Chapter 1: Modules, Comments, and PIP

Type	Lecture
Reviewed	☐

1.1 Writing Your First Python Program

To begin your journey in Python, create a file named `hello.py` with the following code:

```
print("Hello, World")
```

To run it:

- Open your terminal.
- Type: `python hello.py`
- You'll see: `Hello, World`

> Concept Highlight: The print() function is a built-in way to display output in Python.

1.2 Understanding Modules

A **module** is a file containing Python code, such as functions, classes, or variables, which can be reused. It is usually pre-written and can be imported and used in a program.

Why use modules?

- **Organize code** into smaller, manageable pieces.
- **Reuse functionality** without rewriting code.

- **Import built-in or external libraries**.

Types of Modules:

1. **Built-in Modules** (e.g., `os`, `math`, `random`)

   These come with Python and are ready to use.

   Example:

   ```
   import math
   print(math.sqrt(25))  # Output: 5.0
   ```

2. **External Modules** (e.g., `flask`, `numpy`, `pandas`)

   Install them using `pip`:

   Example:

   ```
   pip install requests
   ```

   Then in Python:

   ```
   import requests
   ```

3. **Custom modules**

   You can create your own `.py` files with reusable code.

   Example:

   ```
   # greetings.py
   def say_hello(name):
       print(f"Hello, {name}!")
   ```

   Then in another file:

   ```
   import greetings
   greetings.say_hello("Shreesh")  # Output: Hello, Shreesh!
   ```

> `greetings.py` is the **file name** where you write and save your Python code.
>
> `def` is short for **define** — it's used to **create (define) a function** in Python.

1.3 Using Python as a Calculator (REPL Mode)

What is REPL Mode in Python?

REPL stands for:

> Read – Eval – Print – Loop

It's an interactive mode where you can type Python code **line by line**, and it will execute each line immediately.

Breakdown of REPL:

1. **Read**: Takes the input you type.
2. **Eval**: Evaluates (runs) the input as Python code.
3. **Print**: Displays the result/output.
4. **Loop**: Repeats the process — waits for your next input.

How do you enter REPL Mode?

Type `python` in the terminal and hit Enter to start interactive mode (REPL: Read-Eval-Print Loop).

(Or `python3` depending on your system)

You'll then see something like:

```
>>>
```

Now you can start typing code directly:

```
>>> 2 + 2
4
>>> print("Hello!")
Hello!
```

REPL is great for:

- Quickly testing ideas
- Trying small code snippets
- Learning Python interactively

1.4 Python Comments

Comments in Python are notes **you write in your code to explain what it does**. They are **not executed** by Python. They are just for developers to understand the code better.

Types of Comments in Python:

1. Single-line comments

Use the `#` symbol. Everything after `#` on that line is ignored by Python.

```
# This is a single-line comment
print("Hello, world!")  # This comment is at the end of a line
```

> Shortcut: Use CTRL + / to t**oggle** a line of code into a comment (adds or removes `#` at the beginning).

2. Multi-line comments

Python doesn't have true multi-line comment syntax, but you can write multiple `#` lines or use a **multi-line string** as a "hack" for commenting.

```
# This is a comment
# spread across multiple lines
# to explain something big
```

Or

```
"""
This is a multi-line string,
but can also be used as a comment.
Just remember: it's technically a string,
so avoid using it for actual comments in logic-heavy code.
"""
```

Why use comments?

- To explain **what** your code is doing
- To **remind yourself** why you wrote something a certain way
- To help **others** understand your code easily

Practice Tasks

1. Print the poem “Twinkle Twinkle Little Star.”
2. Use REPL to print the multiplication table of 5.
3. Install an external module and use it (e.g., `pyjokes`, `emoji`).
4. Use the `os` module to list contents of a directory.
5. Add comments to explain your program from #4.

Notes Zone:

The `print()` function is a built-in way to display output in Python.

This means:

- `print()` is a **built-in function** → You can use it without importing anything.
- It's used to **display information on the screen** (usually in the terminal or console).

Example:

```
print("Hello, Shreesh!")
```

Output:

```
Hello, Shreesh!
```

So whenever you want to **see the result** of your code, show a message, or debug something, you use `print()`.

Practice Set:

Q-1: Write a program to print Twinkle twinkle little star poem in python.

There are two ways to do it:

1. Using '#' to print every single line:

```
print("Twinkle, twinkle, little star,")
print("    How I wonder what you are!")
print("        Up above the world so high,")
print("        Like a diamond in the sky.")
print("Twinkle, twinkle, little star,")
print("    How I wonder what you are!")
```

2. Using ''' to print all lines together:

```
print ('''Twinkle, twinkle, little star,
How I wonder what you are!
Up above the world so high,
Like a diamond in the sky.

When the blazing sun is gone,
When he nothing shines upon,
Then you show your little light,
Twinkle, twinkle, all the night.

Then the trav'ller in the dark,
Thanks you for your tiny spark,
He could not see which way to go,
If you did not twinkle so.

In the dark blue sky you keep,
And often thro' my curtains peep,
For you never shut your eye,
Till the sun is in the sky.

'Tis your bright and tiny spark,
Lights the trav'ller in the dark:
Tho' I know not what you are,
Twinkle, twinkle, little star.''')
```

Q: Print the table of 5.

