AI ASSISTED CODING

LAB ASSIGNMENT: 1.3

NAME: Likhitha Pothunuri

ROLL NO: 2403A510D1

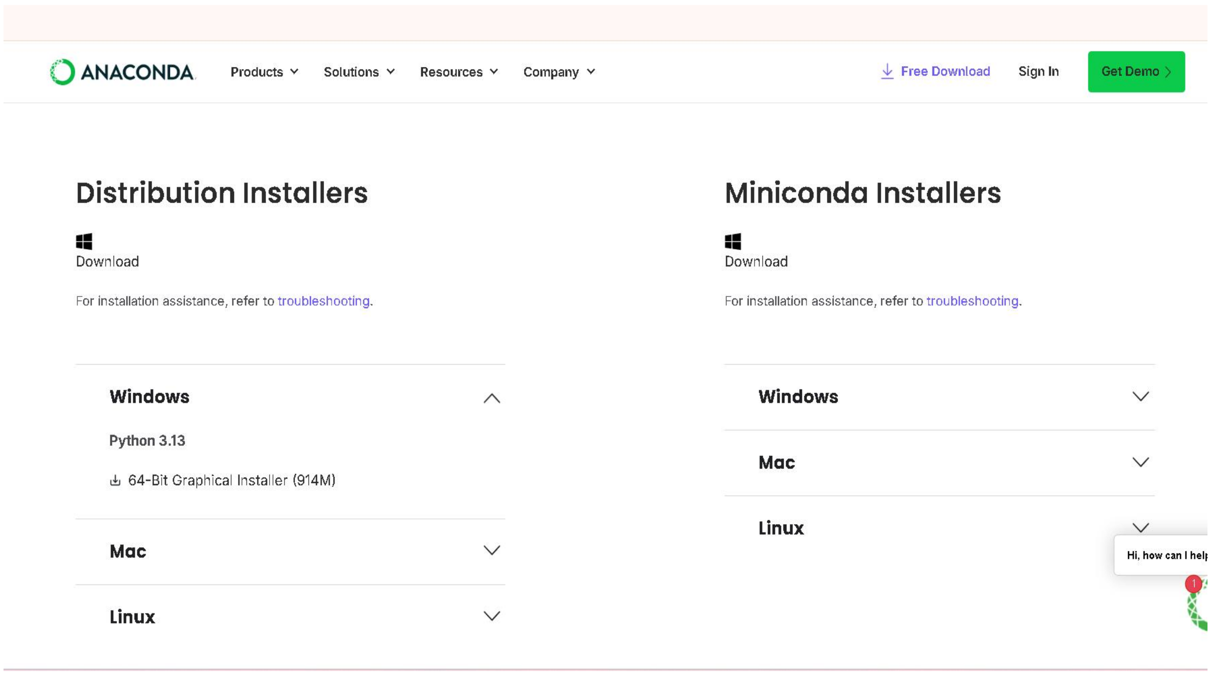
BATCH: 05

BRANCH: CSE

SUB: AI ASSISTED CODING

TASK - 01

Screen shot-1



Distribution Installers

Download

For installation assistance, refer to troubleshooting.

Windows

Python 3.13

64-Bit Graphical Installer (914M)

Mac

Linux

Miniconda Installers

Download

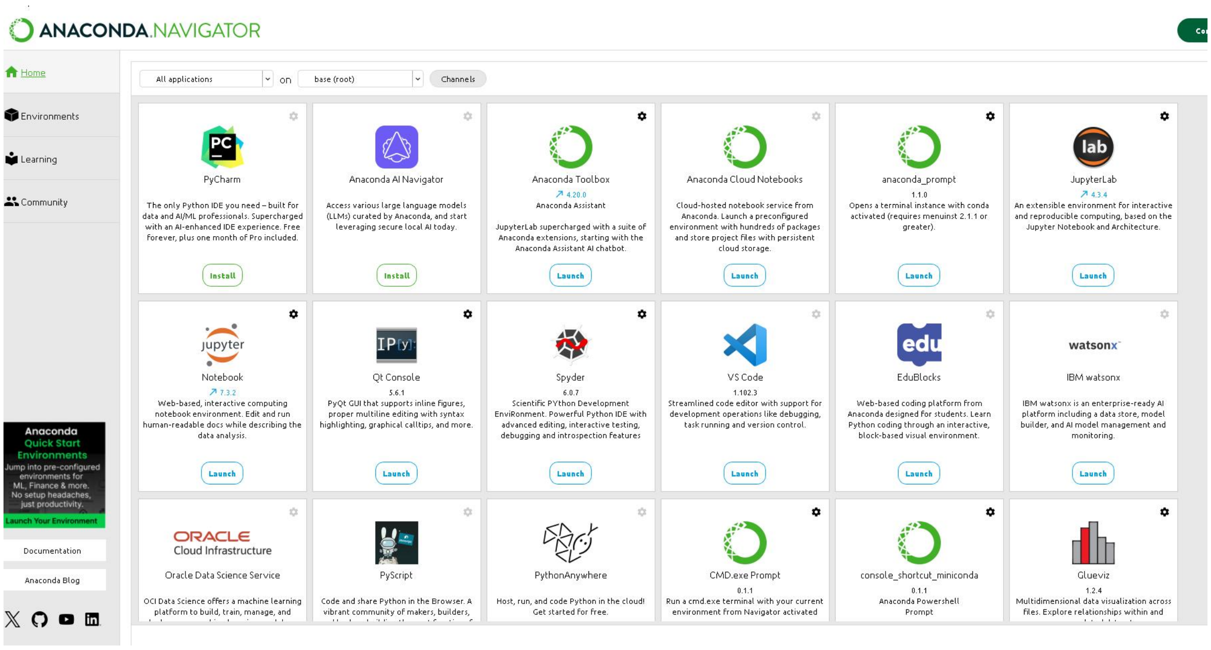
For installation assistance, refer to troubleshooting.

Windows

Mac

Linux

Screen shot-2



PyCharm

The only Python IDE you need – built for data and AI/ML professionals. Supercharged with an AI-enhanced IDE experience. Free forever, plus one month of Pro included.

Install

Anaconda AI Navigator

Access various large language models (LLMs) curated by Anaconda, and start leveraging secure local AI today.

Install

Anaconda Toolbox

4.20.0

Anaconda Assistant

JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.

Launch

Anaconda Cloud Notebooks

Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.

Launch

anaconda_prompt

1.1.0

Opens a terminal instance with conda activated (requires menuinst 2.1.1 or greater).

Launch

JupyterLab

4.3.4

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch

Notebook

7.3.2

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch

Qt Console

5.6.1

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch

Spyder

6.0.7

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Launch

VS Code

1.102.3

Streamlined code editor with support for development operations like debugging, task running and version control.

