The fundamental principles of Object-Oriented Programming (OOP) are:

1. **Encapsulation**: This principle refers to the bundling of data and methods (functions) that operate on that data within a single class. The data is hidden from the outside world and can only be accessed through the methods of the class. This helps to maintain data integrity and reduces the risk of unintended changes.

2. **Inheritance**: This principle allows a subclass to inherit the properties and methods of its parent class. This means that the subclass can use the code of its parent class without having to write it all over again. This also helps to create a hierarchical structure of classes, where each class inherits from its parent.

3. **Polymorphism**: This principle refers to the ability of objects of different classes to be treated as if they are of the same class. This means that objects can be passed as arguments to a function or method without knowing their specific class. Polymorphism is achieved through inheritance and interfaces.

4. **Abstraction**: This principle refers to the ability to create a simplified representation of a real-world object, process or system. Abstraction allows developers to focus on the essential features of an object or system, while ignoring irrelevant details. This helps to reduce complexity and increase code reusability.

These four principles are often referred to as the SOLID principles of OOP, and they are considered to be the foundation of modern software development.

I know we are just briefly covering them today, but trust me you will see them again. So I figured I'd familiarize them to you early on, so you can catch on to these things when you're writing your own code.

So you want to talk about Magic methods

Method	Desc
__init__	Initializer, creates a new instance of a class
__str__	String representation, meant for readable printing
__repr__	Machine readable representation of a type, useful for debugging and representing all data of a class object
__add__, __sub__, __mul__	Overload math operations for classes,
__lt/le__, __eq/ne__, __gt/ge__	Overload comparison operators for classes

Magic methods are what allow for ‘syntactic sugar’ such as being able to cast between types, being able to use +, -, >, ==, etc between specific objects, which is extremely useful for object oriented programming especially when dealing with objects with several different attribute types

For example if two objects have 2 parameters, a string and an int you would not want to add the strings , because that would just concatenate them, rather when you invoke + you would want it to add the integer values of the objects and return that, this is what you would specify in your __add__ method