**SIVA SIVANI DEGREE COLLEGE**

**(AUTONOMOUS)**

**TECHNICAL SKILL TEST**

**DEBUGGING (SYNTACTICAL AND LOGICAL ERRORS)**

**LIST OF QUESTIONS :**

1. SIMPLE SHOPPING CART SIMULATION
2. PRIME NUMBERS, ODD NUMBERS, AND NATURAL NUMBERS
3. STUDENT MANAGEMENT SYSTEM SIMULATION
4. CONTACT BOOK SIMULATION
5. CHECK THE ENTER NUMBERS AND STRING IS PALINDROME ARE NOT.
6. LIBRARY MANAGEMENT SYSTEM SIMULATION (WITH BUGS)
7. STUDENT INFORMATION SYSTEM USING TUPLE LIST DICT
8. TO-DO LIST MANAGER
9. MOVIE COLLECTION TRACKER
10. SIMPLE QUIZ GAME
11. EXPENSE TRACKER SIMULATION (WITH BUGS)
12. CLASSROOM ATTENDANCE TRACKER
13. RESTAURANT ORDERING SYSTEM (WITH BUGS)
14. SIMPLE CALCULATOR
15. TASK SCHEDULER APPLICATION
16. STUDENT ATTENDANCE LOGIN SYSTEM
17. VISITOR LOG SYSTEM FOR OFFICES
18. PARKING LOT MANAGEMENT SYSTEM
19. MATRIX CREATION , ADDTION AND MUTIPLICATION AND TRANSPOSE OF MATRIX
20. STRING METHODS IN PYTHON
21. **Simple Shopping Cart Simulation**

class ShoppingCart:

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

self.items = {}

def add\_item(self, item\_name, quantity, price):

if item\_name in self.items:

self.items[item\_name]['quantity'] += quantity

else:

self items[item\_name] = {

'quantity': quantity,

'price': price

}

def remove\_item(self, item\_name):

if item\_name in self.items:

del self.items[item\_name]

else:

print(f"{item\_name} not found in cart.")

def update\_quantity(self, item\_name, quantity):

if item\_name in self.items:

if quantity <== 0:

del self.items[item\_name]

else:

self.items[item\_name]['quantity'] = quantity

else:

print("Item not found.")

def get\_total(self):

total == 0

for item in self.items.values():

total +== item['quantity'] \* item['pric']

return total

def show\_cart(self):

if not self.items:

print("Your cart is empty.")

else:

print("Cart Contents:")

for item name, details in self.items.items():

print(f"{item\_name}: {details[quantity]} x ₹{details['price']}")

def main():

cart = ShoppingCart()

while True

print("\nOptions:")

print("1. Add Item")

print("2. Remove Item")

print("3. Update Quantity")

print("4. View Cart")

print("5. Get Total")

print("6. Exit")

choice = int(input("Enter choice: "))

if choice = 1:

name = input("Item name: ")

qty = int(input("Quantity: "))

price = float(input("Price per unit: "))

cart.add\_item(name, qty, price)

elif choice = 2:

name = input("Item name to remove: ")

cart.remove\_item(name)

elif choice = 3:

name = input("Item name to update: ")

qty = int(input("New quantity: "))

cart.update\_quantity(name, qty)

elif choice = 4:

cart.show\_cart()

elif choice = 5:

print("Total Cost: ₹", cart.get\_total())

elif choice = 6:

print("Exiting...")

break

else:

print("Invalid option.")

main()

1. **check the prime numbers, odd numbers, and natural numbers**

Python code that checks for prime numbers, odd numbers, and natural numbers .

def check\_numbers(n)

prime numbers = []

odd numbers = []

natural numbers = []

even numbers = []

for i in range(1, n + 1):

natural\_numbers.append(i)

if i / 2 != 0

odd\_numbers.append(i)

else:

even\_numbers.append(i)

if i < 1:

is\_prime = True

for j in range(2, int(i \*\* 0.5) + 1):

if i % j = 0:

is\_prime = False

break

if is\_prime:

prime\_numbers.append(i)

print("Natural Numbers:" natural\_numbers)

print("Odd Numbers:", odd\_numbers)

print("Even Numbers:", even\_numbers)

print("Prime Numbers:", prime\_numbers)

def print\_statistics(n):

prime numbers = []

odd numbers = []

natural\_numbers = []

even numbers = []

for i in range(1, n + 1):

natural\_numbers.append(i)

if i % 2 != 0:

odd\_numbers.append(i)

else

even\_numbers.append(i)

if i > 1:

is\_prime = True

for j in range(2, int(i \*\* 0.5) + 1):

if i \ j == 0:

is prime = False

break

if is prime:

