Introduction to Python: From Machine State to Mindset

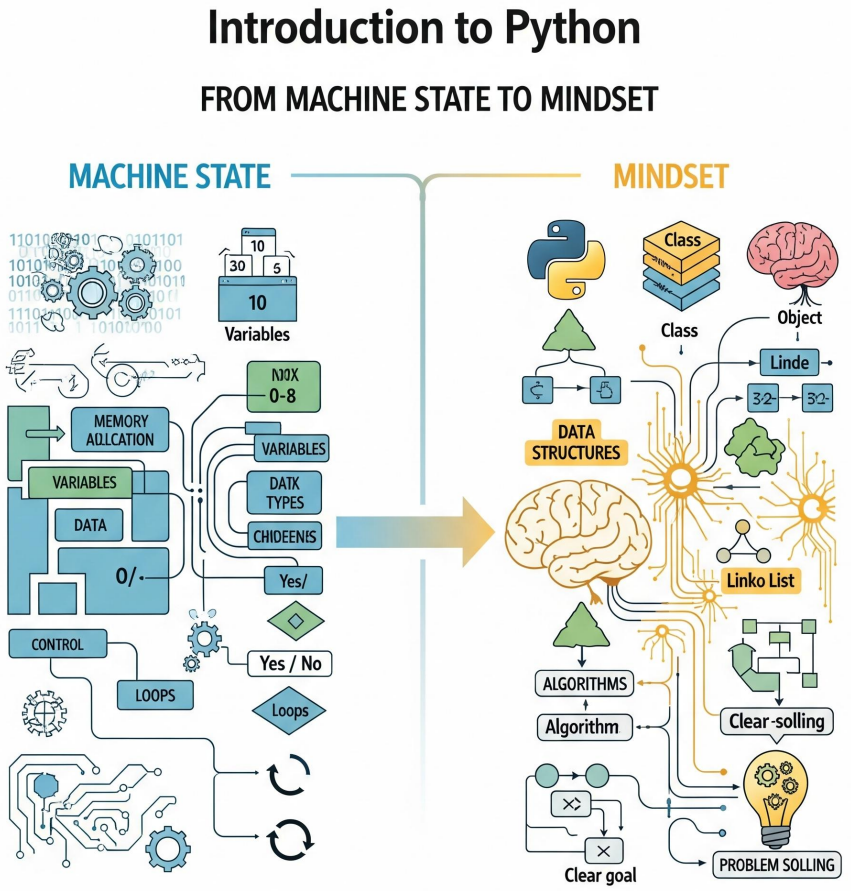
Discover how Python turns ideas into code and why it's ideal for beginners.

Abhishek Kr Singh
SERC, IIIT Hyderabad

Lecture - 1

Introduction to Python
Programming Course

" (Python) Programming Done Right "



Who are we?

Dr. Abhishek Kr Singh

Assistant Professor, Ph.D (TIFR, Mumbai)

Software Engineering Research Center, IIIT-H

Research Interests:Formal methods, Programming languages, Software engineering, and Trustworthy AI.

Dr. Sai Anirudh Karre

Guest Faculty, Ph.D (IIIT Hyderabad)

Software Engineering Research Centre (SERC), IIIT-H

Data Product Manager, Phenom Inc.

Research Interests: Software Engineering, Human Computer Interaction, Virtual Reality

Thank You!

Course Outline: https://sai11101989.github.io/cs6302_ssd_monsoon2025.html

INTRODUCTION

CURRENT AFFILIATION:
Assistant Proffessor, SERC, IIIT Hyderabad

RESEARCH DOMAINS:

- Formal Methods
- Semantics of Programming Languages
- Software Engineering
- Trustworthy AI

Abhiskek Kr Singh!

QUICK PAST AFFILIATIONS:

Senior Research Fellow
School of Computing, NUS
→ Postodacral Rescarcher
Tel Aviv University
→ Assistant Professor
BITS Pilani Goa Campus
→ PhD in CS,
TIFR Mumbai

INTRODUCTION

CURRENT AFFILIATION:
Assistant Proffessor, SERC, IIIT Hyderabad

This is my first Lecture at IIIT Hyderabad

Abhiskek Kr Singh!

QUICK PAST AFFILIATIONS:

Senior Research Fellow
School of Computing, NUS
→
Postodacral Rescarcher
Tel Aviv University
→
Assistant Professor
BITS Pilani Goa Campus
→
PhD in CS,
TIFR Mumbai

Python Programming Fundamentals

Done Right!

Initial State

No prior programming knowledge

Core Concepts

Learn syntax, semantics, and error types

Python's Role

Understand Python's modern coding role

Versatility

Explore Python's multi-paradigm nature

Proficient Coder

Able to write and validate code

Python Programming Fundamentals

Done Right!

