AI_AC(ASS-5.5)

BOLAGANI SUSHANTH

2403A52L11 – B50

Task-1:

Use AI to generate two solutions for checking prime

numbers:

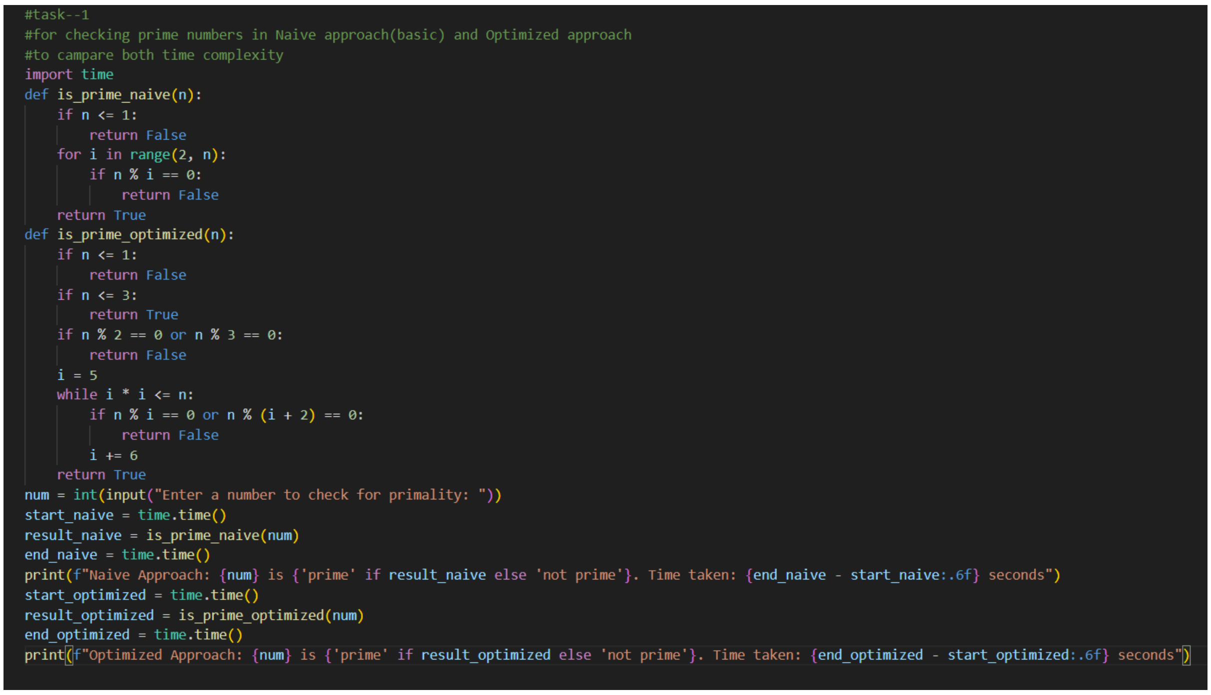
• Naive approach(basic)

• Optimized approach

Prompt: for checking prime numbers in Naive approach(basic) and Optimized approach

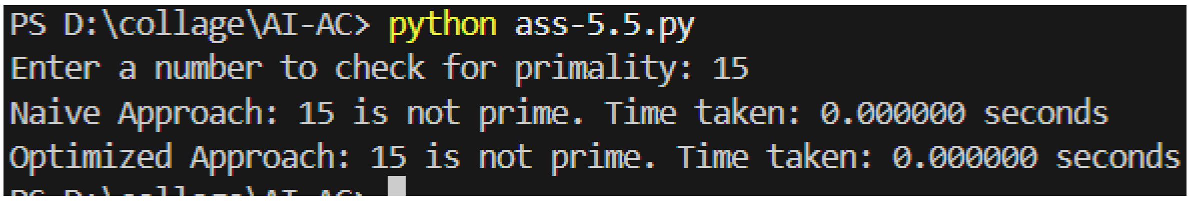
to compare both time complexity

code:



```python
#task--1
#for checking prime numbers in Naive approach(basic) and Optimized approach
#to campare both time complexity
import time
def is_prime_naive(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
def is_prime_optimized(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True
num = int(input("Enter a number to check for primality: "))
start_naive = time.time()
result_naive = is_prime_naive(num)
end_naive = time.time()
print(f"Naive Approach: {num} is {'prime' if result_naive else 'not prime'}. Time taken: {end_naive - start_naive:.6f} seconds")
start_optimized = time.time()
result_optimized = is_prime_optimized(num)
end_optimized = time.time()
print(f"Optimized Approach: {num} is {'prime' if result_optimized else 'not prime'}. Time taken: {end_optimized - start_optimized:.6f} seconds")
```

