Session 01:

*# Create a student class with properities (id, name, age) and function sayname() that displays "Welcome + the name" then instanciate two objects and use the to print all properties and methods of the class*

Output:

A screenshot of a computer program

Description automatically generated

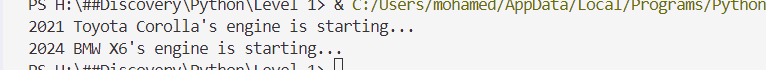
Session 02:

*# Create the Vehicle class with a constructor and a method  // properities: make, model, year // method---prints that "{make} + {model} + {year} the engine has started"*

*# Create a dictionary that hold the same properties of the viechle (make, model, year)*

*# Creat 2 ojects from the Vehichle class*

output



Session 03:

*# Create a class with encapsulation then: create 2 other classes that inherits the super class //*

*# Creating instance from the two Childs and use them.*

Output:

A screenshot of a computer code

Description automatically generated

SESSION 04:

 **Create a Vehicle class** with the following:

* **Private Attribute:** \_speed (integer) to represent the vehicle's speed.
* **Method:** move() to print a message that the vehicle is moving at a certain speed.
* **Getter and Setter:** get\_speed() to retrieve the speed and set\_speed(speed) to set the speed.

 **Create a Car class** that inherits from Vehicle:

* **Method:** move() to print a specific message like "The car is driving at *speed* km/h."

 **Create a Bicycle class** that also inherits from Vehicle:

* **Method:** move() to print a specific message like "The bicycle is pedaling at *speed* km/h."

 **Polymorphism in Action:**

* Create instances of Car and Bicycle, set their speeds, and call the move() method on both. Show how the same move() method behaves differently depending on the vehicle.

 **Abstraction:**

* Discuss how the Vehicle class hides the complexity of setting and getting speed, and how each vehicle can focus on its specific way of moving.

Output:

A close up of a text

Description automatically generated

SESSION 05:

 **Create an Animal class** with basic attributes and methods:

* **Attributes:** name (string), sound (string)
* **Method:** make\_sound() that prints the sound the animal makes

 **Create a Lion class** that inherits from Animal. Add:

* **Additional Attribute:** roar\_volume (integer)
* **Method:** roar() that prints the roar volume

 **Create a Zoo class** to use composition:

* **Attribute:** animals (a list to hold instances of Animal)
* **Method:** add\_animal(animal) to add an animal to the zoo
* **Method:** show\_animals() to print the names and sounds of all animals in the zoo

 **Hands-on Activity:**

* **Inheritance:** Show how Lion inherits from Animal and extends its functionality.
* **Composition:** Show how Zoo uses a list of Animal instances to manage multiple animals.

Output:

A white background with black text

Description automatically generated

SESSION 06: Project:

 **Create a Animal class**:

* **Attributes:** name (string), sound (string), \_age (integer, private)
* **Methods:**
  + make\_sound() prints the sound the animal makes.
  + get\_age() returns the age of the animal.
  + set\_age(age) sets the age of the animal.

 **Create a Lion class** that inherits from Animal:

* **Additional Attribute:** roar\_volume (integer)
* **Methods:**
  + roar() prints the roar volume.

 **Create a Elephant class** that inherits from Animal:

* **Additional Attribute:** trunk\_length (integer)
* **Methods:**
  + trumpet() prints a message about the trumpet sound.

 **Create a Zoo class** using composition:

* **Attribute:** animals (a list to hold instances of Animal)
* **Methods:**
  + add\_animal(animal) adds an animal to the zoo.
  + show\_animals() prints the names and sounds of all animals in the zoo.

 **Hands-on Activity**:

* Create instances of Lion, Elephant, and add them to the Zoo.
* Demonstrate how to use methods from different classes.
* Show how Lion and Elephant share common functionalities from Animal, but have their unique features.

A black screen with a black background

Description automatically generated