Introduction to Object Oriented Programming (OOP) & concepts in Python

Deepan Pitchairaj

NIMHANS, India

29/06/2024

Programming

Definition

Programming is the process of writing **a set of instructions** that can manipulate the microprocessor inputs in a certain way to achieve a specific final output

Imagine it like this:

- you're the chef, and the computer is your kitchen assistant. You need to tell it exactly how to prepare a dish (or perform a task) step by step, leaving no room for confusion.
- Just like a recipe guides you in baking a cake, programming provides precise instructions for the computer to follow and accomplish useful tasks.

So, in a nutshell, programming is all about communicating with computers to make them do what we want!

Behind the stage

Let's try to understand what's happening in the background

- Data isn't just a treasure trove for statisticians and data scientists; it's also a siren call for the computer science realm.
- All Things Are Number certainly in the computer world
- Input - Process - Output (IPO)

Algorithms + Data structures = Programs
by Wirth, Niklaus

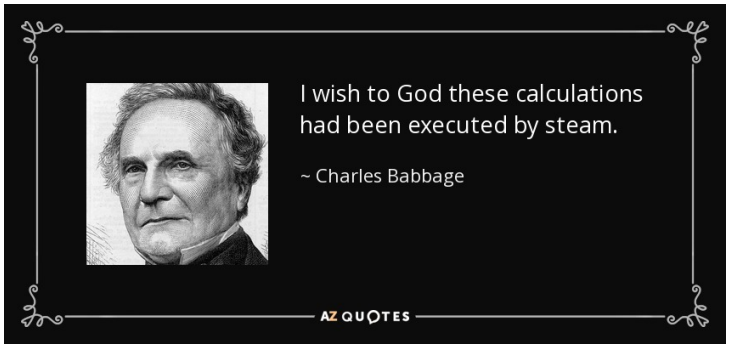


Figure 1: Charles Babbage

Programming paradigms - "styles"

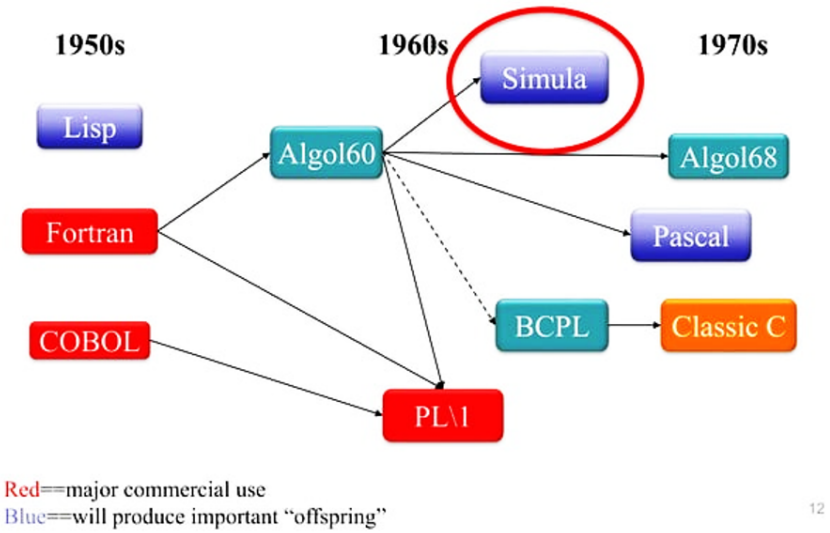
Programming paradigms are **different ways or styles** in which a given program or programming language can be organized. Each paradigm consists of certain structures, features, and opinions about how common programming problems should be tackled.

Popular Programming paradigms:

- Imperative Programming
- Procedural Programming
- Functional Programming
- Declarative Programming
- **Object Oriented Programming (OOP)**

OOP - Origin

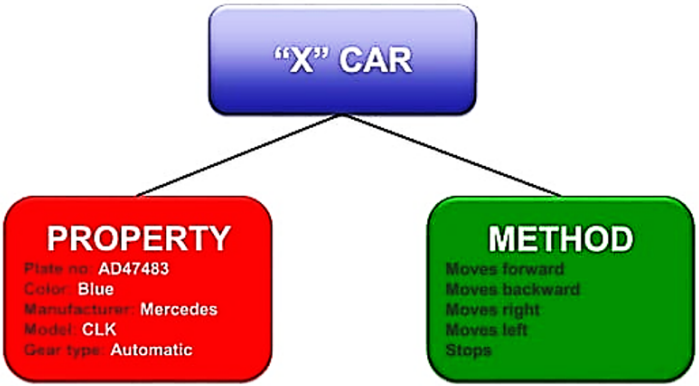
Early Programming Languages



- Simula is considered the first object-oriented programming (OOP) language, created in the 1960s by Norwegian computer scientists Ole-Johan Dahl and Kristen Nygaard.
- Simula aimed primarily at simulations modeling real-world systems from a user's point of view

Classes & Objects

Think of the **Attributes** and **Functions** of a car (Object)