Launch

EduBlocks

Web-based coding platform from Anaconda designed for students. Learn Python coding through an interactive, block-based visual environment.

Launch

IBM watsonx

IBM watsonx is an enterprise-ready AI platform including a data store, model builder, and AI model management and monitoring.

Launch

Anaconda Quick Start Environments

Jump into pre-configured environments for ML, Finance & more. No setup headaches, just productivity.

Launch Your Environment

Oracle Data Science Service

OCI Data Science offers a machine learning platform to build, train, manage, and

PyScript

Code and share Python in the Browser. A vibrant community of makers, builders,

PythonAnywhere

Host, run, and code Python in the cloud! Get started for free.

CMD.exe Prompt

0.1.1

Run a cmd.exe terminal with your current environment from Navigator activated

console_shortcut_miniconda

0.1.1

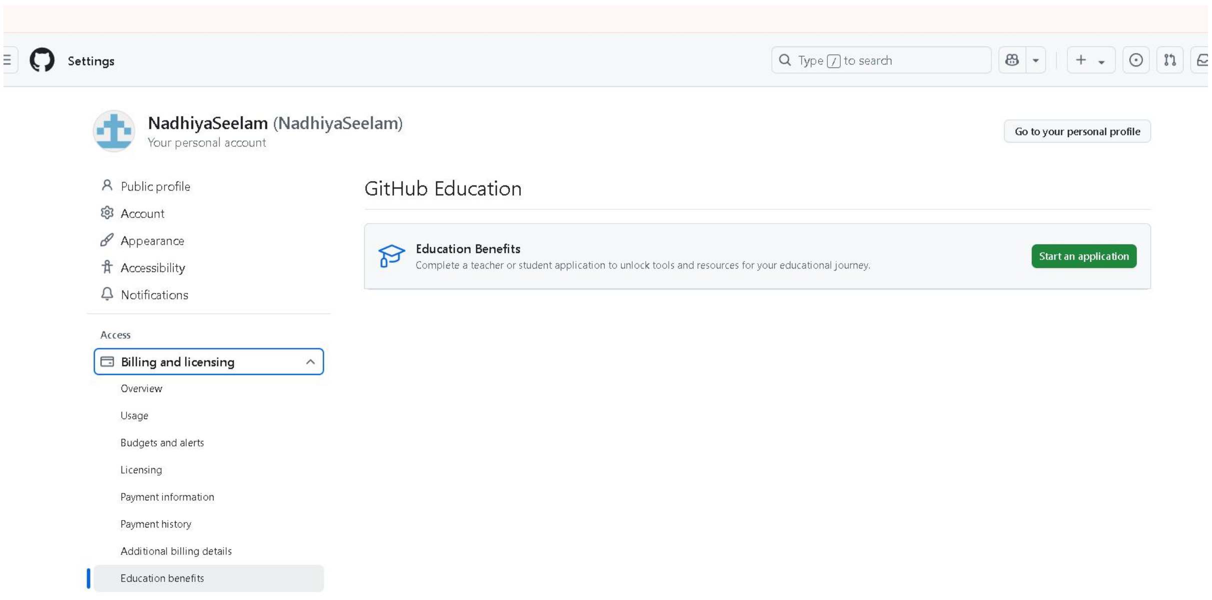
Anaconda Powershell Prompt

Glueviz

1.2.4

Multidimensional data visualization across files. Explore relationships within and

Screen shot-3


Settings

NadhiyaSeelam (NadhiyaSeelam)
Your personal account

Go to your personal profile

Public profile
Account
Appearance
Accessibility
Notifications

Access

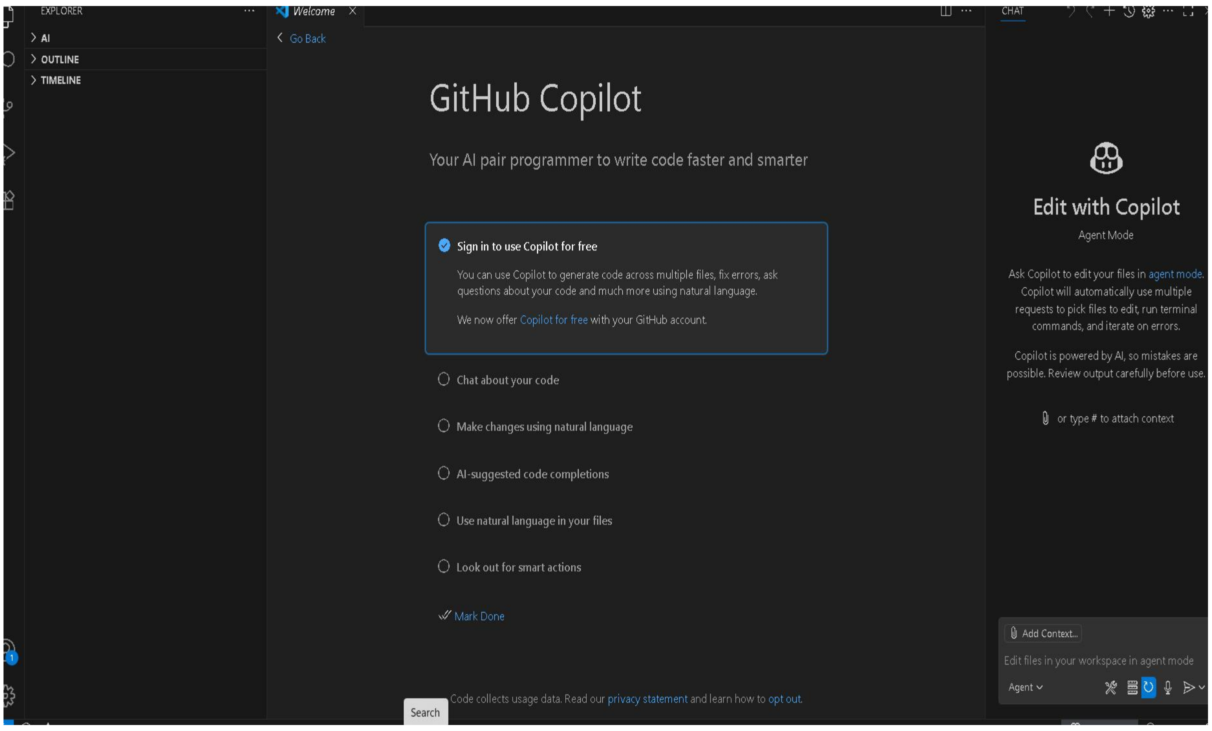
Billing and licensing
Overview
Usage
Budgets and alerts
Licensing
Payment information
Payment history
Additional billing details
Education benefits

GitHub Education

Education Benefits
Complete a teacher or student application to unlock tools and resources for your educational journey.

Start an application

Screen shot-4


GitHub Copilot

Your AI pair programmer to write code faster and smarter

Sign in to use Copilot for free

You can use Copilot to generate code across multiple files, fix errors, ask questions about your code and much more using natural language.

