Q1. Create a class Book that has three attributes: title, author, and price. Write a method display() that prints the details of the book. Create two instances of the class and display their details.

class Book:

def \_\_init\_\_(self, title, author, price):

self.title = title

self.author = author

self.price = price

def display(self):

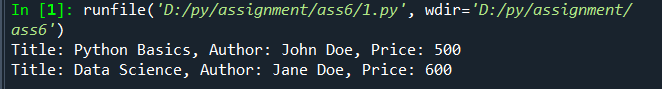
print(f"Title: {self.title}, Author: {self.author}, Price: {self.price}")

book1 = Book("Python Basics", "John Doe", 500)

book2 = Book("Data Science", "Jane Doe", 600)

book1.display()

book2.display()



Q2. Create a class Student with attributes name, age, and grade. Write a method show\_details() to display the student's details, and a method is\_passed() that returns True if the grade is greater than or equal to 40, otherwise False.

# -\*- coding: utf-8 -\*-

"""

Created on Wed Oct 23 08:37:43 2024

@author: ADMIN

"""

class Student:

def \_\_init\_\_(self, name, age, grade):

self.name = name

self.age = age

self.grade = grade

def show\_details(self):

print(f"Name: {self.name}, Age: {self.age}, Grade: {self.grade}")

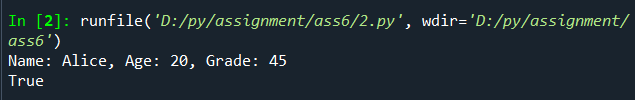
def is\_passed(self):

return self.grade >= 40

student = Student("Alice", 20, 45)

student.show\_details()

print(student.is\_passed())



Q3. Create a class Car that has attributes make, model, and year. Write a method display() to print the details. Modify the car's year and display the details again.

# -\*- coding: utf-8 -\*-

"""

Created on Wed Oct 23 08:37:50 2024

@author: ADMIN

"""

class Car:

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

self.make = make

self.model = model

self.year = year

def display(self):

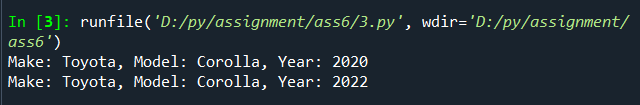
print(f"Make: {self.make}, Model: {self.model}, Year: {self.year}")

car = Car("Toyota", "Corolla", 2020)

car.display()

car.year = 2022

car.display()



Q4. Create a class BankAccount with attributes account\_number, name, and balance. The constructor should initialize these values. Write methods to deposit() an amount, withdraw() an amount (only if sufficient balance is available), and display the balance.

# -\*- coding: utf-8 -\*-

"""

Created on Wed Oct 23 08:38:03 2024

@author: ADMIN

"""

class BankAccount:

def \_\_init\_\_(self, account\_number, name, balance):

self.account\_number = account\_number

self.name = name

self.balance = balance

def deposit(self, amount):

self.balance += amount

def withdraw(self, amount):

if self.balance >= amount:

self.balance -= amount

else:

print("Insufficient balance")

def display(self):

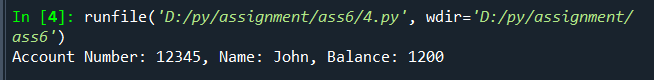
print(f"Account Number: {self.account\_number}, Name: {self.name}, Balance: {self.balance}")

account = BankAccount(12345, "John", 1000)

account.deposit(500)

account.withdraw(300)

account.display()



Q5. Create a class Employee with attributes name, position, and salary. The constructor should initialize these values. Write methods to display the employee details and provide an increment in salary by a given percentage.

# -\*- coding: utf-8 -\*-

"""

Created on Wed Oct 23 08:38:13 2024

@author: ADMIN

"""

class Employee:

def \_\_init\_\_(self, name, position, salary):

self.name = name

self.position = position

self.salary = salary

def display(self):

print(f"Name: {self.name}, Position: {self.position}, Salary: {self.salary}")

def increment\_salary(self, percentage):

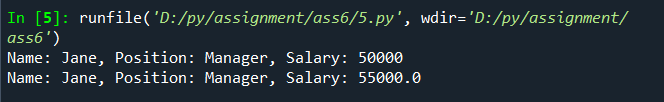
self.salary += self.salary \* (percentage / 100)

employee = Employee("Jane", "Manager", 50000)

employee.display()

employee.increment\_salary(10)

employee.display()



Q6. Create a class MovieTicket with attributes movie\_name, seat\_no, and price. Write methods to display the ticket details and check if a ticket is affordable based on a budget.