Classes & Objects

- **Class:** A class is a user-defined data type. It consists of data members and member functions, which can be accessed and used by creating an instance of that class. It represents the set of properties or methods that are common to all objects of one type.
- **Object:** An Object is **an instance of a Class.** When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. An object has an identity, state, and behavior. Each object contains data and code to manipulate the data. .
- A class is like **a blueprint for an object.**
  An Object is like **a product of the blueprint.**

OOP - Example

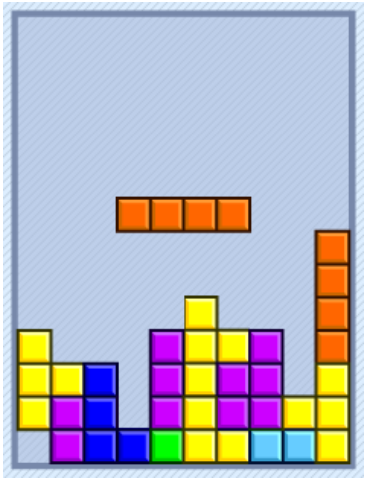


Figure 2: Tetris

In Tetris, really **the only object is a Tetromino**. It has states of:

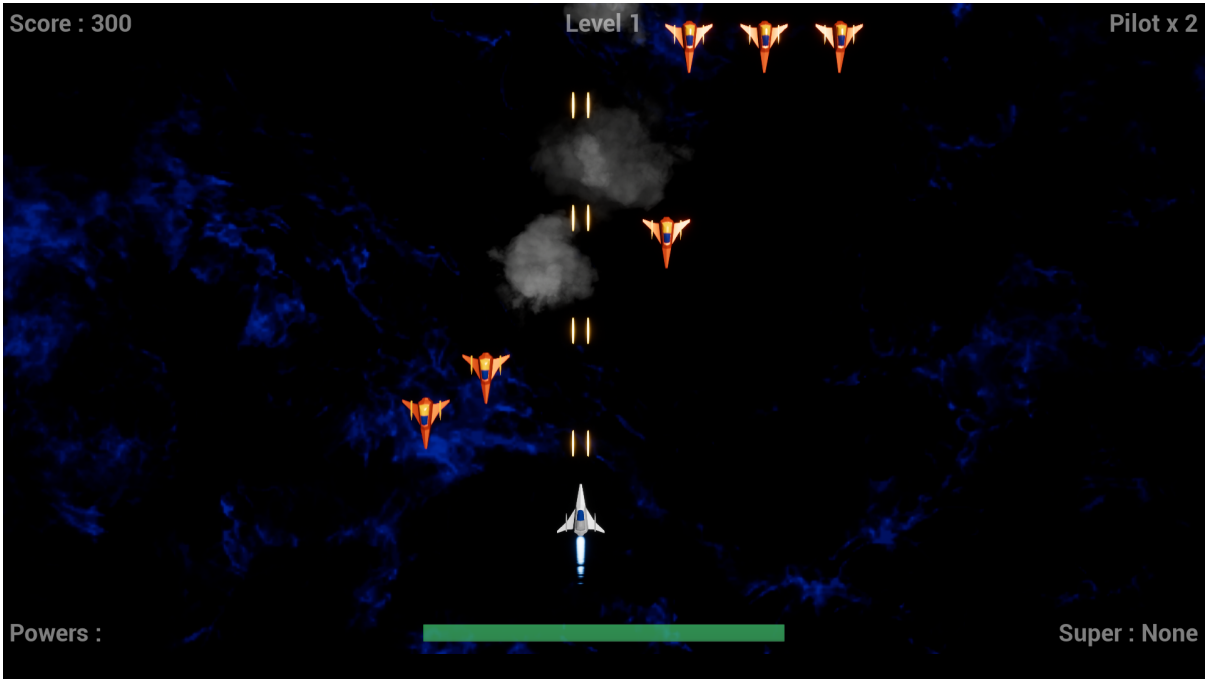
- rotation (in 90 degree units)
- shape
- color

and behaviors of:

- falling
- moving (sideways)
- rotating

Time to Think

Let's build a logic for this game!



Advantages & Disadvantages

Advantages

- Code Maintenance: Changes and bug fixes can be made to specific objects or classes without affecting other parts of the system, reducing errors and improving debugging.
- Code Reusability: OOP encourages the development of reusable code elements, saving time and improving system reliability.
- Better Problem Solving: OOP models real-world systems, allowing developers to create intuitive solutions that closely mimic real-world circumstances.

Disadvantages

- Steeper learning curve: OOP introduces complex concepts like classes, objects, inheritance, and polymorphism, making it harder for new programmers to grasp and apply them effectively.
- Increased complexity: OOP's emphasis on modularity and code organization can make larger projects more challenging to understand and maintain.
- Performance overhead: OOP languages often have a performance cost compared to procedural languages due to the additional abstraction layers introduced by objects and encapsulation.

Top OOP languages