


# Threading in Python


Understanding Threading with Examples



# What is Threading?

- ▶ – A thread is a lightweight process that can run concurrently with other threads.
  - ▶ – Threads share the same memory space, making communication between them easy.
  - ▶ – Threading is used to run multiple tasks simultaneously, ideal for I/O-bound operations.
- 

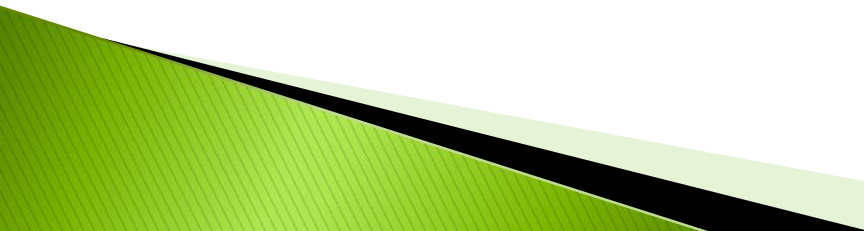
# Why Use Threading?

- ▶ – Perform multiple tasks at the same time without blocking.
  - ▶ – Useful for I/O-bound tasks (e.g., file I/O, network operations).
  - ▶ – Threads allow parallel execution of tasks in the background.
- 

# Basic Example of Threading

- import threading
- import time
- def print\_numbers():
- for i in range(5):
- print(f"Number: {i}")
- time.sleep(1)
- def print\_letters():
- for letter in ['A', 'B', 'C', 'D', 'E']:
- print(f"Letter: {letter}")
- time.sleep(1.5)
- # Create and start threads
- thread1 = threading.Thread(target=print\_numbers)
- thread2 = threading.Thread(target=print\_letters)
- thread1.start()
- thread2.start()
- thread1.join()
- thread2.join()

# Threading with Arguments

- `import threading`
  - `import time`
  - `def greet(name, times):`
  - `for _ in range(times):`
  - `print(f"Hello, {name}!")`
  - `time.sleep(1)`
  - `# Create and start threads with arguments`
  - `thread1 = threading.Thread(target=greet, args=("Alice", 3))`
  - `thread2 = threading.Thread(target=greet, args=("Bob", 5))`
  - `thread1.start()`
  - `thread2.start()`
  - `thread1.join()`
  - `thread2.join()`
- 

# Synchronization with Locks

- `import threading`
- `counter = 0`
- `lock = threading.Lock()`
- `def increase_counter():`
  - `global counter`
  - `for _ in range(1000):`
    - `with lock:`
      - `counter += 1`
- `# Create and start threads`
- `thread1 = threading.Thread(target=increase_counter)`
- `thread2 = threading.Thread(target=increase_counter)`
- `thread1.start()`
- `thread2.start()`
- `thread1.join()`
- `thread2.join()`
- `print(f"Final counter value: {counter}")`

# Summary

- ▶ – Threads allow multiple tasks to run concurrently.
  - ▶ – Use threading for I/O-bound tasks (network, file I/O).
  - ▶ – Synchronization with locks is crucial for shared resources.
- 