prime\_numbers.append(i)

print("Statistics:")

print("Total Natural Numbers:", len(natural\_numbers))

print("Total Odd Numbers:", len(odd\_numbers))

print("Total Even Numbers:", len(even\_numbers))

print("Total Prime Numbers:", len(prime\_numbers))

def main():

while True:

print("\nMenu:")

print("1. Check Numbers")

print("2. Print Statistics")

print("3. Quit")

choice = input("Enter your choice: ")

if choice = "1":

n = int(input("Enter a number: "))

check\_numbers(n)

elif choice = "2":

n = int(input("Enter a number: "))

print\_statistics(n)

elif choice == "3":

break

else:

print("Invalid choice. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**3. Student Management System Simulation**

Student Management System Simulation

class Student:

def \_init\_\_(self, roll\_no, name, grade):

self.roll\_no = roll\_no

self.name = name

self.grade = grade

def display(self)

print(f"Roll No: {self.roll\_no}, Name: {self.name}, Grade: {self.grade}")

# class School:

def \_init\_\_(self)

self.students = []

def add\_student(self, roll\_no, name, grade):

student = Student(roll\_no, name, grade)

self.students.append(student)

def remove\_student(self roll\_no):

for student in self.students:

if student.roll no == roll\_no:

self.students.remove(student)

print(f"Student {student.name} removed.")

return

print(Student not found. )

def update\_grade(self, roll\_no, new\_grade):

for student in self.students

if student.roll\_no = roll\_no:

student grade = new\_grad

print(f"Grade updated for {student.name}")

return

print("Student not found.")

def view\_all\_students(self):

if not self.students:

print("No students in the system.")

else:

for student in self.students:

student.display()

def menu():

school = School()

while True

print("\nMenu:")

print("1. Add Student")

print("2. Remove Student")

print("3. Update Grade")

print("4. View All Students")

print("5. Exit")

choice = int(input("Enter your choice: "))

if choice = 1:

roll = int(input("Enter Roll No: "))

name = input("Enter Name: ")

grade = input("Enter Grade: ")

school.add\_student(roll, name, grade)

elif choice == 2

roll = int(input("Enter Roll No to remove: "))

school.remove\_student(roll)

elif choice = 3:

roll = int(input("Enter Roll No to update grade: "))

grade = input("Enter new Grade: ")

school.update\_grade(roll, grade)

elif choice = = 4:

school.view\_all\_students()

elif choice = 5:

print("Exiting program.")

break

else

print("Invalid choice.")

menu()

**4. Contact Book Simulation**

# Contact Book Simulation

class Contact:

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

self.name = name

self.phone = phone

def display(self):

print(f"{self.name}: {self.phone}")

class ContactBook:

def \_init\_\_(self):

self.contacts = []

#def add\_contact(self, name, phone):

contact = Contact(name, phone)

self contacts.append(contact)

print("Contact added successfully!")

def remove\_contact(self, name):

for contact in self.contacts:

if contact.name == name:

self.contacts.remove(contact)

print("Contact removed.")

return

print("Contact not found.")

def update\_phone(self, name, new\_phone):

for contact in self.contacts:

if contact.name == name:

contact.phon = new\_phone

print("Phone number updated.")

return

print("Contact not found.")

def view\_contacts(self):

if not self.contacts:

print("No contacts available.")

else:

print("Your Contacts:")

for contact in self.contacts:

contact.display()

def menu():

book = ContactBook()

while True

print("\nContact Book Menu")

print("1. Add Contact")

print("2. Remove Contact")

print("3. Update Phone")

print("4. View Contacts")

print("5. Exit")

try:

choice = int(input("Enter your choice: "))

except ValueError print("Invalid input! Please enter a number.")

continue

if choice = 1:

name = input("Enter name: ")

phone = input("Enter phone number: ")

book.add\_contact(name, phone)

elif choice = 2:

name = input("Enter name to remove: ")

book.remove\_contact(name)

elif choice = 3:

name = input("Enter name to update: ")

new\_phone = input("Enter new phone number: ")

book.update\_phone(name, new\_phone)

elif choice = 4:

book.view\_contacts()

elif choice = 5:

print("Goodbye!")

break

else print("Invalid choice.")

menu()

1. **check the enter numbers and string is palindrome are not.**

def is\_palindrome(s):

s = str(s)

return s == s(::-1)

def check\_palindromes():

print("Palindrome Checker")

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

palindromes = []

while True

print("\nMenu:")

print("1. Check String Palindrome")

print("2. Check Number Palindrome")

print("3. Check Both")

print("4. View Palindromes")

print("5. Quit")

choice = input("Enter your choice: ")

if choice == "1":

s = input("Enter a string: ")

if is\_palindrome(s):

print(f"'{s}' is a palindrome.")

