

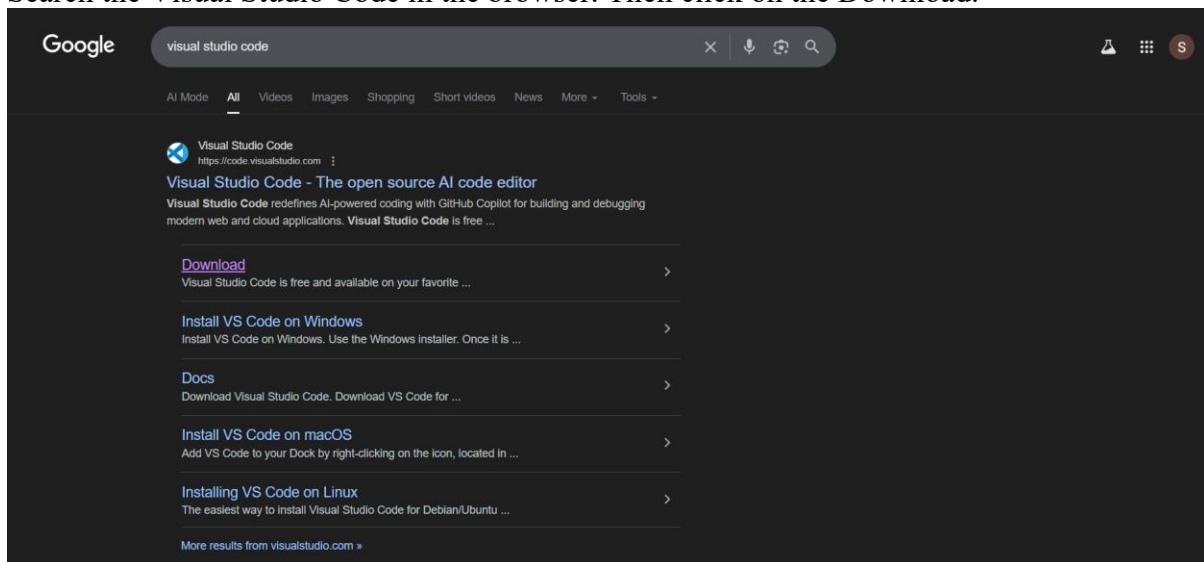
School of Computer Science and Artificial Intelligence

Lab Assignment 1.5

Program	: B. Tech (CSE)
Course Title	: AI Assisted Coding
Course Code	: 23CS002PC304
Semester	: III
Academic Session	: 2025-2026
Name of Student	: Sruthi
Enrollment No.	: 2403A51L10
Batch No.	: 51
Date	: 09-10-2026

