Introduction to Python Language

Application 06- Programming Session

1. Write the output of the following code.

```
def modify(a, b):
    a.append(100)
    b = b + [200]
    print("Inside a:", a)
    print("Inside b:", b)

x = [1, 2]
y = [1, 2]

modify(x, y)
print("Outside x:", x)
print("Outside y:", y)
```

1. ANSWER

```
Inside a: [1, 2, 100]
Inside b: [1, 2, 200]
Outside x: [1, 2, 100]
Outside y: [1, 2]
```

`a.append()` → change the original list

`b = b + [200]` → creates a new list, `y` in the outside isn't change.

2. A research drone collects the temperature of an area in Celsius and sends it to your program.
Write a Python program that:

- Asks the user for a temperature in Celsius
- Converts the Celsius reading to Fahrenheit (fahrenheit = celsius * 9 / 5 + 32)
- Classifies the environment :

➢ " Critical Heat Zone " if temperature > 90°F
➢ " Freezing Hazard Zone " if temperature < 30°F
➢ Otherwise " Stable Zone " output will be used by the drone's emergency system.

```
# A2 – Weather Drone System

celsius = float(input("Enter the temperature in Celsius from the drone: "))

fahrenheit = celsius * 9 / 5 + 32
print(f"Temperature in Fahrenheit: {fahrenheit:.1f} °F")

if fahrenheit > 90:
    print("Zone Status: Critical Heat Zone")
elif fahrenheit < 30:
    print("Zone Status: Freezing Hazard Zone")
else:
    print("Zone Status: Stable Zone")
```

3. You are developing a fitness tracker. Users enter the number of calories burned after each exercise session. The user may enter:

- A number → valid session
- An empty input → end of day

Write a program that:

- Repeatedly asks for calorie values
- Computes the total and average calories burned in the day

```
# A3- Method 1 - Fitness App Statistics

tot_cal = 0.0
session_count = 0
while True:
    entry = input("Enter calories burned this session (press Enter to finish): ")
    if entry == "":
        break
    calories = float(entry)
    tot_cal=tot_cal + calories
    session_count += 1

if session_count > 0:
    ave_cal = tot_cal / session_count
    print("\nDaily Summary:")
    print("Sessions:", session_count)
    print("Total calories burned:", tot_cal)
    print("Average calories per session:", ave_cal)
else:
    print("No session data was entered.")
```

```
# A3- Method 2 - Fitness App Statistics

total_calories = 0.0
session_count = 0

while True:
    entry = input("Enter calories burned this session (press Enter to finish): ")
    if entry == "":
        break
    try:
        calories = float(entry)
        total_calories += calories
        session_count += 1
    except ValueError:
        print("Invalid input. Please enter a number or press Enter to stop.")

if session_count > 0:
    average_calories = total_calories / session_count
    print("\nDaily Summary:")
    print("Sessions:", session_count)
    print("Total calories burned:", total_calories)
    print("Average calories per session:", average_calories)
else:
    print("No session data was entered.")
```

4. A university's digital portal needs to automatically generate usernames.
Write a program that:

- Takes the student's full name as input
- Creates a username using :
  - First initial (indexing)
  - First 7 letters of last name (slicing)
- Converts everything to lowercase
- Example:
  Input: "Sarah Wellington"
  Output: swelling

```
# A4 - Online Portal: Username Creation

full_name = input("Enter student's full name (e.g., Sarah Wellington): ")

# Split into parts
parts = full_name.split()

# Defensive check
if len(parts) < 2:
    print("Please enter at least a first name and a last name.")
else:
    first_name = parts[0]
    last_name = parts[1]

    first_initial = first_name[0].lower()
    last_part = last_name[:7].lower()

    username = first_initial + last_part
    print("Generated username:", username)
```

5. A digital library stores short text files describing books. Given a file bookinfo.txt, write a program that:
Write a program that:

- Opens the file
- Counts how many lines, words, and characters it contains
- Prints a summary report for librarians. This helps librarians track metadata quality.

Use the data set below: save it as "bookinfo.txt« in the same directory with the program.

Welcome to the Gaziantep University Digital Library.
This system provides access to electronic books, journals, theses, and technical reports.
Users can search by title, author, keyword, or publication year.

The library supports students and researchers in engineering, natural sciences, and social sciences.
Core collections include materials on mechanics, materials science, computer science, and data analysis.
Many resources are available as full-text PDF, while others provide abstracts and bibliographic information.

Digital systems and data processing play an important role in modern libraries.
Usage statistics, access logs, and recommendation systems help librarians improve services.
Search algorithms are constantly updated to provide faster and more relevant results.

Each digital item in the collection has a unique identifier.
Metadata records store information such as title, author, year, language, and subject area.
Well-structured metadata makes it easier to integrate the library with external databases and search engines.

Gaziantep University Library collaborates with international publishers and open-access platforms.
Students can access many resources from outside the campus network using secure authentication.
Workshops on literature search, reference management, and research data management are offered every semester.

Digital transformation in libraries is an ongoing process.
Librarians, IT staff, and academic departments must work together to design effective systems.
Good digital library design focuses on usability, accessibility, and long-term preservation of information.

123 456

```
# A5 - Digital Library Analyzer

filename = "bookinfo.txt"   # Make sure this file exists in the same folder

line_count = 0
word_count = 0
char_count = 0

try:
    with open(filename, "r", encoding="utf-8") as f:
        for line in f:
            line_count += 1
            words = line.split()
            word_count += len(words)
            char_count += len(line)  # counts all characters including spaces and '\n'

    print("Library Text Summary:")
    print("Lines     :", line_count)
    print("Words     :", word_count)
    print("Characters:", char_count)

except FileNotFoundError:
    print(f"File '{filename}' not found. Please check the file name and path.")
```

6. A smart home system logs energy consumption readings (one number per line) in a file called energy.txt. Write:

1. A function check_power(num) that returns "Even" if the reading is even and "Odd" if it is odd.
2. A program that reads all energy readings from the file and prints results in the format:

```
45 → Odd
102 → Even
```

This helps engineers detect abnormal fluctuations.

```
# Q6 – Smart Home: Even/Odd Power Checker

def check_power(num):
    """Return 'Even' if num is even, otherwise 'Odd'."""
    if num % 2 == 0:
        return "Even"
    else:
        return "Odd"

def main():
        try:
        with open("energy.txt", "r", encoding="utf-8") as f:
            for line in f:
                line = line.strip()
                if not line:           # skip empty lines
                    continue
                try:
                    value = int(line)
                    status = check_power(value)
                    print(f"{value} → {status}")
                except ValueError:
                    print(f"Invalid reading in file: '{line}' (not an integer)")
    except FileNotFoundError:
        print(f"File '{filename}' not found. Please check the file name and path.")
**  if __name__ == "__main__":
    main()
```

** This line controls when the main( ) function should run.
It ensures that the code inside main( ) runs only when the file is executed directly, and NOT when it is imported by another Python file.

__name__ is a special variable that Python sets depending on how the file is used.

- If the file is run directly->Python assigns : "_main_« thus condition becomes TRUE.
- If the file is imported into another program-> Python sets "smart_home« thus the condition becomes FALSE

This prevents the program from running automatically when the module is reused.