- What Python is and why it's popular
- Where Python is used (web, automation, data, AI, scripting)
- Using the Python interpreter (REPL)
- Choosing and setting up a code editor (VS Code)
- Running Python files

Developer Tools & Code Quality

- Linting (finding errors and bad practices)
- Formatting code automatically
- How Python code is executed internally
- Python implementations (CPython, PyPy)

Basics of Python Syntax

- Comments
- Variables and naming rules
- Data types (int, float, string, boolean)

Strings

- Creating strings
- Escape sequences (`\n`, `\t`, etc.)
- String concatenation
- Formatted strings (f-strings)
- Common string methods (`upper`, `lower`, `find`, `replace`, etc.)

Numbers

- Integers and floats
- Arithmetic operators
- Operator precedence
- Math operations
- Type conversion (`int()`, `float()`, `str()`)

Comparison & Logic

- Comparison operators (`==`, `!=`, `<`, `>`, `<=`, `>=`)
- Logical operators (`and`, `or`, `not`)
- Short-circuit evaluation
- Chaining comparisons

Conditional Statements

- `if`, `elif`, `else`
- Nested conditionals

- Ternary (one-line if-else)

Loops

- `for` loops
- `range()`
- Iterating over strings and lists
- `for-else` construct
- Nested loops
- `while` loops
- Infinite loops
- Loop control concepts

Iterables

- Understanding iterables
- Lists, ranges, strings (conceptual level)
- Iteration mechanics

Functions

- Defining functions (`def`)
- Calling functions
- Parameters and arguments
- Positional arguments
- Keyword arguments
- Default arguments
- Functions with return values
- Functions without return values
- `*args` and `**kwargs`

Mutable vs Immutable Objects

- **Mutable objects** (can be changed after creation like lists, dictionaries)
- **Immutable objects** (cannot be changed directly like ints, strings, tuples)

List Comprehensions

Shows how to use list comprehensions a concise way to build lists from other iterables.

Example pattern:

```
new_list = [expression for item in iterable if condition]
```

This replaces longer loop-based constructions.

Function Argument & Parameter Types

Breaks down different ways to pass information into functions:

- Positional parameters
- Default values
- Keyword arguments
- Possibly variable argument lists (conceptually)

`if __name__ == "__main__"`

Explains what this special Python construct means:

- It controls whether code runs when a file is executed directly
- Helps separate **runtime code from importable modules**

Global Interpreter Lock (GIL)

Introduces the concept of the **GIL**, which affects how Python executes threads:

- Only one thread executes Python bytecode at a time
- Important for understanding performance and concurrency limitations