**Python Loops — Seminar 6**

**Note: When you know conditions, loops  you know all about programming !**

## 1. Warm-Up  Revision of the lesson

**Recall**

- for variable in iterable: executes block once per element.
- range(stop) → 0…stop-1
- range(start, stop, step) → start…stop-1 by step
- list(range(...)) converts to a list.

**Exercises  ( 5 minutes)  Do not take too long here please it should be almost automatic**

1. Print the numbers 0–9.
2. Print even numbers from 2 to 20 using range(start, stop, step).
3. Print squares of numbers 1–10 in format $n^2$ = result.
4. Count backwards from 10 down to 1.

---

## 2. Iterating over Lists & Strings

**Recall**

- You can loop directly over elements or via their indices.
- Use len(list) when looping by index.

**Exercises**

1. Given fruits = ["apple", "banana", "cherry"]:
    - Print each fruit on its own line.
    - Print each fruit with its length.
2. Using indices: change each fruit to uppercase **in-place** in the list.
3. Loop through a string "LoopingIsFun" and print only the vowels.

4. Combine lists:
   prices = [10, 20, 30], items = ["pen","book","bag"]
   Print pen → 10 USD etc.

---

**3. while Loops “think deeper about forming the correct condition**

**Recall**

- while condition: repeats while the condition is True.
- Useful when the number of iterations isn’t fixed.

**Exercises**

1. Ask the user for numbers until they enter 0; print the sum of all non-zero numbers entered.
2. Ask for a password; keep asking until they type "openSesame".
3. Count down from a user-given number to 1, then print "Blastoff!".
4. Compute factorial of a number using only while (no built-in factorial).

---

**4. Loop Control: break and continue**

**Recall that this is not a good practice and should be used only when strictly necessary**

- break stops the nearest loop immediately.
- continue skips the rest of the current iteration.

**Exercises**

1. Iterate numbers 1–20: stop printing entirely when you reach 13.
2. Iterate numbers 1–20: skip all multiples of 3 but print the rest.
3. Search a string for the first uppercase letter; print its index or -1 if none (stop at first found).

---

## 5. Nested Loops

**Recall that executing nested loops is costly**

- A loop inside another runs its full cycle for every outer-loop iteration.

**Exercises**

1. Print full multiplication tables from 1×1 to 10×10 in neat aligned columns.
2. Make a 5×5 grid of * characters.
3. From a list of lists:
4. data = [["Anna", 19, "A"], ["Bob", 22, "B"], ["Cara", 21, "A"]]

Print each row’s data on a single line like Anna | 19 | A.

5. Create a right-angled triangle pattern of numbers:
6. 1
7. 1 2
8. 1 2 3
9. ...

---

## 6. Warm up Challenges

**Challenge 1 — Uppercase Extractor**

Input a text; collect all uppercase letters into a **new list** and print them joined as a string.

---

**Challenge 2 — List Normalizer**

Given scores = [12, 55, 37, 90, 47], loop and convert every score below 50 to 50 in place.

---

**Challenge 3 — Word Counter**

Prompt the user for a sentence. Count how many words (split by spaces) have length ≥ 5.

---

**Challenge 4 — Duplicate Finder**

Given nums = [2,4,7,4,9,2,7,3], print which numbers appear more than once **and** how many times.

---

**Challenge 5 — Histogram of Characters**

Ask for a word; print a vertical histogram of each character's occurrence (example for "hello").

---

**Challenge 6 — Prime Numbers up to N**

Ask for N; print all prime numbers up to N.
(Hint: for each candidate, test divisibility using an inner for loop.)

---

**7. Better formed Problems**

**Capstone 1 — Sudoku Row Check**

Given a 9×9 grid as a list of lists of integers, use **loops** to validate that each row has the numbers 1–9 with no duplicates. Print any row index that fails.

---

**Capstone 2 — Tic-Tac-Toe Winner**

Ask for a 3×3 board stored as nested lists with "X", "O" or "".
Check rows, columns, and diagonals using loops to detect if either player has won.

---

**Capstone 3 — Spiral Printer (Stretch Goal)**

Given N, print numbers from 1 to $N^2$ laid out in an **N×N spiral**.
*(This pushes them to reason about indices and nested loops.)*

## Some classic real problems to have a go on:

### 1. ATM Cash Dispenser

- Input: an amount of money (positive integer, e.g. 376).
- Output: how many **100**, **50**, **20**, **10**, **5**, **1** bills to return.
- Must always give the *minimum number* of bills.

### 2. Bus Ticket Validation

- Ask a user to enter the 8-digit ticket number as a string.
- If the sum of the **first 4 digits** equals the sum of the **last 4 digits**, print “Lucky ticket”, otherwise “Regular ticket”.

---

### 3. Inventory Restocker

- Start with a list:
- stock = [["apple", 12], ["banana", 4], ["orange", 0]]
- Ask the manager for a minimum stock threshold (e.g. 5).
- Loop through the list:
  - If quantity is below threshold, print “Restock <item>”.
  - Otherwise print “OK”.

---

### 4. Gradebook Analyzer

- Input: a list of student scores (or ask user to enter them one by one until they enter -1).
- Compute:
  - class **average**
  - **highest** and **lowest** score
  - count of scores >= 50

### 5. Parking Lot Fee Calculator

- Each car pays:

- First 2 hours: \$5/hour
- After that: \$3/hour

- Ask for arrival and departure times (whole hours, 0–23), loop until user enters exit.
- For each car, compute fee and print total for the day.

---

### 6. Text Analyzer

- Ask user for a paragraph.
- Produce:
    - number of **characters**
    - number of **words**
    - number of **sentences** (split on ., !, ?)
    - the **most frequent letter** (ignore case)

---

### 7. Temperature Logger

- Keep asking the user to enter daily temperatures until they type done.
- Store them in a list.
- At the end:
    - print **average**
    - print **days above average**
    - print the **hottest** and **coldest** day

---

### 8. Mini Banking Ledger

- Keep a running balance starting at 0.
- Loop offering actions:
  D → deposit, W → withdraw, B → show balance, Q → quit.

- Reject withdrawals that would make the balance negative.

---

**9. Simple Password Checker**

- Ask for a password input repeatedly until:
  - it’s at least 8 characters,
  - contains at least one digit,
  - contains at least one uppercase letter.

---

**10. Toll-Booth Counter**

- Vehicles come through with a type: car, truck, bike, done.
- Charge: car $2, truck $5, bike $1.
- Loop until done, keep separate counts and the total revenue.

---

**11. Library Late Fee**

- For each book returned, ask: days late.
- Compute fee:
  - first 5 days: $0.50/day
  - next 5 days: $1/day
  - beyond 10 days: $2/day

---

**12. Currency Breakdown with Input Validation**

- Ask for an amount until a valid positive integer is entered.
- Show how to break it into bills and coins of available denominations.

---

**13. Basic Encryption / Decryption**

- Ask the user for a message and a numeric key.
- Produce an **encoded message** where each letter is shifted forward by the key (Caesar cipher).
- Allow decoding by shifting back.

---

### 14. Word Guessing Game (Hangman-lite)

- Pick a secret word from a predefined list.
- Display underscores for hidden letters.
- Let the player guess letters in a loop:
  - reveal correct letters
  - track wrong guesses
- End when all letters are guessed or max attempts exceeded.

---

### 15. Mini-Calendar Days-in-Month

- Ask the user for a month (1-12) and a year.
- Using rules for leap years, print the number of days in that month.