Initial State

No prior programming knowledge

Core Concepts

Learn syntax, semantics, and error types

Python's Role

Understand Python's modern coding role

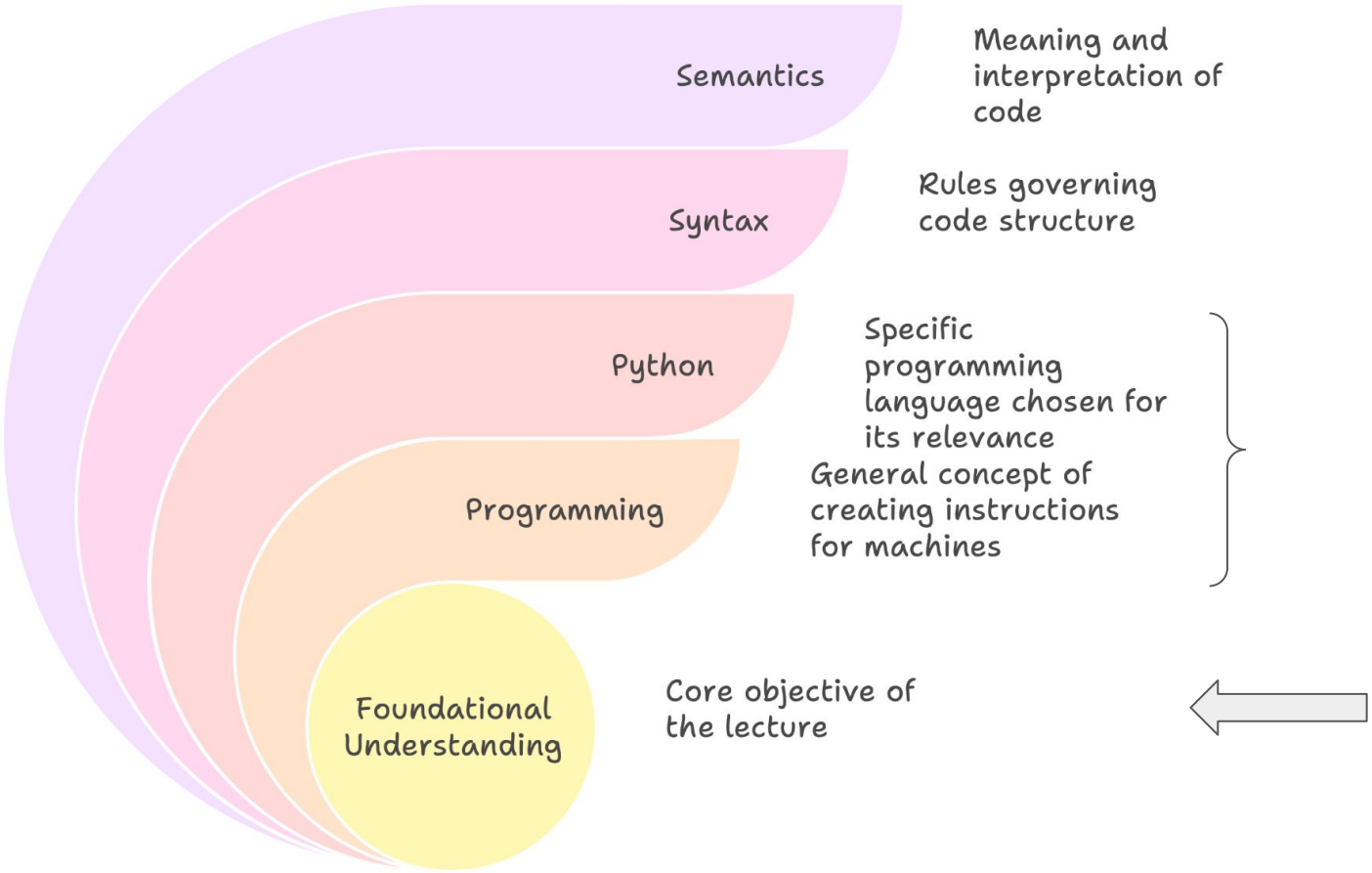
Versatility

Explore Python's multi-paradigm nature

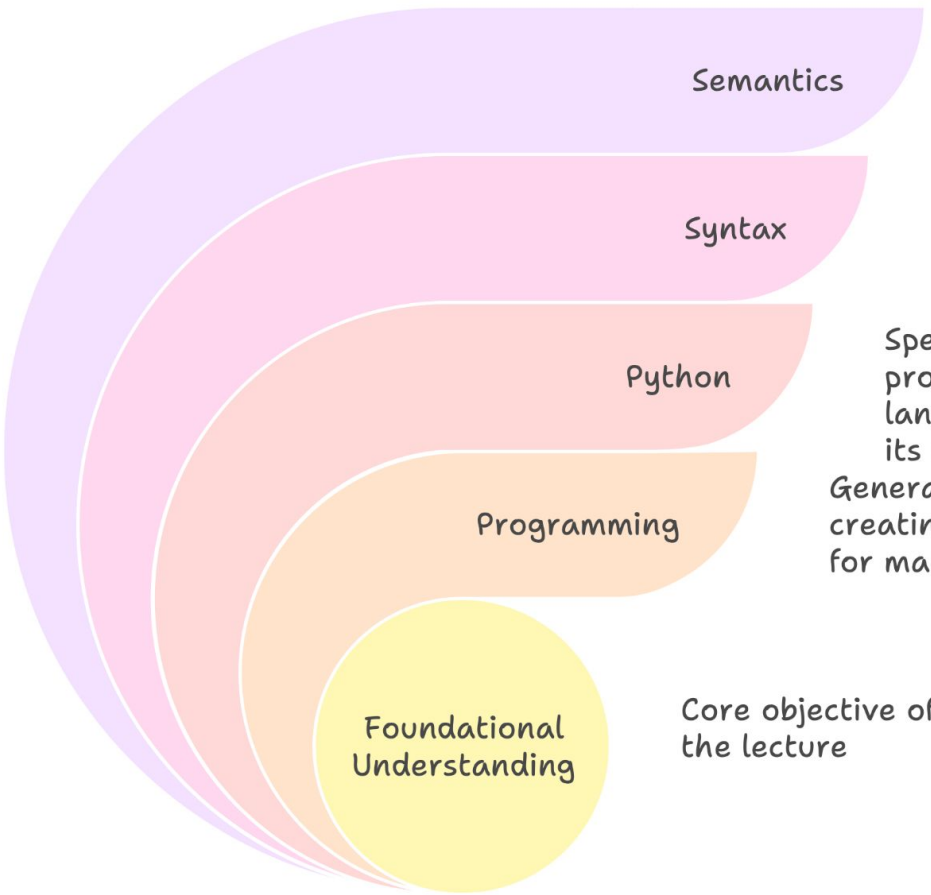
Proficient Coder

Able to write and validate code

Understanding Programming with Python



Understanding Programming with Python



Meaning and interpretation of code

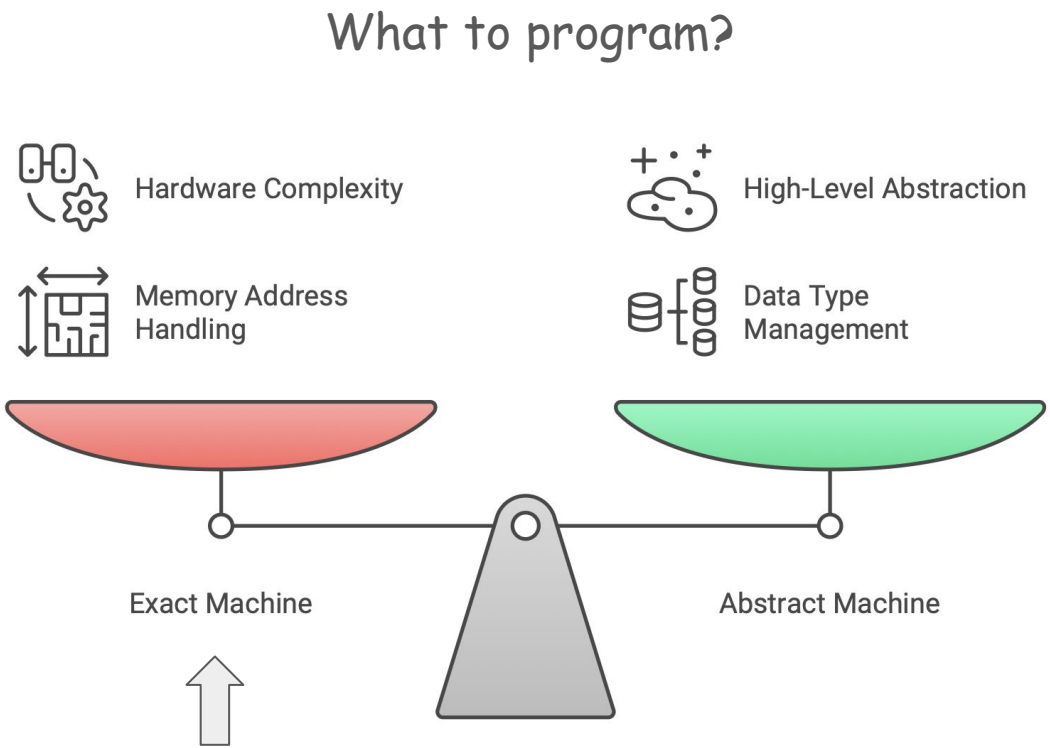
Rules governing code structure

Specific programming language chosen for its relevance

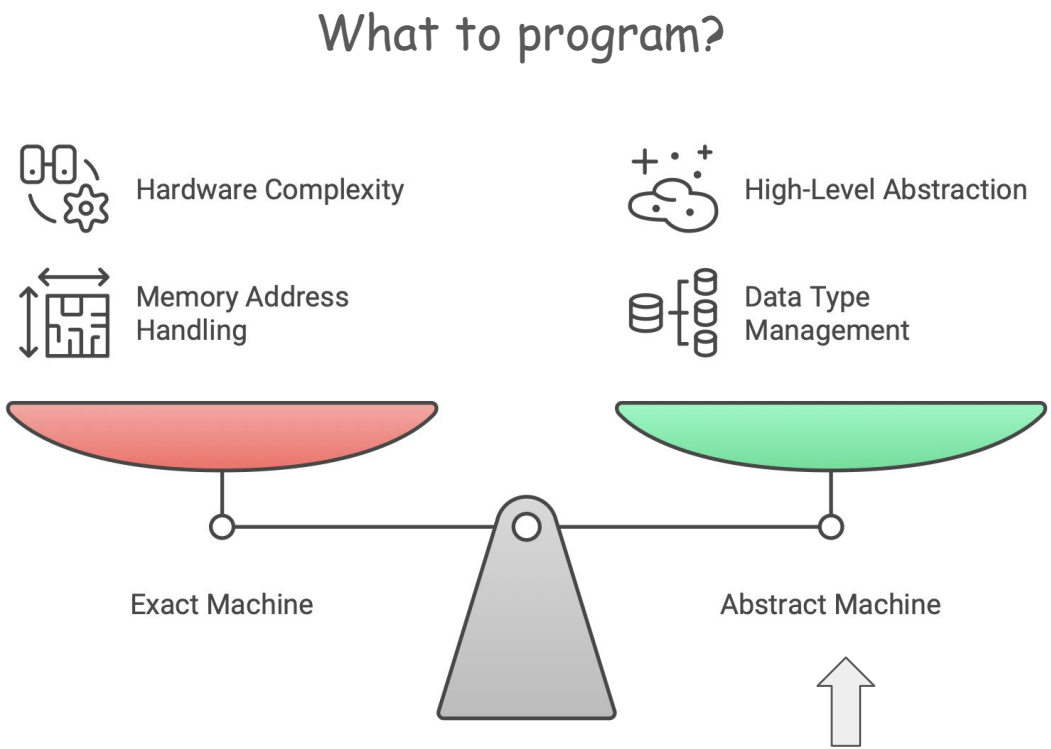
General concept of creating instructions for machines

