**PYTHON: Unit- 2/OOPconcept**

**CODE:**

import re

from abc import ABC, abstractmethod

# Abstract Class

class Transport(ABC):

@abstractmethod

def display\_info(self):

pass

# Base Class

class Vehicle(Transport):

def \_\_init\_\_(self, make, model, year):

self.make = make

self.model = model

self.set\_year(year)

def set\_year(self, year):

# Regular Expression to validate that year is a four-digit number

if not re.match(r'^\d{4}$', str(year)):

raise ValueError("Year must be a four-digit number.")

self.year = year

def display\_info(self):

print(f"Vehicle Info: {self.year} {self.make} {self.model}")

# Derived Class

class Car(Vehicle):

def \_\_init\_\_(self, make, model, year, number\_of\_doors):

super().\_\_init\_\_(make, model, year)

self.number\_of\_doors = number\_of\_doors

def display\_info(self):

print(f"Car Info: {self.year} {self.make} {self.model} with {self.number\_of\_doors} doors")

# Exception Handling and Demonstration

try:

# Creating instances

vehicle = Vehicle("Toyota", "Corolla", 2020)

car = Car("Honda", "Civic", 2018, 4)

# Displaying information (Polymorphism)

vehicle.display\_info()

car.display\_info()

# Testing exception handling with an invalid year

invalid\_vehicle = Vehicle("Ford", "Mustang", "20AB")

except ValueError as e:

print(f"Error: {e}")

# Validating the correct year format with regular expressions

try:

correct\_vehicle = Vehicle("Tesla", "Model 3", 2022)

correct\_vehicle.display\_info()

except ValueError as e:

print(f"Error: {e}")

**OUTPUT:**