We now offer Copilot for free with your GitHub account.

Chat about your code
Make changes using natural language
AI-suggested code completions
Use natural language in your files
Look out for smart actions

Mark Done

Code collects usage data. Read our privacy statement and learn how to opt out.

Edit with Copilot
Agent Mode

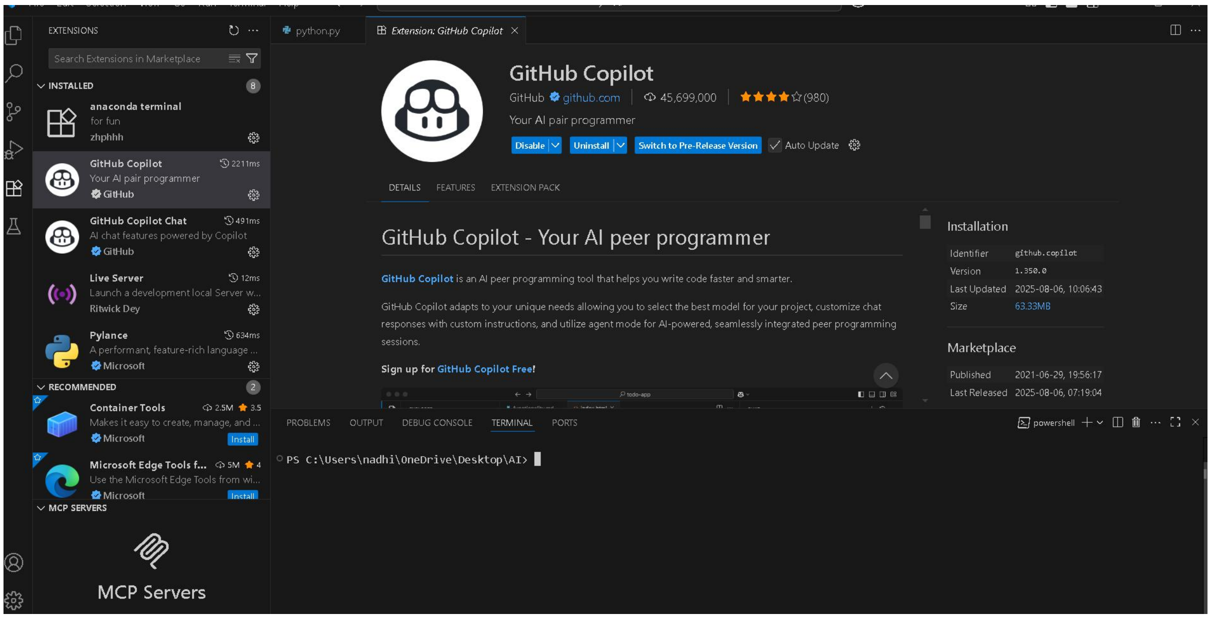
Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

Add Context...
Edit files in your workspace in agent mode
Agent

Screen shot-5



EXTENSIONS

Search Extensions in Marketplace

INSTALLED 8

anaconda terminal
for fun
zhphhh

GitHub Copilot 2211ms
Your AI pair programmer
GitHub

GitHub Copilot Chat 491ms
AI chat features powered by Copilot
GitHub

Live Server 12ms
Launch a development local Server w...
Ritwick Dey

Pylance 634ms
A performant, feature-rich language ...
Microsoft

RECOMMENDED 2

Container Tools 2.5M 3.5
Makes it easy to create, manage, and ...
Microsoft Install

Microsoft Edge Tools f... 5M 4
Use the Microsoft Edge Tools from wi...
Microsoft Install

MCP SERVERS

MCP Servers

python.py | Extension: GitHub Copilot

GitHub Copilot

GitHub github.com | 45,699,000 | ★★★★☆ (980)

Your AI pair programmer

Disable | Uninstall | Switch to Pre-Release Version | Auto Update

DETAILS FEATURES EXTENSION PACK

GitHub Copilot - Your AI peer programmer

GitHub Copilot is an AI peer programming tool that helps you write code faster and smarter.

GitHub Copilot adapts to your unique needs allowing you to select the best model for your project, customize chat responses with custom instructions, and utilize agent mode for AI-powered, seamlessly integrated peer programming sessions.

Sign up for GitHub Copilot Free!

Installation

Identifier	github.copilot
Version	1.350.0
Last Updated	2025-08-06, 10:06:43
Size	63.33MB

