**Group 5 Report on Installation and Implementation of C++ and Python Programs**

**PHS781 – Numerical and Computational Methods**

**[Instructor: Prof O.E Oyewande]**

***Participant:***

|  |  |  |
| --- | --- | --- |
| S/N | **Name** | **Program** |
| 1. | OYELEYE Favour O. | Solid State Physics |
| 2. | AJIBADE Opeyemi D. | Theoretical Physics |
| 3. | KANU Faith C. | Radiation & Health Physics |
| 4. | ADEJARE Taiwo D. | Radiation & Health Physics |
| 5. | EHINLAFA Kayode T. | Radiation & Health Physics |
| 6. | ADEBOYE Wemimo | Solid State Physics |
| 7. | LAMIDI Habeeb A. | Atmospheric Physics |
| 8. | AKINWALE Oluwafemi V. | Applied Geophysics |
| 9. | OLALEMI Oluwaseyi O. | Atmospheric Physics |

**Section 1: Installation of C++ and Python**

***\*\*\*\*Note: All installations and activities are done/ carried out on a Windows 10 64-bit laptop***

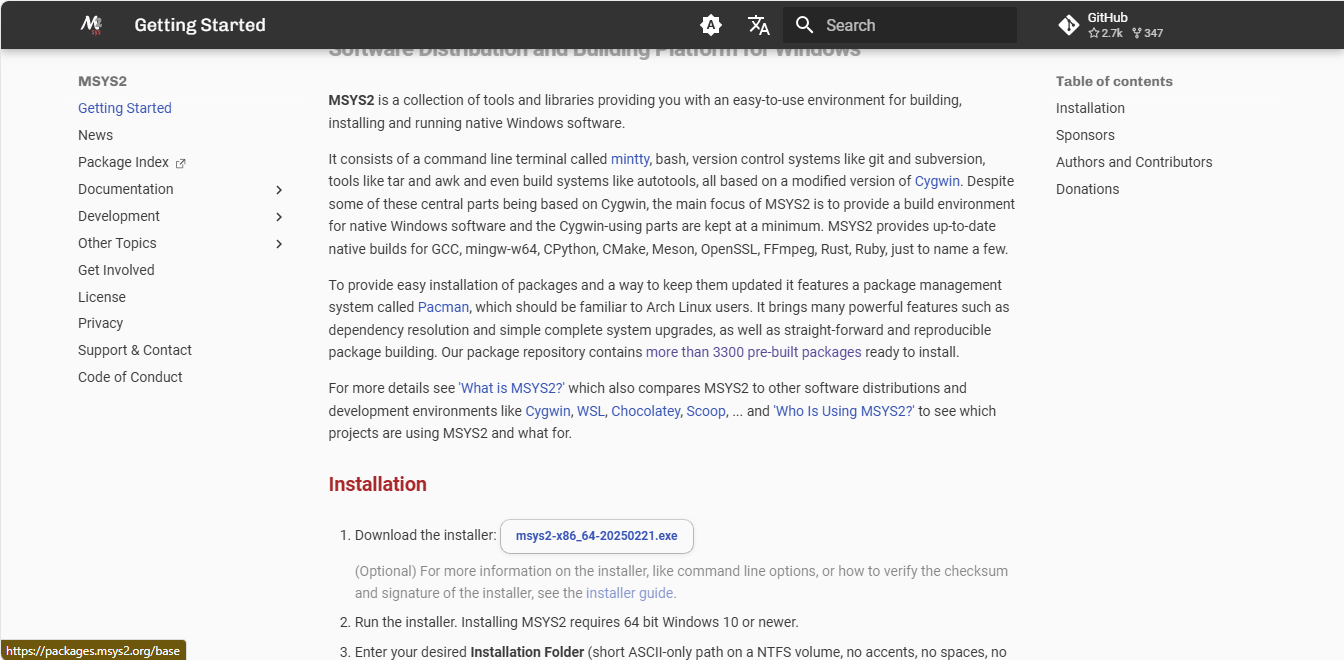
**1.1 Installing C++ (g++) on Windows**

To install C++ with GCC on your Windows computer, use the **MinGW-w64** toolchain, which provides a Windows port of the GNU Compiler Collection (GCC).

**Step 1: Download MSYS2 Installer**

First, you need to download the MSYS2 installer. MSYS2 is a software distribution and building platform for Windows that makes it easy to install MinGW-w64.