Output:



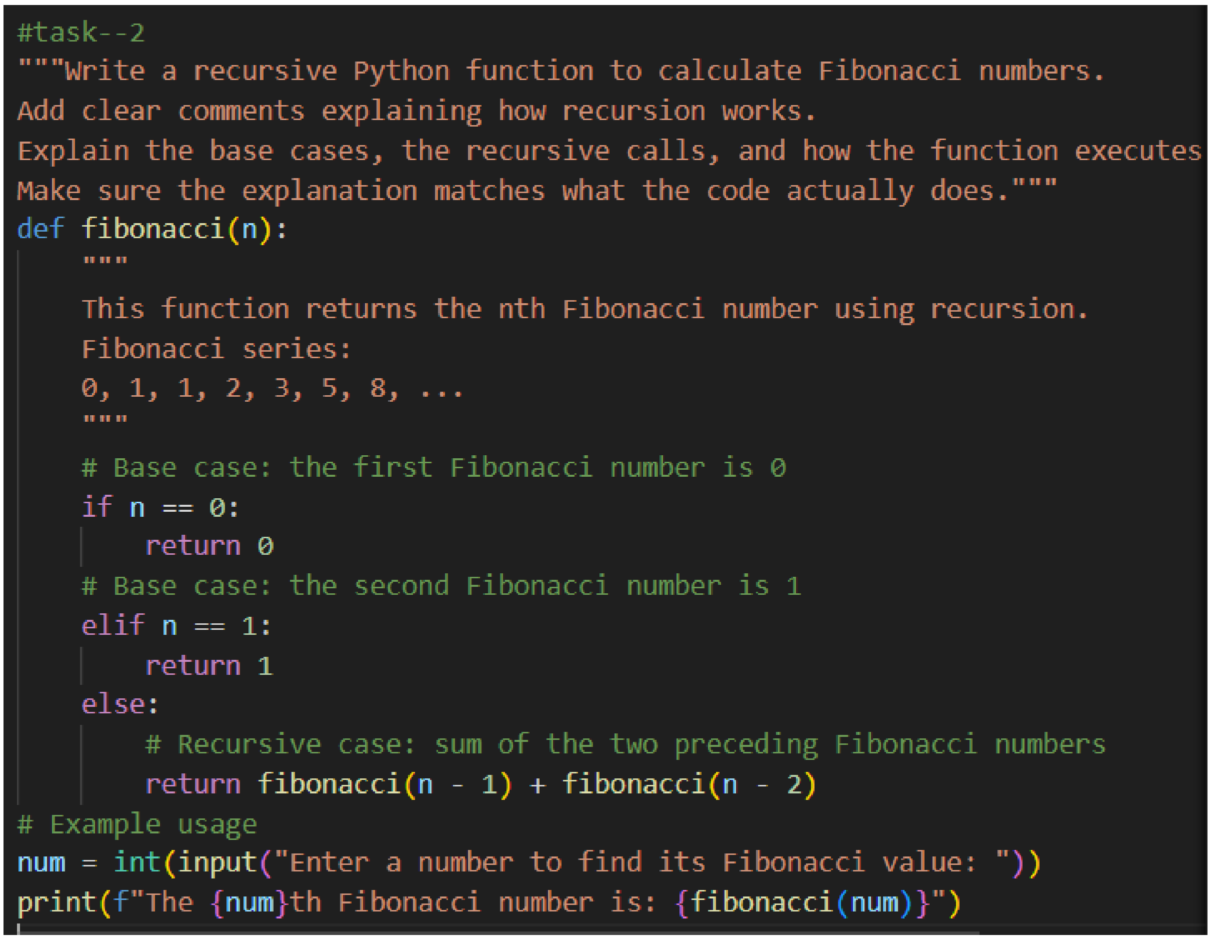
```
PS D:\collage\AI-AC> python ass-5.5.py
Enter a number to check for primality: 15
Naive Approach: 15 is not prime. Time taken: 0.000000 seconds
Optimized Approach: 15 is not prime. Time taken: 0.000000 seconds
```