Core objective of the lecture

Balancing Hardware Complexity with Abstraction in Python



Balancing Hardware Complexity with Abstraction in Python



Computer Program Abstraction

Abstract Machine

High-level programming environment

Data Structures

Organized data containers

Data Types

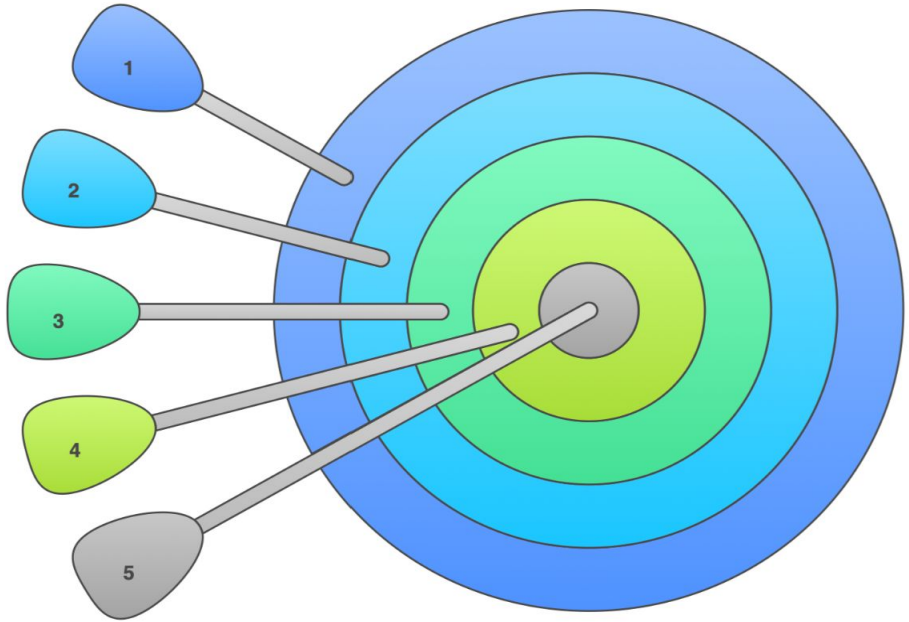
Categories of data

Variables

Named data containers

Exact Machine

Physical hardware components



Computer Program Abstraction

Abstract Machine

High-level programming environment

Data Structures

Organized data containers

Data Types

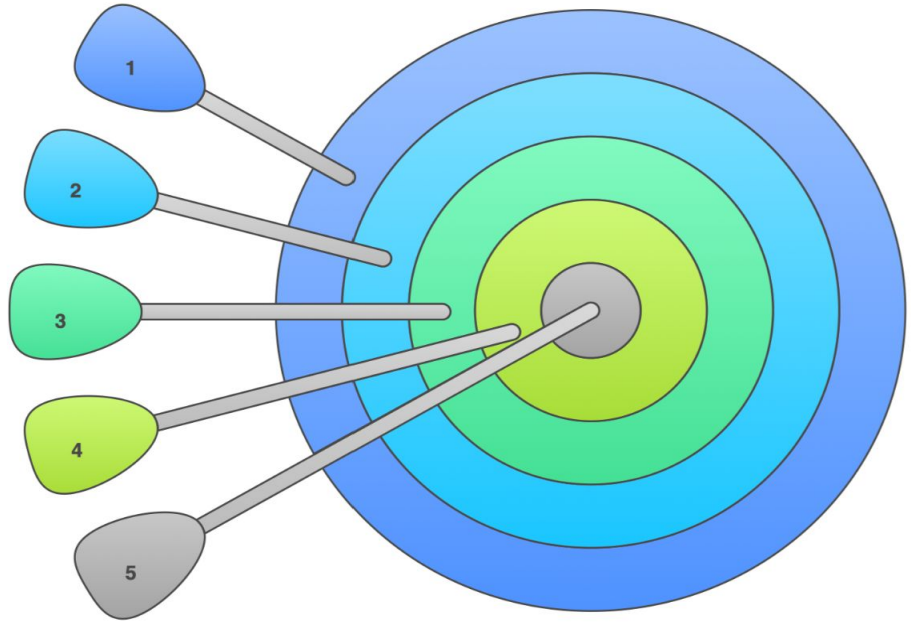
Categories of data

Variables

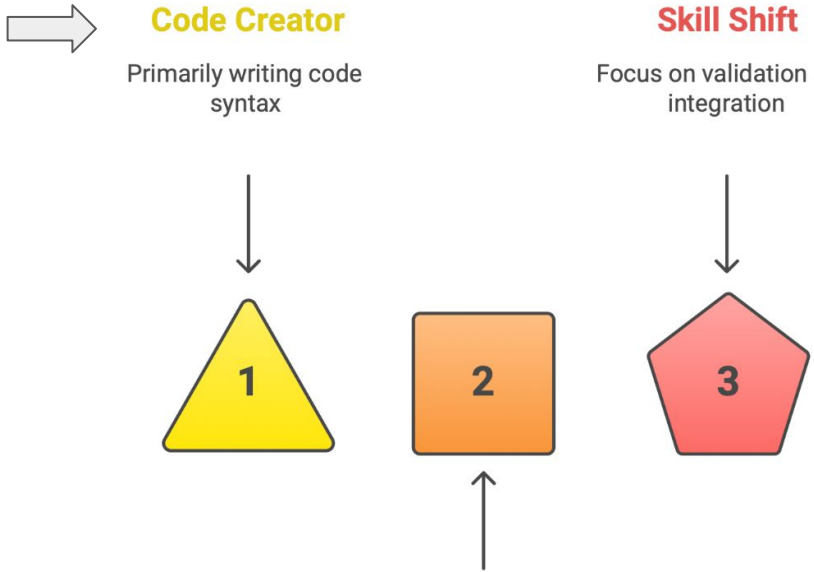
Named data containers

Exact Machine

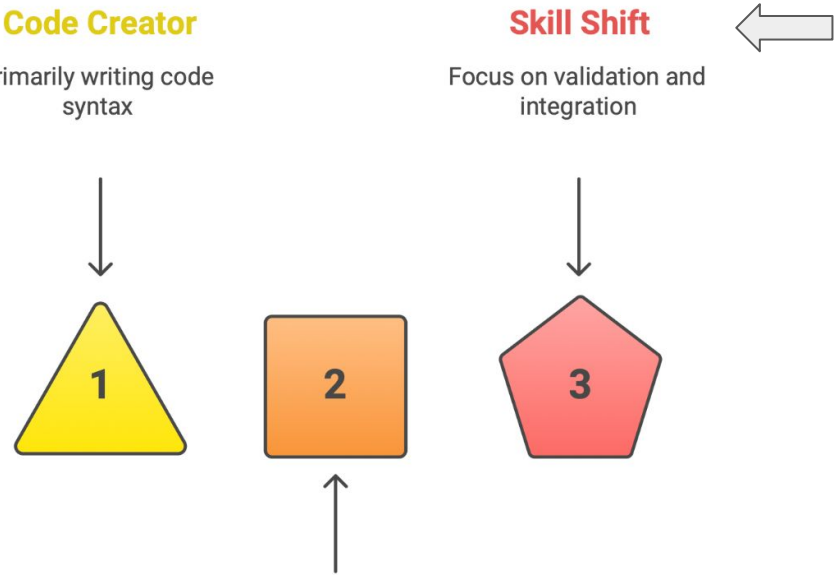
Physical hardware components



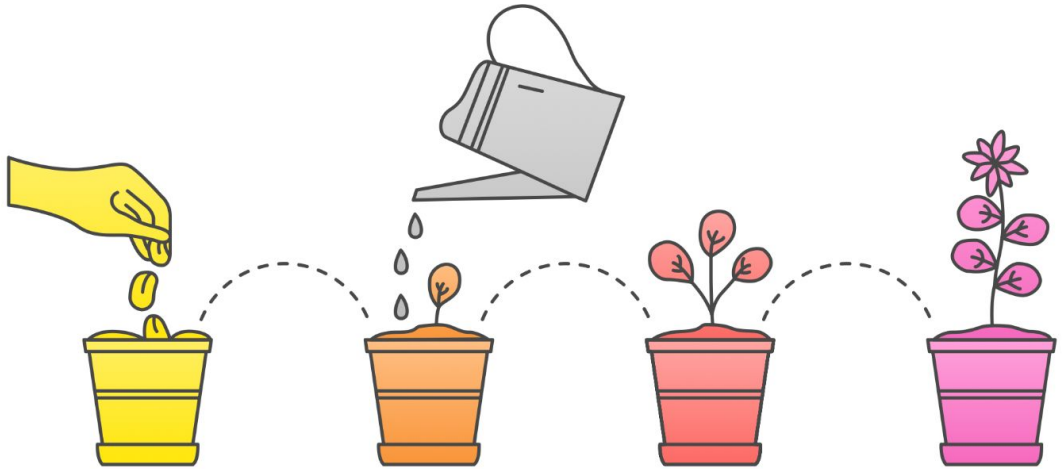
The Evolving Role of the Coder



The Evolving Role of the Coder



The Evolving Role of the Coder



Code Creator

Primarily writing code syntax

LLM Automation

Initial code generation is automated

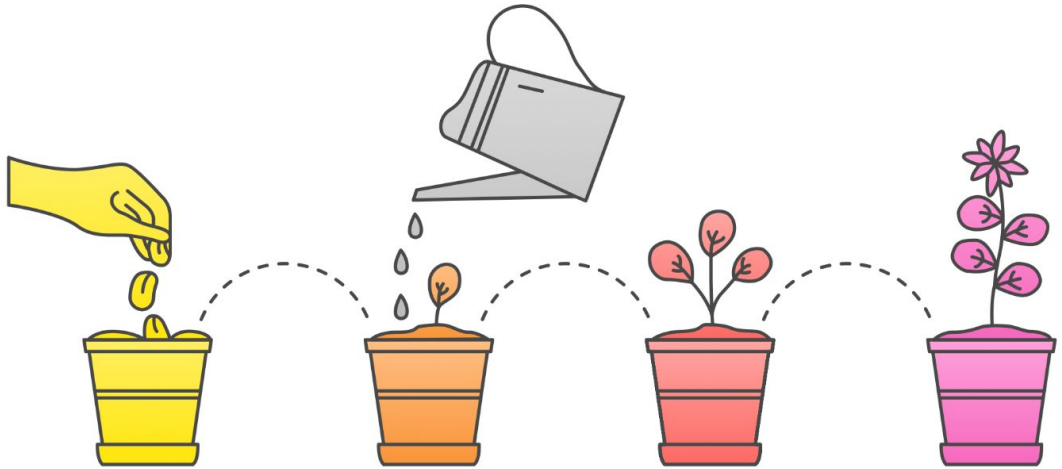
Skill Shift

Focus on validation and integration

Code Validator

Ensuring code quality and integration

The Evolving Role of the Coder



Code Creator

Primarily writing code syntax