# -\*- coding: utf-8 -\*-

"""

Created on Wed Oct 23 08:38:21 2024

@author: ADMIN

"""

class MovieTicket:

def \_\_init\_\_(self, movie\_name, seat\_no, price):

self.movie\_name = movie\_name

self.seat\_no = seat\_no

self.price = price

def display(self):

print(f"Movie: {self.movie\_name}, Seat No: {self.seat\_no}, Price: {self.price}")

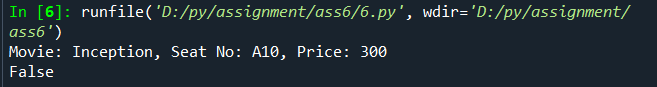
def is\_affordable(self, budget):

return self.price <= budget

ticket = MovieTicket("Inception", "A10", 300)

ticket.display()

print(ticket.is\_affordable(250))



Q7. Create a class Laptop with attributes brand, model, and price. Write methods to display the laptop details and check if the price is above a certain value

# -\*- coding: utf-8 -\*-

"""

Created on Wed Oct 23 08:38:31 2024

@author: ADMIN

"""

class Laptop:

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

self.brand = brand

self.model = model

self.price = price

def display(self):

print(f"Brand: {self.brand}, Model: {self.model}, Price: {self.price}")

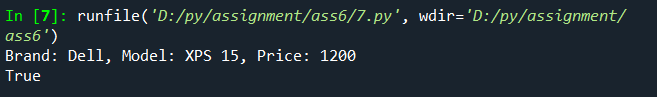
def is\_price\_above(self, value):

return self.price > value

laptop = Laptop("Dell", "XPS 15", 1200)

laptop.display()

print(laptop.is\_price\_above(1000))



. Q8. Create a class Person with attributes name and age, and a method display(). Create another class Student that inherits from Person and adds an attribute grade. Write a method show\_student() to display student details.

# -\*- coding: utf-8 -\*-

"""

Created on Wed Oct 23 08:38:41 2024

@author: ADMIN

"""

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def display(self):

print(f"Name: {self.name}, Age: {self.age}")

class Student(Person):

def \_\_init\_\_(self, name, age, grade):

super().\_\_init\_\_(name, age)

self.grade = grade

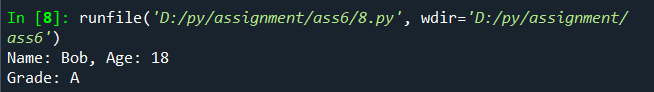
def show\_student(self):

self.display()

print(f"Grade: {self.grade}")

student = Student("Bob", 18, "A")

student.show\_student()



Q9. Create two base classes Father and Mother, both with attributes and methods. Create a derived class Child that inherits from both Father and Mother, combining their attributes and methods.

# -\*- coding: utf-8 -\*-

"""

Created on Fri Oct 25 08:59:05 2024

@author: ADMIN

"""

class Father:

def \_\_init\_\_(self, father\_name):

self.father\_name = father\_name

def father\_details(self):

print(f"Father's Name: {self.father\_name}")

class Mother:

def \_\_init\_\_(self, mother\_name):

self.mother\_name = mother\_name

def mother\_details(self):

print(f"Mother's Name: {self.mother\_name}")

class Child(Father, Mother):

def \_\_init\_\_(self, father\_name, mother\_name, child\_name):

Father.\_\_init\_\_(self, father\_name)

Mother.\_\_init\_\_(self, mother\_name)

self.child\_name = child\_name

def child\_details(self):

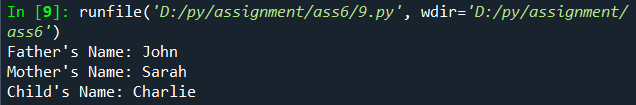
self.father\_details()

self.mother\_details()

print(f"Child's Name: {self.child\_name}")

child = Child("John", "Sarah", "Charlie")

child.child\_details()



Q10. Create a class Employee with a method show\_details(). Create two derived classes Developer and Manager that inherit from Employee and override the method to display their own details.

# -\*- coding: utf-8 -\*-

"""

Created on Fri Oct 25 08:59:24 2024

@author: ADMIN

"""

class Employee:

def show\_details(self):

print("Employee Details")

class Developer(Employee):

def show\_details(self):

print("Developer Details")

class Manager(Employee):

def show\_details(self):

print("Manager Details")

dev = Developer()

man = Manager()

dev.show\_details()

man.show\_details()

