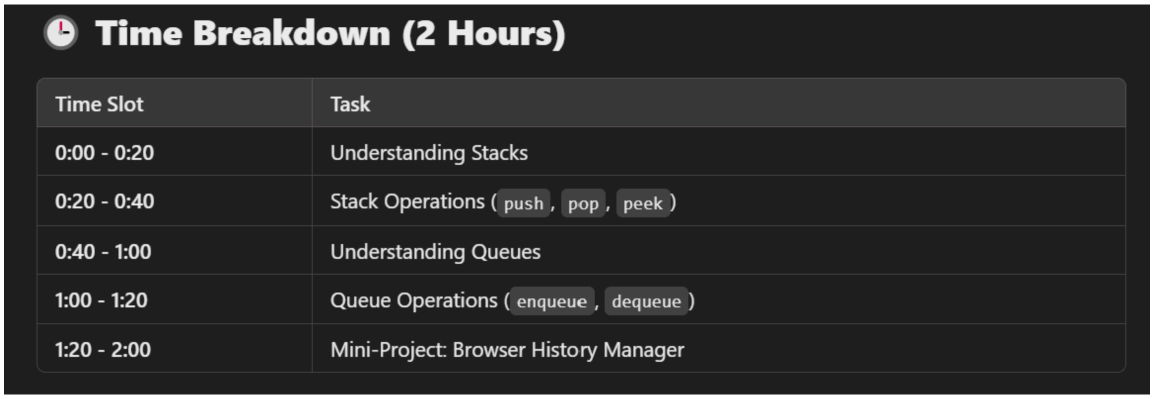
Day 13 of 6 weeks Python course:



🕒 Time Breakdown (2 Hours)

Time Slot	Task
0:00 - 0:20	Understanding Stacks
0:20 - 0:40	Stack Operations (`push`, `pop`, `peek`)
0:40 - 1:00	Understanding Queues
1:00 - 1:20	Queue Operations (`enqueue`, `dequeue`)
1:20 - 2:00	Mini-Project: Browser History Manager

1️⃣ What is a Stack? ✅ Definition: A Stack is a Last In, First Out (LIFO) data structure, meaning the last item added is the first one to be removed. ✅ Real-World Examples: •Undo/Redo in text editors •Browser history (Back button) •Call stack in recursion ✅ Stack Representation:Top → [5, 4, 3, 2, 1] (5 is the last item added, so it will be removed first) 2️⃣ Stack Operations (push, pop, peek) ✅ Stack Implementation Using List

In [6]:
```python
stack = []

# Push (Add to stack)
stack.append(10)
stack.append(20)
stack.append(30)

# Pop (Remove from stack)
print(stack.pop())  # Output: 30
print(stack.pop())  # Output: 20

# Peek (View top element)
print(stack[-1])  # Output: 10
```

```
30
20
10
```

✅ Stack Implementation Using Class

In [8]:
```python
class Stack:
    def __init__(self):
        self.stack = []

    def push(self, item):
        self.stack.append(item)

    def pop(self):
        if not self.is_empty():
            return self.stack.pop()
        return "Stack is empty!"

    def peek(self):
        if not self.is_empty():
            return self.stack[-1]
        return "Stack is empty!"
    def is_empty(self):
        return len(self.stack) == 0

stack = Stack()
stack.push(10)
stack.push(20)
print(stack.pop())  # Output: 20
print(stack.peek())  # Output: 10
```

```
20
10
```

💡 append() for push(), pop() for pop(), [-1] for peek(). ✅ Definition: A Queue is a First In, First Out (FIFO) data structure, meaning the first item added is the first one to be removed. ✅ Real-World Examples: •Line at a ticket counter •CPU scheduling •Printer job scheduling ✅ Queue Representation:Front → [1, 2, 3, 4, 5] → Rear (1 is the first item added, so it will be removed first) 4️⃣ Queue Operations (enqueue, dequeue) ✅ Queue Implementation Using List

In [14]:
```
queue = []

# Enqueue (Add to queue)
queue.append(10)
queue.append(20)
queue.append(30)

# Dequeue (Remove from queue)
print(queue.pop(0))  # Output: 10
print(queue.pop(0))  # Output: 20
```

```
10
20
```

✅ Queue Implementation Using Class

In [16]:
```
class Queue:
    def __init__(self):
        self.queue = []

    def enqueue(self, item):
        self.queue.append(item)

    def dequeue(self):
        if not self.is_empty():
            return self.queue.pop(0)
        return "Queue is empty!"

    def front(self):
        if not self.is_empty():
            return self.queue[0]
        return "Queue is empty!"

    def is_empty(self):
        return len(self.queue) == 0

queue = Queue()
queue.enqueue(10)
queue.enqueue(20)
```