LLM Automation

Initial code generation is automated

Skill Shift

Focus on validation and integration

Code Validator

Ensuring code quality and integration

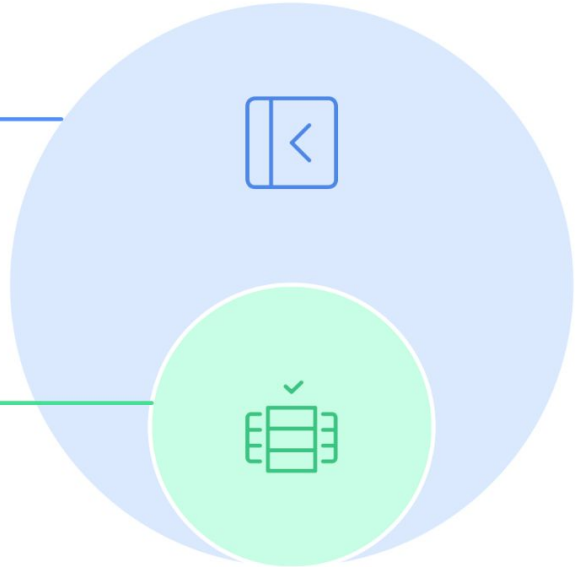
Understanding Program Structure

Syntax

Rules for correctly structured programs

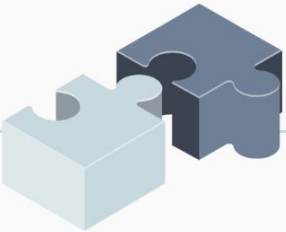
Semantics

The meaning and effect of code execution



Syntax: The Grammar That Governs Code Structure

A finite representation of an infinite set of valid programs



1

Syntax as Grammar

Syntax sets the rules for writing code, just like grammar governs sentence structure. Correct syntax ensures the code 'makes sense' to the computer.

2

Correct vs. Incorrect Syntax

Just as 'The cat sat on the mat' is a proper sentence and 'Mat the on sat cat the' isn't, Python requires strict rules like indentation, colons after if statements, and matching parentheses to avoid errors.

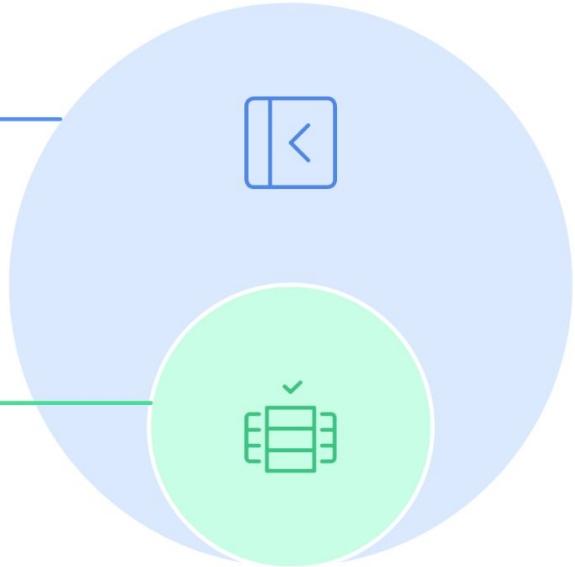
Understanding Program Structure

Syntax

Rules for correctly structured programs

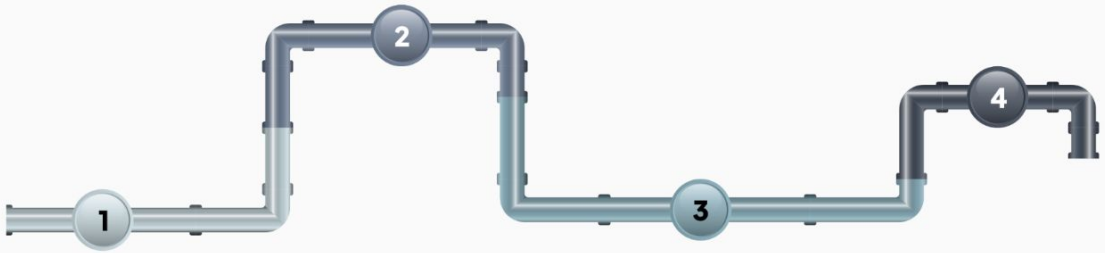
Semantics

The meaning and effect of code execution



Semantics: What Your Code Actually Does

Explore how step-by-step execution reveals code meaning and effects



x = 5

Variable x is assigned the value 5.

