Tasks – String Operations: Concatenation, Repetition, and str()

Practice joining strings with `+` (concatenation), repeating strings with `*` (repetition), and building strings from other values with `str()`. Create each file, run it, and check the output.

Run scripts with: `python3 script_name.py`

Part 1 – Concatenation (+)

Task 1.1 – Two strings with + (`str_concat_basic.py`)

- Create `str_concat_basic.py`.
- Assign `"Hello"` to one variable and `"World"` to another. Concatenate them with `+` and print the result (one word or with a space: `"Hello " + "World"` or add `" "` between).

Expected output (example):

```
Hello World
```

Task 1.2 – Concatenate three parts (`str_concat_three.py`)

- Create `str_concat_three.py`.
- Build one string from three parts (e.g. first name, space, last name) using `+`. Print the full string.

Expected output (example):

```
John Doe
```

Task 1.3 – Concatenate and store (`str_concat_store.py`)

- Create `str_concat_store.py` .
- Assign `"Py"` and `"thon"` to two variables. Concatenate them and store the result in a third variable. Print that variable.

Expected output:

```
Python
```

Task 1.4 – += with strings (`str_concat_plus_eq.py`)

- Create `str_concat_plus_eq.py` .
- Assign `"Hello"` to a variable. Use `+= "!"` then `+= " World"` . Print the variable after each step (or only at the end).

Expected output (if you print at the end):

```
Hello! World
```

Part 2 – Repetition (*)

Task 2.1 – String times number (`str_repeat_basic.py`)

- Create `str_repeat_basic.py` .
- Print the result of `"a" * 5` (string repeated 5 times).

Expected output:

```
aaaaa
```

Task 2.2 – Variable and repetition (`str_repeat_var.py`)

- Create `str_repeat_var.py`.
- Assign a short string (e.g. `"-"`) to a variable. Print that variable repeated 10 times (e.g. `s * 10`). Use it to make a simple “line” of dashes.

Expected output:

```
----------
```

Task 2.3 – Repetition then concatenate (`str_repeat_concat.py`)

- Create `str_repeat_concat.py`.
- Build one string: repeat `"="` three times, then concatenate a word (e.g. `" Hi "`), then repeat `"="` three times. Print the result (e.g. `=== Hi ===`).

Expected output (example):

```
=== Hi ===
```

Task 2.4 – Number * string (`str_repeat_number.py`)

- Create `str_repeat_number.py`.
- Use a variable for the number (e.g. `n = 4`) and one for the character (e.g. `c = "*"`). Print `c * n` to get `****`.

Expected output (example):

```
****
```

Part 3 – str()

Task 3.1 – int to string for concatenation (`str_from_int.py`)

- Create `str_from_int.py`.

- Assign the integer `42` to a variable. Use `str()` to convert it and concatenate with a label (e.g. `"Answer: " + str(42)`). Print the result.

Expected output:

```
Answer: 42
```

Task 3.2 – float to string (`str_from_float.py`)

- Create `str_from_float.py`.
- Assign `3.14` to a variable. Build a string like `"Pi is " + str(3.14)` and print it.

Expected output:

```
Pi is 3.14
```

Task 3.3 – Variable number in a message (`str_var_message.py`)

- Create `str_var_message.py`.
- Assign a number (int or float) to a variable. Build one string that includes that number in a sentence using `str()`. Print the string (e.g. "You have 5 items.").

Expected output (example):

```
You have 5 items.
```

Task 3.4 – Concatenation + str() + repetition (`str_combined.py`)

- Create `str_combined.py`.
- Build a string that: starts with a repeated character (e.g. `"-" * 3`), then a label and a number using `str()` (e.g. `" Count: " + str(10)`), then the same repeated character again. Print one line like `--- Count: 10 ---`.

Expected output (example):

```
--- Count: 10 ---
```

Done

You've used: **concatenation** (`+` , `+=`), **repetition** (`string * number`), and **str()** to build strings from numbers and combine them with other text.