#PYTHONTIPS

CONCURRENCY VS PARALLELISM

UNRAVELING THE KEY DIFFERENCES

@VEDANTSOLANKI

CONCURRENCY

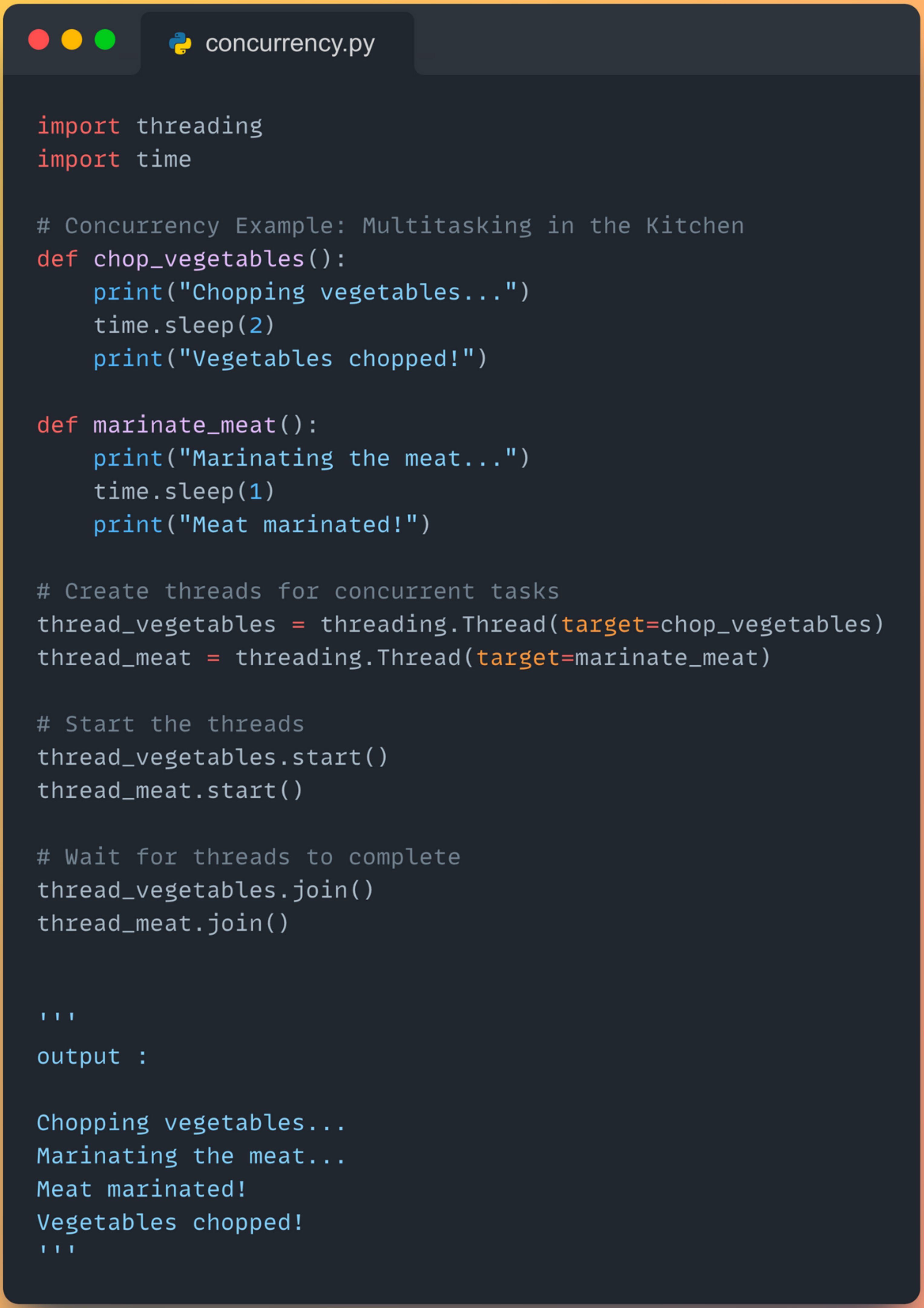
- Concurrency is like multitasking in the kitchen, where you can work on multiple tasks simultaneously without doing them at the exact same time.
- Imagine you're cooking a delicious meal. While you chop vegetables, you can also marinate the meat, boil water, and prepare the ingredients concurrently. Each task progresses independently, and you switch between them efficiently.