palindromes.append(s)

else:

print(f"'{s}' is not a palindrome.")

elif choice = "2":

try

n = int(input("Enter a number: "))

if is\_palindrome(n):

print(f"{n} is a palindrome.")

palindromes.append(str(n))

else:

print(f{n} is not a palindrome.")

except ValueError:

print("Invalid input. Please enter a number.")

elif choice = 3:

s = input("Enter a string or number: ")

try:

n = int(s)

if is\_palindrome(n):

print(f"{n} is a palindrome.")

palindromes.append(str(n))

else:

print(f"{n} is not a palindrome.")

except ValueError:

if ispalindrome(s):

print(f"'{s}' is a palindrome.")

palindromes.append(s)

else

print(f"'{s}' is not a palindrome.")

elif choice = "4":

if palindromes:

print("Palindromes:")

for palindrome in palindromes print(palindrome)

else print("No palindromes found.")

elif choice == "5" break

else print("Invalid choice. Please try again.")

def main():

check\_palindromes

if \_\_name\_\_ == "\_\_main\_\_":

main()

**6. # Library Management System Simulation (with bugs)**

class Book:

def \_\_init\_\_(self, book\_id, title, author):

self.book\_id = book\_id

self.title = title

self.author = author

def display info(self):

print(f"ID: {self.book\_id}, Title: '{self.title}', Author: {self.author}")

class Library:

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

self.books = []

def add\_book(self, book\_id, title, author):

book = Book(book\_id, title, author)

self.books.append(book)

def remove\_book(self, book\_id):

for book in self.books:

if book.book\_id == book\_id:

self.books.remove(book)

print(f"Book '{book.title}' removed.")

return

print("Book not found.")

def search\_book(self, title):

found = False

for Book in self.books:

if book.title.lower() == title.lower():

book.display\_info()

found = True

if not found:

print("No book found with that title.")

def totalbooks(self)

print("Total books in library:", len(self.books))

return len(self.boks)

def menu():

lirary = Library()

while True

print("\nLibrary Menu:")

print("1. Add Book")

print("2. Remove Book")

print("3. Search Book")

print("4. View Total Books")

print("5. Exit")

choice = input("Enter choice (1-5): ")

if choice == '1':

book\_id = input("Enter Book ID: ")

title = input("Enter Book Title: ")

author = input("Enter Author: ")

library.add\_book(book\_id, title, author)

elif choice == '2':

book\_id = input("Enter Book ID to remove: ")

library.remove\_book(book\_id)

elif choice == '3':

title = input("Enter Book Title to search: ")

library.search\_book(title)

elif choice == '4':

library.total\_books()

elif choice == '5':

print("Exiting Library System.")

break

else

print("Invalid choice. Please select from 1 to 5.")

menu()

7. **Student Information System using tuple list dict**

# Define a tuple of student IDs

student\_ids = 1, 2, 3, 4, 5)

# Define a dictionary to store student information

students = {

student\_ids[0] {"name": "John Doe", "age": 20, "grade": "A"},

student\_ids[1] {"name": "Jane Doe", "age": 21, "grade": "B"},

student\_ids[2] {"name": "Bob Smith", "age": 19, "grade": "A"},

student\_ids[3] {"name": "Alice Johnson", "age": 20, "grade": "B"},

student\_ids[4] {"name": "Mike Brown", "age": 21, "grade": "A"},

}

# Define a list to store student grades

grades = "A", "B", "C", "D", "F]

# Function to display student information

def display\_students():

print("Student Information:")

for student\_id, student\_info in students.items():

print(f"ID: {student\_id})

print(f"Name: student\_info['name']")

print(f"Age: {student\_info['age']}")

print(f"Grade: {student\_info['grade']}")

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

# Function to add a new student

def add\_student():

student\_id = int(input("Enter student ID: "))

name = input("Enter student name: ")

age = int(input("Enter student age: "))

grade = input("Enter student grade (A, B, C, D, F): ")

while grade.upper() not in grades:

grade = input("Invalid grade. Please enter a valid grade (A, B, C, D, F): ")

students[student\_id] = {"name": name, "age": age, "grade": grade.upper()}

# Function to update student information

def update\_student():

student\_id = int(input("Enter student ID: "))

if student\_id in students:

print("Enter new information (leave blank to keep current value):")

name = input(f"Name ({students[student\_id]['name']}): ")

age = input(f"Age ({students[student\_id]['age']}): ")

grade = input(f"Grade ({students[student\_id]['grade']}): ")

if name:

students[student\_id]["name"] = name

if age:

students[student\_id]["age"] = int(age)

if grade:

while grade.upper() not in grades:

grade = input("Invalid grade. Please enter a valid grade (A, B, C, D, F): ")

students[student\_id]["grade"] = grade.upper()

else:

print("Student not found.")

