Python Best Practices

Ines Ben Amor

What is OO?

Code always grows.

Code always changes.

Refactoring

https://www.youtube.com/watch?v=EmTKMRcVqI8 - 3:02 to 13:08

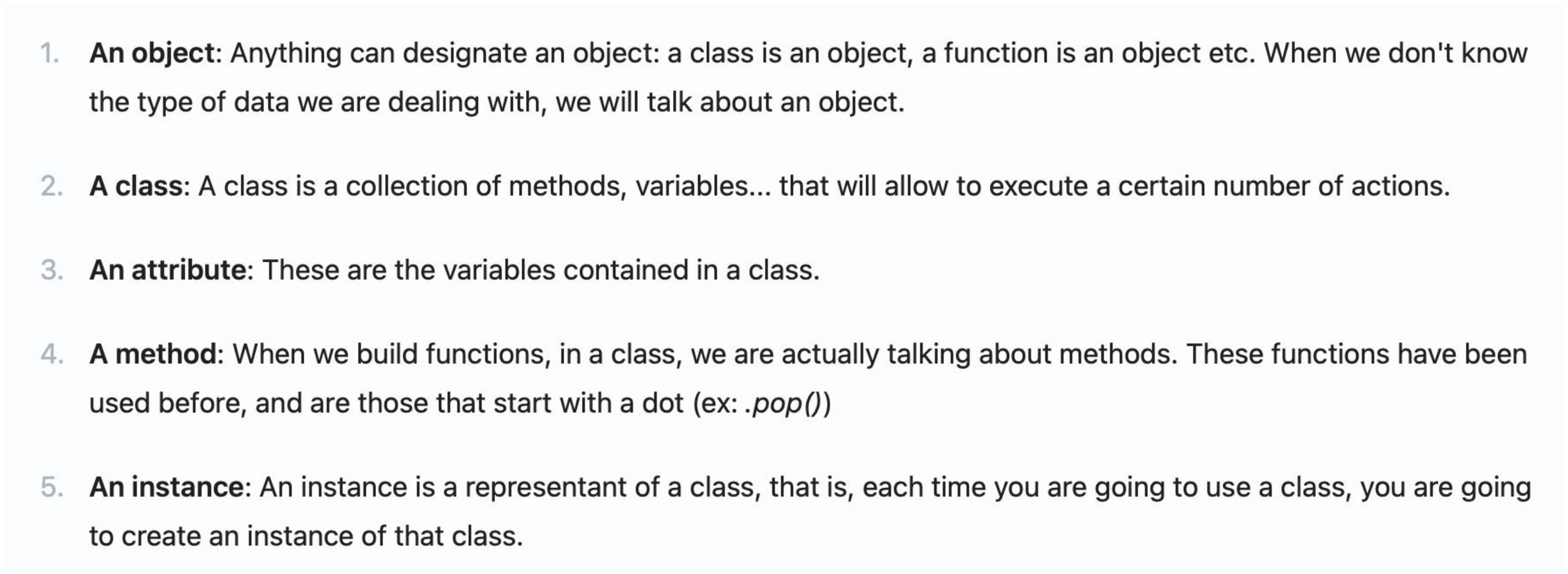
Why OOP?

- It makes it so much easier to maintain and update existing code.
- Provides code reusability.
- It is best for real-life problems.
- Provides a modular structure.
- It is easy to debug.
- In comparison to others (functional or structural), it is much faster and more efficient to use.

The 4 pillars of OOP

1. Inheritance
2. Encapsulation
3. Abstraction
4. Polymorphism

The OOP Structure



1. **An object**: Anything can designate an object: a class is an object, a function is an object etc. When we don't know the type of data we are dealing with, we will talk about an object.

2. **A class**: A class is a collection of methods, variables... that will allow to execute a certain number of actions.

3. **An attribute**: These are the variables contained in a class.

4. **A method**: When we build functions, in a class, we are actually talking about methods. These functions have been used before, and are those that start with a dot (ex: *.pop()*)

5. **An instance**: An instance is a representant of a class, that is, each time you are going to use a class, you are going to create an instance of that class.

Attributes / Methods

Cars have ***data***: like number of wheels, number of doors, seating capacity

Data → Attributes

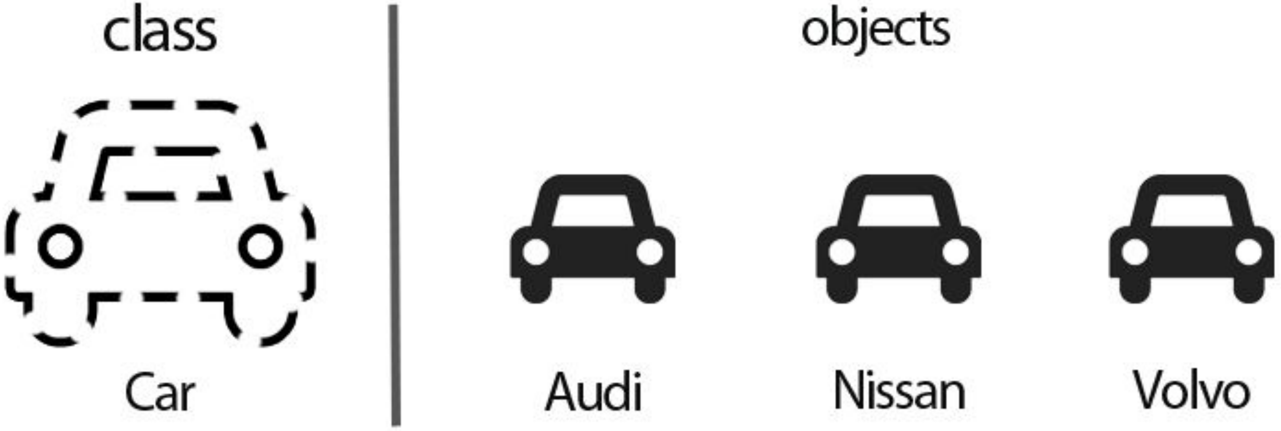
Cars have ***behavior***: accelerate, stop, show how much fuel is missing and so many other.

Behavior → Methods

Class

A **Class** is the blueprint from which individual objects are created.

Class /Instance / Object



instance

What is self?

https://www.youtube.com/watch?v=oaiQ5hYKHTE