PARALLELISM

- Parallelism is like collaborating with others in the kitchen, where each person focuses on a specific task, and multiple tasks are executed simultaneously.
- In parallelism, you can divide the tasks among a group of people. For example, one person chops vegetables, another grills the meat, and someone else boils the pasta. Everyone works together independently, and all tasks progress simultaneously.

#PYTHONTIPS

CONCURRENCY EXAMPLE


concurrency.py

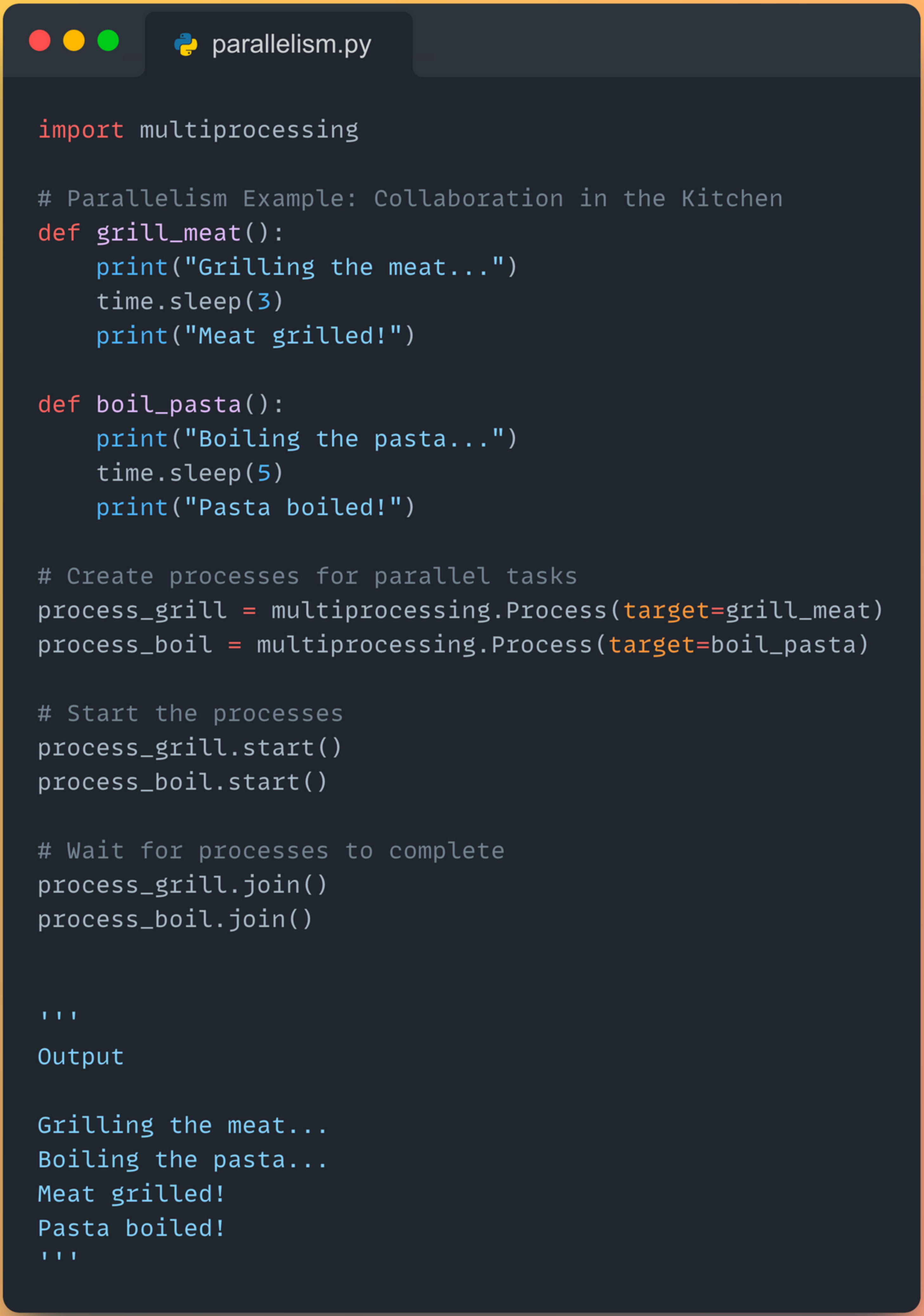
```python
import threading
import time

# Concurrency Example: Multitasking in the Kitchen
def chop_vegetables():
    print("Chopping vegetables...")
    time.sleep(2)
    print("Vegetables chopped!")

def marinate_meat():
    print("Marinating the meat...")
    time.sleep(1)
    print("Meat marinated!")

# Create threads for concurrent tasks
thread_vegetables = threading.Thread(target=chop_vegetables)
thread_meat = threading.Thread(target=marinate_meat)

# Start the threads
thread_vegetables.start()
thread_meat.start()

# Wait for threads to complete
thread_vegetables.join()
thread_meat.join()


'''
output :

Chopping vegetables...
Marinating the meat...
Meat marinated!
Vegetables chopped!
'''
```

