K. JASHWANTH

2303A52300

BATCH – 42

Task 1: Student Performance Evaluation System

PROMPT:

Write a python code for student performance evaluation system two rite the display_details method and to check_performance method using loops

CODE

C: > Users > DELL > Downloads > #Create the skeleton of a Python class n.py > ...

```python
# write the display_details method
# write the check_performance method using if-else
class Student:
    def __init__(self, name, age, grade):
        self.name = name
        self.age = age
        self.grade = grade

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

    def check_performance(self):
        if self.grade >= 90:
            return "Excellent"
        elif self.grade >= 75:
            return "Good"
        elif self.grade >= 60:
            return "Average"
        else:
            return "Poor"


# User input
name = input("Enter student name: ")
age = int(input("Enter student age: "))
grade = int(input("Enter student grade: "))

student1 = Student(name, age, grade)

student1.display_details()
print("Performance Status:", student1.check_performance())
```

OUTPUT:

```
C:\Users\DELL>C:/Users/DELL/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/DELL/Downloads/#Create the skeleton of a
Enter student name: harsha
Enter student age: 20
Enter student grade: 66
Name: harsha
Age: 20
Grade: 66
Performance Status: Average

C:\Users\DELL>
```

JUSTIFICATION:

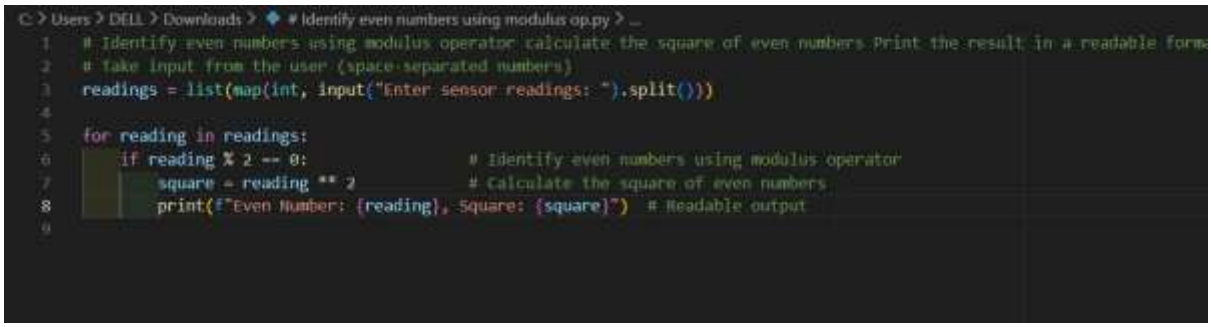
This program uses a class to store student details like name and marks.
It displays the details and checks the student's performance based on class average.

Task 2: Data Processing in a Monitoring System

PROMPT:

Write a python code for monitoring system prompt Identify even numbers using modulus operator calculate the square of even numbers Print the result in a readable format Take input from the user

CODE:



OUTPUT:

JUSTIFICATION:

The code checks each sensor reading one by one.
Only even readings are processed, and their squares are displayed.

Task 3: Banking Transaction Simulation

PROMPT

Write a python code for deposit money into the account, withdraw money from the account , use if-else to check for sufficient balance prevent withdrawal when balance is insufficient.

CODE:

```python
# create a method to deposit money into the account , create a method to withdraw money from the account , use if-else to check for sufficient ba
class BankAccount:
    def __init__(self, account_holder, balance):
        self.account_holder = account_holder
        self.balance = balance

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

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew {amount}. Remaining balance: {self.balance}")
        else:
            print("Insufficient balance. Withdrawal denied.")
# User input
account_holder = input("Enter account holder name: ")
initial_balance = float(input("Enter initial balance: "))
account = BankAccount(account_holder, initial_balance)
account.deposit(float(input("Enter amount to deposit: ")))
account.withdraw(float(input("Enter amount to withdraw: ")))
```

OUTPUT

```
C:\Users\DELL>C:/Users/DELL/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/DELL/Downloads/#create a mmthod to deposit money into the a
ccount.py"
Enter account holder name: harsha
Enter initial balance: 5000
Enter amount to deposit: 4500000
Deposited 4500000.0. Current balance: 4505000.0
Enter amount to withdraw: 4500000
Withdrew 4500000.0. Remaining balance: 5000.0
```

JUSTIFICATION:

The program simulates basic banking operations like deposit and withdrawal.
It prevents withdrawal when the account balance is not sufficient.

Task 4: Student Scholarship Eligibility Check

PROMPT:

write a python code for student scholar eligibility program, use a while loop to iterate through the list of students, check if the student's score is greater than 75, if eligible, print the student's name, handle index properly to avoid infinite loop.

CODE:

```python
#write a python code for student scholar eligibility program, use a while loop to iterate through the list of students , check if the student's score
students = []

try:
    n = int(input("Enter number of students: "))
except ValueError:
    print("Please enter a valid number")
    exit()

for _ in range(n):
    name = input("Enter student name: ")
    score = int(input("Enter student score: "))
    students.append({"name": name, "score": score})

i = 0
print("Eligible students:")
while i < len(students):
    if students[i]["score"] > 75:
        print(students[i]["name"])
    i += 1
```

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER

Enter student name: harsha
Enter student score: 79
Enter student name: naziya
Enter student score: 35
Eligible students:
harsha
```

JUSTIFICATION:

The program loops through student records stored in a list.
It identifies and prints students who are eligible for scholarship based on score**.**

Task 5: Online Shopping Cart Module

PROMPT:

Write a python code to create a Shopping Cart class with an empty list to store items where each item has name, price, and quantity; create methods to add items (accept name, price, quantity and append to list), remove items by name using a loop, calculate total bill using a loop (price * quantity), apply conditional discount if total exceeds a certain amount using if-else, and display user-friendly messages.

CODE:

```python
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_item(self, name, price, quantity):
        self.items.append({
            "name": name,
            "price": price,
            "quantity": quantity
        })
        print(f"Item '{name}' added to cart.")

    def remove_item(self, name):
        for item in self.items:
            if item["name"] == name:
                self.items.remove(item)
                print(f"Item '{name}' removed from cart.")
                return
        print("Item not found in cart.")

    def calculate_total(self):
        total = 0
        for item in self.items:
            total += item["price"] * item["quantity"]

        if total > 1000:
            discount = total * 0.10
            total -= discount
            print("Discount applied: 10%")

        return total


# ----- User Input Section -----
cart = ShoppingCart()
```

```python
# ----- User Input Section -----
cart = ShoppingCart()

n = int(input("Enter number of items to add: "))

for _ in range(n):
    name = input("Enter item name: ")
    price = float(input("Enter item price: "))
    quantity = int(input("Enter quantity: "))
    cart.add_item(name, price, quantity)

remove = input("Do you want to remove an item? (yes/no): ").lower()
if remove == "yes":
    name = input("Enter the name of the item to remove: ")
    cart.remove_item(name)

total_bill = cart.calculate_total()
print("Total Bill Amount:", total_bill)
```

OUTPUT:

JUSTIFICATION

The program manages items added to a shopping cart.
It calculates the total amount and applies discount when applicable.