Tasks – while Loops

Practice `while` loops: condition, counter, accumulation, and input inside a loop. Create each file, run it, and check the output.

Run scripts with: `python3 script_name.py`

Task 1 – Simple while with counter (`while_count.py`)

- Create `while_count.py` .
- Assign `0` to a variable `count` . While `count < 5` , print `count` , then add 1 to `count` (e.g. `count += 1`). Run the script.

Expected output:

```
0
1
2
3
4
```

Task 2 – Print 1 to N (`while_one_to_n.py`)

- Create `while_one_to_n.py` .
- Ask for a positive integer `n` with `input()` and convert to `int` . Use a variable `i = 1` . While `i <= n` , print `i` , then do `i += 1` . Run and enter e.g. 4.

Expected output (if you enter 4):

```
Enter n: 4
1
2
```

```
3
4
```

Task 3 – Sum 1 to N (`while_sum.py`)

- Create `while_sum.py` .
- Read an integer `n` (e.g. 5). Use a loop: variable `i = 1` , variable `total = 0` . While `i <= n` , add `i` to `total` , then `i += 1` . After the loop, print `total` (should be 1+2+3+4+5 = 15 for n=5).

Expected output (n = 5):

```
Enter n: 5
15
```

Task 4 – Countdown (`while_countdown.py`)

- Create `while_countdown.py` .
- Assign `5` to a variable. While it is greater than 0, print the variable, then subtract 1. After the loop print `"Done"` (or "Go!").

Expected output:

```
5
4
3
2
1
Done
```

Task 5 – Input until valid (`while_input_valid.py`)

- Create `while_input_valid.py` .

- Ask for a number between 1 and 10. If the user enters something outside that range, ask again (use a `while` loop). When they enter a valid number, print it and exit the loop. Use a variable to store the number and another or the same for “valid” (e.g. keep looping while not valid).

Expected output (example – user types 0, then 15, then 7):

```
Enter a number (1-10): 0
Enter a number (1-10): 15
Enter a number (1-10): 7
You chose 7
```

Task 6 – Shopping total (while + input + variables + str) (`while_shopping.py`)

- Create `while_shopping.py`.
- In a loop: ask for item name (e.g. "Item name (or 'done' to finish):"). If the user types `"done"`, exit the loop. Otherwise ask for price (float), add the price to a running total, and print a line like "Added [name]: [price]. Running total: [total]" using variables and `str()` so the numbers show correctly. After the loop print "Total: " + str(total). Combines: while, input(), float(), if and break, variables, string concatenation with str().

Expected output (example – bread 2.5, milk 1.2, done):

```
Item name (or 'done' to finish): bread
Price: 2.5
Added bread: 2.5. Running total: 2.5
Item name (or 'done' to finish): milk
Price: 1.2
Added milk: 1.2. Running total: 3.7
Item name (or 'done' to finish): done
Total: 3.7
```

Task 7 – Simple calculator menu (while + input + if/elif/else + operators) (`while_calc_menu.py`)

- Create `while_calc_menu.py`.
- In a loop: print a menu "1=Add 2=Subtract 3=Multiply 4=Quit" and read the user's choice (int). If 4, break. If 1: read two numbers, print their sum. If 2: read two numbers, print their difference. If 3: read two numbers, print their product. Else print "Invalid option". Use variables for choice and for the two numbers. Combines: while, input(), int(), if/elif/else, arithmetic operators.

Expected output (example – 1, then 10 and 3, then 4):

```
1=Add 2=Subtract 3=Multiply 4=Quit
Choice: 1
First number: 10
Second number: 3
13
1=Add 2=Subtract 3=Multiply 4=Quit
Choice: 4
Bye
```

Task 8 – Password retry (while + input + conditions + break) (`while_password.py`)

- Create `while_password.py`.
- Set a secret password in a variable (e.g. `"python"`). Use a variable `attempts = 0`. While `attempts < 3`: ask for the password with input(). If it matches the secret, print "Welcome" and break. Else print "Wrong. Attempts left: X" (use a variable or expression for X, e.g. `2 - attempts` before you add 1), then do `attempts += 1`. After the loop, if the user never succeeded (e.g. check `attempts == 3`), print "Locked". Combines: while, input(), comparison (==), if/else, break, variables, +=.

Expected output (example – wrong twice, then correct):

```
Password: abc
Wrong. Attempts left: 2
Password: xyz
Wrong. Attempts left: 1
Password: python
Welcome
```

Task 9 – Number stats: count, sum, average (while + input + break + conditions) (`while_stats.py`)

- Create `while_stats.py` .
- Use variables `total = 0` and `count = 0` . In a loop, read a number (int). If it is 0, break. Otherwise add it to `total` , add 1 to `count` , and continue. After the loop print "Count: X, Sum: Y" using the variables. If `count > 0` , compute average as `total / count` (use float) and print "Average: Z". If count is 0, print "No numbers entered". Combines: while, input(), int(), break, accumulation, if/else, division, print with variables and str().

Expected output (example – 10, 20, 30, 0):

```
Enter a number (0 to finish): 10
Enter a number (0 to finish): 20
Enter a number (0 to finish): 30
Enter a number (0 to finish): 0
Count: 3, Sum: 60
Average: 20.0
```

Task 10 – Student name and average of N scores (while + input + variables + str) (`while_student_avg.py`)

- Create `while_student_avg.py` .
- Ask for the student's name (input, store in a variable). Ask "How many scores?" and read an integer `n` . Use a counter `i = 0` and `total = 0` . While `i < n` : ask for "Score [i+1]:", read the score (int or float), add it to total, then `i += 1` . After the loop compute `average = total / n` . Print one line like "Student: [name], Average: [average]" using string concatenation and `str()` (e.g. `"Student: " + name + ", Average: " + str(round(average, 1))`). Combines: input(), variables, while with counter, int()/float(), accumulation, division, str() and concatenation.

Expected output (example – name "Ali", 3 scores 80, 90, 70):

```
Student name: Ali
How many scores? 3
Score 1: 80
```

```
Score 2: 90
Score 3: 70
Student: Ali, Average: 80.0
```

Done

You've used: `while` with counter, input, break, and accumulation; combined with variables, if/elif/else, operators, str() and string concatenation, and conditions for real small programs (shopping total, menu calculator, password retry, stats, student average).