y = 10

Variable y is assigned the value 10.

z = x + y

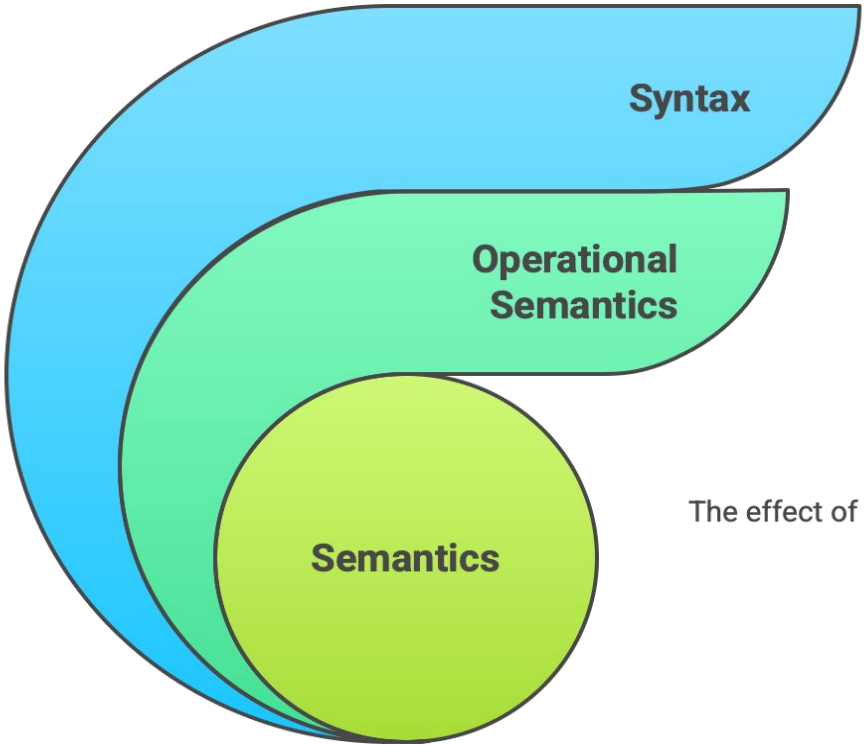
Variable z is calculated as the sum of x and y, resulting in 15.

print(z)

The value of z (15) is output to the console.

This step-by-step execution process is called **operational semantics**. It focuses on how each line changes the program's state, helping you understand the real effects of your code beyond its syntax.

