# 🖥️ Computer Science Worksheet: While Loops (Workbook)

## Primer: Understanding While Loops

A **while loop** repeats a block of code **as long as a condition is true**.  
Think of it as:

“While this condition is true, keep doing these steps.”

### Parts of a While Loop

1. **Initialization** – set a starting value.
2. **Condition Check** – the rule that decides whether to continue.
3. **Body** – the instructions that repeat.
4. **Update Step** – changes a variable so the loop can end.

**Example: Counting from 1 to 5**

counter = 1  
while counter <= 5:  
 print(counter) # body  
 counter = counter + 1 # update

⚠️ If you forget the update step, the loop never stops (infinite loop).

## How to Use a Trace Table

A **trace table** helps predict and check variable values each time through a loop.

Steps:

* Write each **iteration number** in the first column.
* Record the values of important variables (before/after updates).
* Fill in row by row as if you are the computer.

👉 This is a **dry run** — very useful for spotting mistakes.

# Core Exercises (Everyone Completes These)

### Exercise 1 – Counting Up

**Goal:** Learn basic initialization, condition, and update.

**Task:** Write a program that prints numbers 1–10.

**Steps:**

1. Start with counter = 1.
2. While counter <= 10, print it.
3. Increase counter by 1.

**Trace Table Instructions:**  
Fill in the number printed at each iteration.

| Iteration | Counter Printed |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| … |  |
| 10 |  |

### Exercise 2 – Counting Down

**Goal:** See how loops can run backwards.

**Task:** Write a program that prints numbers 20 down to 1.

**Steps:**

1. Start counter at 20.
2. While counter > 0, print it.
3. Decrease by 1.

**Trace Table Instructions:**  
Predict the value printed at each iteration.

| Iteration | Counter Printed |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| … |  |
| 20 |  |

### Exercise 3 – Sum of Numbers

**Goal:** Practice the accumulation pattern.

**Task:** Add the numbers 1 → 100 and print the total.

**Steps:**

1. Start counter = 1 and total = 0.
2. Each loop: total = total + counter.
3. Stop when counter > 100.

**Trace Table Instructions:**  
Dry-run the first 5 iterations — before and after the addition.

| Counter | Total Before | Total After |
| --- | --- | --- |
| 1 | 0 |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

### Exercise 4 – Multiplication Table

**Goal:** Work with user input and output.

**Task:** Ask the user for a number, print its multiplication table up to ×12.

**Steps:**

1. Start counter = 1.
2. While counter <= 12: print number × counter.
3. Increase counter.

### Exercise 5 – Running Average

**Goal:** Learn sentinel-controlled loops (loops that end with a special input).

**Task:** Ask the user repeatedly for numbers. Stop if they enter 0. Print the average.

**Steps:**

1. Set total = 0, count = 0.
2. Input first number.
3. While number != 0:
   * Add to total.
   * Increase count.
   * Input next number.
4. Print average = total / count.

# Extension Exercises (Optional Challenges)

### Extension 1 – Guess the Number

**Goal:** Practice while + if/else.

**Task:**  
Choose a secret number (e.g. 23). Keep asking the user for guesses until they get it correct.

* Too low → print “Too low.”
* Too high → print “Too high.”
* Correct → print “Correct!” and stop.

**Steps:**

1. Set a secret number.
2. Ask for a guess.
3. While guess != secret:
   * Compare to secret, give hint.
   * Ask again.
4. At the end: congratulate the user.

**Hint:** The condition is while guess != secret:.

### Extension 2 – Factor Finder

**Goal:** Use loops with mathematical conditions.

**Task:** Ask the user for a number n. Print all its factors.  
Example: Input = 12 → Output: 1, 2, 3, 4, 6, 12.

**Steps:**

1. Input number n.
2. Start counter = 1.
3. While counter <= n:
   * If n % counter == 0, print it.
   * Increase counter.

**Trace Table Instructions:**  
Do a dry run for n = 6. Fill in each row:

* Current counter.
* Was the condition true?
* Did it print?

| Counter | (6 % counter == 0?) | Printed? |
| --- | --- | --- |
| 1 | Yes/No |  |
| 2 | Yes/No |  |
| 3 | Yes/No |  |
| 4 | Yes/No |  |
| 5 | Yes/No |  |
| 6 | Yes/No |  |

### Extension 3 – Simple Menu System

**Goal:** Control program flow with while + branching.

**Task:** Create a menu-based program that repeats until Exit.

1. Add two numbers  
2. Multiply two numbers  
3. Exit

* If choice = 1 → ask for two numbers, print sum.
* If choice = 2 → ask for two numbers, print product.
* If choice = 3 → stop program.
* Else → print “Invalid choice.”

**Steps:**

1. Set choice = 0.
2. While choice != 3:
   * Print menu.
   * Ask for user’s choice.
   * Handle it with if/elif/else.
3. End when user picks option 3.

**Hint:** This is also a **sentinel loop** — it ends when the sentinel value (3) is entered.

# Final Notes

* ALWAYS check: **Initialize → Condition → Body → Update**.
* **Trace tables** are a powerful debugging tool — use them before coding.
* Start small, test step by step.
* Avoid infinite loops with a proper update step.