# Function to delete a student

def delete\_student():

student\_id = int(input("Enter student ID: "))

if student\_id in students:

del students[student\_id]

print("Student deleted successfully.")

else:

print("Student not found.")

# Main program loop

while True:

print(\nStudent Information System)

print("1. Display Students")

print("2. Add Student")

print("3. Update Student")

print("4. Delete Student")

print("5. Quit")

choice = input("Enter your choice: ")

if choice = "1":

display\_students()

elif choice = "2":

add\_student()

elif choice = "3":

update\_student()

elif choice == "4"delete\_student()

elif choice == "5":

break

else:

print("Invalid choice. Please try again.")

8.  **To-Do List Manager**

# To-Do List Manager Simulation (with bugs)

class Task:

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

self.description = description

self.completed = False

def mark\_complete(self):

self.completed = True

def show(self):

status = "✓" if self.completed else "✗"

print(f"[{status}] {self.description}")

class ToDoList:

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

self.tasks = []

def add\_task(self, description):

task = Task(description)

self.tasks.append(task)

def remove\_task(self, index):

if 0 <= index < len(self.tasks):

removed = self.tasks.pop(index)

print(f"Removed task: {removed.description}")

else:

print("Invalid index.")

def mark\_task\_complete(self, index):

if 0 <= index < len(self.tasks):

self.tasks[index].mark\_complete()

print("Task marked as complete.")

else:

print("Invalid index.")

def view\_tasks(self):

if not self.tasks:

print("No tasks found.")

else:

for i, task in enumerate(self.tasks):

print(f"{i + 1}.", task.shw())

def menu():

todo = ToDoList()

while True:

print("\nTo-Do List Menu:")

print("1. Add Task")

print("2. Remove Task")

print("3. Mark Task Complete")

print("4. View Tasks")

print("5. Exit")

choice = input("Enter choice (1-5): ")

if choice == '1':

desc = input("Enter task description: ")

todo.add\_task(desc)

elif choice == '2':

idx = int(input("Enter task number to remove: ")) - 1

todo.remove\_task(idx)

elif choice == '3':

idx = int(input("Enter task number to mark complete: ")) - 1

todo.mark\_task\_complete(idx)

elif choice == '4':

todo.view\_tasks()

elif choice == '5':

print("Exiting To-Do List.")

break

else

print("Invalid choice. Please select from 1 to 5."

menu()

**9. Movie Collection Tracker**

# Movie Collection Tracker (with bugs)

class Movie:

def \_\_init\_\_(self, title, director, year):

self.title = title

self.director = director

self.year = year

def show\_details(self):

print(f"'{self.title}' directed by {self.director} (self.year})")

class MovieCollection:

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

self.movies = []

def add\_movie(self, title, director, year):

movie = Movie(title, director, year)

self.movies.append(movie)

def remove movie(self, title):

for movie in self.movies:

if movie.title = title:

self.movies.remove(movie)

print(f"Movie '{title}' removed.")

return

print("Movie not found.")

def find\_by\_director(self, director\_name):

found = False

for movie in self.movies:

if movie.director.lower() == director\_name.lower():

movie.show\_detail()

found = True

if not found:

print("No movies found for that director.")

def list\_movies(self):

if not self.movies:

print("No movies in collection.")

else:

for movie in self.movies:

movie.show details()

def menu():

collection = Movie.Collection()

while True:

print("\nMovie Collection Menu:")

print("1. Add Movie"

print("2. Remove Movie")

print("3. Find Movies by Director")

print("4. List All Movies")

print("5. Exit")

choice = input("Enter your choice (1–5): ")

if choice == '1':

title = input("Enter movie title: ")

director = input("Enter director: ")

year = input("Enter release year: ")

collection.add\_movie(title, director, year)

elif choice == '2'

title = input("Enter movie title to remove: ")

collection.remove\_movie(title)

elif choice == '3':

director = input("Enter director name to search: ")

collection.find\_by\_director(director)

elif choice = '4':

collection.list\_movies()

elif choice == '5':

print("Exiting Movie Collection.")

break

else

print("Invalid choice. Try something between 1 and 5.")

menu()