1. **Go to the MSYS2 website:** <https://www.msys2.org/>

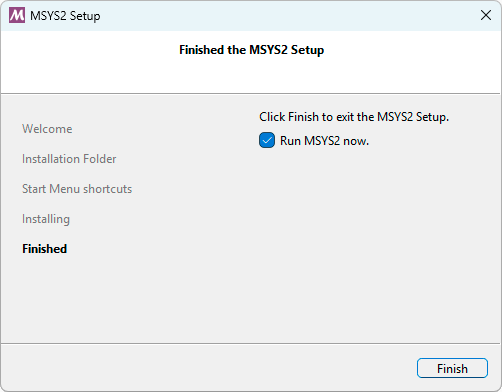
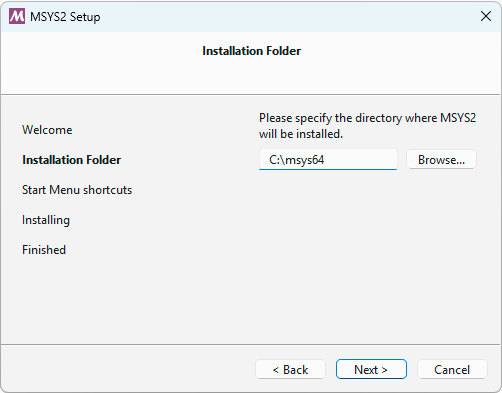


1. **Download the installer:** Click on the prominent download link for the installer (usually named something like [**msys2-x86\_64-20250221.exe**](https://github.com/msys2/msys2-installer/releases/download/2025-02-21/msys2-x86_64-20250221.exe)).

**Step 2: Run the MSYS2 Installer**

Once the download is complete, run the executable file.

1. **Launch the installer:** Double-click the downloaded file to start the installation.
2. **Follow the on-screen instructions:** The default settings are generally fine. It's recommended to keep the default installation directory (C:\msys64).

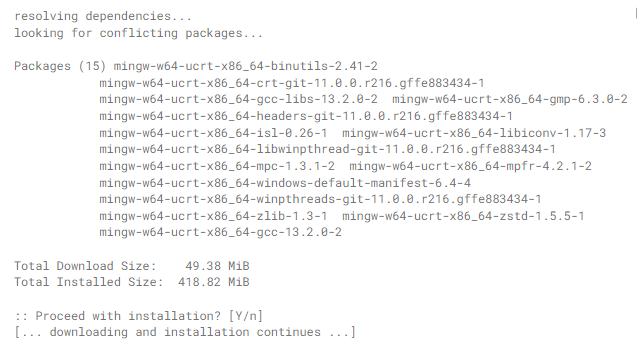


**Step 3: Install GCC (G++) via MSYS2 Terminal**

After the initial setup, a command-line terminal for MSYS2 will open. If it doesn't open automatically, you can run it from the Start Menu by searching for "MSYS2 MSYS".