Task-2: (Transparency in Recursive Algorithms)

Objective: Use AI to generate a recursive function to calculate

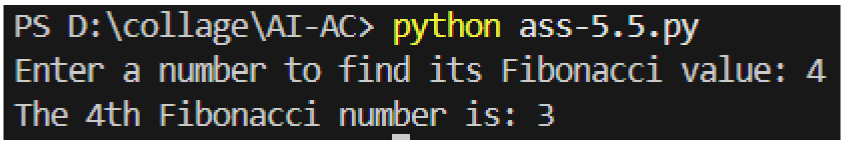
Fibonacci numbers.

Prompt and Code:



```
#task--2
"""Write a recursive Python function to calculate Fibonacci numbers.
Add clear comments explaining how recursion works.
Explain the base cases, the recursive calls, and how the function executes
Make sure the explanation matches what the code actually does."""
def fibonacci(n):
    """
    This function returns the nth Fibonacci number using recursion.
    Fibonacci series:
    0, 1, 1, 2, 3, 5, 8, ...
    """
    # Base case: the first Fibonacci number is 0
    if n == 0:
        return 0
    # Base case: the second Fibonacci number is 1
    elif n == 1:
        return 1
    else:
        # Recursive case: sum of the two preceding Fibonacci numbers
        return fibonacci(n - 1) + fibonacci(n - 2)
# Example usage
num = int(input("Enter a number to find its Fibonacci value: "))
print(f"The {num}th Fibonacci number is: {fibonacci(num)}")
```

Output:



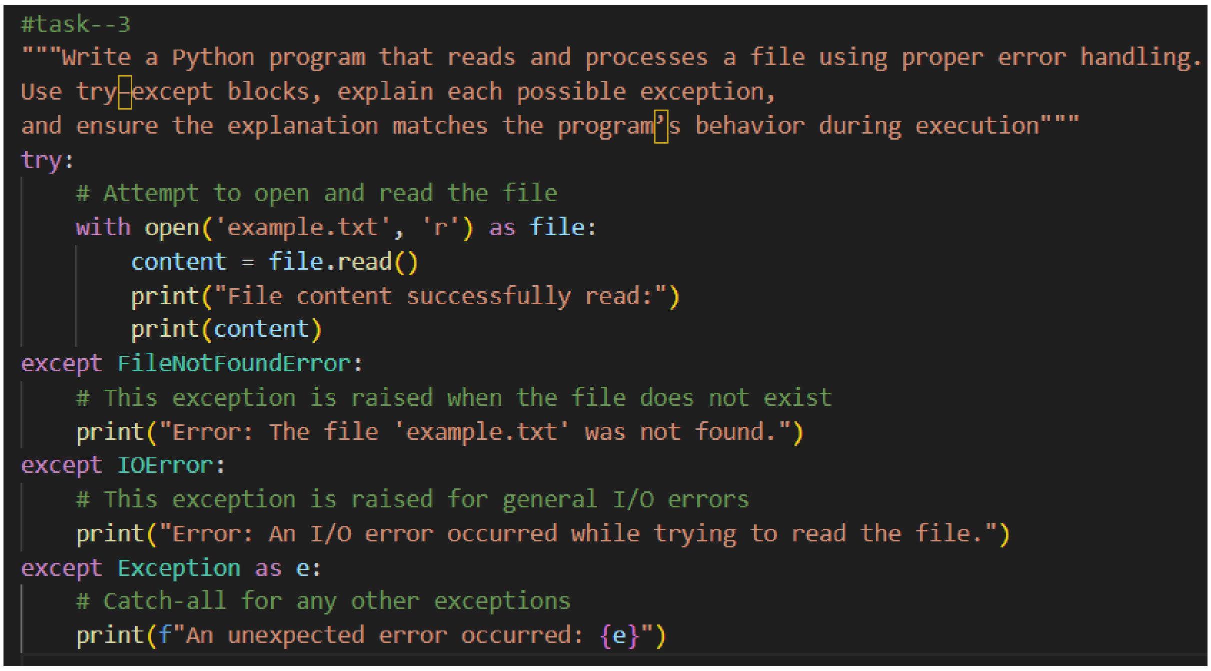
```
PS D:\collage\AI-AC> python ass-5.5.py
Enter a number to find its Fibonacci value: 4
The 4th Fibonacci number is: 3
```

