CHAPTER-09: CLASSES & INHERITANCE

CREATING CLASSES

- We learned how to define our own classes to store information and behavior, rather than using built-in data structures like lists and dictionaries.

- Classes have attributes, which are variables that store information.

- Classes have methods, which are functions that define the behavior of the class.

- The `__init__ () ` method is a special method used to create instances of a class with the desired attributes.

MODIFYING ATTRIBUTES

- We can modify the attributes of an instance directly, by accessing them using dot notation.

- We can also write methods that modify the attributes of an instance, which can include validation or other logic.

INHERITANCE

- Inheritance allows us to create new classes based on existing ones, inheriting their attributes and methods.

- Child classes can add or override attributes and methods from their parent classes.

- Inheritance simplifies creating related classes by avoiding duplication of common code.

COMPOSITION

- Composition involves using instances of one class as attributes in another class.

- This keeps each class focused on a specific set of responsibilities and avoids creating overly complex classes.

ORGANIZING CLASSES

- Classes can be stored in modules, which are Python files containing definitions for classes, functions, and variables.

- Importing classes from modules into the files where they'll be used helps keep projects organized.

- We saw an example using the `OrderedDict` class from the `collections` module.

STYLING CONVENTIONS

- Python has conventions for styling classes, such as using CamelCase for class names.
- Following these conventions makes our code more readable and maintainable.