PYTHON CASE STUDIES WITH SOLUTIONS

1. Case Study: ATM Simulation System

Problem Statement

Develop an ATM simulation that allows users to:

- Check balance
- Deposit money
- Withdraw money
- Exit

Steps to Solve

1. Define an initial balance.
2. Create a menu-driven system to perform transactions.
3. Ensure withdrawal does not exceed balance.
4. Exit the program when the user chooses.

Solution (Code)

```
python
CopyEdit
class ATM:
    def __init__(self, balance=1000):
        self.balance = balance

    def check_balance(self):
        print(f"Your balance: ${self.balance}")

    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited: ${amount}")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient funds!")
        else:
            self.balance -= amount
            print(f"Withdrawn: ${amount}")

def main():
    atm = ATM()
    while True:
        print("\n1. Check Balance\n2. Deposit\n3. Withdraw\n4. Exit")
        choice = input("Enter choice: ")

        if choice == "1":
            atm.check_balance()
        elif choice == "2":
            amt = float(input("Enter deposit amount: "))
            atm.deposit(amt)
```

```
        elif choice == "3":
            amt = float(input("Enter withdrawal amount: "))
            atm.withdraw(amt)
        elif choice == "4":
            print("Thank you for using the ATM!")
            break
        else:
            print("Invalid choice! Try again.")

main()
```

2. Case Study: E-commerce Order Management

Problem Statement

Create an **Order Management System** for an e-commerce platform. The system should allow:

- Adding products to a cart
- Viewing the cart
- Checking out (calculating total price)

Steps to Solve

1. Define a `Product` class.
2. Create a shopping cart to store items.
3. Provide options to add/view/checkout.

Solution (Code)

```
python
CopyEdit
class Product:
    def __init__(self, name, price):
        self.name = name
        self.price = price

class ShoppingCart:
    def __init__(self):
        self.cart = []

    def add_product(self, product):
        self.cart.append(product)
        print(f"{product.name} added to cart!")

    def view_cart(self):
        if not self.cart:
            print("Cart is empty!")
        else:
            print("\nShopping Cart:")
            total = 0
            for p in self.cart:
                print(f"- {p.name}: ${p.price}")
                total += p.price
```

```
            print(f"Total: ${total}")

    def checkout(self):
        if not self.cart:
            print("Cart is empty!")
        else:
            self.view_cart()
            print("Proceeding to checkout...")

def main():
    cart = ShoppingCart()
    products = {
        "1": Product("Laptop", 1000),
        "2": Product("Headphones", 150),
        "3": Product("Mouse", 50),
    }

    while True:
        print("\n1. Add Laptop ($1000)\n2. Add Headphones ($150)\n3. Add
Mouse ($50)\n4. View Cart\n5. Checkout\n6. Exit")
        choice = input("Enter choice: ")

        if choice in products:
            cart.add_product(products[choice])
        elif choice == "4":
            cart.view_cart()
        elif choice == "5":
            cart.checkout()
            break
        elif choice == "6":
            print("Thank you for shopping!")
            break
        else:
            print("Invalid choice!")

main()
```

3. Case Study: Student Grade Management System

Problem Statement

Develop a system to manage student grades:

- Add student grades
- View student grades
- Calculate the average grade

Steps to Solve

1. Create a dictionary to store student grades.
2. Provide options to add, view, and calculate average.
3. Use a loop for interaction.

Solution (Code)

```
python
CopyEdit
class GradeSystem:
    def __init__(self):
        self.grades = {}

    def add_grade(self, name, grade):
        self.grades[name] = grade
        print(f"Added: {name} - {grade}")

    def view_grades(self):
        if not self.grades:
            print("No grades available!")
        else:
            print("\nStudent Grades:")
            for name, grade in self.grades.items():
                print(f"{name}: {grade}")

    def calculate_average(self):
        if not self.grades:
            print("No grades available!")
        else:
            avg = sum(self.grades.values()) / len(self.grades)
            print(f"Class Average: {avg:.2f}")

def main():
    system = GradeSystem()

    while True:
        print("\n1. Add Grade\n2. View Grades\n3. Calculate Average\n4.
Exit")
        choice = input("Enter choice: ")

        if choice == "1":
            name = input("Enter student name: ")
            grade = float(input("Enter grade: "))
            system.add_grade(name, grade)
        elif choice == "2":
            system.view_grades()
        elif choice == "3":
            system.calculate_average()
        elif choice == "4":
            print("Exiting Grade System.")
            break
        else:
            print("Invalid choice!")

main()
```

4. Case Study: Hospital Patient Management

Problem Statement

Create a hospital management system that:

- Adds new patients
- Displays patient details
- Deletes patients

Steps to Solve

1. Use a dictionary to store patient records.
2. Implement add, view, and delete functions.

Solution (Code)

```
python
CopyEdit
class Hospital:
    def __init__(self):
        self.patients = {}

    def add_patient(self, id, name, age, disease):
        self.patients[id] = {"Name": name, "Age": age, "Disease": disease}
        print(f"Patient {name} added!")

    def view_patients(self):
        if not self.patients:
            print("No patients registered!")
        else:
            print("\nPatient Records:")
            for id, details in self.patients.items():
                print(f"ID: {id} - {details}")

    def remove_patient(self, id):
        if id in self.patients:
            del self.patients[id]
            print("Patient removed!")
        else:
            print("Patient not found!")

def main():
    hospital = Hospital()

    while True:
        print("\n1. Add Patient\n2. View Patients\n3. Remove Patient\n4.
Exit")
        choice = input("Enter choice: ")

        if choice == "1":
            id = input("Enter Patient ID: ")
            name = input("Enter Name: ")
            age = input("Enter Age: ")
            disease = input("Enter Disease: ")
            hospital.add_patient(id, name, age, disease)
        elif choice == "2":
            hospital.view_patients()
        elif choice == "3":
```

```
            id = input("Enter Patient ID to remove: ")
            hospital.remove_patient(id)
        elif choice == "4":
            print("Exiting Hospital System.")
            break
        else:
            print("Invalid choice!")

main()
```