1. **Install the MinGW-w64 toolchain:** You will probably want to install some tools like the MinGW-w64 GCC to start compiling projects. Run the following command:

```$ pacman -S mingw-w64-ucrt-x86\_64-gcc```



When prompted, press **Enter** to select all the packages in the group. Type **Y** and press **Enter** to confirm the installation.

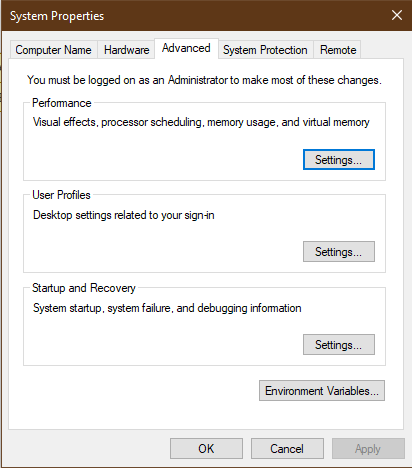
**Step 4: Add GCC to the Windows PATH Environment Variable**

To use GCC from the standard Windows Command Prompt or PowerShell, you need to add its location to the Windows PATH.

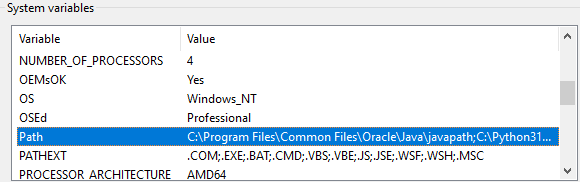
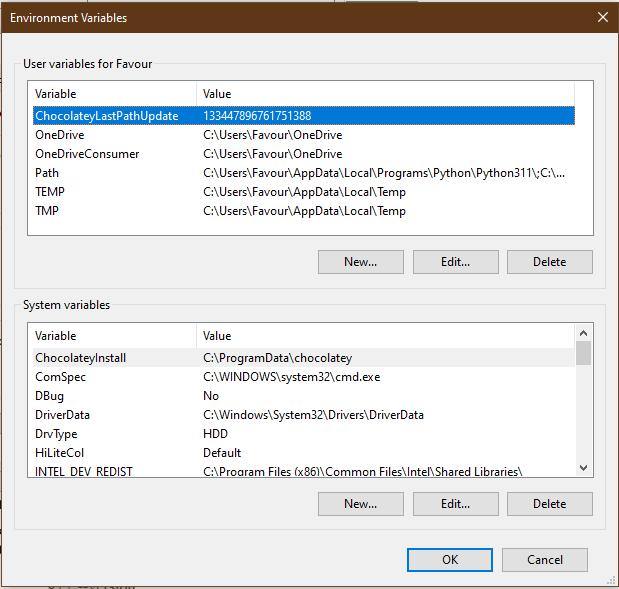
1. Find the bin folder: The C++ compiler (g++.exe) and other tools are now located in a bin folder within your MSYS2 installation directory. The typical path will be:

C:\msys64\ucrt64\bin

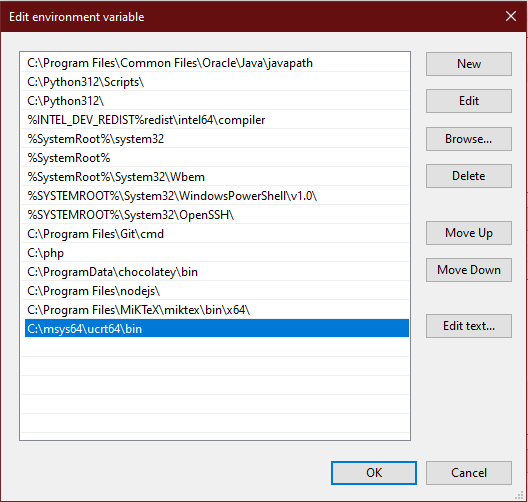
1. **Edit the Environment Variables:**
   * Press the **Windows key**, type env, and select **"Edit the system environment variables"**.



* + In the System Properties window that opens, click the **"Environment Variables..."** button.



* + In the "System variables" section, find and select the **Path** variable, then click **"Edit..."**.



* + Click **"New"**2 and paste the path to the bin folder: C:\msys64\ucrt64\bin.
  + Click **OK** on all the open windows to save the changes.

**Step 5: Verify the Installation**

To ensure everything is working correctly, open a new Command Prompt or PowerShell window (it's essential to open a new one after changing the PATH).

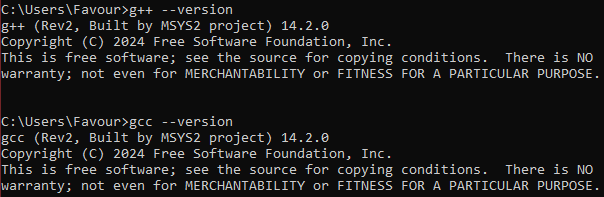
1. **Check the g++ version:** Type the following command and press Enter.

``g++ --version``

1. **Check the gcc version:** You can also check the GCC version.

Bash

``gcc –version``



If the installation was successful, you will see the version information for the g++ and gcc compilers printed in the terminal. You are now ready to compile C++ programs from the command line!

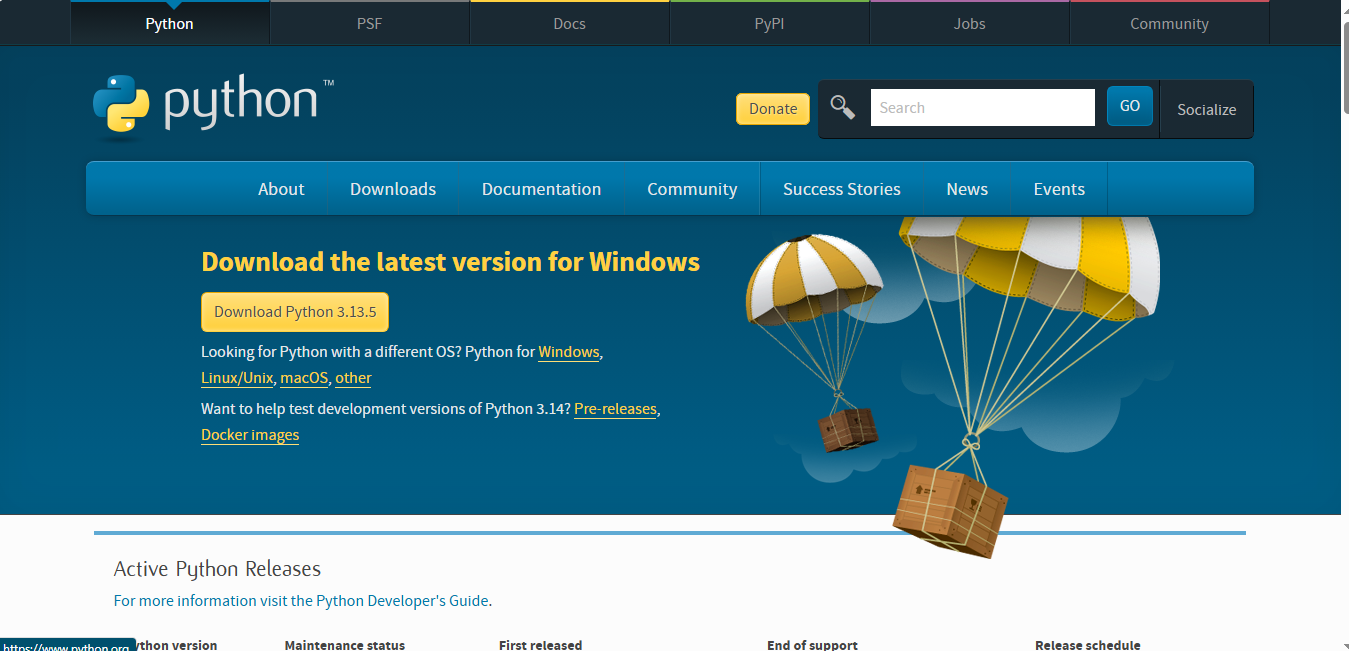