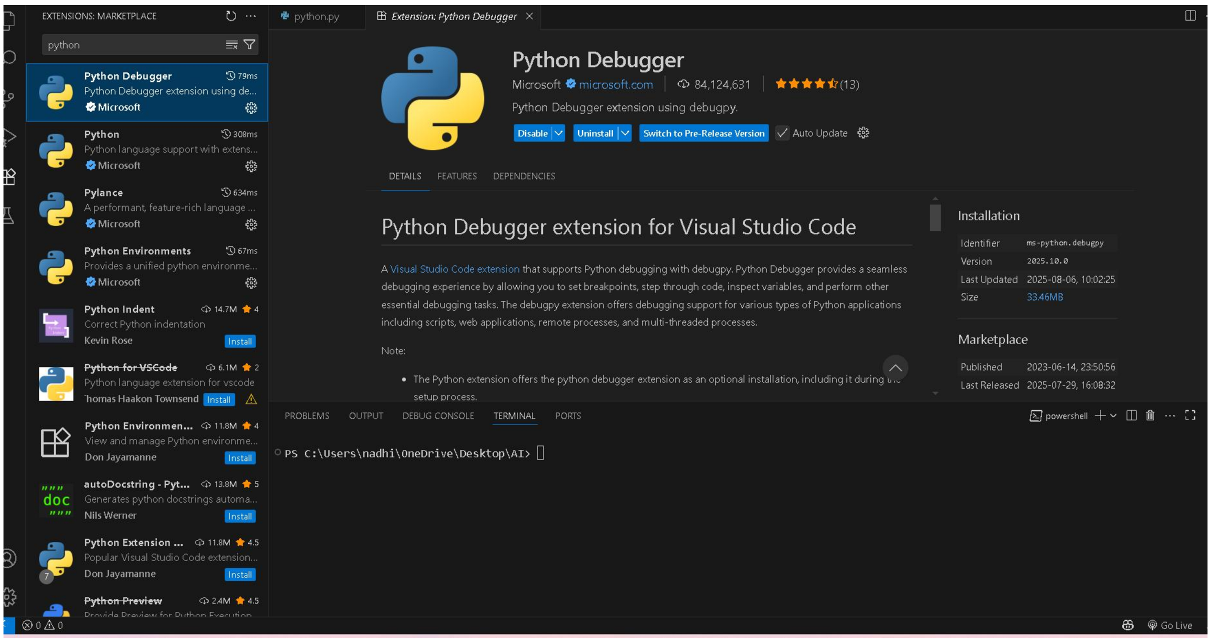
Marketplace

Published	2021-06-29, 19:56:17
Last Released	2025-08-06, 07:19:04

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS C:\Users\nadhi\OneDrive\Desktop\AI>
```

Screen shot-6



EXTENSIONS: MARKETPLACE

python

Python Debugger 79ms
Python Debugger extension using de...
Microsoft

Python 308ms
Python language support with extens...
Microsoft

Pylance 634ms
A performant, feature-rich language ...
Microsoft

Python Environments 67ms
Provides a unified python environme...
Microsoft

Python Indent 14.7M 4
Correct Python indentation
Kevin Rose Install

~~Python for VSCode~~ 6.1M 2
Python language extension for vscode
Thomas Haakon Townsend Install

Python Environmen... 11.8M 4
View and manage Python environme...
Don Jayamanne Install

autoDocstring - Pyt... 13.8M 5
Generates python docstrings automa...
Nils Werner Install

Python Extension ... 11.8M 4.5
Popular Visual Studio Code extension...
Don Jayamanne Install

~~Python Preview~~ 2.4M 4.5
Provide Preview for Python Execution

python.py | Extension: Python Debugger

Python Debugger

Microsoft microsoft.com | 84,124,631 | ★★★★½ (13)

Python Debugger extension using debugpy.

Disable | Uninstall | Switch to Pre-Release Version | Auto Update

DETAILS FEATURES DEPENDENCIES

Python Debugger extension for Visual Studio Code

A Visual Studio Code extension that supports Python debugging with debugpy. Python Debugger provides a seamless debugging experience by allowing you to set breakpoints, step through code, inspect variables, and perform other essential debugging tasks. The debugpy extension offers debugging support for various types of Python applications including scripts, web applications, remote processes, and multi-threaded processes.

Note:

- The Python extension offers the python debugger extension as an optional installation, including it during the setup process.

Installation

Identifier	ms-python.debugpy
Version	2025.10.0
Last Updated	2025-08-06, 10:02:25
Size	33.46MB

Marketplace

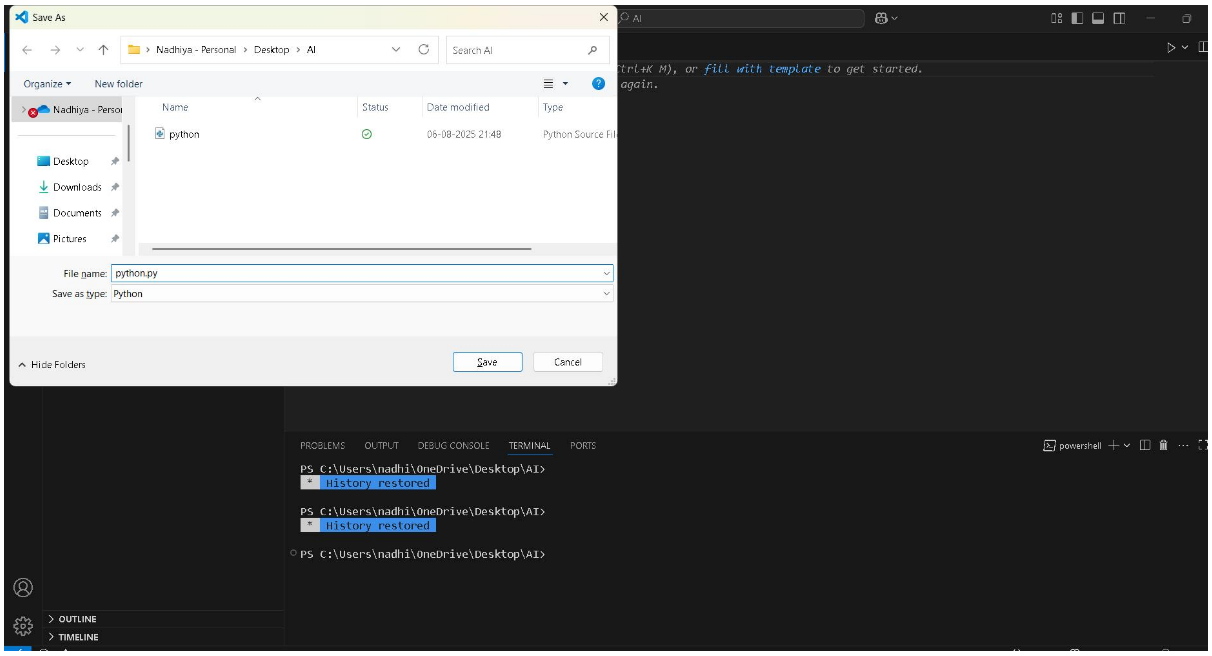
Published	2023-06-14, 23:50:56
Last Released	2025-07-29, 16:08:32

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS C:\Users\nadhi\OneDrive\Desktop\AI>
```

0 ⚠ 0 Go Live

Screen shot-7



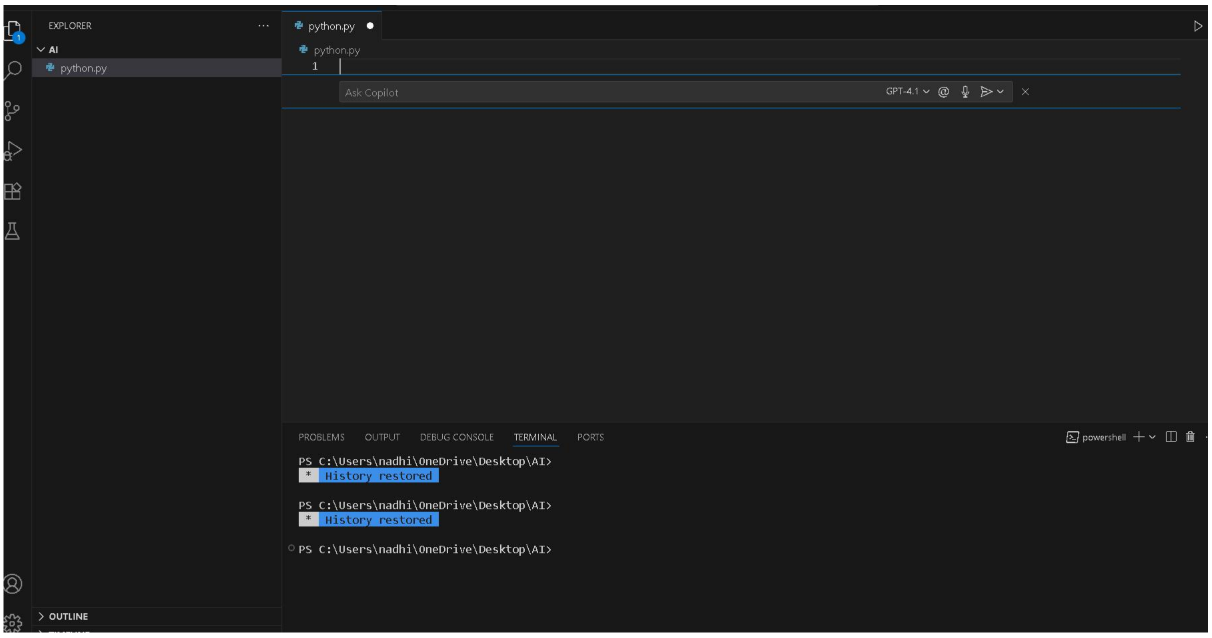
Name	Status	Date modified	Type
python	✓	06-08-2025 21:48	Python Source Fil