Understanding Code Structure and Meaning

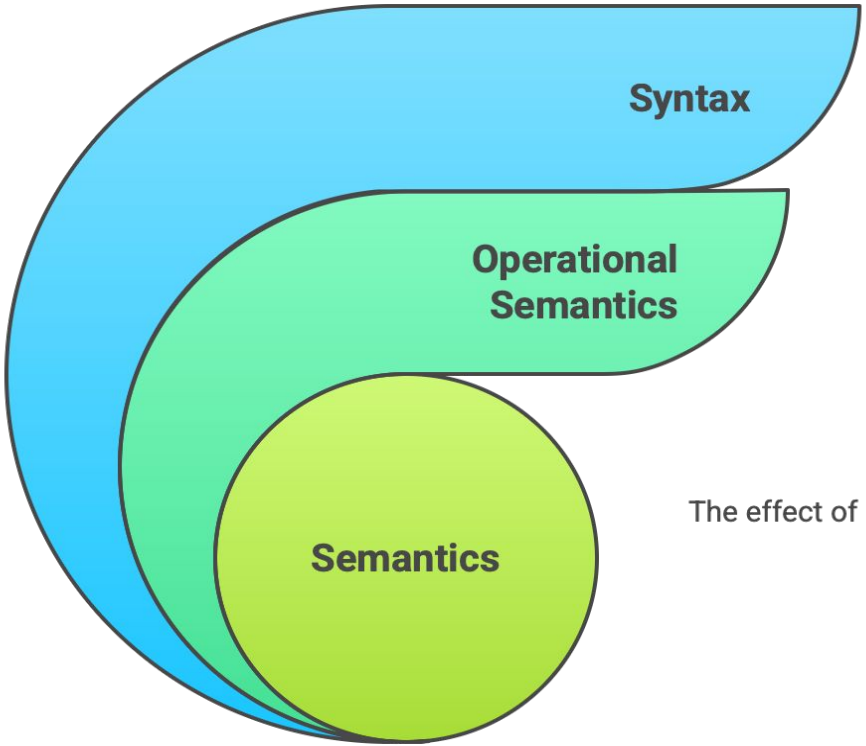


Rules for correct program structure

Meaning through execution observation

The effect of code execution

Understanding Code Structure and Meaning

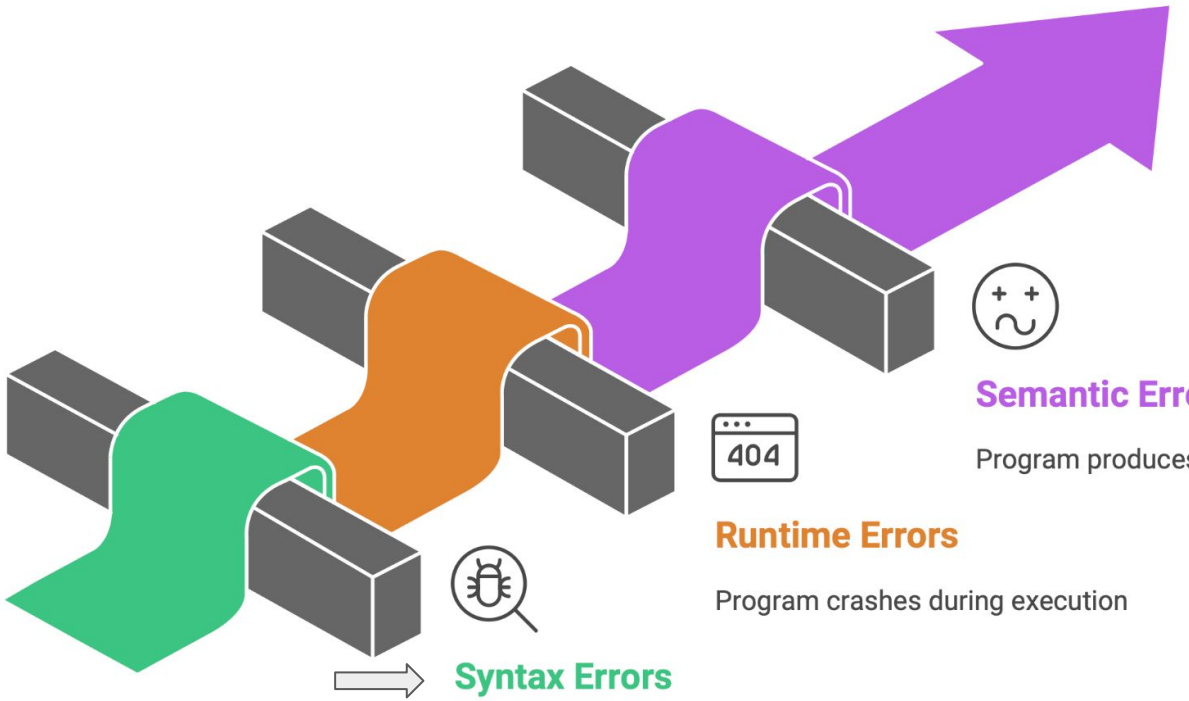


Rules for correct program structure

Meaning through execution observation

The effect of code execution

Types of Programming Errors



Semantic Errors

Program produces wrong result

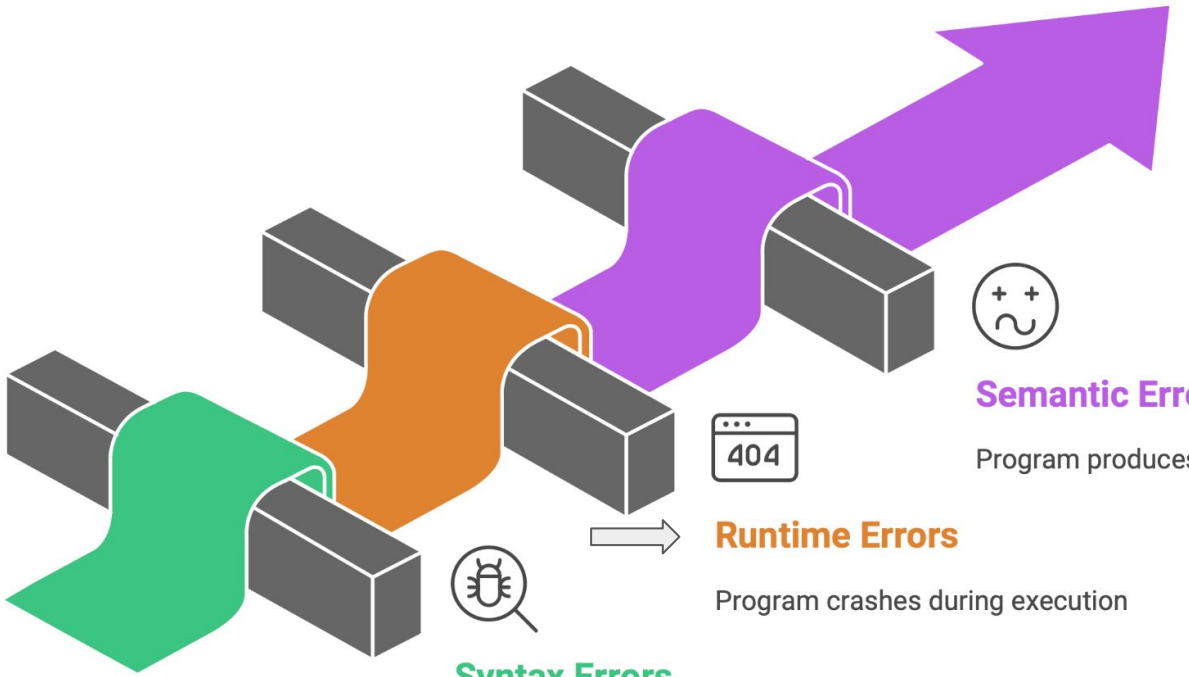
Runtime Errors

Program crashes during execution

Syntax Errors

Interpreter refuses to run code

Types of Programming Errors



Semantic Errors

Program produces wrong result

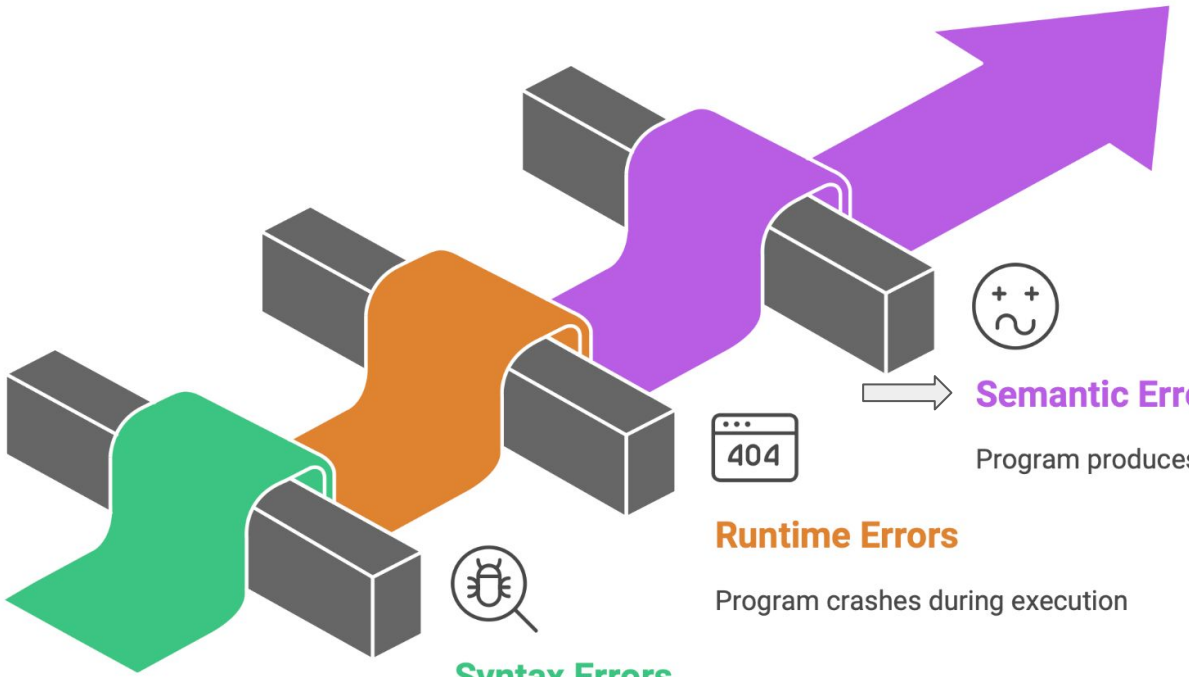
Runtime Errors

Program crashes during execution

Syntax Errors

Interpreter refuses to run code

Types of Programming Errors



Semantic Errors

Program produces wrong result

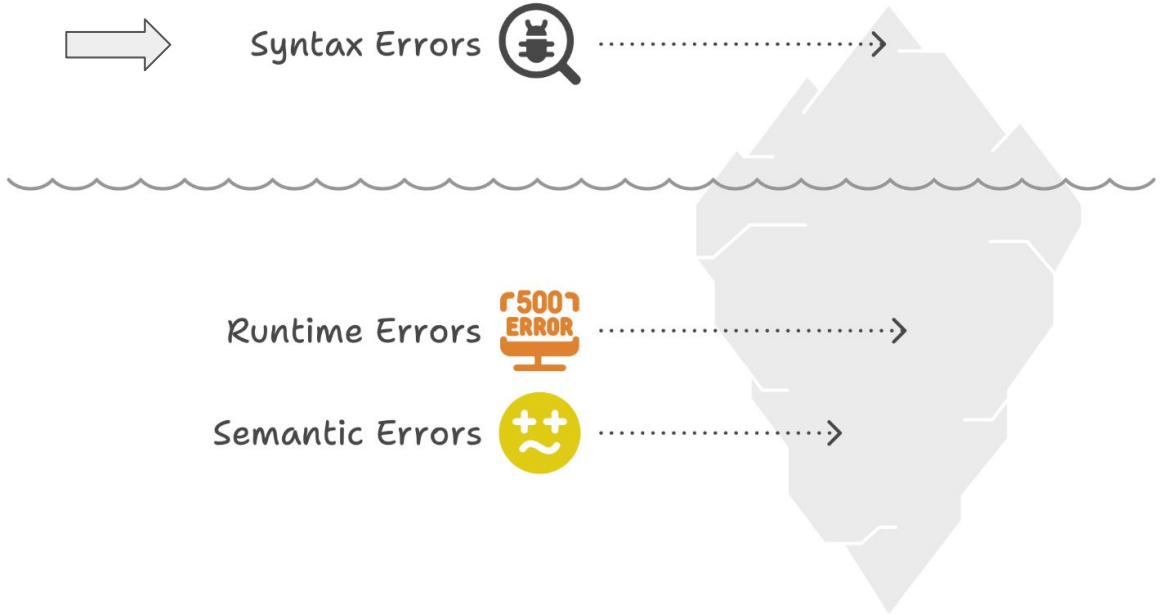
Runtime Errors

Program crashes during execution

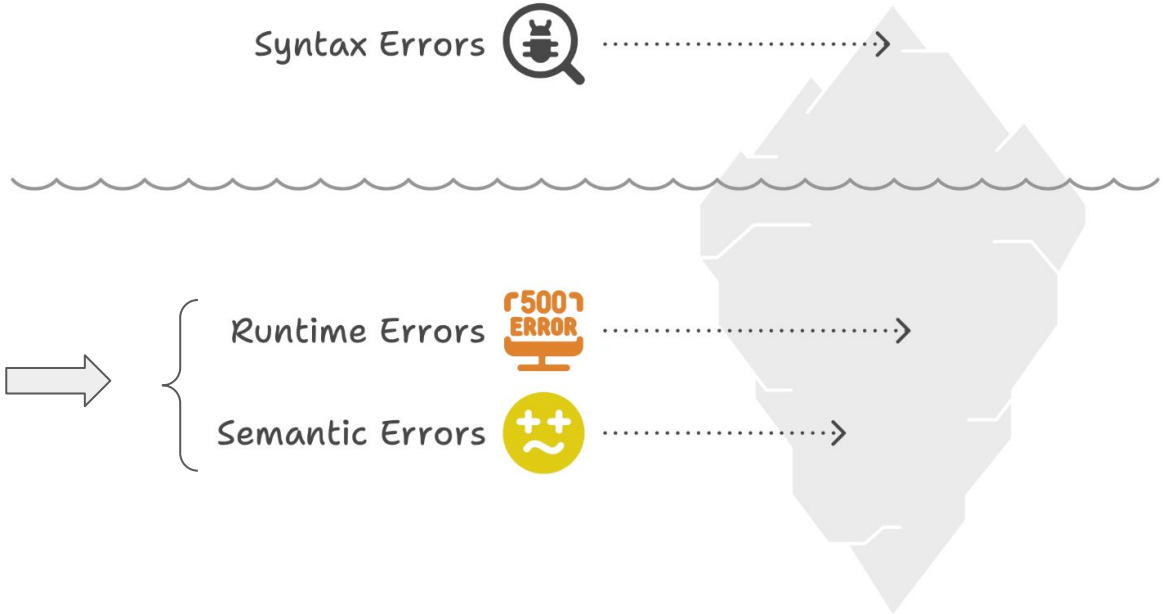
Syntax Errors

Interpreter refuses to run code

Unveiling the Depths of Programming Errors



Unveiling the Depths of Programming Errors



Python's Core Elements

Zen Principles

Emphasizes readability, simplicity, and explicitness in coding.

Multi-Paradigm Nature

Blends imperative, object-oriented, and functional programming styles.

Batteries-Included Ecosystem

Offers a comprehensive standard library and extensive third-party modules.

Python's Core Elements

Readability counts—write code that's easy to read and understand, improving collaboration and maintenance.