**1.2 Installing Python**

**Part 1: Installing Python**

The official installer from the Python Software Foundation is the best way to set it up.

**Step 1: Download the Python Installer**

1. **Open your web browser** and go to the official Python website's download page for Windows: <https://www.python.org/downloads/windows/>
2. Then Click on the **"DownloadPython 3.13.15"** botton to download the executable file (.exe).

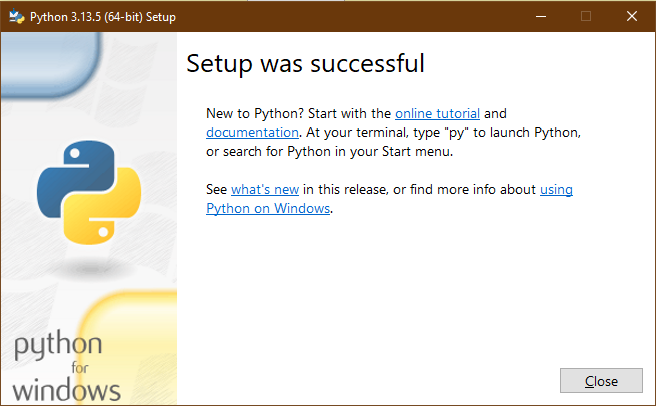


**Step 2: Run the Python Installer**

1. Once the download is complete, open your Downloads folder and **double-click the installer file** to run it.
2. On the first screen of the installer, it is **very important** to check the box at the bottom that says **"Add Python to PATH"**. This will allow you to run Python from the command prompt easily.



1. After checking the box, click on **"Install Now"**. This will install Python with the recommended settings.
2. The installation process will begin. It should only take a few minutes.
3. Once the installation is complete, you will see a "Setup was successful" message. You can now close the installer.



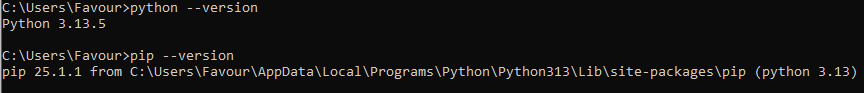
**Step 3: Verify the Python Installation**

1. To make sure Python was installed correctly, open the **Command Prompt**. You can do this by pressing the **Windows key**, typing cmd, and pressing **Enter**.
2. In the Command Prompt window, type the following command and press **Enter**:

``python --version``

1. If Python is installed and added to the PATH correctly, you will see the version number you just installed (e.g., Python 3.13.5).
2. You can also check the Python Package Installer (pip) version with:

``pip --version``



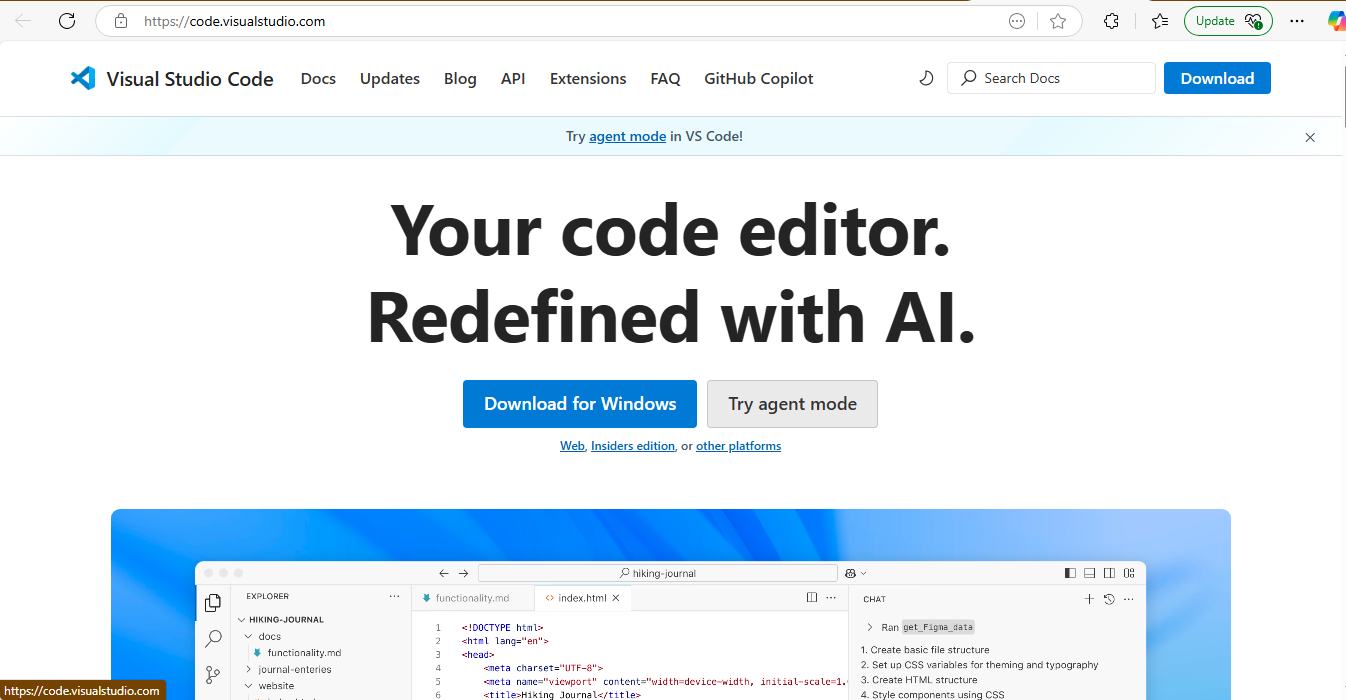
You have now successfully installed Python on your Windows 10 machine.

**Part 2: Installing Visual Studio Code (VS Code)**

VS Code is a lightweight but powerful code editor from Microsoft that is excellent for Python development.

