AI Lab – Week 3 (2 Hours)

Course: Artificial Intelligence (BS-CS Semester 3)
Topic: Lists, Dictionaries & AI-Style Decisions
File Name: AI_Lab3_YourName.ipynb

📑 Lab Instructions

- Open **Anaconda → Jupyter Notebook**.
- Save as:

```
AI_Lab3_YourName.ipynb
```

- For each task:
  - Read the **description** carefully.
  - Write your solution in the **code cell** below.
  - Add comments (# ...) in your code to explain steps.
- Show your work to me your master **in class**.
- ❌ No email/WhatsApp submissions.
- ⚠️ Work not done in lab = your responsibility (lab counts in final marks).

🎯 Objectives

- Learn how to use **lists and dictionaries** for data storage.
- Apply **loops + if/else conditions** on real-world–like problems.
- Write **functions** that behave like small "decision-making" AI programs.

✅ Task 1 — Student Marks & Grades

Description:
In this task, you will create a **dictionary** where student names are keys and their marks are values. Using this dictionary, you will write a function that goes through each student and decides their **grade** (A, B, C, or Fail) based on their marks.

👉 This helps you understand how to **store structured data** and how to apply **if/else conditions** inside a loop to make decisions (just like an AI system evaluates input and produces results).

```python
# --- Task 1: Student Marks & Grades ---

# Step 1: Create a dictionary with student names and their marks
students = {"Ali": 85, "Sara": 67, "Bilal": 45}

# Step 2: Define a function to check grades for each student
def print_results(data):
    """
    This function takes a dictionary of student:marks
    and prints each student's marks with grade.
    """
    # Loop through dictionary using items() to get (name, marks)
    for name, marks in data.items():
        # Apply grading rules using if/elif/else
        if marks >= 80:
            grade = "A"
        elif marks >= 60:
            grade = "B"
        elif marks >= 50:
            grade = "C"
        else:
            grade = "Fail"

        # Print result for each student
        print(f"{name}: {marks} -> Grade: {grade}")

# Step 3: Call the function
print_results(students)
```

👉 **Practice Variation:** Add code to also print the **class average** at the end.

✅ Task 2 — Word Counter

Description:
This task will train you to process **text data**. You will ask the user for a sentence, then use a **dictionary** to count how many times each word appears. This is an important skill in **Natural Language Processing (NLP)**, where AI models often need to analyze and understand text.

👉 For example, if you type "AI is fun and AI is powerful," the program should count how many times "AI," "is," "fun," etc. appear.

```python
# --- Task 2: Word Counter ---

# Step 1: Take input sentence from user
sentence = input("Enter a sentence: ")

# Step 2: Split sentence into list of words
words = sentence.split()

# Step 3: Create an empty dictionary for counts
counter = {}

# Step 4: Loop through words
for w in words:
    w = w.lower()  # Convert to lowercase for consistency
    if w in counter:
        # If word already exists, increment count
        counter[w] += 1
    else:
        # If word appears first time, set count = 1
        counter[w] = 1

# Step 5: Print dictionary of word counts
print("Word counts:", counter)
```

👉 **Practice Variation:** Print the **most frequent word** from the dictionary.

✅ Task 3 — Mini Chatbot

Description:
Here you will build a **rule-based chatbot** using a dictionary. The dictionary will store some common phrases as keys (like “hi”, “bye”, “how are you”), and the chatbot will respond with predefined answers.

👉 This shows how **AI assistants** are built at a basic level — they check for known inputs and provide matching outputs.

python

```python
# --- Task 3: Simple Chatbot ---

# Step 1: Create dictionary of possible responses
responses = {
    "hi": "Hello! How can I help you?",
    "bye": "Goodbye! Have a nice day!",
    "how are you": "I’m doing great, thanks!"
}

# Step 2: Take input from user
user = input("You: ").lower()   # lowercase for matching

# Step 3: Match user input with dictionary keys
if user in responses:
    # If user input exists, print the response
    print("Bot:", responses[user])
else:
    # If input not found, show default response
    print("Bot: Sorry, I don’t understand.")
```

👉 **Practice Variation:** Add **3 more inputs and responses.**

✅ Task 4 — Movie Recommendation

Description:
In this task, you will create a **movie recommendation system**. Movies will be stored in a dictionary grouped by genre (action, comedy, sci-fi). When the user enters a genre, the program will randomly recommend a movie.