Task-3:(Transparency in Error Handling)

Task: Use AI to generate a Python program that reads a file and

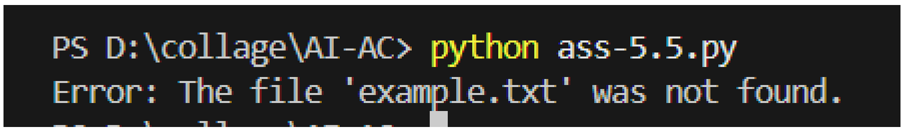
processes data

Prompt and code:



```python
#task--3
"""Write a Python program that reads and processes a file using proper error handling.
Use try-except blocks, explain each possible exception,
and ensure the explanation matches the program's behavior during execution"""
try:
    # Attempt to open and read the file
    with open('example.txt', 'r') as file:
        content = file.read()
        print("File content successfully read:")
        print(content)
except FileNotFoundError:
    # This exception is raised when the file does not exist
    print("Error: The file 'example.txt' was not found.")
except IOError:
    # This exception is raised for general I/O errors
    print("Error: An I/O error occurred while trying to read the file.")
except Exception as e:
    # Catch-all for any other exceptions
    print(f"An unexpected error occurred: {e}")
```

Output:

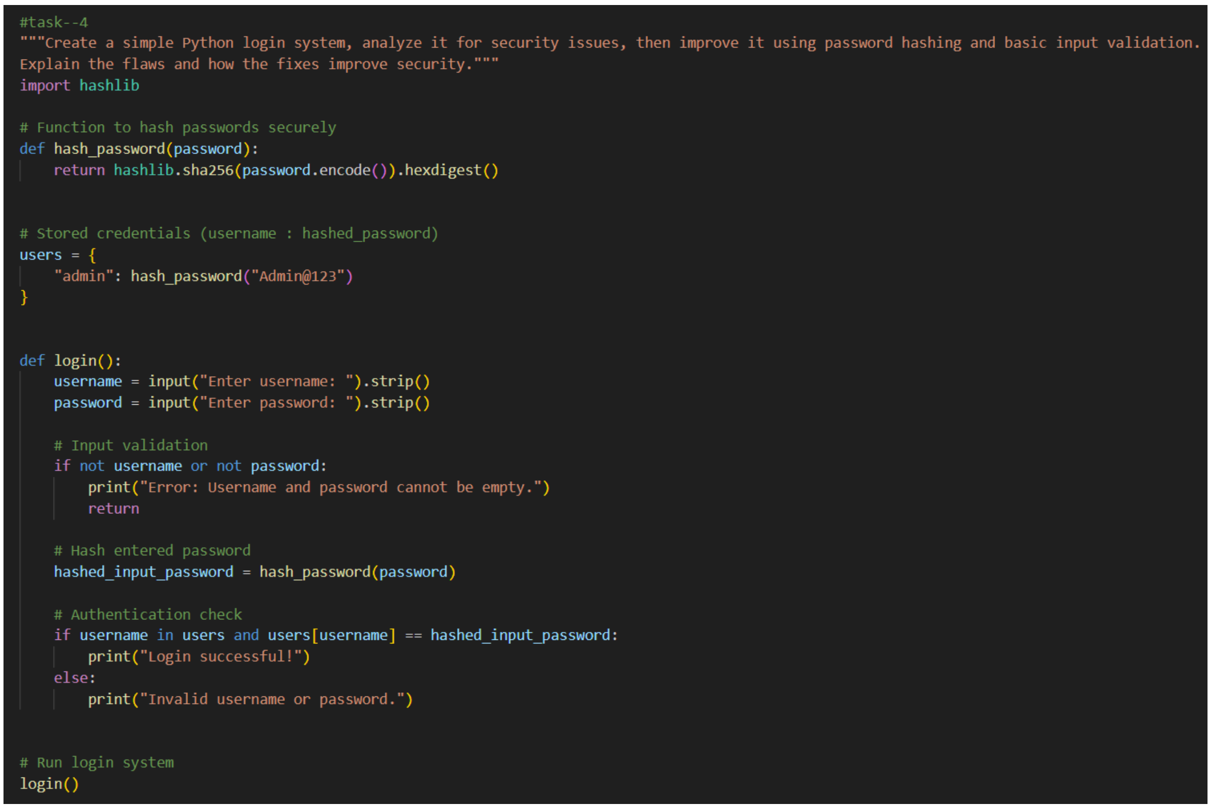


```
PS D:\collage\AI-AC> python ass-5.5.py
Error: The file 'example.txt' was not found.
```