**Step 1: Download the VS Code Installer**

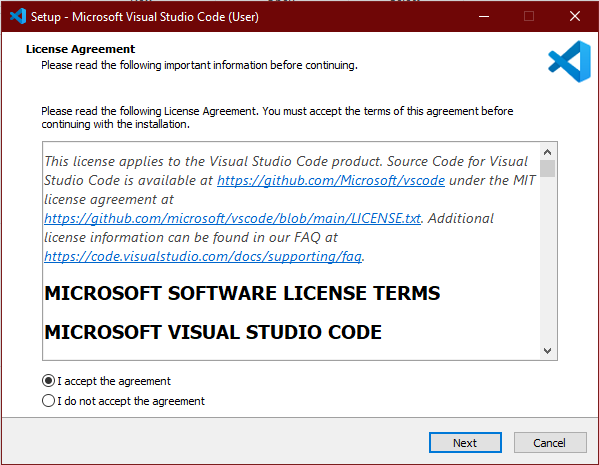
1. **Open your web browser** and go to the official Visual Studio Code website: <https://code.visualstudio.com/>
2. The website will automatically detect that you are on Windows and should prominently display a download button for the "System Installer".



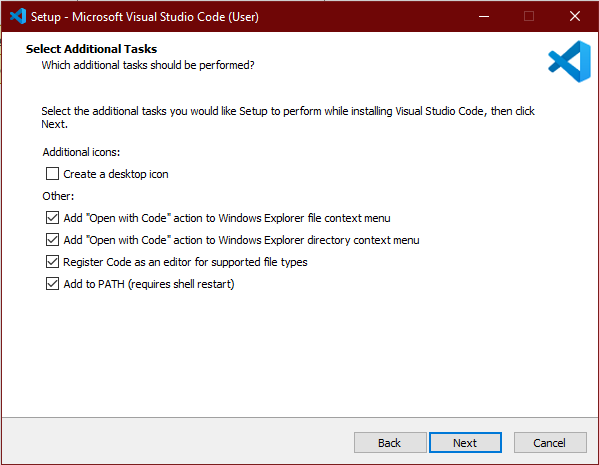
1. Click the **"Download for Windows"** button to download the installer.

**Step 2: Run the VS Code Installer**

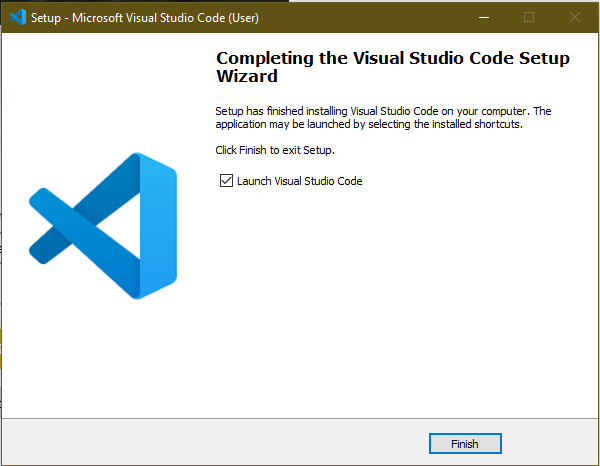
1. After the download is finished, find the installer file in your Downloads folder and **double-click it**.
2. **Accept the license agreement** and click "Next".



1. You can leave the installation location as the default and click "Next".



1. You can also leave the Start Menu folder as the default and click "Next".
2. On the "Select Additional Tasks" screen, it is highly recommended to check the following boxes:
   * **"Create a desktop icon"** (for easy access).
   * **"Add 'Open with Code' action to Windows Explorer file context menu"**.
   * **"Add 'Open with Code' action to Windows Explorer directory context menu"**.
   * **"Register Code as an editor for supported file types"**.1
   * Ensure that **"Add to PATH"** is checked (it usually is by default).
3. Click **"Next"** and then **"Install"**.
4. Once the installation is complete, you can leave the "Launch Visual Studio Code" box checked and click **"Finish"**.

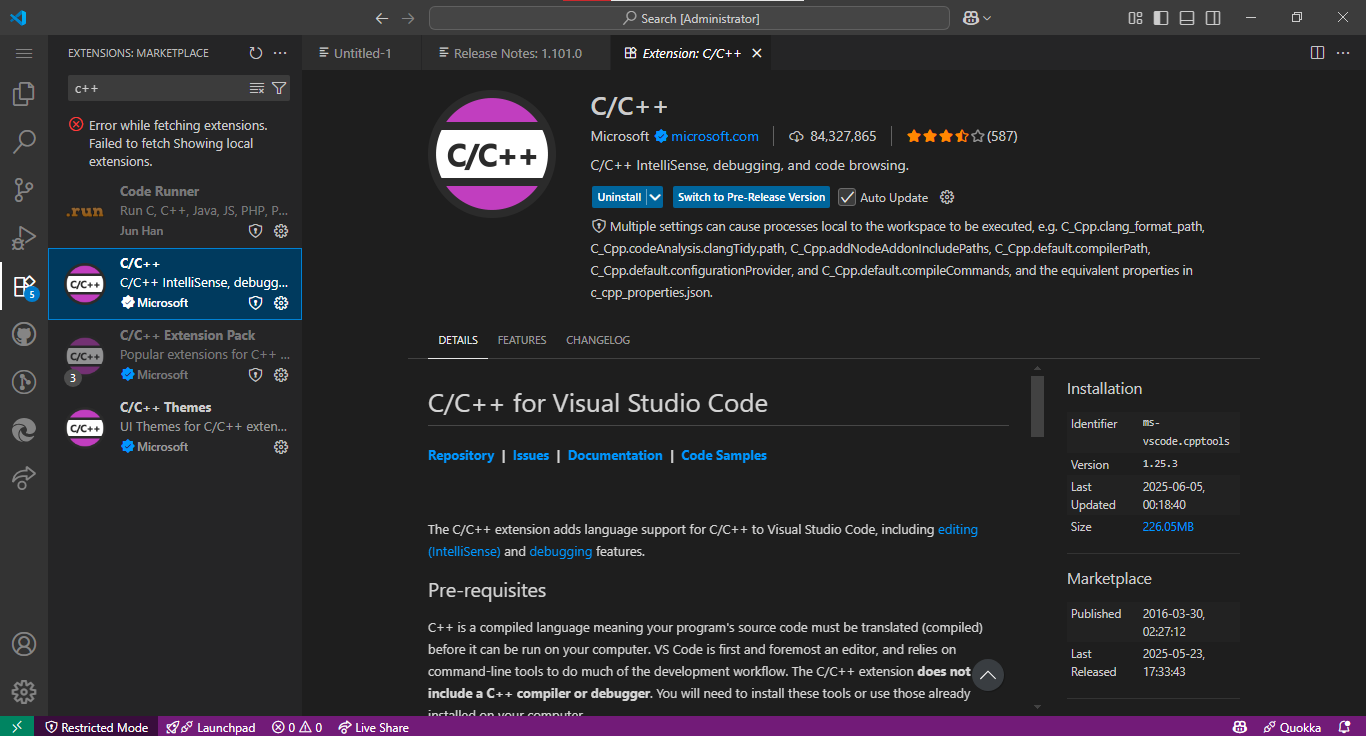


**Step 3: Install the Python/C++ Extension for VS Code**

To turn VS Code into a Python/C++ IDE, you need to install the official Python extension from Microsoft.

1. When VS Code opens, look at the left-hand sidebar. Click on the **Extensions** icon (it looks like four squares, with one flying off).
2. In the search bar at the top of the Extensions view, type **Python/C++**.
3. The first result should be the one named **"Python/C++"** and published by **Microsoft**. It will have a blue checkmark.