Task 0

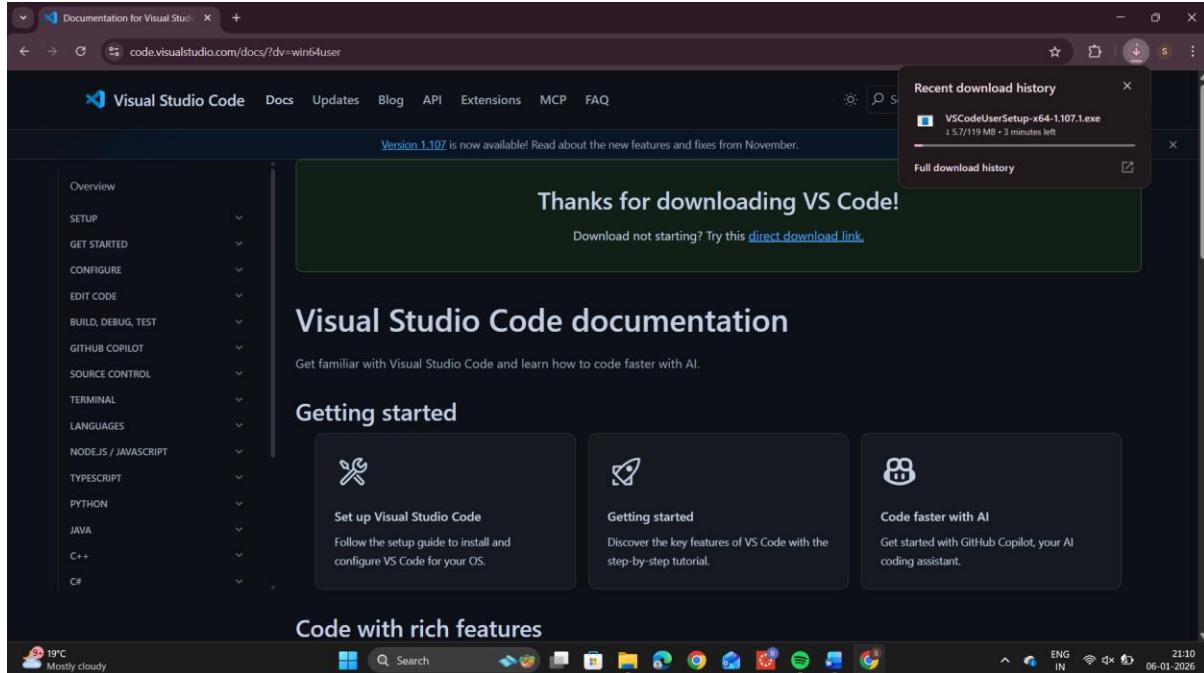
Search the Visual Studio Code in the browser. Then click on the Download.