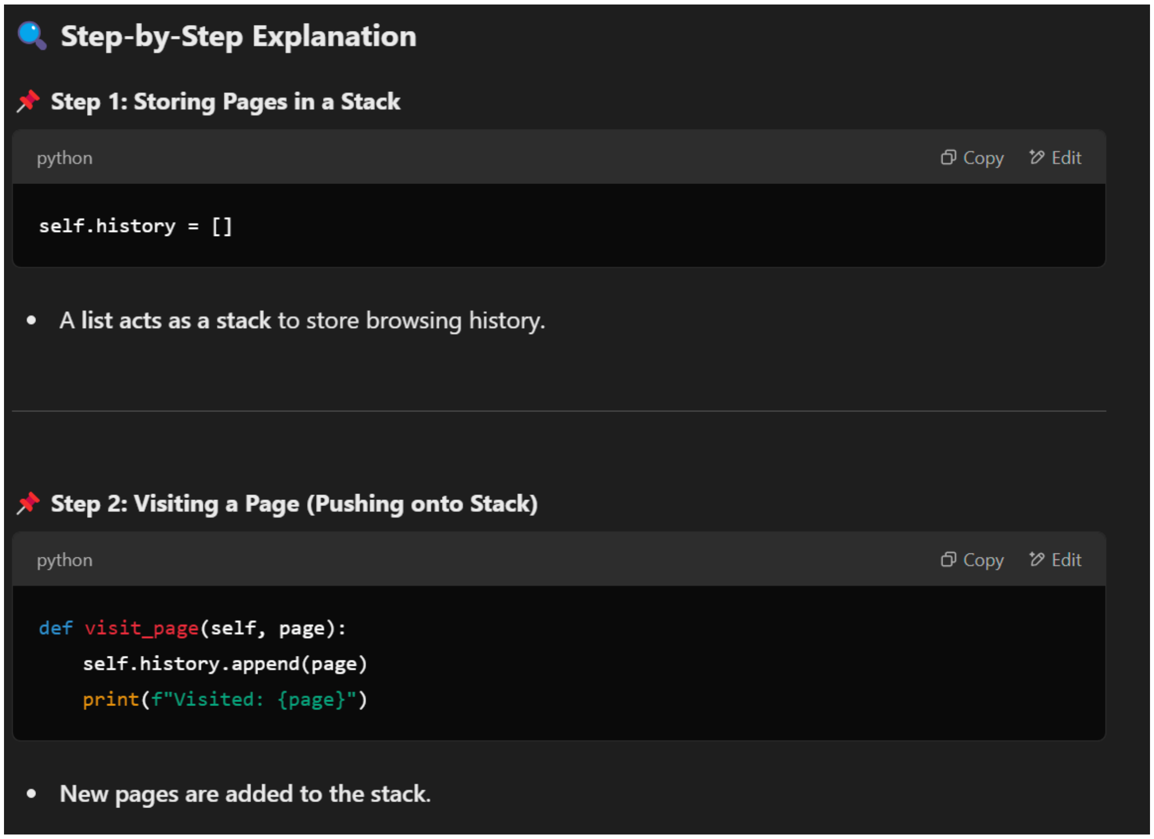
```
print(queue.dequeue())  # Output: 10
print(queue.front())  # Output: 20
```

```
10
20
```

💡 Use append() for enqueue() and pop(0) for dequeue(). 🎯 Mini-Project: Browser History Manager (Stack Implementation) 📌 Project Goal •Track visited web pages. •Allow users to go back (undo navigation). •Use a Stack (push() for new pages, pop() for going back). 🖥️ Code Implementation

In [20]:
```
class BrowserHistory:
    def __init__(self):
        self.history = []

    def visit_page(self, page):
        self.history.append(page)
        print(f"Visited: {page}")

    def go_back(self):
        if len(self.history) > 1:
            self.history.pop()  # Remove last visited page
            print(f"Going back to: {self.history[-1]}")
        else:
            print("No more history to go back to!")

    def show_history(self):
        print("\n📜 Browsing History 📜")
        for page in reversed(self.history):
            print(page)

# Create browser history object
browser = BrowserHistory()

# Menu-driven program
while True:
    print("\n🌍 Browser History Manager 🌍")
    print("1. Visit Page")
    print("2. Go Back")
    print("3. View History")
    print("4. Exit")

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

    if choice == "1":
        page = input("Enter website URL: ")
        browser.visit_page(page)
    elif choice == "2":
        browser.go_back()
    elif choice == "3":
        browser.show_history()
    elif choice == "4":
        print("Exiting Browser History Manager. Goodbye!")
        break
    else:
        print("Invalid choice! Please enter 1-4.\n")
```

```
🌍 Browser History Manager 🌍
1. Visit Page
2. Go Back
3. View History
4. Exit
Visited: https://chat.deepseek.com/

🌍 Browser History Manager 🌍
1. Visit Page
2. Go Back
3. View History
4. Exit
No more history to go back to!

🌍 Browser History Manager 🌍
1. Visit Page
2. Go Back
3. View History
4. Exit
📜 Browsing History 📜
https://chat.deepseek.com/

🌍 Browser History Manager 🌍
1. Visit Page
2. Go Back
3. View History
4. Exit
Exiting Browser History Manager. Goodbye!
```



