

TASK-1

TODOLIST

Since the command-line interface version is simpler to use and comprehend, let's start there.

Key Features:

- Add tasks to the to-do list.
- View the list of tasks.
- Update tasks (mark as complete, edit description, etc.).
- Delete tasks.
- Save tasks to a file (for persistence between sessions).
- Optionally, add due dates or priorities.

A To-Do List Application's Basic Design Using a CLI(Command line Interface)
Structure of Data: Lists of dictionaries, each containing a task's description, status, and priority, would be the most basic data structure for describing tasks.

Creating a CLI based application using python programming

CODE:

```
import json

# Load tasks from file

def load_tasks(filename='tasks.json'):

    try:

        with open(filename, 'r') as file:

            tasks = json.load(file)

    except FileNotFoundError:
```

```

        tasks = []

    return tasks

# Save tasks to file

def save_tasks(tasks, filename='tasks.json'):

    try:

        with open(filename, 'w') as file:

            json.dump(tasks, file, indent=4)

            print("Tasks saved successfully.")

        except Exception as e:

            print(f"Error saving tasks: {e}")

# Add a new task

def add_task(tasks):

    description = input("Enter task description: ")

    tasks.append({

        'description': description,

        'completed': False

    })

    print(f"Task '{description}' added.")

# Display all tasks

def display_tasks(tasks):

    if not tasks:

        print("No tasks in the list.")

```

else:

for idx, task in enumerate(tasks, start=1):

status = "✓" if task['completed'] else "✗"

print(f"{idx}. {task['description']} - {status}")

Mark a task as complete

def complete_task(tasks):

display_tasks(tasks)

try:

task_num = int(input("Enter the task number to mark as complete: "))

if 0 < task_num <= len(tasks):

tasks[task_num - 1]['completed'] = True

print(f"Task {task_num} marked as complete.")

else:

print("Invalid task number.")

except ValueError:

print("Please enter a valid number.")

Delete a task

def delete_task(tasks):

display_tasks(tasks)

try:

task_num = int(input("Enter the task number to delete: "))

if 0 < task_num <= len(tasks):

```

    task = tasks.pop(task_num - 1)

    print(f'Task '{task['description']}' deleted.")

else:

    print("Invalid task number.")

except ValueError:

    print("Please enter a valid number.")

# Main function to drive the program

def main():

    tasks = load_tasks()

    while True:

        print("\nTo-Do List Application")

        print("1. Add a Task")

        print("2. View All Tasks")

        print("3. Complete a Task")

        print("4. Delete a Task")

        print("5. Exit")

        choice = input("Choose an option: ")

        if choice == '1':

            add_task(tasks)

        elif choice == '2':

            display_tasks(tasks)

```

```

elif choice == '3':

    complete_task(tasks)

elif choice == '4':

    delete_task(tasks)

elif choice == '5':

    # Ask for confirmation before exiting

    confirm = input("Are you sure you want to exit? (y/n): ").lower()

    if confirm == 'y':

        save_tasks(tasks) # Save tasks before exit

        print("Goodbye!")

        break # Exit the loop and program

    else:

        print("Exit canceled.")

else:

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

if __name__ == "__main__":

    main()

```

OUTPUT:

To-Do List Application

1. Add a Task
2. View All Tasks
3. Complete a Task

4. Delete a Task

5. Exit

Choose an option: 1

Enter task description: Writing Records,Cleaning the room

Task 'Writing Records,Cleaning the room' added.

To-Do List Application

1. Add a Task

2. View All Tasks

3. Complete a Task

4. Delete a Task

5. Exit

Choose an option: 2

1. Writing Records,Cleaning the room - **X**

To-Do List Application

1. Add a Task

2. View All Tasks

3. Complete a Task

4. Delete a Task

5. Exit

Choose an option: 3

1. Writing Records,Cleaning the room - **X**

Enter the task number to mark as complete: 1

Task 1 marked as complete.

To-Do List Application

1. Add a Task
2. View All Tasks
3. Complete a Task
4. Delete a Task
5. Exit

Choose an option: 4

1. Writing Records,Cleaning the room - ✓

Enter the task number to delete: 1

Task 'Writing Records,Cleaning the room' deleted.

To-Do List Application

1. Add a Task
2. View All Tasks
3. Complete a Task
4. Delete a Task
5. Exit

Choose an option: 5

Are you sure you want to exit? (y/n): Yes

Exit canceled.

Explanation:

- **load_tasks():** This function attempts to load the list of tasks from a file (tasks.json). If the file doesn't exist, it initializes an empty list.
- **save_tasks():** It saves the list of tasks to a file in JSON format, which allows persistence between sessions.
- **add_task():** Takes user input to add a new task to the list.
- **display_tasks():** Displays all tasks with a status indicator (✓ for complete, ✗ for incomplete).
- **complete_task():** Allows the user to mark a task as complete by its number.
- **delete_task():** Removes a task from the list.
- **main():** The main function that drives the menu-based interface, allowing the user to choose different options.

TASK-2

CALCULATOR

This is a straightforward Python calculator software that can handle addition, subtraction, multiplication, and division, among other fundamental arithmetic operations. When the user enters two numbers and an operation, the application calculates the outcome and shows it.