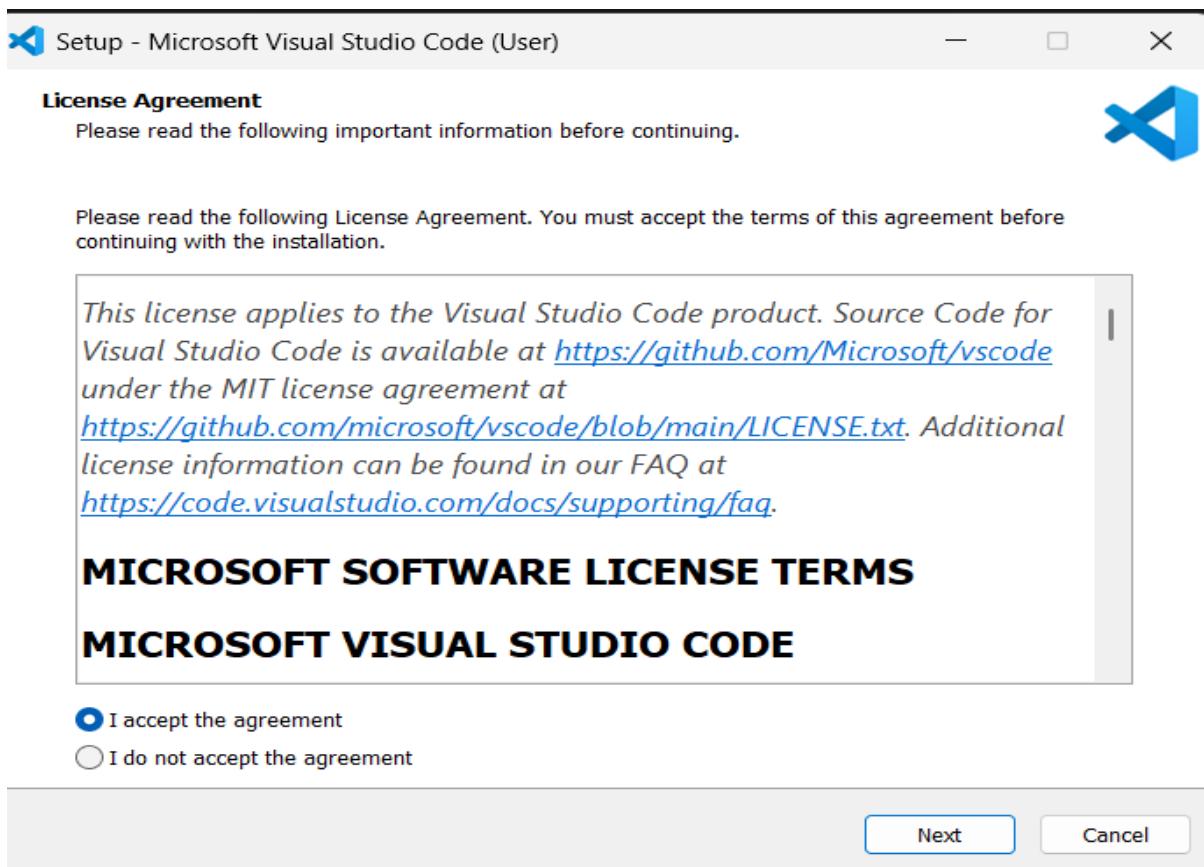
Click on the OS your desktop or laptop



A download .exe file you can see.

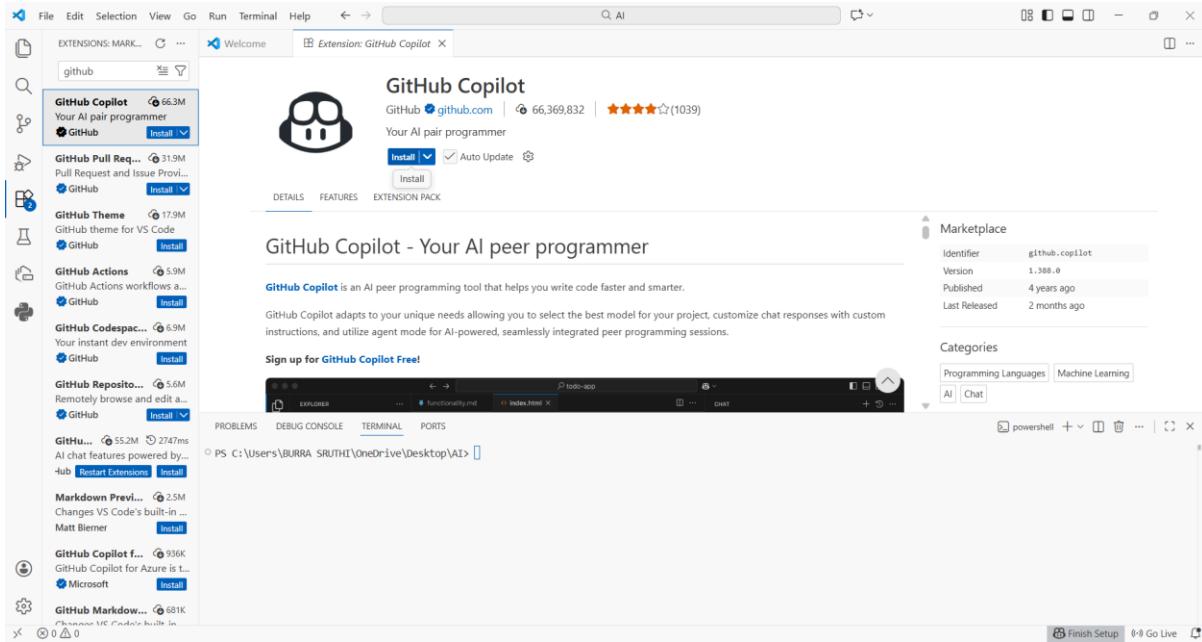


After the downloading the .exe file completed. Click on the .exe file then you will see the below window . Then click on the accept the agreement click next.