File name: python.py
Save as type: Python

```
trl+K M), or fill with template to get started.
 again.
```

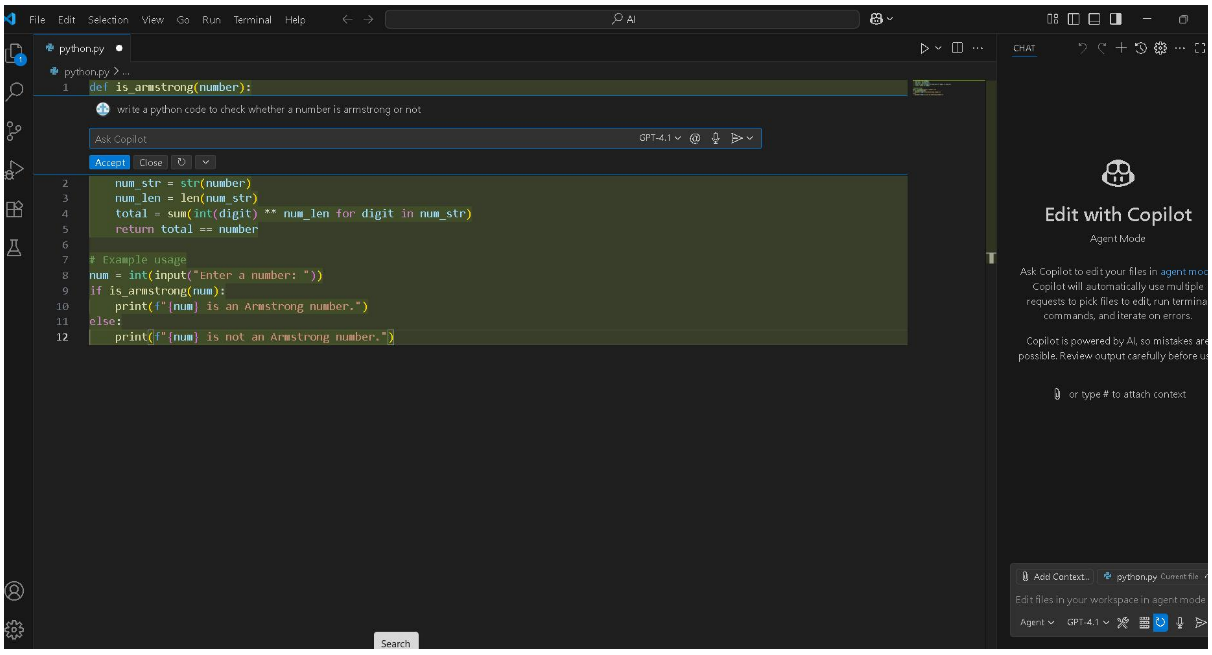
```
PS C:\Users\nadhi\OneDrive\Desktop\AI>
 *  History restored

PS C:\Users\nadhi\OneDrive\Desktop\AI>
 *  History restored

PS C:\Users\nadhi\OneDrive\Desktop\AI>
```

Screen shot-8



```
PS C:\Users\nadhi\OneDrive\Desktop\AI>
 *  History restored

PS C:\Users\nadhi\OneDrive\Desktop\AI>
 *  History restored

PS C:\Users\nadhi\OneDrive\Desktop\AI>
```

Screen shot-9



write a python code to check whether a number is armstrong or not

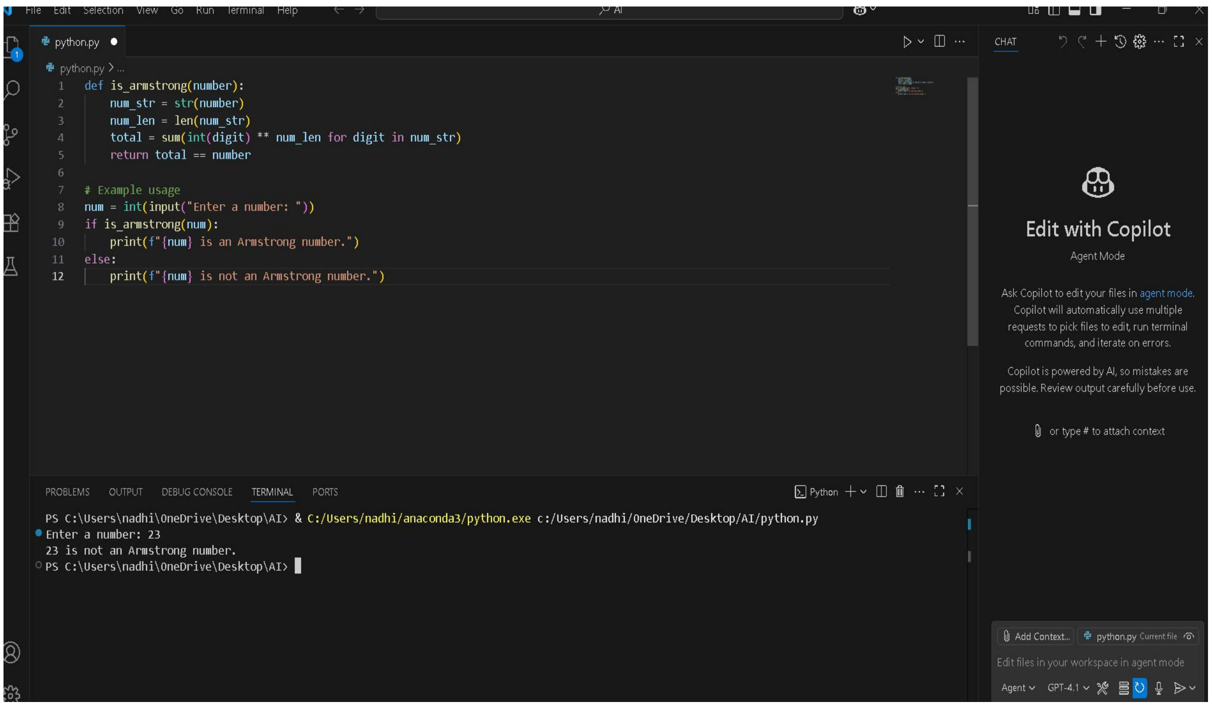
```python
def is_armstrong(number):
    num_str = str(number)
    num_len = len(num_str)
    total = sum(int(digit) ** num_len for digit in num_str)
    return total == number

# Example usage
num = int(input("Enter a number: "))
if is_armstrong(num):
    print(f"{num} is an Armstrong number.")
else:
    print(f"{num} is not an Armstrong number.")
```

Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

Screen shot-9


File Edit Selection View Go Run Terminal Help
python.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
CHAT
Edit with Copilot
Agent Mode
Add Context...
python.py Current file
Edit files in your workspace in agent mode
Agent
GPT-4.1

```python
def is_armstrong(number):
    num_str = str(number)
    num_len = len(num_str)
    total = sum(int(digit) ** num_len for digit in num_str)
    return total == number

# Example usage
num = int(input("Enter a number: "))
if is_armstrong(num):
    print(f"{num} is an Armstrong number.")
else:
    print(f"{num} is not an Armstrong number.")
```

```
PS C:\Users\nadhi\OneDrive\Desktop\AI> & C:/Users/nadhi/anaconda3/python.exe c:/Users/nadhi/OneDrive/Desktop/AI/python.py
Enter a number: 23
23 is not an Armstrong number.
PS C:\Users\nadhi\OneDrive\Desktop\AI>
```

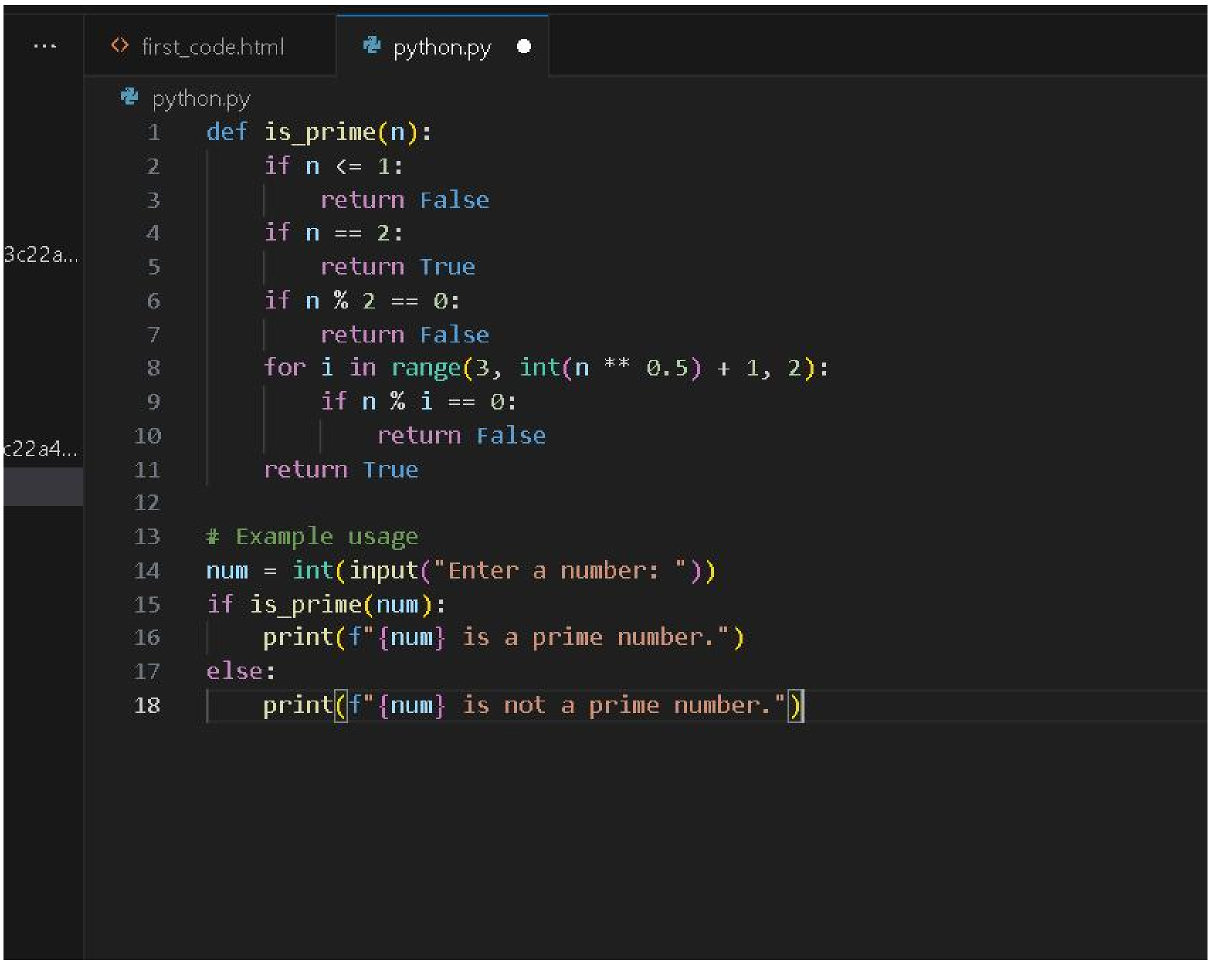
Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

TASK – 02

Prompt : Write a python code to check whether a number is prime or not.



```python
def is_prime(n):
    if n <= 1:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    for i in range(3, int(n ** 0.5) + 1, 2):
        if n % i == 0:
            return False
    return True

# Example usage
num = int(input("Enter a number: "))
if is_prime(num):
    print(f"{num} is a prime number.")
else:
    print(f"{num} is not a prime number.")
```

