Day 8 of 6 weeks Python course:

Time Breakdown (2 Hours)

Time Slot	Task
0:00 - 0:20	Introduction to Functions
0:20 - 0:40	Function Arguments & Return Values
0:40 - 1:00	Lambda (Anonymous) Functions
1:00 - 1:20	Recursive Functions
1:20 - 2:00	Mini-Project: Simple To-Do List

1️⃣ What are Functions? ✅ Definition: A function is a block of reusable code that performs a specific task. ✅ Syntax:

In [ ]:
```
def function_name():
    # Code block
```

✅ Example Usage:

In [1]:
```
def greet():
    print("Hello, David!")

greet()  # Function call
```

```
Hello, David!
```

✅ Why Use Functions? 1.Avoid code repetition. 2.Make code modular. 3.Easier debugging. 2️⃣ Function Arguments & Return Values ✅ Functions with Parameters

In [5]:
```
def greet(name):
    print(f"Hello, {name}!")

greet("Ananya")
```

```
Hello, Ananya!
```

✅ Functions with Multiple Parameters:

In [7]:
```
def add(a, b):
    return a + b

result = add(10, 5)
print("Sum:", result)
```

```
Sum: 15
```

✅ Default Parameters:

In [11]:
```
def greet(name="Guest"):
    print(f"Hello, {name}!")

greet()  # Uses default value
greet("David")  # Overrides default value
```

```
Hello, Guest!
Hello, David!
```

In [ ]:
```
3️⃣ Lambda (Anonymous) Functions

✅ What is a Lambda Function?


1.A one-line function without a name.

2.Used for short & simple operations.

✅ Syntax:
```

In [13]:
```
lambda arguments: expression
```

Out[13]:
```
<function __main__.<lambda>(arguments)>
```

✅ Example Usage:

In [17]:
```
square = lambda x: x * x
print(square(5))  # Output: 25
```

```
25
```

✅ Lambda with Multiple Arguments:

In [21]:
```
add = lambda a, b: a + b
print(add(3, 4))  # Output: 7
```

```
7
```

4️⃣ Recursive Functions ✅ What is Recursion? 1.A function calls itself to solve a smaller subproblem. ✅ Factorial Example Using Recursion

In [25]:
```
def factorial(n):
    if n == 1:
        return 1
    return n * factorial(n - 1)

print(factorial(5))  # Output: 120
```

```
120
```

✅ Fibonacci Series Using Recursion:

In [29]:
```
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)

print(fibonacci(6))  # Output: 8
```

```
8
```

🎯 Mini-Project: Simple To-Do List using Functions 📌 Project Goal 1.Allow users to add, view, and remove tasks. 2.Store tasks in a list. 3.Use functions to manage tasks. 🖥️ Code Implementation:

In [31]:
```
# To-Do List Storage
tasks = []

# Function to add a task
def add_task():
    task = input("Enter a new task: ")
```

```
    tasks.append(task)
    print(f"Task '{task}' added successfully!")

# Function to view tasks
def view_tasks():
    if not tasks:
        print("No tasks found.")
    else:
        print("\n📌 To-Do List 📌")
        for i, task in enumerate(tasks, start=1):
            print(f"{i}. {task}")

# Function to delete a task
def delete_task():
    view_tasks()
    if tasks:
        try:
            task_no = int(input("Enter task number to delete: "))
            removed = tasks.pop(task_no - 1)
            print(f"Task '{removed}' deleted successfully!")
        except (IndexError, ValueError):
            print("Invalid task number!")

# Main Menu
while True:
    print("\n📝 To-Do List Manager 📝")
    print("1. Add Task")
    print("2. View Tasks")
    print("3. Delete Task")
    print("4. Exit")

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

    if choice == "1":
        add_task()
    elif choice == "2":
        view_tasks()
    elif choice == "3":
        delete_task()
    elif choice == "4":
        print("Exiting To-Do List. Goodbye!")
        break
    else:
        print("Invalid choice! Please enter a valid option.")
```

```
📝 To-Do List Manager 📝
1. Add Task
2. View Tasks
3. Delete Task
4. Exit
```

```
Task 'talk to mom' added successfully!

📝 To-Do List Manager 📝
1. Add Task
2. View Tasks
3. Delete Task
4. Exit
Task 'learn python' added successfully!

📝 To-Do List Manager 📝
1. Add Task
2. View Tasks
3. Delete Task
4. Exit
📌 To-Do List 📌
1. talk to mom
2. learn python

📝 To-Do List Manager 📝
1. Add Task
2. View Tasks
3. Delete Task
4. Exit
📌 To-Do List 📌
1. talk to mom
2. learn python
Task 'talk to mom' deleted successfully!

📝 To-Do List Manager 📝
1. Add Task
2. View Tasks
3. Delete Task
4. Exit
Exiting To-Do List. Goodbye!
```

🔍 Step-by-Step Explanation

📌 Step 1: Storing Tasks in a List

python Copy Edit

```python
tasks = []
```

- Uses a **list** to store tasks.

📌 Step 2: Adding a Task

python Copy Edit

```python
def add_task():
    task = input("Enter a new task: ")
    tasks.append(task)
    print(f"Task '{task}' added successfully!")
```

- Takes user input and **appends it to the list.**

📌 Step 3: Viewing Tasks

python Copy Edit

```python
def view_tasks():
    if not tasks:
        print("No tasks found.")
    else:
        print("\n📌 To-Do List 📌")
        for i, task in enumerate(tasks, start=1):
            print(f"{i}. {task}")
```

- Uses a **loop to print tasks**.

📌 Step 4: Deleting a Task

python Copy Edit

```python
def delete_task():
    view_tasks()
    if tasks:
        try:
            task_no = int(input("Enter task number to delete: "))
            removed = tasks.pop(task_no - 1)
            print(f"Task '{removed}' deleted successfully!")
        except (IndexError, ValueError):
            print("Invalid task number!")
```

- **Gets task number**, removes the selected task.
- Uses **error handling** to prevent crashes.

📌 Step 5: Using a Loop for Menu

python Copy Edit

```python
while True:
    print("\n📝 To-Do List Manager 📝")
    print("1. Add Task")
    print("2. View Tasks")
    print("3. Delete Task")
    print("4. Exit")

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

    if choice == "1":
        add_task()
    elif choice == "2":
        view_tasks()
    elif choice == "3":
        delete_task()
    elif choice == "4":
        print("Exiting To-Do List. Goodbye!")
        break
    else:
        print("Invalid choice! Please enter a valid option.")
```

- Runs **until the user exits.**

- Runs **until the user exits.**

✅ Example Output:

pgsql Copy Edit

```
📝 To-Do List Manager 📝
1. Add Task
2. View Tasks
3. Delete Task
4. Exit

Enter your choice (1-4): 1
Enter a new task: Learn Python
Task 'Learn Python' added successfully!

Enter your choice (1-4): 2

📌 To-Do List 📌
1. Learn Python

Enter your choice (1-4): 3
Enter task number to delete: 1
Task 'Learn Python' deleted successfully!
```

📌 Summary of Day 8 ✔ Learned Python Functions ✔ Practiced Function Arguments, Lambda, Recursion ✔ Completed a Mini-Project: To-Do List Manager