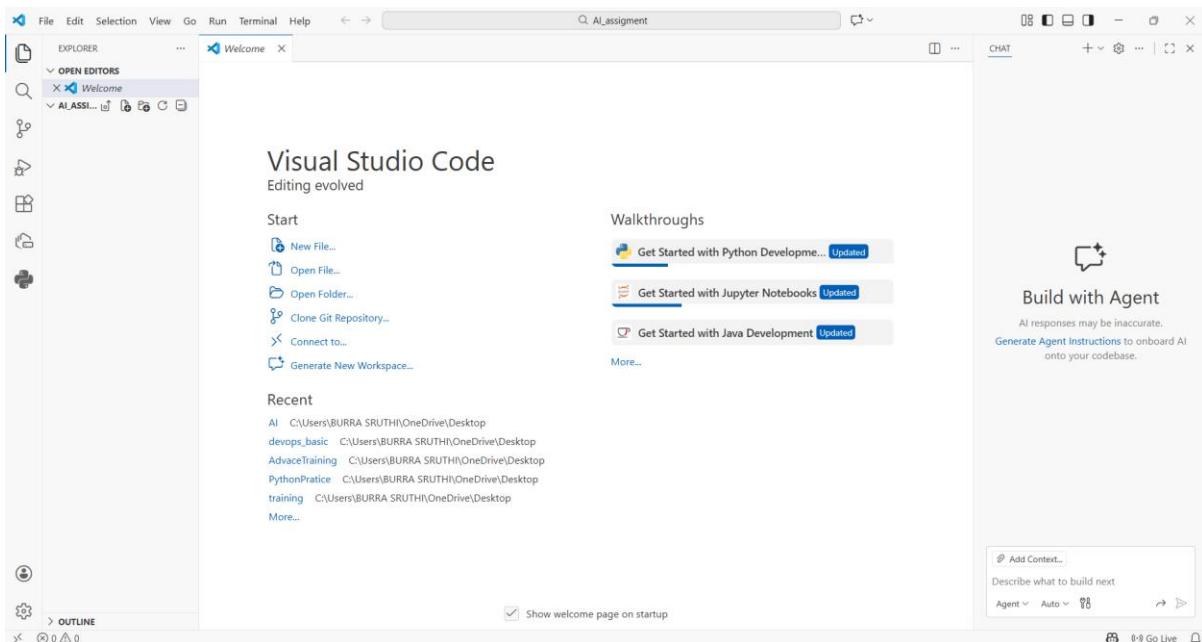


After that click next and then click on install. After the completion of download. Open the vs code

Click on the extension then search for GitHub Copilot. Click on the install



After the installation completed . sign in into the github copilot using the account of github. Create a new folder on desktop open it in the visual studio.



Create a python program

Task 1

Prompt: Write the python program to reverse a string without using functions

Code:

```
string = "Hello, World!"  
reversed_string = ""  
for char in string:  
    reversed_string = char + reversed_string  
print("The reversed string is:", reversed_string)
```

Output:

```
The reversed string is: !dlrow ,olleH
```

Explanation:

The program reverses a string using a for loop. The original string is stored in a variable, and an empty string is initialized to store the reversed result. Each character of the string is added to the beginning of the new string during iteration, which reverses the order of characters. Finally, the reversed string is printed.

Task 2

Prompt: Write a python program to reverse a string without using functions and in optimized way

Code:

```
string = "Hello, World!"  
reversed_string = string[::-1]  
print("The reversed string is:", reversed_string)
```

Output:

```
The reversed string is: !dlrow ,olleH
```

Explanation:

In this program, the string "Hello, World!" is reversed using Python slicing. The slicing syntax `[::-1]` starts from the end of the string and moves backward, creating a reversed copy of the string. The reversed string is then stored in the variable `reversed_string` and displayed using the `print()` function.

Task 3

Prompt: Write a python program to reverse a string using recursion

Code:

```
def reverse_string_recursive(s):
    if len(s) == 0:
        return s
    else:
        return s[-1] + reverse_string_recursive(s[:-1])
string = "Hello, World!"
reversed_string = reverse_string_recursive(string)
print("The reversed string is:", reversed_string)
```

Output:

```
The reversed string is: !dlrow ,olleH
```

Explanation:

This program reverses a string using recursion. The function `reverse_string_recursive()` takes a string as input. If the string is empty, it returns the string (base case). Otherwise, the function returns the last character of the string and recursively calls itself with the remaining part of the string excluding the last character. This process continues until the entire string is reversed. Finally, the reversed string is printed.

