*Inheritance– Week 5*

Written by: Sergio Luilly Cabrera Dorado

1. What is inheritance?

Inheritance is a concept in OOP (object-oriented programming) that allows you to create new classes using classes that are already present in the project you are doing. The original class is called the parent class (superclass) and a new class is called the daughter (subclass).

The "child" class automatically inherits the attributes and methods of the "parent" class, being able to add new methods or make modifications to those already created previously.

1. Main benefit

The most important benefit is the reuse of code already created previously, saving time and resources, since, instead of rewriting the same code in different parts of the same project, for this reason common behaviors or functions can be defined only once, in the parent class (superclass) and then inherited to the specific child classes that require it.

1. Practical application

A common example is that of a parent class: vehicle, with the attributes: make, model and methods: accelerate, brake. These are inherited from the daughter classes, which for practical purposes of this example would be: car, motorcycle, truck, although each of the daughter classes can add characteristics that are specific to each one.

1. Code

* The parent class is the vehicle class, while the child’s classes are: car and motorcycle.
* Specific methods for cars: open\_trunk and overwrite the accelerate class, as can be presented
* While the methods for the motorcycle class is do\_wheelie.
* What is inherited by both classes is, make and model, which applies perfectly to each class, since you can have a motorcycle and a car of the Suzuki brand, but one can be a GN 125H (motorcycle) and the other JIMNY 5P GLX MT (car).

# PARENT CLASS

class Vehicle:

def \_\_init\_\_(self, brand, model):

self.brand = brand

self.model = model

self.speed = 0

def accelerate(self, increment):

self.speed += increment

return f"Speed increased to {self.speed} km/h"

def brake(self, decrement):

self.speed -= decrement

return f"Speed reduced to {self.speed} km/h"

# FIRST CHILD CLASS (that inherits from Vehicle)

class Car(Vehicle): # Here's the inheritance

def \_\_init\_\_(self, brand, model, num\_doors):

super().\_\_init\_\_(brand, model) # Calls parent's constructor

self.num\_doors = num\_doors # Car-specific attribute

# Car-specific method

def open\_trunk(self):

return "Trunk opened"

# We can override parent methods

def accelerate(self, increment):

# Add specific behavior

result = super().accelerate(increment)

return f"Car: {result} with engine roar"

# SECOND CHILD CLASS

class Motorcycle(Vehicle):

def \_\_init\_\_(self, brand, model, engine\_size):

super().\_\_init\_\_(brand, model)

self.engine\_size = engine\_size

def do\_wheelie(self):

return "Doing a wheelie!"

# CODE USAGE

my\_car = Car("Suzuki", " JIMNY 5P GLX MT ", 4)

my\_motorcycle = Motorcycle("Suzuki", "Gn 125H", 125)

print(my\_car.accelerate(20)) # Inherited from Vehicle

print(my\_car.open\_trunk()) # Specific to Car

print(my\_motorcycle.do\_wheelie()) # Specific to Motorcycle