📘 AI Lab – Week 4 (2 Hours)

Course: Artificial Intelligence (BS-CS Semester 3)
File Name: `AI_Lab4_YourName.ipynb`

What you'll practice today

- Functions to structure programs
- Loops & Conditions for logic
- NumPy basics for simple AI-style data analysis
- Small rule-based decisions

> Please read **Lecture 4** (algorithms, linear search/balanced parentheses, OOP encapsulation) and **Lecture 6** (stacks, precedence, infix→postfix, postfix evaluation) before the lab.

Lab Instructions

- Open Anaconda → **Jupyter Notebook**.
- Save this notebook as **`AI_Lab4_YourName.ipynb`**.
- For each task:
  - Read the description block.
  - Run the solution cell to see an example output.
  - Then **modify or extend** the code as instructed by me.
- **Comment your code** with `#` to explain each step.
- Show your work in class (no email/WhatsApp submissions unless told).

🎯 Objectives

- Practice **functions** to organize code.
- Apply **loops + conditions** to solve problems.
- Use **NumPy** to analyze small arrays.
- Build a tiny **rule-based** suggestion system.

🔹 Task 1 — Function Practice: Student Grading

Goal: Write a reusable function `get_grade(marks)` that maps numeric marks to letter grades.

What to learn here:

- How to define a function with parameters

- How to return values
- How to loop over a list and call your function repeatedly

In [7]:
```python
# --- Task 1 Solution (Fully Commented) ---

def get_grade(marks):
    """Return a letter grade based on numeric marks.

    Parameters
    ----------
    marks : int or float
        The marks for a student (0-100).

    Returns
    -------
    str
        One of 'A', 'B', 'C', or 'Fail' according to specified thresholds.
    """
    # Use descending thresholds (highest first) to avoid overlapping conditions.
    if marks >= 80:
        return "A"            # 80–100 → A
    elif marks >= 60:
        return "B"            # 60–79  → B
    elif marks >= 40:
        return "C"            # 40–59  → C
    else:
        return "Fail"         # <40    → Fail

# Example data to demonstrate function usage
marks_list = [95, 72, 38, 60, 45]

# Loop through the list and print grade for each entry
for m in marks_list:
    # f-string shows both the marks and the computed grade
    print(f"Marks: {m} → Grade: {get_grade(m)}")
```

```
Marks: 95 → Grade: A
Marks: 72 → Grade: B
Marks: 38 → Grade: Fail
Marks: 60 → Grade: B
Marks: 45 → Grade: C
```

🔹 Task 2 — Text Cleaner (NLP Pre-processing Step)

Goal: Clean text so it is easier for AI models to process.

What to learn here:

- Lowercasing text
- Removing punctuation
- Splitting a sentence into tokens (words)

Instructions:

1. Convert input sentence to **lowercase**.
2. Remove punctuation characters.
3. Split the result into words.

In [2]:
```python
# --- Task 2 Solution (Fully Commented) ---

import string  # Provides string.punctuation (all common punctuation chars)

# Example input sentence; replace with input() to read from user if needed
sentence = "Hello, Students! AI is amazing, isn't it?"

# 1) Lowercase conversion
sentence_lower = sentence.lower()

# 2) Remove punctuation by replacing each punctuation char with empty string
cleaned = sentence_lower
for p in string.punctuation:
    cleaned = cleaned.replace(p, "")

# 3) Tokenization: split the cleaned sentence into words on whitespace
words = cleaned.split()

# Show results at each stage for clarity
print("Original:", sentence)
print("Lower   :", sentence_lower)
print("Cleaned :", cleaned)
print("Words   :", words)
```

```
Original: Hello, Students! AI is amazing, isn't it?
Lower   : hello, students! ai is amazing, isn't it?
Cleaned : hello students ai is amazing isnt it
Words   : ['hello', 'students', 'ai', 'is', 'amazing', 'isnt', 'it']
```

🔹 Task 3 — NumPy Array Basics

Goal: Practice basic statistics using NumPy arrays (useful for AI data analysis).

What to learn here:

- Creating arrays
- Calculating sum/mean/min/max
- Using boolean masks to filter values

In [3]:
```python
# --- Task 3 Solution (Fully Commented) ---

import numpy as np  # NumPy is the fundamental package for scientific computing

# Create a NumPy array of numeric values (10 elements)
arr = np.array([10, 20, 30, 40, 50, 60, 70, 80, 90, 100])

# Compute basic statistics using NumPy vectorized methods
arr_sum = arr.sum()       # sum of all elements
arr_mean = arr.mean()     # arithmetic mean
arr_max = arr.max()       # maximum value
arr_min = arr.min()       # minimum value
```

```
# Build a boolean mask: True where element is strictly greater than mean
mask_above_mean = arr > arr_mean

# Use the mask to filter elements
above_mean_values = arr[mask_above_mean]

# Display results
print("Array         :", arr)
print("Sum           :", arr_sum)
print("Mean          :", arr_mean)
print("Max / Min     :", arr_max, "/", arr_min)
print("Above Mean?   :", mask_above_mean)  # shows True/False per element
print("Values > mean :", above_mean_values)
```

```
Array         : [ 10  20  30  40  50  60  70  80  90 100]
Sum           : 550
Mean          : 55.0
Max / Min     : 100 / 10
Above Mean?   : [False False False False False  True  True  True  True  True]
Values > mean : [ 60  70  80  90 100]
```

🔹 Task 4 — Mini Rule-Based System: Health Tips

Goal: Build a tiny rule-based AI that suggests a tip based on how the user feels.

