `input_total.py`.

- Ask for price (float) and quantity (int). Compute total = price * quantity and print it (e.g. "Total: 14.5").

**Expected output (example – 2.5 and 4):**

```
Price: 2.5
Quantity: 4
Total: 10.0
```

---

## Task 5.2 – Remainder (modulo) from input (`input_remainder.py`)

- Create `input_remainder.py`.
- Ask for two integers: dividend and divisor. Print the remainder (use `%`). Example: 17 and 5 → 2.

**Expected output (example):**

```
Dividend: 17
Divisor: 5
Remainder: 2
```

---

## Task 5.3 – Greeting with name and number (`input_greeting.py`)

- Create `input_greeting.py`.
- Ask for name (string) and a number. Print a greeting that includes both (e.g. "Hello, Alice! Your number is 42."). Use variables and one or more `print()` calls with `sep` / `end` as needed.

**Expected output (example):**

```
Your name: Alice
Your number: 42
Hello, Alice! Your number is 42.
```

---

## Done

You’ve used: `input()` with and without a prompt, storing input in variables, converting with `int()` and `float()`, using input in expressions and with operators, and combining input with print, `sep`, `end`, variables, and comments.