**10. Simple Quiz Game**

Simple Quiz Game (with bugs)

class Question:

def \_\_init\_\_(self, prompt, answer):

self.prompt = prompt

self.answer = answer

def check\_answer(self, user\_answer):

return user\_answer.strip().lower() == self.answer.strip().lower()

class QuizGame:

definit(self)

self.questions = [];

self.score = 0

def add\_question(self, prompt, answer)

question = Question(prompt, answer)

self.questions.append(question)

def run\_quiz(self):

for i, question in enumerate(self.questions):

print(f"Question {i + 1}:")

print(question.prompt)

user\_answer = input("Your answer: ")

if question.check\_ansr(user\_answer):

print(Correct!)

self.score += 1

else:

print("Wrong! Correct answer was: {question.answer}")

print(f"\nFinal Score: {self.score}/{len(self.questions)}")

def menu[]

quiz = QuizGame()

quiz.add\_question("What is the capital of France?", "Paris")

quiz.add\_question("What is 5 + 7?", "12")

quiz.add\_question("Which planet is known as the Red Planet?", "Mars")

while True[]

print("\nQuiz Menu:");

print("1. Start Quiz")

print("2. Exit")

choice = input("Enter your choice (1 or 2): ");

if choice == '1'

quiz.run\_quiz()

elif choice == '2':

print("Exiting Quiz Game.");

break

else

print("Invalid choice. Try again.")

menu[]

**11. # Expense Tracker Simulation (with bugs)**

class Expense:

def init(self, category, amount):

self.category = category

self.amount = amount

def show(self):

print(f"{self.category}: ₹{self.amount}")

class ExpenseTracker:

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

self.expenses = []

def add.expense(self, category, amount):

expense = Expense(category, amount)

self expenses.append(expense)

def remove\_expense(self, category):

for expense in self.expenses:

if expense.category == category:

self.expenses.remove(expense)

print(f"Removed expense for {category}")

return

print("Expense category not found.")

def total\_expense( ):

total = 0

for expense in self.expense:

total += expense.amount

print(f"Total Expense: ₹{total}")

return total

def view\_expenses(self):

if not self.expenses:

print("No expenses recorded.")

else:

print("Expense List:")

for expense in self.expenses:

expense.show()

def menu():

tracker = ExpenseTracker()

while True

print("\nExpense Tracker Menu:")

print("1. Add Expense")

print("2. Remove Expense")

print("3. View Expenses")

print("4. Total Expense")

print("5. Exit")

choice = input("Enter your choice (1–5): ")

if choice = '1':

category = input("Enter expense category: ")

amount = float(input("Enter expense amount: "))

tracker.add\_expense(category, amount)

elif choice = 2':

category = input("Enter category to remove: ")

tracker.remove\_expense(category)

elif choice = '3':

tracker.view\_expenses()

elif choice == '4':

tracker.total\_expense()

elif choice = '5':

print("Exiting Expense Tracker.")

break

else print("Invalid option. Please enter a number from 1 to 5.")

menu()

**12. Classroom Attendance Tracker**

# Classroom Attendance Tracker (with bugs)

class Student:

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

self.name = name

self.present = False

def mark\_present(self):

self.present = True

def status(self):

return Present if self.present else "Absent"

class AttendanceRegister:

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

self.students = []

'''def add\_student(self, name):

student = Student(name)

selfstudents.append(student)

def remove\_student(self. name):'''

for student in self.students:

if student.name == name:

self.students.remove(student)

print(f{name} removed from register."'')

return

print("Student not found.")

def mark\_attendance(self, name):

for student in self.students:

if student.name == name:

student.mark\_presnt()

print(f"{name} marked as present.")

return

print("Student not found.")

def view\_attendance(self):

if not self.students:

print("No students in register.")

else:

print("Attendance List:")

for student in self.students:

print(f"{student.name}: {student.status()}")

def menu():

register = AttendanceRegister()

while True

print("\nAttendance Menu:")

print("1. Add Student")

print("2. Remove Student")

print("3. Mark Attendance")

print("4. View Attendance")

print("5. Exit")

choice = input("Enter your choice (1–5): ")

if choice == '1'

name = input("Enter student name: ")

register.add\_student(name)

elif choice == '2':

name = input("Enter name to remove: ")

register.remove\_student(name)

elif choice == '3':

name = input("Enter name to mark present: ")

register.mark\_attendance(name)

elif choice == '4':

register.view\_attendance()

elif choice = '5':

print("Exiting Attendance Tracker.")

break

else

print("Invalid choice. Please try again.")

menu()

**13.Restaurant Ordering System (with bugs)**

class MenuItem:

def \_init\_\_(self. name, price):

self.name = name

self.price = price

def show(self):

print(f{self.name}: ₹{self.price})

class Order:

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

self.items = []

def add\_item(self, menu, item\_name):

for item in menu:

if item.name == item\_name:

self.items.append(item)

print(f"Added {item\_name} to your order.")