What to learn here:

- Mapping keywords to responses using a dictionary
- Applying simple if/else logic on user input

In [4]:
```
# --- Task 4 Solution (Fully Commented) ---

# Example user input (replace with input(...) if you want interactive behavior)
feeling = "hungry"   # try values like "tired", "hungry", "sad", "happy", or som

# Dictionary maps known feelings to suggested responses
responses = {
    "tired": "Take a short rest and drink water.",
    "hungry": "Have a balanced snack (fruit + nuts) or a healthy meal.",
    "sad": "Take a short walk and talk to a friend.",
    "happy": "Great! Keep smiling and spread positivity."
}

# Normalize the feeling string to lowercase to match dictionary keys
feeling_key = feeling.lower()

# Select a response based on the input; provide a default if not found
if feeling_key in responses:
    print("Feeling    :", feeling)
    print("Suggestion :", responses[feeling_key])
else:
    print("Feeling    :", feeling)
    print("Suggestion : I'm not sure about this feeling. Try 'tired', 'hungry',
```

```
Feeling    : hungry
Suggestion : Have a balanced snack (fruit + nuts) or a healthy meal.
```

🔹 Task 5 — NumPy Decision: Even/Odd Analyzer

Goal: Use NumPy to generate a sequence and analyze parity (even/odd).

What to learn here:

- Creating ranges with `np.arange`
- Building boolean masks for conditions
- Counting elements that match a condition

In [5]:
```python
# --- Task 5 Solution (Fully Commented) ---

import numpy as np

# Create integers from 1 to 20 (inclusive of 1, exclusive of 21)
arr = np.arange(1, 21)

# Boolean masks: True where condition holds
even_mask = (arr % 2 == 0)
odd_mask = ~even_mask  # logical NOT of even_mask

# Filter arrays using masks
evens = arr[even_mask]
odds = arr[odd_mask]

# Count evens and odds using len(...) or .size
n_evens = len(evens)
n_odds = len(odds)

# Display results
print("All numbers :", arr)
print("Even mask   :", even_mask)
print("Odd mask    :", odd_mask)
print("Evens       :", evens, "→ count =", n_evens)
print("Odds        :", odds,  "→ count =", n_odds)
```

```
All numbers : [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
Even mask   : [False  True False  True False  True False  True False  True False
True
 False  True False  True False  True False  True]
Odd mask    : [ True False  True False  True False  True False  True False  True
False
  True False  True False  True False  True False]
Evens       : [ 2  4  6  8 10 12 14 16 18 20] → count = 10
Odds        : [ 1  3  5  7  9 11 13 15 17 19] → count = 10
```

🔹 Wrap-Up Challenge — AI Data Summary

Goal: Simulate student marks and compute a simple data summary like an AI analytics step.

What to learn here:

- Generating random integers with NumPy
- Using array methods for summary statistics
- Filtering values above the average

In [6]:
```
# --- Wrap-Up Challenge Solution (Fully Commented) ---

import numpy as np

# Set a random seed so results are repeatable (optional)
np.random.seed(42)

# Generate random marks for 10 students between 0 and 100 (inclusive of 0, exclu
marks = np.random.randint(0, 101, size=10)

# Compute summary statistics
avg = marks.mean()
topper = marks.max()
lowest = marks.min()

# Boolean mask for students scoring above average
above_avg_mask = marks > avg
above_avg_scores = marks[above_avg_mask]

# Display the full analysis
print("Marks array          :", marks)
print("Class Average        :", round(avg, 2))
print("Topper Score         :", topper)
print("Lowest Score         :", lowest)
print("Above-Average Mask   :", above_avg_mask)
print("Above-Average Scores :", above_avg_scores)
```

```
Marks array          : [51 92 14 71 60 20 82 86 74 74]
Class Average        : 62.4
Topper Score         : 92
Lowest Score         : 14
Above-Average Mask   : [False  True False  True False False  True  True  True  Tr
ue]
Above-Average Scores : [92 71 82 86 74 74]
```

✅ Submission Checklist

- This notebook is saved as `AI_Lab4_YourName.ipynb`.
- Each task block has:
  - A clear **description** (markdown cell).
  - A **fully commented** code solution (code cell).
  - Visible **outputs** after running the cell.
- Be ready to show and explain your code to me.