**OOPS CONCEPTS**

**ABSTRACT CLASSES AND INTERFACES**

Using Abstract Classes and Interfaces we can define a contract for our child classes that is if we create a class XYZ and if we define an abstract method, that abstract method will not have any implementation using abstract method decorator.

Once its done, any child class that extends or inherits this parent class will have to implement the defined method.

We cannot create an instance of an abstract class.

You can have methods in an abstract class that can have implementation by typically at least one method in an abstract class is a abstract method.

**ABSTRACT CLASSES**

**Code:**

from abc import abstractstaticmethod, ABC

# CLASS SHOULD INHERIT FROM A CLASS CALLED ABC

class **Volvo**(ABC):

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

*self*.carMake=make

*self*.carModel=model

*self*.carYear=year

def **carShowroom**(*self*):

print(*"Showrooms are .... 1. JoeDuffy, 2. Volvo"*)

def **carStartStop**(*self*):

print(*"Car can be Started and Stopped using Key"*)

#MARK THIS METHOD WITH @abstractmethod annotation from abc

*@abstractstaticmethod*

def **carSafety**(*self*):

#IF WE GIVE PASS OR ANYTHING WITH "" OR "XYWUEROITU£" THEN ITS REFERRED AS ABSTRACT CLASS

pass

#INHERITING THE PARENT CLASS IN CHILD CLASS

class **XCFuel**(Volvo):

# ALSO THE 3 FILEDS ON PARENT CLASS

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

# FIRST LINE OF CHILD CLASS CONSTRUCTOR SHOULD BE TO INVOKE THE PARENT CLASS CONSTRUCTOR

#Volvo.\_\_init\_\_(self, make, model, year)

#SUPER CLASS DELCERATION

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

#CHILD CLASS VALUES

*self*.carSeater=seater

*self*.carFuelType=fuelType

def **carDescription**(*self*):

print(*"Car models available only in Petrol and Disel"*)

#OVERRIDING THE PARENT METHOD IN CHILD CLASS

def **carStartStop**(*self*):

#USING SUPER CLASS TO CALL THE PARENT CLASS carStartStop METHOD

super().carStartStop()

# WILL OVERRIDE THE PARENT CLASS carStartStop METHOD BY THIS CLASS

print(*"Car can be Started and Stopped using Start Stop Button"*)

#IF THE METHOD IS NOT INVOKED, THEN THE ERROR WILL THROW

def **carSafety**(*self*):

print(*"All the Volvo Cars are enabled with Safety feature....."*)

#CREATE ANOTHER CHILD CLASS AND PROCESS AS ABOVE

class **XCElectric**(Volvo):

# ALSO THE 3 FILEDS ON PARENT CLASS

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

# FIRST LINE OF CHILD CLASS CONSTRUCTOR SHOULD BE TO INVOKE THE PARENT CLASS CONSTRUCTOR

#Volvo.\_\_init\_\_(self, make, model, year)

#SUPER CLASS DELCERATION

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

#CHILD CLASS VALUES

*self*.carElectric=electric

def **carDescription**(*self*):

print(*"Car models available only in Electric"*)

#OVERRIDING THE PARENT METHOD IN CHILD CLASS

def **carStartStop**(*self*):

print(*"Car can be Started and Stopped using Phone"*)

#IF THE METHOD IS NOT INVOKED, THEN THE ERROR WILL THROW

def **carSafety**(*self*):

print(*"All the Volvo Cars are enabled with Safety feature....."*)

print(*"VOLVO - FUEL TYPE VEHICLE DETAILS"*)

print(*"=================================="*)

xcfuel1 = XCFuel(5,*"Petrol"*,*"Volvo"*,*"XC40"*,2023)

#xcfuel2 = XCFuel(5,"Disel","Volvo","XC40",2023)

#xcfuel3 = XCFuel(7,"Petrol","Volvo","XC60",2024)

#xcfuel4 = XCFuel(7,"Petrol","Volvo","XC90",2024)

print(*"Car Brand : "*, xcfuel1.carMake)

print(*"Car Model : "*, xcfuel1.carModel)

print(*"Car Year : "*, xcfuel1.carYear)

print(*"Car Seater : "*, xcfuel1.carSeater)

print(*"Car Fuel Type : "*, xcfuel1.carFuelType)

print(*"Calling Class from Parent Class"*)

print(*"-------------------------------"*)

xcfuel1.carShowroom()

print(*"Calling Class from Child Class"*)

print(*"-------------------------------"*)

xcfuel1.carDescription()

print(*"Calling Class from Child Class using overriding"*)

print(*"-----------------------------------------------"*)

xcfuel1.carSafety()

print()

print(*"VOLVO - FUEL TYPE ELECTRIC VEHICLE DETAILS"*)

print(*"=========================================="*)

xelec1 = XCElectric(*"Recharge"*,*"Volvo"*,*"XCE-90"*,2024)