4. Click the **"Install"** button next to the Python/C++ extension. It will be downloaded and installed automatically.





**Section 2: Input and Output Code in C++ and Python**

**2.1 Python Version**

# Simple Python program to take input and give output

def main():

    name = input("Enter your name: ")

    age = int(input("Enter your age: "))

    print(f"\nHello, {name}!")

    print(f"You are {age} years old.")

    print(f"In 5 years, you will be {age + 5} years old.")

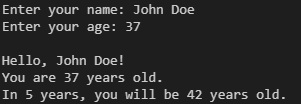
if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Explanation:**

* input() is used to accept user input.
* int() converts string input to integer.
* f"..." is used for formatted output.
* Run the code in an interactive terminal/ Dedicated terminal

**Output:**

****

**2.2 C++ Version**

// Simple C++ program to take input and give output

#include <iostream>

#include <string>

using namespace std;

int main()

{

    string name;

    int age;

    cout << "Enter your name: ";

    getline(cin, name); // use getline to accept full names

    cout << "Enter your age: ";

    cin >> age;

    cout << "\nHello, " << name << "!" << endl;

    cout << "You are " << age << " years old." << endl;

    cout << "In 5 years, you will be " << age + 5 << " years old." << endl;

    return 0;

}

///  To run the program use the bellow comand

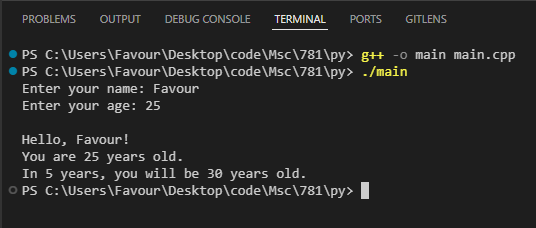
///  g++ -o main main.cpp

///  ./main

**Explanation:**

* getline() reads full name including spaces.
* cin is used for numeric input.
* cout is used to print messages.

**Output:**

****

**Section 3: Gauss-Jordan Elimination Code**

**3.1 Python Version**

def gauss\_jordan(augmented, n):

    for i in range(n):

        # Find the row with the maximum value in the current column

        max\_row = max(range(i, n), key=lambda r: abs(augmented[r][i]))

        if abs(augmented[max\_row][i]) < 1e-12:

            raise ValueError("Matrix is singular or nearly singular.")

        # Swap current row with the row with the max pivot