Simple is better than complex—choose straightforward solutions to keep code clean and efficient.

Explicit is better than implicit—make your code's intentions clear to avoid confusion and errors.

These guiding principles explain Python's popularity with beginners and experts, promoting maintainable and elegant code.

Zen Principles

Emphasizes readability, simplicity, and explicitness in coding.

Multi-Paradigm Nature

Blends imperative, object-oriented, and functional programming styles.

Batteries-Included Ecosystem

Offers a comprehensive standard library and extensive third-party modules.

Python's Core Elements

Programming Paradigms

Imperative: Write sequences of commands that change program state

Object-Oriented: Structure code with classes and objects

Functional: Use functions as first-class and higher-order objects

Zen Principles

Emphasizes readability, simplicity, and explicitness in coding.

Multi-Paradigm Nature

Blends imperative, object-oriented, and functional programming styles.

Batteries-Included Ecosystem

Offers a comprehensive standard library and extensive third-party modules.

Python's Core Elements

Popular Packages & Use Cases

numpy: Efficient numerical computing

pandas: Powerful data analysis

Extensive ecosystem supporting **data science** and **machine learning**

Zen Principles

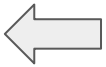
Emphasizes readability, simplicity, and explicitness in coding.

Multi-Paradigm Nature

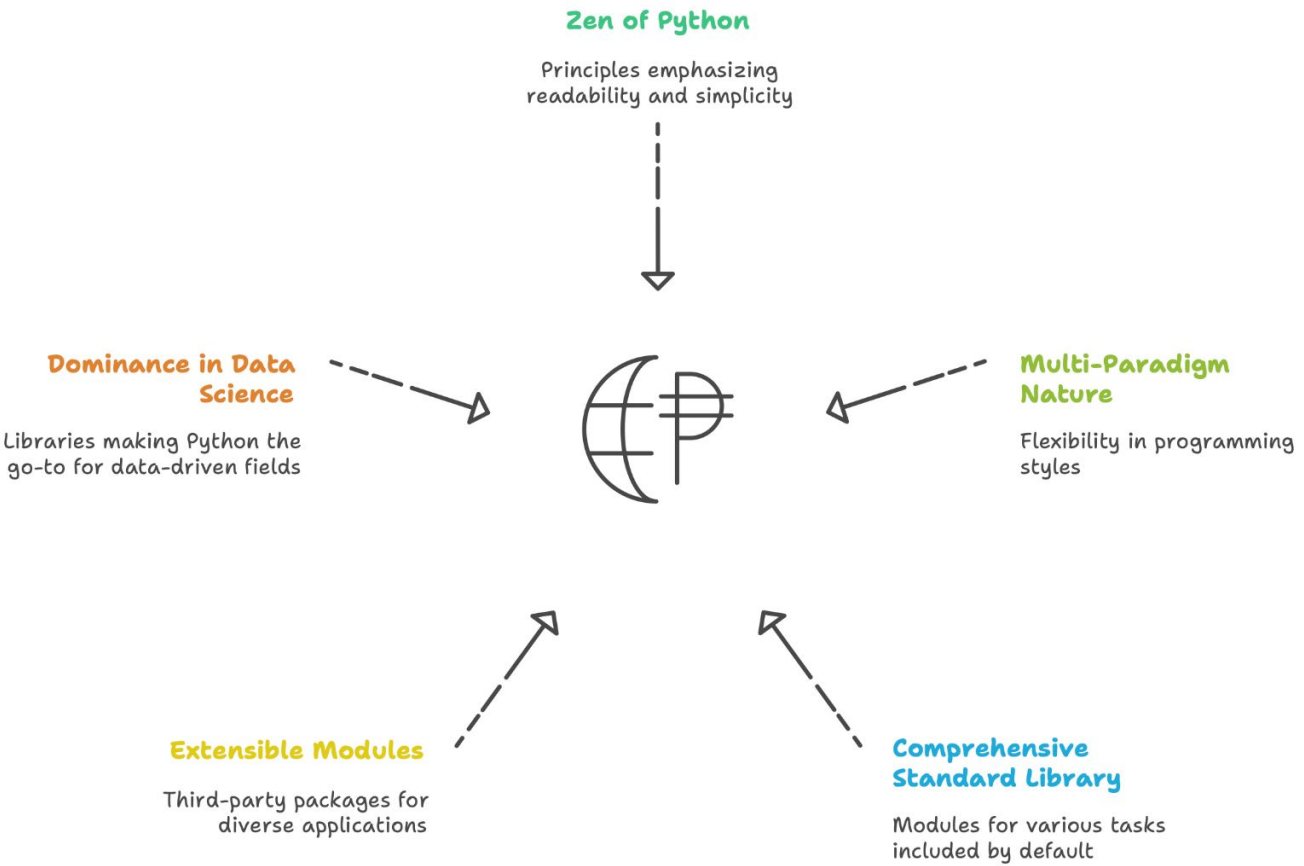
Blends imperative, object-oriented, and functional programming styles.

