Lesson Plan – Python Lesson 1 (1 Hour)

Objective

By the end of this lesson, students will:

- Understand what Python is and how to set it up.
- Write and run their first Python program.
- Understand variables, operators, and data types.
- Manipulate strings and numbers in Python.
- Build a simple project: a **Task Logger**.

1. Setup & Environment (5 min)

- Download Python from python.org/downloads.
- Install with the default wizard.
- Open **IDLE**. Explain what REPL is (Read-Evaluate-Print Loop).

Quick Demo:

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

2. Functions & Errors (5 min)

- Explain that functions are “mini-programs” that do a job.
- Show how to “call” them with parentheses.

Example:

```
print("This is a function call")
```

- Introduce errors:

```
>>> prnt("oops")   # typo
```

Discuss **traceback** and runtime errors.

3. Variables & Operators (10 min)

- Variables are “boxes” that hold values.

Example:

```
>>> name = "Alice"
>>> age = 25
```

- Operators (focus on = assignment operator).
- Introduce comments (`# this is a comment`).

4. Data Types & Strings (10 min)

- Strings, integers, floats.
- Use `type()` function.
- String delimiters (`"  "` vs `'  '`).
- Escape sequences (`\n`, `\"`).
- `len()` for string length.

Example:

```
>>> alphabet = "abcdef"
>>> print(alphabet[0])    # a
>>> print(alphabet[-1])   # f
>>> print(alphabet[0:3])  # abc
```

- Explain immutability:

```
>>> s = "hello"
>>> s[0] = "H"   # error
```

5. String Methods (5 min)

Show useful methods:

```
>>> msg = " hello world "
>>> print(msg.strip())
>>> print(msg.upper())
>>> print(msg.startswith("h"))
```

6. Arithmetic & Numbers (10 min)

- Integers vs floats.
- Casting with `int()`, `float()`, `str()`.
- f-strings for formatting.
- Arithmetic operators: `+ - * / // % **`.
- Floating point error demo:

```
>>> print(0.1 + 0.2)  # not exactly 0.3
```

Math functions: `round()`, `abs()`, `pow()`.
Complex numbers: `.real`, `.imag`.

7. Mini-Project: Task Logger (15 min)

Guide students through building:

```
>>> # Get task input from user
>>> task_name = input("Enter the task name: ")
>>> difficulty = input("Enter the difficulty (1-5): ")
>>>
>>> # Clean up task name
>>> task_name = task_name.strip()
>>>
>>> # Calculate a simple score
>>> score = len(task_name) * int(difficulty)
>>>
>>> # Display result
>>> print(f"Task: {task_name.upper()}")
>>> print(f"Score: {score}")
```

Stretch Challenge: Validate inputs are integers.

Wrap-Up (5 min)

- Recap: variables, strings, operators, input/output.
- Homework: play with `print()`, `len()`, string methods, and math operators.

Teacher Guide / Aide

Timing Breakdown

- Setup & Hello World → 5 min
- Functions & Errors → 5 min
- Variables & Operators → 10 min
- Data Types & Strings → 10 min
- String Methods → 5 min
- Arithmetic & Numbers → 10 min
- Project → 15 min
- Wrap-up → 5 min

Teaching Tips

- **Engagement**: After every new concept, ask learners to try a quick variation. Example: after showing `print("Hello")`, ask them to print their name.
- **Error Normalization**: Encourage mistakes! Deliberately create syntax/runtime errors and walk through them.
- **Analogy**: Variables = “labeled boxes.” Strings = “words/sentences inside quotes.”
- **Hands-On**: Every section should have a live coding demo + student practice.
- **Stretch for faster learners**:
    - Add more methods (`.replace()`, `.find()`).
    - Expand project: store multiple tasks in a list.

Common Pitfalls

- Forgetting quotes in strings → `NameError`.
- Misusing `=` vs `==`.
- Mixing string and int without casting.

- Off-by-one indexing errors.

Materials Needed

- Projector/laptop to demo IDLE.
- Handout/slide with key operators and string methods (optional).