🔍 Step-by-Step Explanation

📌 Step 1: Storing Pages in a Stack

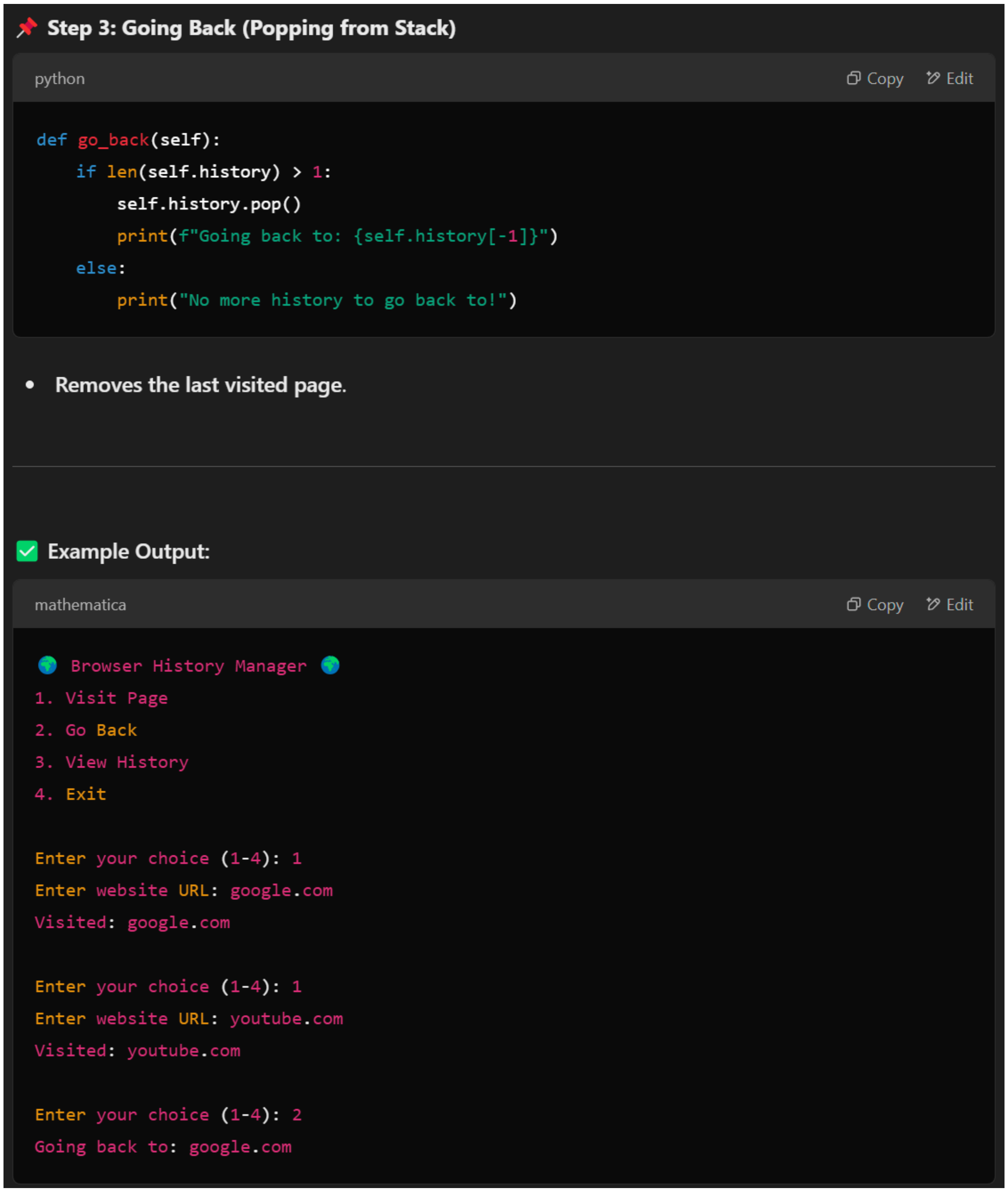
```python
self.history = []
```

- **A list acts as a stack** to store browsing history.

📌 Step 2: Visiting a Page (Pushing onto Stack)

```python
def visit_page(self, page):
    self.history.append(page)
    print(f"Visited: {page}")
```

- **New pages are added to the stack.**



📌 **Step 3: Going Back (Popping from Stack)**

python

```python
def go_back(self):
    if len(self.history) > 1:
        self.history.pop()
        print(f"Going back to: {self.history[-1]}")
    else:
        print("No more history to go back to!")
```

- **Removes the last visited page.**

✅ **Example Output:**

mathematica

```
🌍 Browser History Manager 🌍
1. Visit Page
2. Go Back
3. View History
4. Exit

Enter your choice (1-4): 1
Enter website URL: google.com
Visited: google.com

Enter your choice (1-4): 1
Enter website URL: youtube.com
Visited: youtube.com

Enter your choice (1-4): 2
Going back to: google.com
```

📌 Summary of Day 13 ✔ Learned Stacks & Queues ✔ Practiced Stack & Queue Operations ✔ Completed a Mini-Project: Browser History Manager