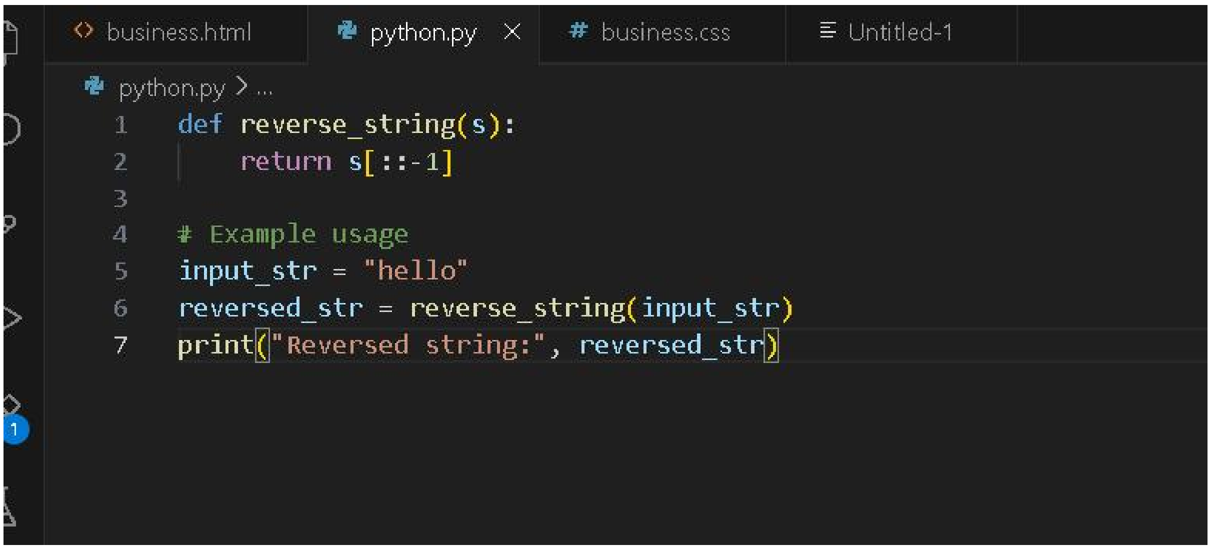
Output: 2 is prime number

Explanation:

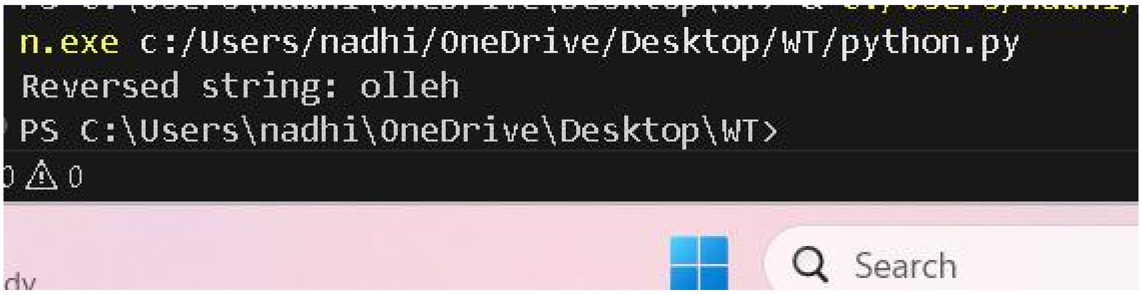
- he code checks if a given number is prime.
- A prime number is a number greater than 1 that has no divisors other than 1 and itself.
- The code usually:
    - Returns False if the number is less than or equal to 1.

- Loops from 2 up to the square root of the number.
- If the number is divisible by any of these, it returns False.
- If no divisors are found, it returns True.

Task-3:



```python
def reverse_string(s):
    return s[::-1]

# Example usage
input_str = "hello"
reversed_str = reverse_string(input_str)
print("Reversed string:", reversed_str)
```



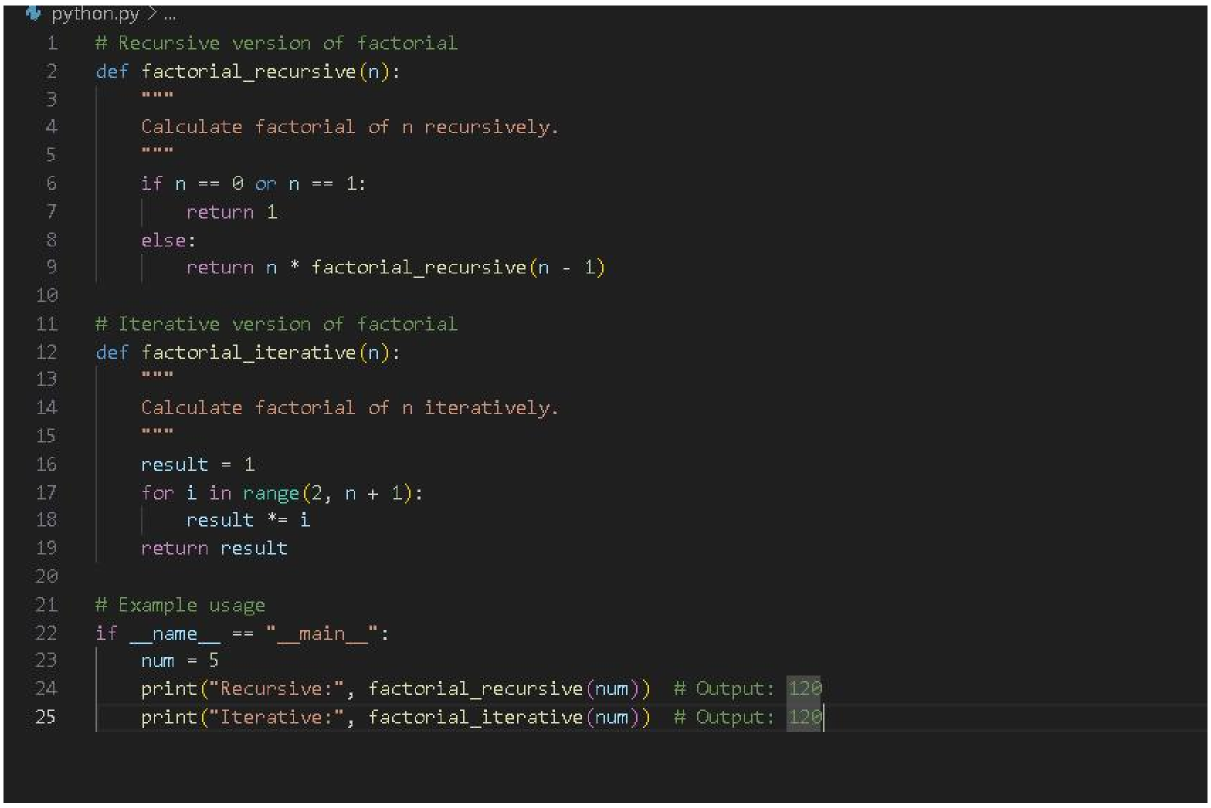
```
n.exe c:/Users/nadhi/OneDrive/Desktop/WT/python.py
Reversed string: olleh
PS C:\Users\nadhi\OneDrive\Desktop\WT>
```

Explanation:

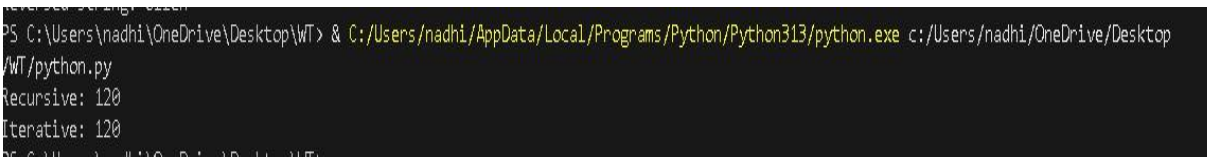
- The function reverse_string(s) takes a string s as input and returns its reverse using slicing (s[::-1]).
- The example usage sets input_str to "hello".