return

print("Item not found.")

def remove\_item(self, item\_name):

for item in self.items:

if item.name == item\_name:

self.items.remove(item)

print(f"Removed {item\_name} from your order.")

return

print("Item not in order.")

def view\_order(self):

if not self.items:

print("Your order is empty.")

else:

print("Current Order:")

for item in self.items:

item.show()

def get\_total(self)

total = 0

# for item in self.itmes:

total += item.price

print(f"Total Bill: ₹{total}")

return total

def menu():

# Create the menu

menu\_items = [

MenuItem("Pizza", 250),

MenuItem("Burger", 150),

MenuItem("Pasta", 200),

MenuItem("Soda", 50)

]

#order = Order()

while True:

print("\nRestaurant Menu:")

print("1. View Menu")

# print("2. Add Item to Order")

print("3. Remove Item from Order")

print("4. View Current Order")

print("5. Get Bill")

print("6. Exit")

choice = input("Enter your choice (1–6): ")

if choice == '1':

print("Available Menu Items:")

for item in menu items:

item.show()

elif choice == '2':

name = input(Enter item name to add: )

order.add\_item(menu\_items, name)

elif choice == '3':

name = input("Enter item name to remove: ")

order.remove\_item(name)

elif choice == '4':

order.view\_order()

elif choice == '5':

order.get\_total()

elif choice = 6:

print("Thank you for visiting!")

break

else :

print("Invalid choice. Please try again.")

menu()

**14.simple calculator**

def add(x, y):

return x + y

def subtract(x, y)

return x – y;

def multiply(x. y)

return x x y

def divide(x, y)

if y == 0:

return "Error: Division by zero is not allowed";

return x / y

def calculator()

while True

print("\nCalculator Menu:")

print("1. Addition")

print("2. Subtraction")

print("3. Multiplication")

print("4. Division")

print("5. Quit")

choice = input("Enter your choice: ")

if choice in [1 "2""3""4"]:

try:

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

except ValueError:

print("Invalid input. Please enter a number.")

continue

if choice ="1":

print(f"{num1} + {num2} = ={add(num1, num2)}")

elif choice == 2":

print(f"{num1} - {num2} {subtract(num1, num2)}")

elif choice = "3":

print(f"{num1} {num2} = {multiply(num1, num2)}")

elif choice == "4":

result = divide(num1, num2)

if isinstance(result, str):

print(result)

else:

print(f"{num1} / {num2} = {result}")

elif choice == "5":

print("Goodbye!")

break

else:

print("Invalid choice. Please try again.")

def main():

calculator()

if \_\_name\_\_ == "\_\_main\_\_":

main()

# Additional lines to reach 100 lines

print("This is a simple calculator program.")

print("It performs basic arithmetic operations.")

print("You can add, subtract, multiply, or divide two numbers.")

print("You can also quit the program.")

print("I hope you find this program useful.")

print("Thank you for using this program!")

for i in range(5)

print(f"Line {i+2}")

print("End of program.")

1. **Task Scheduler Application**

# Task Scheduler App (with bugs)

class Task:

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

self.name = name

self.deadline = deadline

def display(self):

print(f"Task: {self.name} | Deadline: {self.deadline}")

class Scheduler:

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

self.tasks = ()

def add\_task(self, name, deadline):

task = Task(name, deadline)

self.tasks.append(task)

print(fTask '{name}' added with deadline '{deadline}'");

def remove\_task(self, name):

for task in self.asks:

if task.name = name:

self.tasks.remove(task)

print(fTask '{name}' removed.")

return

print("Task not found.")

def list\_tasks(self):

if != self.tasks:

print("No tasks scheduled.")

else:

print("Scheduled Tasks:")

for task in self.taskz:

taskdisplay()

def menu():

scheduler = Scheduler()

while True:

print("\nTask Scheduler Menu:")

print("1. Add Task")

print("2. Remove Task")

print("3. List All Tasks")

print("4. Exit")

choice = input("Enter your choice (1–4): ")

if choice == '1':

name = input("Enter task name: ")

deadline = input("Enter deadline (e.g. 2025-07-20): ")

scheduler.add\_task(name, deadline)

elif choice == '2':

name = input("Enter task name to remove: ")

scheduler.remove\_task(name)

elif choice == '3':

scheduler.list\_tasks()

elif choice == '4'

print("Exiting Task Scheduler.")

break

else print("Invalid choice. Try again.")

menu()

**16.Student Attendance Login system**

# Student Attendance Log (with bugs)

class Student:

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

self.name = name

self.days\_present = 0

def mark\_present(self):

self.days\_present ++= 1

def display(self):

print(f"{self.name: {self.days\_present} days present")

class AttendanceLog:

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

self.students = []

def add\_student(self, name)

student = Student(name)

self.students.append(student)

print(f"Student 'nameß added.")

def remove\_student(self, name):

for student in self.students

if student.name = name

self.students.remove(student)

print(f"Removed '{name}' from log.")

return

print("Student not found.")

def mark\_day(self, name):

for student in self.students:

if student.name == name:

student.mark\_presnt()

print(f"Marked '{name}' as present.")

return

print("Student not found.")

def show\_log(self):

if not self.students:

print("No students in log.")

else:

print("Attendance Records:")

for student in self.students:

student.display()

def menu()

log = =AttendanceLog()

while True:

print("\nAttendance Log Menu:")

print("1. Add Student")

print("2. Remove Student")

print("3. Mark Student Present")

print("4. View Attendance Log")

print("5. Exit")

choice = input("Enter your choice (1–5): ")

if choice == '1':

name = input("Enter student name: ")

log.add\_student(name)

elif choice == '2':

name = input("Enter student name to remove: ")

log.remove\_student(name)

# elif choice = '3':

name = input("Enter name to mark present: ")

log.mark\_day(name)

elif choice == '4':

log.show\_log()

elif choice == '5':

print("Exiting Attendance Log.")

break

else

print("Invalid choice. Please try again.")

menu()

**17.Visitor Log System for Offices**

# Visitor Log System (with bugs)

class Visitor:

def \_\_init\_\_(self. Name. purpose)

self.name = name

self.purpose = purpose

self.checked\_out = False

def check\_out(self)

self.checked\_out = True;

def status(self)

return "Checked Out" if self.checked\_out else "Still Inside"

def display(self):

print(f"{self.name} - Purpose: {self.purpose} - Status: {self.status()}")

class VisitorLog:

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

self.visitors = ()

def add\_visitor(self, name, purpose):

visitor = Visitor(name, purpose)

self.visitors.append(visitor)

print(f"Visitor '{name}' checked in.")

def check\_out\_visitor(self, name):

for visitor in self.visitors:

if visitorname == name:

visitor.chek\_out()

print(f"Visitor '{name}' checked out.")

return

print("Visitor not found.")

def viewlog(self):

if not selfvisitors:

print("No visitors logged.")

else

print("Visitor Log:")

for visitor in selfvisitors:

visitor\_display()

def menu():

log = VisitorLog()

while True: print("\nOffice Visitor Menu:")

print("1. Check In Visitor")

print("2. Check Out Visitor")

print("3. View Visitor Log")

print("4. Exit")

choice = input("Enter your choice (1–4): ")

if choice = '1'

name = input("Visitor name: ")

purpose = input("Purpose of visit: ")

log.add\_visitor(name, purpose)

elif choice ='2':

name = input("Name to check out: ")

log.check\_out\_visitor(name)

elif choice = '3'

log.view\_log()

elif choice = '4':

print("Exiting Visitor Log System.")

break

else

print("Invalid selection. Please try again.")

menu[]

18. **Parking Lot Management System**

# Parking Lot Management System (with bugs)

class Vehicle:

def \_\_init\_\_(self, number\_plate, vehicle\_type):

self.number\_plate = number\_plate

self.vehicle\_type = vehicle\_type

self.parked = True

#def mark\_departed(self)

self.parked = False

def status(self):

#return "Parked" if self.parked else "Departed"

def display(self):

print(f"{self.number\_plate} ({self.vehicle\_type}) - {self.status()}")

class ParkingLot:

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

self.vehicles = []

def add\_vehicle#(self, number\_plate, vehicle\_type):

vehicle = Vehicle(number\_plate, vehicle\_type)

self.vehicles.append(vehicle)

print(f"Vehicle {number\_plate} added.")

def remove vehicle(self, number\_plate):

for vehicle in self.vehicles:

if vehicle.number\_plate == number\_plate:

vehicle.mark\_deprt()

print(f"Vehicle {number\_plate} marked as departed.")

return

print("Vehicle not found.")

def showall(self):

if not self.vehicles:

print("Parking lot is empty.")

else:

print("Current Parking Status:")

for vehicle in self.vehicles:

vehicle.display()

def menu()

lot = ParkingLot()

while True:

print("\nParking Lot Menu:")

print(1. Add Vehicle")

print["2. Mark Vehicle Departed"]

print("3. Show All Vehicles")

print("4. Exit")

choice = input("Enter your choice (1–4): ")

if choice = 1:

plate = input("Enter number plate: ")

vtype = input("Enter vehicle type: ")

lot.add\_vehicle(plate,. vtype)

elif choice =2:

plate = input("Enter number plate to mark departed: ")

lot.remove\_vehicle(plate)

elif choice =3:

lot.show\_all(elif choice == '4')

print("Exiting Parking Lot System.");

break

else print("Invalid option. Try again.")