A simple way is to print everything one by one:

```
print("5 × 1 = 5")
print("5 × 2 = 10")
print("5 × 3 = 15")
```

```
print("5 × 4 = 20")
print("5 × 5 = 25")
print("5 × 6 = 30")
print("5 × 7 = 35")
print("5 × 8 = 40")
print("5 × 9 = 45")
print("5 × 10 = 50")
```

Another way is to print using loops like ‘while’ and ‘for’:

Using ‘while’ loop:

```
i = 1
while i <= 10:
    print("5 x", i, "=", 5 * i)
    i = i + 1
```

What this does:

- Loops from **1 to 10**
- Prints the multiplication table of **5**
- Uses **string concatenation** (not f-string) for output formatting
- Multiplies **5 by each number in the loop** and displays the result

Using ‘for’ loop:

```
for i in range(1, 11):
    print(f"5 x {i} = {5 * i}")
```

What this does:

- Loops from 1 to 10
- Uses f-string formatting for readable output
- Multiplies 5 by each number in the loop

Future Task:

1. Make it dynamic — ask the user for a number and print its multiplication table.
2. Learn f-string.

Q: Install an external module and use it to perform an operation of your interest.

Module: `pyjokes`

The `pyjokes` module gives your Python code a sense of humor! It lets you fetch programmer-friendly jokes instantly.

Why It's Great for Beginners:

- Super lightweight and easy to use
- No setup needed beyond installation
- Great practice for working with external modules
- Fun way to learn Python while laughing!

Step 1: Choose and Install a Module

A good beginner-friendly module is 'pyjokes' (it gives random programming jokes).

- In your terminal or VS Code terminal, run:

```
pip install pyjokes
```

Step 2: Use the Module in a Python File

Create a Python file (e.g., `Problem3.py`) and write:

```
import pyjokes
```

```
joke = pyjokes.get_joke()
print("Here's a joke for you:")
print(joke)
```

> **Here is what this code does:**
> import pyjokes → Importing the 'pyjokes' module which provides programming-related jokes.
> joke = pyjokes.get_joke() → Getting a random joke and storing it in the variable 'joke'.
> print("Here's a joke for you:") → Printing an introductory message.
> print(joke) → Printing the actual joke fetched from pyjokes.

Step 3: Run the File

In your terminal:

```
python joke.py
```

You'll see a random joke printed to the screen!

Why This is Useful

- Shows how to install third-party packages with pip
- Teaches how to import and use external libraries
- Helps build confidence with simple fun projects

Here's another **basic and fun Python module** to try:

Module: `emoji`

The `emoji` module lets you print emojis in your terminal using simple codes.

Why It's Great for Beginners:

- Easy to install and use
- Helps understand how strings and Unicode work
- Adds fun to learning

Step 1: Install the Module

Open your terminal and run:

```
pip install emoji
```

Step 2: Try It in a Python File

Create a file called `emojis.py` and add:

```
import emoji

print(emoji.emojize("Python is fun :snake:", language='alias'))
print(emoji.emojize("Keep going! :rocket:", language='alias'))
print(emoji.emojize("You're doing great! :thumbs_up:", language='alias'))
```

▶️ Step 3: Run It

```
python emojis.py
```

You should see:

```
Python is fun 🐍
Keep going! 🚀
You're doing great! 👍
```

Module: `pyttsx3`

The `pyttsx3` module lets your Python code **speak aloud using text-to-speech**—no internet required!

Why It's Great for Beginners:

- Super easy to set up and use
- Works **offline**
- Introduces you to **Python modules** and **speech synthesis**
- Great for adding accessibility or fun to your projects

Step 1: Install the Module

Open your terminal or command prompt and run:

```
pip install pyttsx3
```

Step 2: Try It in a Python File

Create a file called `speak.py` and add:

```
import pyttsx3

engine = pyttsx3.init()
engine.say("Hey I am good")
engine.runAndWait()
```

Step 3: Run It

In your terminal, run:

```
python speak.py
```

🎧 You should hear your computer say:

"Hey I am good"

Module: `os`

The `os` module allows your Python programs to interact with your computer's operating system — like working with files and directories.

Why It's Great for Beginners:

- Comes built-in with Python (no installation needed!)
- Lets you explore your own file system with code
- Helps you learn how programs interact with folders and files
- Foundation for automation and scripting tasks

Step 1: No Installation Needed

The `os` module is part of Python's standard library — just import and use!

Step 2: Try It in a Python File

Create a file called `list_dir.py` and add:

```
import os  # Import the os module

# Get current working directory
print("Current Directory:", os.getcwd())

# List all files and folders in the current directory
contents = os.listdir()
print("Contents of the directory:")
for item in contents:
    print("-", item)
```

Step 3: Run It

In your terminal or command prompt, run:

```
python list_dir.py
```

You should see output like:

```
Current Directory: C:\Users\Shreesh\Documents
Contents of the directory:
- file1.txt
- photos
- script.py
- README.md
```