Task 4

Prompt: Write a python program to reverse a string with and without function

Code:

```
#without function
string = "Hello, World!"
reversed_string = ""
for char in string:
    reversed_string = char + reversed_string
print("Reversed string without function:", reversed_string)
#with function
def reverse_string(s):
    return s[::-1]
input_string = "Hello, World!"
reversed_str = reverse_string(input_string)
print("Reversed string with function:", reversed_str)
```

Output:

```
Reversed string without function: !dlroW ,olleH  
Reversed string with function: !dlroW ,olleH
```

Explanation:

In the first part, the string is reversed **without using a function**. An empty string is initialized, and a for loop is used to traverse each character of the original string. Each character is added at the beginning of the new string, which results in the reversed string. In the second part, the string is reversed **using a function**. The function uses Python slicing `[::-1]` to reverse the string in a simple and efficient way. The reversed string is then returned and printed.

Both methods produce the same output, but using a function makes the code more reusable and concise.

Comparative Analysis: With Function vs Without Function

Criteria	Without Function	With Function
Code Clarity	Logic is explicit and easy to understand step by step.	Code is concise and clean, but slicing may be less intuitive for beginners.
Reusability	Code cannot be reused easily and must be rewritten each time.	Highly reusable; the function can be called multiple times with different inputs.
Debugging Ease	Easier to debug since each step is visible inside the loop.	Slightly harder to debug as the logic is hidden inside slicing.
Suitability for Large-scale Applications	Not suitable, as repeated logic increases code duplication.	More suitable due to modular design and better maintainability.

Task 5

Prompt :Write a Python program to reverse a string using iteration and recursion

Code:

```
# Using iteration
def reverse_string_iterative(s):
    reversed_string = ""
    for char in s:
        reversed_string = char + reversed_string
    return reversed_string

# Using recursion
def reverse_string_recursive(s):
    if len(s) == 0:
        return s
    else:
        return s[-1] + reverse_string_recursive(s[:-1])

# Test the functions
input_string = "Hello, World!"
reversed_iterative = reverse_string_iterative(input_string)
reversed_recursive = reverse_string_recursive(input_string)
print("Reversed string using iteration:", reversed_iterative)
print("Reversed string using recursion:", reversed_recursive)
```

Output:

```
Reversed string using iteration: !dlrow ,olleH
Reversed string using recursion: !dlrow ,olleH
```

Explanation:

This program demonstrates two different methods to reverse a string: **iteration** and **recursion**.

1. Iterative Approach

The function `reverse_string_iterative()` takes a string as input and initializes an empty string to store the reversed result. A for loop iterates through each character of the input string.

During each iteration, the current character is added to the beginning of the new string, which gradually forms the reversed string. The final reversed string is then returned.

2. Recursive Approach

The function `reverse_string_recursive()` reverses the string using recursion. If the length of the string is zero, the function returns the string (base case). Otherwise, it takes the last character of the string and calls itself with the remaining substring excluding the last character. This process continues until the base case is reached, resulting in the reversed string.

3. Function Testing

The input string "Hello, World!" is passed to both functions. The returned reversed strings from the iterative and recursive methods are stored in separate variables and printed. Both methods produce the same output.

Comparative Analysis

Criteria	Iterative Approach	Recursive Approach
Execution Flow	Executes sequentially using a loop.	Executes through repeated function calls until the base case is reached.
Time Complexity	$O(n)$, where n is the length of the string.	$O(n)$, but with additional overhead due to recursive calls.
Performance for Large Inputs	More efficient and faster, as it avoids function call overhead.	Less efficient and may cause stack overflow for very large strings.
Memory Usage	Uses constant extra memory apart from the result string.	Uses extra stack memory for each recursive call.
Appropriate Usage	Suitable for practical and large-scale applications.	Suitable for learning recursion and problem decomposition.