PYTHON CODE:

```
# Simple Calculator with basic arithmetic operations
```



```
def add(x, y):

    return x + y

def subtract(x, y):

    return x - y

def multiply(x, y):

    return x * y

def divide(x, y):

    if y == 0:

        return "Error! Division by zero."

    else:

        return x / y

def calculator():

    print("Welcome to the Simple Calculator")

# Prompt user for input

    try:

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

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

    except ValueError:

        print("Invalid input. Please enter numeric values.")

    return

# Display operation choices

    print("\nSelect operation:")
```

```

print("1. Add")

print("2. Subtract")

print("3. Multiply")

print("4. Divide")

choice = input("Enter your choice (1/2/3/4): ")

# Perform the selected operation

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':

    print(f'{num1} / {num2} = {divide(num1, num2)}')

else:

    print("Invalid choice. Please select a valid operation.")

# Run the calculator

calculator()

```

OUTPUT:

Welcome to the Simple Calculator

Enter the first number: 10

Enter the second number: 15

Select operation:

1. Add
2. Subtract
3. Multiply
4. Divide

Enter your choice (1/2/3/4): 1

$$10.0 + 15.0 = 25.0$$

The Function of the Program:

Functions for Arithmetic Operations:

Add(x, y) is the addition function.
To subtract, use subtract(x, y).
For multiplication, use multiply(x, y). To prevent division by zero mistake, divide(x, y) also verifies if the denominator is zero.

User Input: Two digits must be entered by the user. After that, the user is asked to select an operation (addition, subtraction, multiplication, or division).

Error Handling: An error message appears if the user inputs text that is not a number.

Rather of crashing the application, division by zero is handled and an error message is shown.

Carrying out the Operation:

The relevant arithmetic function is run based on the user's selection, and the outcome is shown.

TASK-3

Rock-Paper-Scissors Game

This Python software replicates the gameplay of the game Rock, Paper, Scissors. In accordance with the game's rules, the software asks the user to select between rock, paper, or scissors, then creates a random selection for the computer. It also offers the ability to play again and score monitoring for several rounds.

PYTHON CODE:

```
import random

# Function to get the computer's choice

def get_computer_choice():

    return random.choice(['rock', 'paper', 'scissors'])

# Function to determine the winner

def determine_winner(user_choice, computer_choice):

    if user_choice == computer_choice:

        return "It's a tie!"

    if (user_choice == 'rock' and computer_choice == 'scissors') or \

        (user_choice == 'scissors' and computer_choice == 'paper') or \

        (user_choice == 'paper' and computer_choice == 'rock'):

        return "You win!"

    else:

        return "Computer wins!"

# Function to display the current score
```

```
def display_score(user_score, computer_score):

    print(f"\nScoreboard:")

    print(f"User: {user_score} | Computer: {computer_score}")

# Main function to drive the game

def play_game():

    user_score = 0

    computer_score = 0

    print("Welcome to Rock, Paper, Scissors!")

    while True:

        # Get the user's choice

        user_choice = input("Enter rock, paper, or scissors (or 'exit' to quit): ").lower()

        if user_choice == 'exit':

            print("Thanks for playing!")

            display_score(user_score, computer_score)

            break

        if user_choice not in ['rock', 'paper', 'scissors']:

            print("Invalid choice! Please choose rock, paper, or scissors.")

            continue

        # Get the computer's choice

        computer_choice = get_computer_choice()

        print(f"Computer chose: {computer_choice}")
```

```
# Determine the winner

result = determine_winner(user_choice, computer_choice)

print(result)

# Update score based on result

if result == "You win!":

    user_score += 1

elif result == "Computer wins!":

    computer_score += 1

# Display the score after each round

display_score(user_score, computer_score)

# Ask if the user wants to play again

play_again = input("\nDo you want to play again? (y/n): ").lower()

if play_again != 'y':

    print("Thanks for playing!")

    display_score(user_score, computer_score)

    break

# Run the game

if __name__ == "__main__":

    play_game()
```

OUTPUT:

Welcome to Rock, Paper, Scissors!

Enter rock, paper, or scissors (or 'exit' to quit): paper

Computer chose: rock

You win!

Scoreboard:

User: 1 | Computer: 0

Do you want to play again? (y/n): no

Thanks for playing!

The Function of the Program:

The user is asked to select from scissors, rock, and paper. Upon entering "exit," the user concludes the game and the score is shown. The random.choice() function is utilized by the computer to make a random selection among rock, paper, or scissors.

Game Theory: These guidelines are used to decide who wins:

Scissors yield to Rock.

Paper is defeated by scissors.

Rock is beaten by Paper.

Should the user and the machine choose the same choice, a tie will occur.

Show Results: The application shows the results of each round, including the user's and the computer's selections, and it declares who won.

Score Monitoring (Optional Feature): Throughout several rounds, the application keeps track of the user's and computer's scores. Following every round, the current score is shown.

Play Once Again:

The user is prompted to continue playing after each round. The game proceeded if the user entered 'y'; if not, it ended.

User Interface: The interface is designed to be easy to use and straightforward, with prompts for every action and feedback given there after.

Explanation:

The random.choice(): method is used by the computer to make a random selection among rock, paper, or scissors.

Game Logic: According to the established rules of the game, the winner is decided. In the event that the user and the machine make the same decision, a tie results.

Easy to Use Interface: After every round, the application shows the options and results, giving users direct feedback.

Score tracking: The scores of both the user and the machine are kept track of across several rounds, with updates made at the conclusion of each game.