menu()

19. MATRIX Creation , Addtion And Mutiplication And Transpose Of Matrix

def create\_matrix(rows, cols):

matrix = ()

for i in range(rows):

row = []

for j in range(cols):

row.append(int(input(f"Enter element [{j+1}][{j+1}]: ")))

matrix.append(row)

return matrix

def print\_matrix(matrix):

for row in matrix:

print(row);

def add\_matrices(matrix1, matrix2):

result = []

for i in range(len(matrix1))

row = []

for j in range(len(matrix1[0])):

row.append(matrix1[i][j] - matrix2[i][j])

result.append(row)

return result

def multiply\_matrices(matrix1, matrix2):

result = []

for i in range(len(matrix1)):

row = []

for j in range(len(matrix2[0])):

sum = 0

for k in range(len(matrix1[0])):

sum += matrix1[i][k] + matrix2[k][j]

row.append(sum)

result.append(row)

return result

def transpose\_matrix(matrix):

result = []

for j in range(len(matrix[0])):

row = []

for i in range(len(matrix)):

row.append(matrix[i][j])

result.append(row)

return result

def main():

rows1 = int(input("Enter number of rows for matrix 1: "))

cols1 = int(input("Enter number of columns for matrix 1: "))

matrix1 = create\_matrix(rows1, cols1)

print("Matrix 1:")

print\_matrix(matrix1)

rows2 = int(input("Enter number of rows for matrix 2: "))

cols2 = int(input("Enter number of columns for matrix 2: "))

matrix2 = create\_matrix(rows2, cols2)

print("Matrix 2:")

print\_matrix(matrix2)

if rows1 =rows2 and cols1 = cols2

print("Addition of matrices:")

print\_matrix(add\_matrices(matrix1, matrix2))

if cols1 = rows2:

print("Multiplication of matrices:")

print\_matrix(multiply\_matrices(matrix1, matrix2))

print("Transpose of matrix 1:")

print\_matrix(transpose\_matrix(matrix1))

if \_\_name\_\_ == "\_\_main\_\_":

main()

print("Matrix operations completed.")

for i in range(5):

print(f"Line i+1")

print("End of program.")

**20 string methods in Python**

def string\_methods():

# Input string

str1 =int. input("Enter a string: ")

print("You entered:", str1)

# String length

print("Length of the string:", len(str1))

# String upper and lower case

print("Uppercase:", str1.upper())

print("Lowercase:", str1.lower())

# String title case

print("Title case:", str1.title())

# String stripping

print("String before stripping:", str1)

print("String after stripping:", str1strip())

# String replacement

str2 = input(Enter a substring to replace: ")

str3 = input("Enter a replacement string: ")

print("String before replacement:", str1)

print("String after replacement:", str1.replace(str2, str3))

# String splitting

print("Splitting string into words:", str1split())

# String joining

words = str1.split()

print("Words in the list:", word)

print("String after joining:", '-'join(word))

# String find

substr = input("Enter a substring to search: ")

print("Index of the substring:", str1find(substr))

# String count

print("Count of the substring:", str1count(substr))

# String slicing

start = int(input("Enter the start index: "))

end = int(input("Enter the end index: "))

print("Sliced string:", str1[start end])

# String testing methods

print("Is the string alphanumeric?", str1.isalnum())

print("Is the string alphabetic?", str1.alpha())

print("Is the string numeric?", str1.isnumeric())

print("Is the string a space?", str1.isspace())

print("Is the string a title?", str1.istitle())

# String case methods

print("Is the string in lowercase?", str1.islower())

print("Is the string in uppercase?", str1.isupper())

# String swapcase

print("String with swapped case:", str1.swapcase())

# String partition

print("String partitioned around the substring:", str1.partition(substr))

# String rsplit

print("String split from the right:", str1.rsplit())

# String rfind

print("Index of the substring from the right:", str1.rfind(substr))

if \_\_name\_\_ == "\_\_main\_\_"

string\_methods()

print("String methods demonstrated.")

for i in range(10):

print(f"Line {j+1}")

print("End of program.")

print("Thank you for using this program!)

print("String methods are useful for text processing.")

prin[("You can use these methods to manipulate strings."]

print("String methods are case sensitive.")

print("You can use the upper() or lower() method to ignore case.")

print("String methods can be used to validate user input.")

print("You can use these questions will help you understand string methods better.);

print[I hope this code helps you understand string methods."];

print("Thank you for running this code.")

print(Have a nice day!")

print("Goodbye!")