TASK-4: (Security in User Authentication)

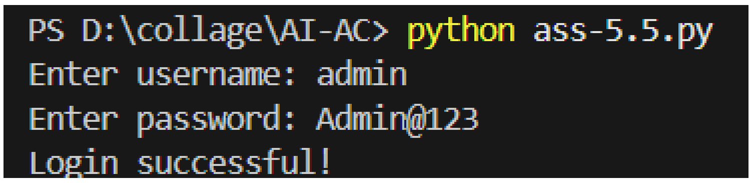
Task: Use an AI tool to generate a Python-based login system.

Prompt and code:



```python
#task--4
"""Create a simple Python login system, analyze it for security issues, then improve it using password hashing and basic input validation.
Explain the flaws and how the fixes improve security."""
import hashlib

# Function to hash passwords securely
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()


# Stored credentials (username : hashed_password)
users = {
    "admin": hash_password("Admin@123")
}


def login():
    username = input("Enter username: ").strip()
    password = input("Enter password: ").strip()

    # Input validation
    if not username or not password:
        print("Error: Username and password cannot be empty.")
        return

    # Hash entered password
    hashed_input_password = hash_password(password)

    # Authentication check
    if username in users and users[username] == hashed_input_password:
        print("Login successful!")
    else:
        print("Invalid username or password.")


# Run login system
login()
```

Output:

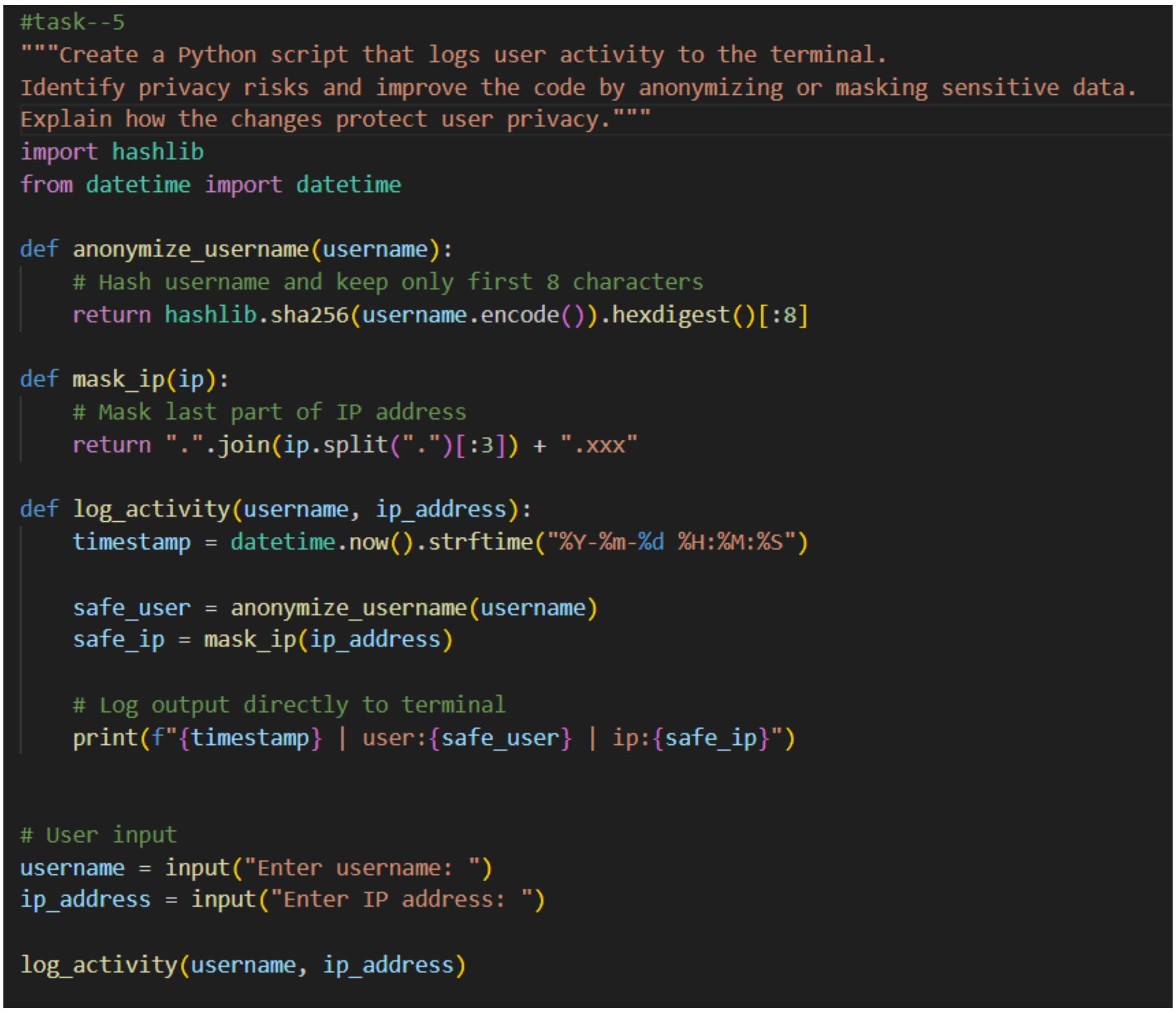


```
PS D:\collage\AI-AC> python ass-5.5.py
Enter username: admin
Enter password: Admin@123
Login successful!
```

Task-5: Privacy in Data Logging

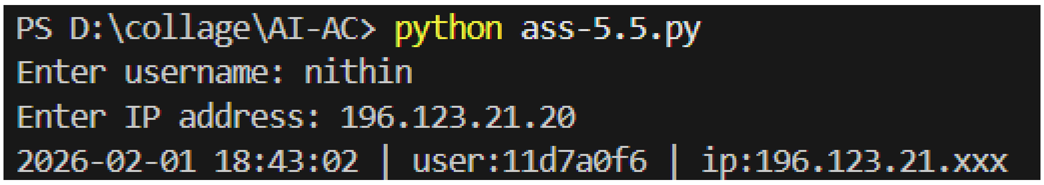
Task: Use an AI tool to generate a Python script that logs user activity (username, IP address, timestamp).

Prompt and code:



```python
#task--5
"""Create a Python script that logs user activity to the terminal.
Identify privacy risks and improve the code by anonymizing or masking sensitive data.
Explain how the changes protect user privacy."""
import hashlib
from datetime import datetime

def anonymize_username(username):
    # Hash username and keep only first 8 characters
    return hashlib.sha256(username.encode()).hexdigest()[:8]

def mask_ip(ip):
    # Mask last part of IP address
    return ".".join(ip.split(".")[:3]) + ".xxx"

def log_activity(username, ip_address):
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    safe_user = anonymize_username(username)
    safe_ip = mask_ip(ip_address)

    # Log output directly to terminal
    print(f"{timestamp} | user:{safe_user} | ip:{safe_ip}")


# User input
username = input("Enter username: ")
ip_address = input("Enter IP address: ")

log_activity(username, ip_address)
```

Output:



```
PS D:\collage\AI-AC> python ass-5.5.py
Enter username: nithin
Enter IP address: 196.123.21.20
2026-02-01 18:43:02 | user:11d7a0f6 | ip:196.123.21.xxx
```