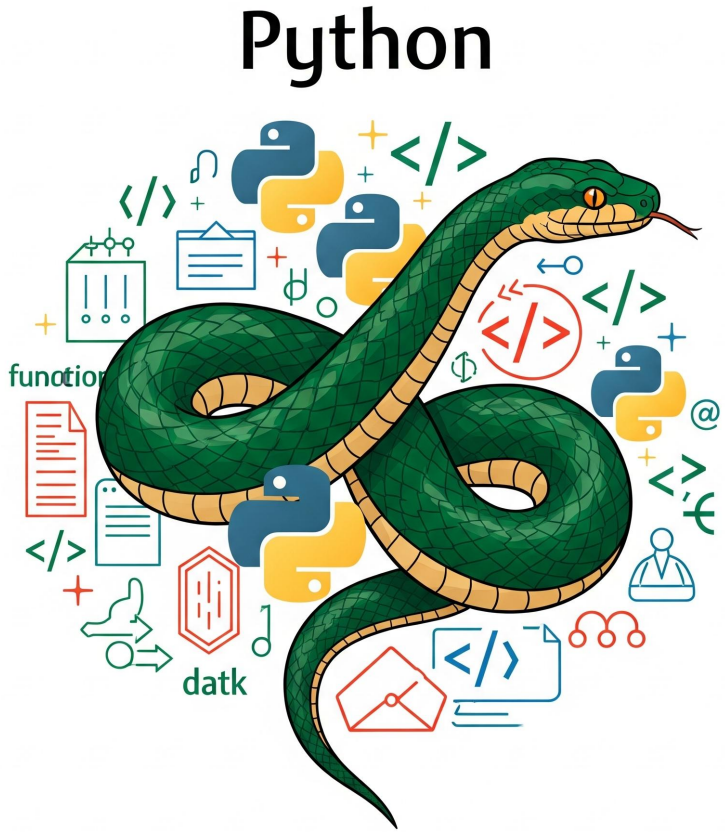
Batteries-Included Ecosystem

Offers a comprehensive standard library and extensive third-party modules.



Factors Contributing to Python's Popularity



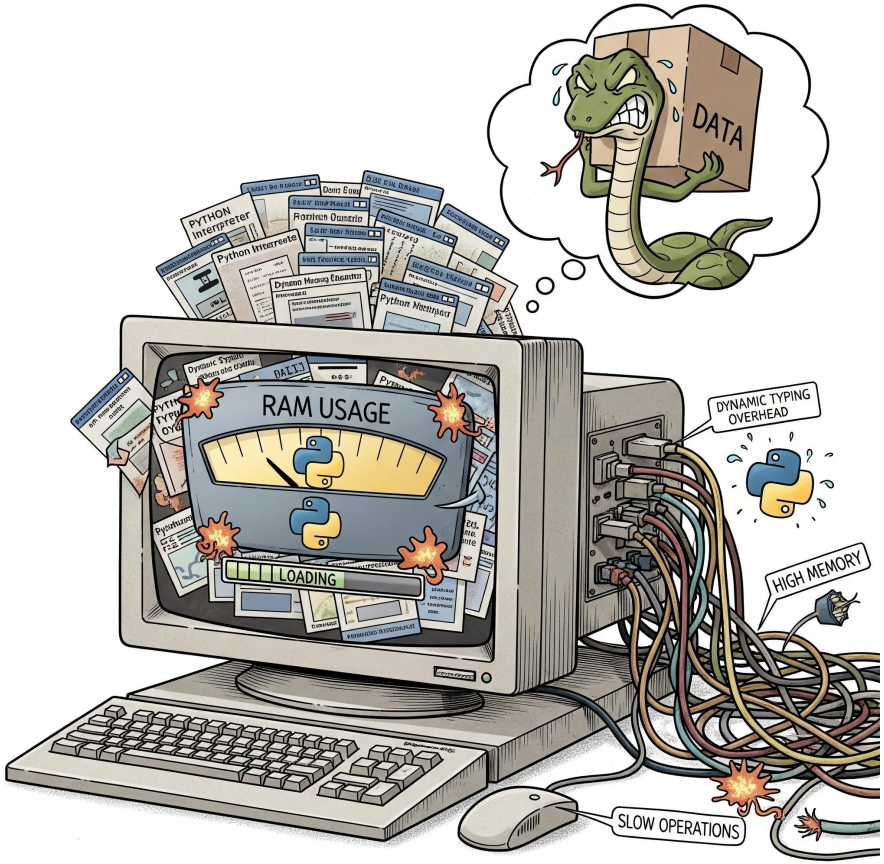


Key Features

- Python offers a simple and readable syntax
- It supports multiple programming paradigms
- Python has an extensive standard library
- It is highly portable across different platforms

Python's Pitfalls

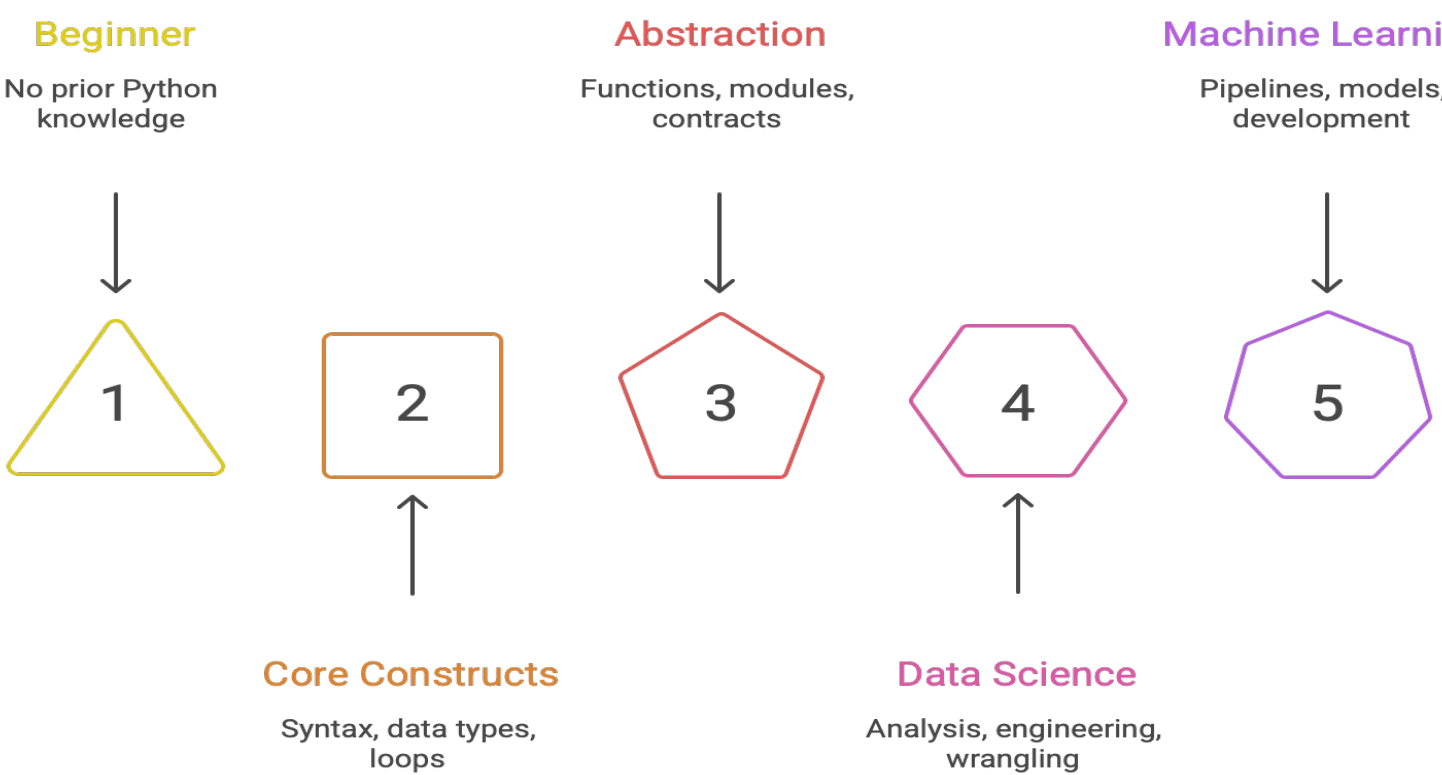
- Python consumes significantly **more (76 times) energy** than C
- **Memory usage** in Python is higher (**2.5 times**) than in C
- Interpreted nature makes Python **inherently slower**
- Dynamic typing prevents **compiler optimizations** in Python



Overcoming Python Disadvantages

- Clever Python use mitigates disadvantages
- **Optional typing** helps reduce effects
- Numba and PyPy offer **just-in-time compilation**
- **Static compilation** improves performance
- **Concurrency** and **efficient data structures** are key
- Graceful **error handling** is essential

Python Programming Journey



Python Programming Course Sequence

Thank You!