- It calls reverse_string(input_str), which returns "olleh", and stores it in reversed_str.
- Finally, it prints Reversed string: olleh to the console

Task-04:



```python
# Recursive version of factorial
def factorial_recursive(n):
    """
    Calculate factorial of n recursively.
    """
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial_recursive(n - 1)

# Iterative version of factorial
def factorial_iterative(n):
    """
    Calculate factorial of n iteratively.
    """
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

# Example usage
if __name__ == "__main__":
    num = 5
    print("Recursive:", factorial_recursive(num))  # Output: 120
    print("Iterative:", factorial_iterative(num))  # Output: 120
```



```
PS C:\Users\nadhi\OneDrive\Desktop\WT> & C:/Users/nadhi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nadhi/OneDrive/Desktop
/WT/python.py
Recursive: 120
Iterative: 120
```

Explanation:

- This function calculates the factorial of n using recursion.
    - If n is 0 or 1, it returns 1 (base case).
    - Otherwise, it returns n * factorial_recursive(n - 1).

- **factorial_iterative(n):**

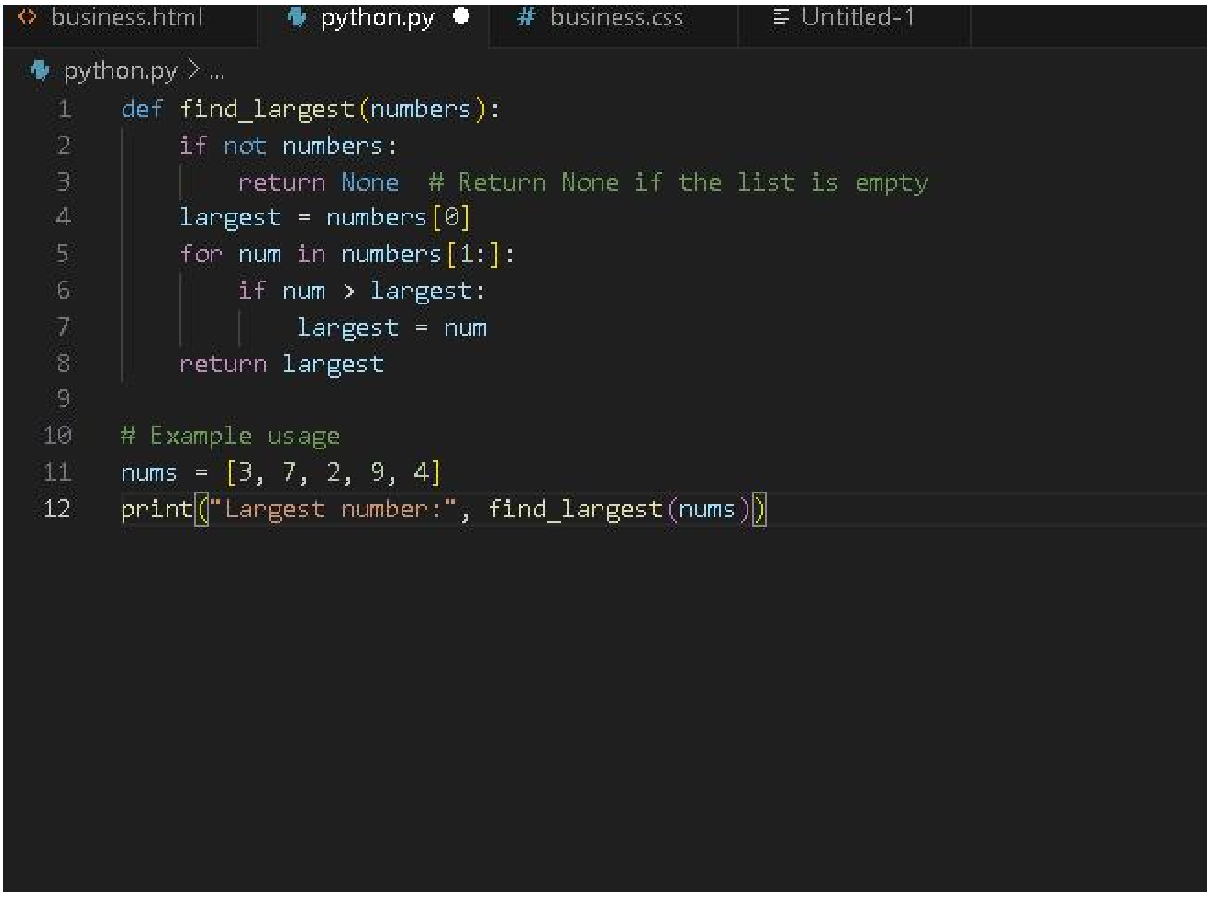
  This function calculates the factorial of n using a loop.

  - It initializes result to 1.
  - Then multiplies result by each number from 2 up to n.

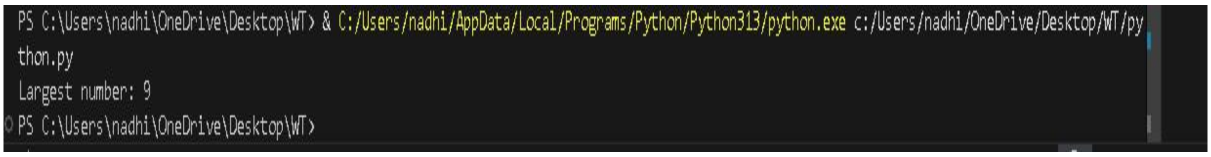
- **Example usage:**

  - If the script is run directly, it sets num = 5.
  - It prints the factorial of 5 using both the recursive and iterative functions.
  - Both methods output 120.

Task-05:



```python
def find_largest(numbers):
    if not numbers:
        return None  # Return None if the list is empty
    largest = numbers[0]
    for num in numbers[1:]:
        if num > largest:
            largest = num
    return largest

# Example usage
nums = [3, 7, 2, 9, 4]
print("Largest number:", find_largest(nums))
```



```
PS C:\Users\nadhi\OneDrive\Desktop\WT> & C:/Users/nadhi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nadhi/OneDrive/Desktop/WT/py
thon.py
Largest number: 9
PS C:\Users\nadhi\OneDrive\Desktop\WT>
```

Explanation:

- **find_largest(numbers):**

  This function takes a list of numbers and returns the largest value.

  - If the list is empty, it returns None.
  - It starts by assuming the first number is the largest.
  - It then loops through the rest of the list, updating largest if it finds a bigger number.

    - Finally, it returns the largest number found.
- **Example usage:**
    - A list nums = [3, 7, 2, 9, 4] is defined.
    - The function is called with this list, and the result is printed.
    - Output: Largest number: 9