**PW SKILLS**

**OOPS TASK ASSIGNMENT**

Q1. Creating a Vehicle class with an `\_\_init\_\_` method:

```python

class Vehicle:

def \_\_init\_\_(self, name\_of\_vehicle, max\_speed, average\_of\_vehicle):

self.name\_of\_vehicle = name\_of\_vehicle

self.max\_speed = max\_speed

self.average\_of\_vehicle = average\_of\_vehicle

# Example usage:

vehicle1 = Vehicle("Car", 200, 30)

print(vehicle1.name\_of\_vehicle) # Output: Car

print(vehicle1.max\_speed) # Output: 200

print(vehicle1.average\_of\_vehicle) # Output: 30

```

Q2. Creating a child class Car with a `seating\_capacity` method:

```python

class Car(Vehicle):

def seating\_capacity(self, capacity):

return f"{self.name\_of\_vehicle} has a seating capacity of {capacity}."

# Example usage:

car1 = Car("SUV", 180, 25)

print(car1.seating\_capacity(7)) # Output: SUV has a seating capacity of 7.

```

Q3. Multiple Inheritance: Multiple inheritance is when a class inherits attributes and methods from more than one parent class.

```python

class A:

def method\_a(self):

print("Method A from class A")

class B:

def method\_b(self):

print("Method B from class B")

class C(A, B):

def method\_c(self):

print("Method C from class C")

# Example usage:

obj = C()

obj.method\_a() # Output: Method A from class A

obj.method\_b() # Output: Method B from class B

obj.method\_c() # Output: Method C from class C

```

Q4. Getter and Setter in Python: Getter and setter methods are used to access and modify the private attributes of a class respectively, providing controlled access to the attributes.

```python

class MyClass:

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

self.\_value = 0

def get\_value(self):

return self.\_value

def set\_value(self, value):

self.\_value = value

# Example usage:

obj = MyClass()

print(obj.get\_value()) # Output: 0

obj.set\_value(10)

print(obj.get\_value()) # Output: 10

```

Q5. Method Overriding in Python: Method overriding occurs when a subclass defines a method with the same name as a method in its superclass, effectively replacing the superclass method.

```python

class Parent:

def show(self):

print("Parent method")

class Child(Parent):

def show(self):

print("Child method")

# Example usage:

obj = Child()

obj.show() # Output: Child method

```