👉 This is a very simple version of **recommender systems** (like Netflix or YouTube), where AI suggests content based on user choice.

python

```python
# --- Task 4: Movie Recommender ---
import random  # Import random library for random.choice()

# Step 1: Create dictionary of genres and movie lists
movies = {
    "action": ["Avengers", "Batman", "Spiderman"],
    "comedy": ["Mr. Bean", "Home Alone", "The Mask"],
    "sci-fi": ["Interstellar", "Inception", "The Matrix"]
}

# Step 2: Ask user for preferred genre
genre = input("Enter genre (action/comedy/sci-fi): ").lower()

# Step 3: Check if genre exists in dictionary
if genre in movies:
    # Suggest a random movie from the selected genre
    print("Watch:", random.choice(movies[genre]))
else:
    # Handle invalid input
    print("Sorry, try action, comedy, or sci-fi.")
```

👉 **Practice Variation:** Add a new genre (e.g., "horror") with at least 2 movies.

✅ Task 5 — Weather-Based Decision

Description:
This task simulates a small AI program that **suggests an activity based on weather conditions**. The user provides weather (sunny, rainy, cold, windy), and the program decides what to do.

👉 This introduces the concept of **decision-making systems**, where input conditions guide the output recommendation.

```python
# --- Task 5: Weather Decision ---

# Step 1: Define function with weather as input
def suggest_activity(weather):
    weather = weather.lower()   # normalize input
    if weather == "sunny":
        return "Go outside."
    elif weather == "rainy":
        return "Take an umbrella."
    elif weather == "cold":
        return "Stay warm at home."
    elif weather == "windy":
        return "Fly a kite."
    else:
        return "Unknown weather condition."

# Step 2: Example test (you can replace with user input)
print(suggest_activity("rainy"))
```

👉 **Practice Variation:** Take **user input** for weather instead of hardcoding.

✅ Task 6 — Mini Data Analyzer

Description:
We will analyze a list of numbers and calculate maximum, minimum, and average values. We'll also print which numbers are above the average. This shows how AI can **extract insights from data**.

```python
# --- Task 6: Data Analyzer ---

# Step 1: Example dataset
numbers = [12, 45, 67, 23, 89, 34]

# Step 2: Define function to analyze numbers
def analyze(nums):
    maximum = max(nums)   # Find largest
    minimum = min(nums)   # Find smallest
    average = sum(nums) / len(nums)   # Calculate mean

    # Print summary
    print("Numbers:", nums)
    print("Max:", maximum)
    print("Min:", minimum)
    print("Average:", average)

    # Find numbers greater than average
    above_avg = [n for n in nums if n > average]
    print("Above average:", above_avg)

# Step 3: Call function
analyze(numbers)
```

👉 **Practice Variation:** Let the user enter their own numbers instead of using the predefined list.

Wrap-Up Challenge — Student Report System

Description:
This final challenge brings everything together. You will build a **student report generator**. Store student names and marks in a dictionary, then print a formatted table showing name, marks, grade, and pass/fail. At the end, also calculate the **class average** and find the **topper**.

👉 This is similar to how AI-powered education systems analyze student performance and generate reports.

python

```python
# --- Wrap-Up: Student Report ---

# Step 1: Create dictionary of students and marks
students = {
    "Ali": 85,
    "Sara": 67,
    "Bilal": 45,
    "Ayesha": 92,
    "Omar": 54
}

# Step 2: Define function to print report
def report(data):
    print("Name\tMarks\tGrade\tResult")
    print("-"*32)

    # Loop through students
    for name, marks in data.items():
        # Decide grade + result
        if marks >= 80:
            grade, result = "A", "Pass"
        elif marks >= 60:
            grade, result = "B", "Pass"
        elif marks >= 50:
            grade, result = "C", "Pass"
        else:
            grade, result = "F", "Fail"

        # Print each row
        print(f"{name}\t{marks}\t{grade}\t{result}")

    # Calculate class average
    avg = sum(data.values()) / len(data)
    # Find topper using max()
    topper = max(data, key=data.get)

    # Print summary
    print("\nClass Average:", round(avg, 2))
    print("Topper:", topper, "with", data[topper], "marks")

# Step 3: Call function
report(students)
```

👉 **Practice Variation:** Accept 5 student names and marks as input, then build and print the report.