

1. Why Python?

Python is often referred to as "executable pseudocode" because its syntax closely resembles the English language.

- **Interpreted Language:** Python executes code line-by-line, which makes debugging easier and faster.
 - **Dynamically Typed:** You don't need to declare variable types (e.g., `x = 10` instead of `int x = 10`).
 - **Large Standard Library:** "Batteries included" philosophy—Python comes with modules for everything from file I/O to HTTP requests.
-

2. Core Python Features

Data Structures (Built-in)

Python provides powerful, flexible ways to store data:

- **Lists:** Ordered, mutable collections: `[1, 2, "Python"]`.
- **Tuples:** Ordered, immutable (cannot be changed): `(10, 20)`.
- **Dictionaries:** Key-value pairs (extremely fast lookups): `{"name": "Alice", "age": 25}`.
- **Sets:** Unordered collections of unique elements.

List Comprehensions

A concise way to create lists.

`squares = [x**2 for x in range(10)]` replaces a four-line for loop.

3. Advanced Python Concepts

To master Python, you must move beyond basic loops and functions into its internal mechanics.

I. Decorators

Decorators allow you to modify the behavior of a function or class without permanently changing its source code. They are essentially wrappers.

Python

```
def my_decorator(func):
```

```
    def wrapper():
```

```
        print("Something is happening before the function is called.")
```

```
        func()
```

```
        print("Something is happening after.")
```

```
    return wrapper
```

```
@my_decorator
```

```
def say_hello():
```

```
    print("Hello!")
```

II. Generators and Iterators

Instead of storing a giant list in memory (which could crash your app), Generators yield items one at a time using the yield keyword.

- Memory Efficiency: They use "lazy evaluation," meaning they only calculate the next value when requested.

III. Context Managers (with statement)

Used for resource management, such as opening files or database connections. It ensures that resources are closed automatically, even if an error occurs.

```
with open('data.txt', 'r') as file: content = file.read()
```

IV. Functional Programming Tools

- Lambda Functions: Anonymous, one-line functions: `add = lambda x, y: x + y`.
 - Map, Filter, and Reduce: Higher-order functions that process collections efficiently.
-

4. Object-Oriented Programming (OOP)

Python is an object-oriented language. Everything—numbers, strings, and functions—is an object.

- Classes and Objects: Blueprints for creating data structures.
 - Inheritance: Creating a new class that inherits attributes from an existing one.
 - Dunder Methods (Magic Methods): Methods like `__init__`, `__str__`, or `__add__` that allow you to define how objects behave with operators like `+` or `print()`.
-

5. Python 3.12+ and Modern Features

The latest versions of Python have introduced significant performance and syntax improvements:

- Type Hinting: Using `def greet(name: str) -> str:` to make code more robust and IDE-friendly.
 - Structural Pattern Matching: The `match-case` statement (similar to `switch-case` in JS), introduced in 3.10.
 - Faster CPython: Massive performance optimizations in the interpreter itself.
 - AsyncIO: A library to write concurrent code using the `async/await` syntax, perfect for network-bound tasks.
-

6. Applications of Python

| Field | Popular Libraries |
|----------------------|-----------------------------------|
| Data Science | Pandas, NumPy, Matplotlib |
| Machine Learning | TensorFlow, PyTorch, Scikit-learn |
| Web Development | Django, Flask, FastAPI |
| Automation/Scripting | Selenium, BeautifulSoup, Scrapy |

7. Best Practices (The Zen of Python)

Type `import this` in a Python terminal to see the core philosophy. Key takeaways include:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Readability counts.