Tasks – break and continue in for Loops

Practice `break` (exit the loop early) and `continue` (skip to the next iteration) inside `for` loops. Create each file, run it, and check the output.

Run scripts with: `python3 script_name.py`

Part 1 – break

Task 1.1 – Break when value found (`for_break_found.py`)

- Create `for_break_found.py`.
- Use `for i in range(1, 11):` and inside the loop: if `i == 5`, print `"Found 5"` and `break`. Otherwise print `i`. Run the script. You should see 1, 2, 3, 4, then "Found 5", and the loop stops (no 6–10).

Expected output:

```
1
2
3
4
Found 5
```

Task 1.2 – Break on input (stop at 0) (`for_break_zero.py`)

- Create `for_break_zero.py`.
- Use a loop that runs at most 10 times: `for i in range(10):`. Inside, read a number with `input()`. If the number is 0, print `"Stopped"` and `break`. Otherwise print the number. So the loop can end early when the user types 0.

Expected output (example – user types 3, 7, 0):

```
Enter a number (0 to stop): 3
3
Enter a number (0 to stop): 7
7
Enter a number (0 to stop): 0
Stopped
```

Task 1.3 – Sum until 0 (break in for) (`for_break_sum.py`)

- Create `for_break_sum.py` .
- Use `total = 0` and `for i in range(100):` (large upper bound). Inside, read a number. If it is 0, `break` . Else add it to total. After the loop print `"Sum: " + str(total)` . Test with e.g. 10, 20, 30, 0.

Expected output (example):

```
Number (0 to finish): 10
Number (0 to finish): 20
Number (0 to finish): 30
Number (0 to finish): 0
Sum: 60
```

Task 1.4 – First number greater than 10 (`for_break_gt10.py`)

- Create `for_break_gt10.py` .
- Read how many numbers to ask for (`n`). Use `for i in range(n):` and read a number each time. If the number is greater than 10, print `"First > 10: " + str(number)` and `break` . If the loop finishes without breaking, print `"None > 10"` . Use a variable for the current number.

Expected output (example – 4 numbers: 3, 7, 15, 2):

```
How many? 4
Number 1: 3
Number 2: 7
Number 3: 15
First > 10: 15
```

Part 2 – continue

Task 2.1 – Skip odd numbers (`for_continue_evens.py`)

- Create `for_continue_evens.py` .
- Use `for i in range(1, 11):` . Inside, if `i % 2 != 0` (odd), use `continue` . Otherwise print `i` . So you only print 2, 4, 6, 8, 10.

Expected output:

```
2
4
6
8
10
```

Task 2.2 – Skip multiples of 3 (`for_continue_skip3.py`)

- Create `for_continue_skip3.py` .
- Use `for i in range(1, 11):` . If `i % 3 == 0` , `continue` . Else print `i` . You should see 1, 2, 4, 5, 7, 8, 10 (3, 6, 9 skipped).

Expected output:

```
1
2
4
5
7
8
10
```

Task 2.3 – Skip negative input (`for_continue_positive.py`)

- Create `for_continue_positive.py` .

- Use `for i in range(5):` and read a number each time. If the number is negative, print `"Skipped"` and `continue`. Otherwise add it to a total and print the total so far. After the loop print the final total. Use variables for the number and total.

Expected output (example – 3, -1, 5, -2, 4):

```
Number 1: 3
Total so far: 3
Number 2: -1
Skipped
Number 3: 5
Total so far: 8
Number 4: -2
Skipped
Number 5: 4
Total so far: 12
Final total: 12
```

Part 3 – break and continue together

Task 3.1 – Stop at "quit", skip empty (`for_break_continue_words.py`)

- Create `for_break_continue_words.py`.
- Use a loop that runs at most 5 times: `for i in range(5):`. Read a word. If the word is `"quit"`, print `"Bye"` and `break`. If the word is empty (e.g. `word == ""`), print `"Skipped"` and `continue`. Otherwise print `"You said: " + word`. Use a variable for the word.

Expected output (example – hello, "", bye, quit):

```
Word: hello
You said: hello
Word:
Skipped
Word: bye
You said: bye
Word: quit
Bye
```

Task 3.2 – Find first valid score (`for_break_continue_valid.py`)

- Create `for_break_continue_valid.py` .
- Use `for i in range(5):` and ask for a score (0–100). If the score is outside 0–100, print `"Invalid, try again"` and `continue` . If valid, print `"Valid: " + str(score)` and `break` . So you keep asking until a valid score is entered (or 5 attempts). Use variables for the score and for the converted int.

Expected output (example – 150, -5, 85):

```
Score (0-100): 150
Invalid, try again
Score (0-100): -5
Invalid, try again
Score (0-100): 85
Valid: 85
```

Task 3.3 – Sum positive numbers, stop at negative (`for_break_continue_sum.py`)

- Create `for_break_continue_sum.py` .
- Use `total = 0` and a loop that runs at most 10 times. Read a number each time. If it is negative, print `"Stopped (negative)"` and `break` . If it is 0, `continue` (don't add, just skip). If positive, add to total. After the loop print `"Sum: " + str(total)` . Test with 5, 0, 3, -1.

Expected output (example):

```
Number: 5
Number: 0
Number: 3
Number: -1
Stopped (negative)
Sum: 8
```

Done

You've used: **break** to exit a for loop early (on a condition or input), **continue** to skip the rest of the iteration, and both in the same loop with input and conditions.