Module 1: Introduction, Setup, and Fundamentals

1.1 Python Fundamentals & History

What is Python?

Python is a high-level, interpreted, general-purpose programming language created by Guido van Rossum and first released in 1991. It is designed for **readability** and features an often-cited simple syntax, making it an excellent choice for beginners.

Feature	Explanation
High-Level	Programmers do not need to manage memory or complex system details (like CPU architecture). Focus is on solving problems, not system management.
Interpreted	The code is executed line-by-line by an interpreter, not compiled into machine code all at once before running. This makes debugging easier.
Dynamically Typed	You don't need to declare a variable's data type (like `int` or `string`) before using it. The type is checked and assigned during runtime.
General-Purpose	Can be used for almost any kind of application, including web development, data analysis, scripting, and more.

Applications

- **Web Development (Backend):** Using frameworks like Django and Flask.
- **Data Science & Machine Learning:** Libraries like NumPy, Pandas, Scikit-learn.
- **Automation/Scripting:** Automating repetitive tasks (file management, system administration).
- **Software Testing:** Used for writing test scripts.

The Python Interpreter (REPL)

Python's interpreter allows you to execute code interactively, line-by-line. This is often called the **Read-Eval-Print-Loop (REPL)**.

- **Read:** Reads the user input.
- **Eval:** Evaluates the input expression.
- **Print:** Prints the result.
- **Loop:** Returns to the beginning to read the next input.

Example (In a terminal):

Python
```
>>> 2 + 3
5
>>> print("Hello")
Hello
```

1.2 Environment Setup

Installation and IDE

1. **Installation:** Download the official installer from python.org. Ensure the option "Add Python to PATH" is checked during installation.
2. **IDE/Code Editor:** While you can use a simple text editor, an **Integrated Development Environment (IDE)** or advanced code editor is highly recommended for writing larger programs.
    - **VS Code:** (Popular code editor)
    - **PyCharm:** (Dedicated Python IDE, good for large projects)

Writing and Executing a .py File

To run a script (a program saved as a file):

1. **Write the code:** Save your code in a file named, for example, first_script.py.

   Python

   ```
   # first_script.py
   print("Executing my first Python script!")
   ```

2. **Execute from terminal:** Navigate to the file's directory and run the command:

   Bash

   ```
   python first_script.py
   ```

Understanding PVM and Bytecode

When you run a Python file, the following happens:

1. **Compilation:** The Python source code (.py file) is first compiled into an intermediate form called **Bytecode**.
2. **Bytecode File:** This bytecode is often saved as a .pyc file (Python Compiled).
3. **Execution:** The **Python Virtual Machine (PVM)** is the runtime engine that executes the bytecode. The PVM is responsible for managing memory and running the program.

1.3 Syntax Basics and Conventions

Keywords and Identifiers

- **Keywords:** Reserved words that have special meaning to the interpreter. You cannot use them as variable names. (e.g., if, for, while, def, class, return).
- **Identifiers:** Names given to variables, functions, classes, etc.
    - **Rules:** Must start with a letter (A-z) or underscore (_). Cannot be a keyword. Case-sensitive (age is different from Age).

Indentation: The Crucial Rule

In many languages, code blocks (like those following an if or for statement) are defined by curly braces ({}). **In Python, code blocks are defined by indentation.**

- All statements within a block **must** have the same level of indentation (usually 4 spaces).
- Inconsistent indentation will result in an IndentationError.

Example:

Python
```
# CORRECT indentation
if True:
    print("This runs")
    print("This also runs because it's at the same level")

# INCORRECT indentation (Syntax Error)
if True:
    print("This runs")
  print("This breaks the code block!") # Two spaces instead of four
```

Comments

Comments are non-executable notes used to explain the code.

- **Single-line comments:** Use the # symbol.

  Python

  ```
  # This is a comment, the interpreter ignores it.
  temperature = 25
  ```

- **Multi-line comments (Docstrings):** Python uses **triple quotes** ("""…""" or '''…''') primarily for Docstrings (documentation strings, see Module 5), but they can also serve as multi-line comments.

Statements and Expressions

- **Statement:** An instruction that the Python interpreter can execute. It performs an action.
    - *Examples:* Variable assignment, printing output, a loop (for i in range(5)).

  Python

```
x = 10              # Assignment statement
print("Hello")      # Print statement
```

- **Expression:** A combination of values, variables, operators, and function calls that the interpreter evaluates to produce a single value.
    - *Examples:* Mathematical calculations, comparisons, function calls.

    Python

    ```
    10 + 5       # Evaluates to 15
    'a' * 3      # Evaluates to 'aaa'
    ```

PEP 8: Code Style Guidelines

PEP 8 is the official style guide for writing readable Python code. Adhering to it makes your code consistent and easier for others (and your future self) to understand.

- **Key Rules:**
    - Use 4 spaces per indentation level.
    - Use meaningful variable names (e.g., `user_name` instead of `u_n`).
    - Limit lines to a maximum of 79 characters.
    - Use blank lines to separate functions and classes.