@VEDANTSOLANKI

- In above example, we have two concurrent tasks: chop_vegetables() and marinate_meat().
- Each task represents a separate thread. When the threads are started, the tasks run concurrently.
- While chopping vegetables, the meat can be marinated simultaneously. The time.sleep() function simulates the time taken to perform each task.
- Once the threads complete their respective tasks, the results are printed.

#PYTHONTIPS

PARALLELISM EXAMPLE



```python
import multiprocessing

# Parallelism Example: Collaboration in the Kitchen
def grill_meat():
    print("Grilling the meat...")
    time.sleep(3)
    print("Meat grilled!")

def boil_pasta():
    print("Boiling the pasta...")
    time.sleep(5)
    print("Pasta boiled!")

# Create processes for parallel tasks
process_grill = multiprocessing.Process(target=grill_meat)
process_boil = multiprocessing.Process(target=boil_pasta)

# Start the processes
process_grill.start()
process_boil.start()

# Wait for processes to complete
process_grill.join()
process_boil.join()


'''
Output

Grilling the meat...
Boiling the pasta...
Meat grilled!
Pasta boiled!
'''
```

@VEDANTSOLANKI

- In above example, we have two parallel tasks: grill_meat() and boil_pasta().
- Each task represents a separate process. The processes are started, allowing the tasks to execute simultaneously and independently.
- The time.sleep() function simulates the time taken to perform each task.
- Once the processes complete their respective tasks, the results are printed.

CONCURRENCY USECASE

- Real-time Chat Applications: In chat applications like WhatsApp or Slack, multiple users can send and receive messages concurrently. Each user's message is processed independently, allowing for smooth and real-time communication.
- E-commerce Websites: On e-commerce platforms, multiple users can browse, search for products, and add items to their shopping carts concurrently. Concurrency ensures that each user's actions are processed independently and accurately.

- Online Collaboration Tools: Platforms like Google Docs, where multiple users can simultaneously edit a document, leverage concurrency to manage concurrent edits and ensure data integrity.
- Online Ticket Booking: When multiple users try to book tickets for a popular event simultaneously, concurrency ensures that each user's booking request is processed independently and in real-time.

PARALLELISM USECASE

- Data Sorting: Parallelism can be employed to sort large datasets concurrently using different algorithms, which significantly reduces the time required for sorting.
- Cryptocurrency Mining: Mining cryptocurrencies like Bitcoin requires solving complex cryptographic puzzles. Miners use parallel processing power to increase their chances of solving these puzzles and earning rewards.

- Video Transcoding: Video streaming services like YouTube or Netflix use parallelism to transcode videos into multiple formats, suitable for different devices and network conditions.
- Batch Image Processing: Image processing applications often apply filters or perform editing operations on multiple images in parallel, significantly speeding up the batch processing of images.

SUMMARY

- Concurrency allows for multitasking and efficient switching between tasks, just like cooking multiple things in the kitchen.
- Parallelism enables true simultaneous execution with independent tasks, similar to collaborating with others in the kitchen.
- Concurrency optimizes responsiveness and resource utilization in I/O-bound tasks.
- Parallelism speeds up CPU-bound tasks by leveraging multiple CPU cores.

@TAG

SOMEONE WHO WILL FIND THIS HELPFUL

FOLLOW FOR MORE !

Vedant Solanki
@vedantsolanki