#xelec2 = XCElectric("Recharge","Volvo","XCE-60",2024)

#xelec3 = XCElectric("Recharge","Volvo","XCE-40",2024)

print(*"Car Brand : "*, xelec1.carMake)

print(*"Car Model : "*, xelec1.carModel)

print(*"Car Year : "*, xelec1.carYear)

print(*"Car Fuel Type : "*, xelec1.carElectric)

print(*"Calling Class from Parent Class"*)

print(*"-------------------------------"*)

xelec1.carShowroom()

print(*"Calling Class from Child Class"*)

print(*"-------------------------------"*)

xelec1.carDescription()

print(*"Calling Class from Child Class using overriding"*)

print(*"-----------------------------------------------"*)

xelec1.carStartStop()

print(*"Calling Class from Child Class using overriding"*)

print(*"-----------------------------------------------"*)

xelec1.carSafety()

print()

**Output:**

VOLVO - FUEL TYPE VEHICLE DETAILS

==================================

Car Brand : Volvo

Car Model : XC40

Car Year : 2023

Car Seater : 5

Car Fuel Type : Petrol

Calling Class from Parent Class

-------------------------------

Showrooms are .... 1. JoeDuffy, 2. Volvo

Calling Class from Child Class

-------------------------------

Car models available only in Petrol and Disel

Calling Class from Child Class using overriding

-----------------------------------------------

All the Volvo Cars are enabled with Safety feature.....

VOLVO - FUEL TYPE ELECTRIC VEHICLE DETAILS

==========================================

Car Brand : Volvo

Car Model : XCE-90

Car Year : 2024

Car Fuel Type : Recharge

Calling Class from Parent Class

-------------------------------

Showrooms are .... 1. JoeDuffy, 2. Volvo

Calling Class from Child Class

-------------------------------

Car models available only in Electric

Calling Class from Child Class using overriding

-----------------------------------------------

Car can be Started and Stopped using Phone

Calling Class from Child Class using overriding

-----------------------------------------------

All the Volvo Cars are enabled with Safety feature.....

**Note:**

Cannot instantiate a abstract class as below:

**Code:**

vol=Volvo(*"Petrol"*,*"Volvo"*,*"XC40"*,2023)

**Error:**

Traceback (most recent call last):

File "C:\Users\kamal\OneDrive\Desktop\Python\Python Examples\n\_oops3\_polymorphism\abstractEgs.py", line 119, in <module>

vol=Volvo("Petrol","Volvo","XC40",2023)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

TypeError: Can't instantiate abstract class Volvo without an implementation for abstract method 'carSafety'

**INTERFACES**

In Python, an Interface is know as abstract class that where all the methods are abstract.

**Code:**

**Output:**

**QUIZ**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**ASSIGNMENT**

**Code:**

from abc import ABC, abstractmethod

class **TouchScreenLaptop**(ABC):

def **\_\_init\_\_**(*self*):

pass

*@abstractmethod*

def **scroll**(*self*):

pass

*@abstractmethod*

def **click**(*self*):

pass

class **HP**(TouchScreenLaptop):

def **\_\_init\_\_**(*self*):

pass

def **scroll**(*self*):

print(*"Scroll Option from HP"*)

print()

def **click**(*self*):

pass

class **Dell**(TouchScreenLaptop):

def **\_\_init\_\_**(*self*):

pass

def **scroll**(*self*):

print(*"Scroll Option from DELL"*)

print()

def **click**(*self*):

pass

class **HPNoteBook**(HP):

def **\_\_init\_\_**(*self*):

pass

def **scroll**(*self*):

print(*"Scroll Option from HP Notebook"*)

print()

def **click**(*self*):

print(*"Click Option from HP Notebook"*)

print()

class **DELLNoteBook**(Dell):

def **\_\_init\_\_**(*self*):

pass

def **scroll**(*self*):

print(*"Scroll Option from Dell Notebook"*)

print()

def **click**(*self*):

print(*"Click Option from Dell Notebook"*)

print()

hp=HP()

hp.scroll()

dl=Dell()

dl.scroll()

hp1 = HPNoteBook()

hp1.click()

hp1.scroll()

dl1 = DELLNoteBook()

dl1.click()

dl1.scroll()

**Output:**

Scroll Option from HP

Scroll Option from DELL

Click Option from HP Notebook

Scroll Option from HP Notebook

Click Option from Dell Notebook

Scroll Option from Dell Notebook