Custom Classes

October 8, 2025

Creating a Basic Custom Class

Objective: To understand how to define a custom class, including attributes and methods, to create objects.

Theory: A **class** is a blueprint for creating objects. It defines the properties (called **attributes**) and behaviors (called **methods**) that all objects of that type will have. A custom class allows you to create your own data types that are tailored to your program's needs.

The __init__ method is a special function called a **constructor**. It's automatically executed when a new object of the class is created. Its purpose is to initialize the object's attributes.

The self parameter in a method definition refers to the instance of the object itself. It allows you to access and modify the object's attributes from within its methods.

Program:

```
# A class named 'Book' to represent a book object.
class Book:
    # The constructor method, used to initialize the object's attributes.
    def __init__(self, title, author, pages):
        self.title = title
        self.author = author
        self.pages = pages
        print(f"A new book '{self.title}' has been created.")

    # A method to display information about the book.
    def display_info(self):
        """Prints the title, author, and number of pages."""
        print(f"Title: {self.title}")
        print(f"Author: {self.author}")
        print(f"Pages: {self.pages}")

# Creating two objects (instances) of the Book class.
```

```
book1 = Book("The Hitchhiker's Guide to the Galaxy", "Douglas Adams", 192)
book2 = Book("Dune", "Frank Herbert", 412)


# Calling the 'display_info' method for each object.
book1.display_info()
print("-" * 20)
book2.display_info()
```

Output:

```
A new book 'The Hitchhiker's Guide to the Galaxy' has been created.
A new book 'Dune' has been created.
Title: The Hitchhiker's Guide to the Galaxy
Author: Douglas Adams
Pages: 192
--------------------
Title: Dune
Author: Frank Herbert
Pages: 412
```

Class and Instance Attributes

Objective: To differentiate between class-level attributes and instance-level attributes and understand their usage.

Theory: Instance attributes are unique to each object and are defined inside the constructor (__init__) using the self keyword.

Class attributes are shared by all objects of the class. They are defined directly within the class body but outside of any methods. They are useful for storing constants or data that is common to all instances.

Program:

```
class Car:
    # A class attribute shared by all Car objects.
```

```
    number_of_wheels = 4

    def __init__(self, make, color):
        # Instance attributes, unique to each object.
        self.make = make
        self.color = color

    def display_info(self):
        print(f"This is a {self.color} {self.make} with {self.number_of_wheels} wheels.")

# Create two different car objects.
car1 = Car("Toyota", "blue")
car2 = Car("Ford", "red")

# Accessing instance attributes.
print(f"Car 1 make: {car1.make}")
print(f"Car 2 make: {car2.make}")

# Accessing the class attribute via the object or the class itself.
print(f"All cars have {car1.number_of_wheels} wheels.")
print(f"Class attribute directly: {Car.number_of_wheels}")

# Changing the class attribute affects all instances.
Car.number_of_wheels = 3
print("\nNumber of wheels has been changed.")

car1.display_info()
car2.display_info()
```

Output:

```
Car 1 make: Toyota
Car 2 make: Ford
All cars have 4 wheels.
Class attribute directly: 4


Number of wheels has been changed.
This is a blue Toyota with 3 wheels.
This is a red Ford with 3 wheels.
```