        if max\_row != i:

            augmented[i], augmented[max\_row] = augmented[max\_row], augmented[i]

        # Normalize pivot row

        pivot = augmented[i][i]

        augmented[i] = [x / pivot for x in augmented[i]]

        # Eliminate other rows

        for k in range(n):

            if k != i:

                factor = augmented[k][i]

                augmented[k] = [a - factor \* b for a, b in zip(augmented[k], augmented[i])]

    return augmented

def main():

    try:

        n = int(input("Enter the size of the matrix (n): "))

        if n <= 0:

            raise ValueError("Matrix size must be positive.")

        print(f"\nEnter the {n}x{n} matrix A row by row (each row has {n} values):")

        A = []

        for i in range(n):

            row = list(map(float, input(f"Row {i+1} of A: ").split()))

            while len(row) != n:

                print(f"Each row must have {n} values.")

                row = list(map(float, input(f"Row {i+1} of A: ").split()))

            A.append(row)

        print(f"\nEnter the RHS vector b (must have {n} values):")

        b = list(map(float, input("b: ").split()))

        while len(b) != n:

            print(f"RHS vector must have {n} values.")

            b = list(map(float, input("b: ").split()))

        # Build augmented matrix: [A | I | b]

        augmented = []

        for i in range(n):

            identity = [1.0 if i == j else 0.0 for j in range(n)]

            augmented.append(A[i] + identity + [b[i]])

        # Perform Gauss-Jordan Elimination with pivoting

        augmented = gauss\_jordan(augmented, n)

        # Output solution vector x

        print("\nSolution Vector x:")

        for i in range(n):

            print(f"x{i+1} = {augmented[i][-1]:.6f}")

        # Output inverse of A

        print("\nInverse of A:")

        for i in range(n):

            print(" ".join(f"{augmented[i][n + j]:.6f}" for j in range(n)))

    except ValueError as e:

        print("Input Error:", e)

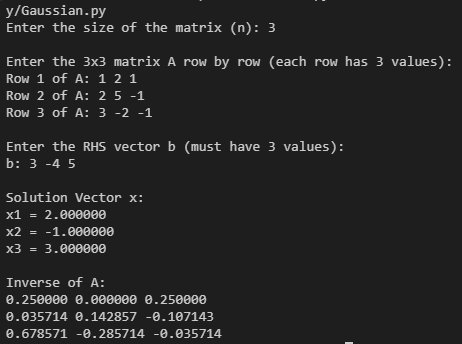
if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Explanation:**

* Performs Gauss-Jordan elimination with row pivoting.
* Normalizes pivot row.
* Eliminates other rows to produce identity matrix and solution.
* Run the code in an interactive terminal/ Dedicated terminal

**Output:**



**3.2 C++ Version**

#include <iostream>

#include <vector>

#include <iomanip>

#include <cmath>

#include <stdexcept>

using namespace std;

void gaussJordan(vector<vector<double>> &augmented, int n)

{

    for (int i = 0; i < n; ++i)

    {

        // Find the row with the max pivot in column i

        int maxRow = i;

        for (int k = i + 1; k < n; ++k)

        {

            if (fabs(augmented[k][i]) > fabs(augmented[maxRow][i]))

            {

                maxRow = k;

            }

        }

        if (fabs(augmented[maxRow][i]) < 1e-12)

        {

            throw runtime\_error("Matrix is singular or nearly singular.");

        }

        // Swap rows if needed

        if (maxRow != i)

        {

            swap(augmented[i], augmented[maxRow]);

        }

        // Normalize pivot row

        double pivot = augmented[i][i];

        for (int j = 0; j < 2 \* n + 1; ++j)

        {

            augmented[i][j] /= pivot;

        }

        // Eliminate other rows

        for (int k = 0; k < n; ++k)

        {

            if (k != i)

            {

                double factor = augmented[k][i];

                for (int j = 0; j < 2 \* n + 1; ++j)

                {

                    augmented[k][j] -= factor \* augmented[i][j];

                }

            }

        }

    }

}

int main()

{

    int n;

    cout << "Enter the size of the matrix (n): ";

    cin >> n;

    if (n <= 0)

    {

        cerr << "Matrix size must be positive." << endl;

        return 1;

    }

    vector<vector<double>> A(n, vector<double>(n));

    vector<double> b(n);

    cout << "\nEnter the " << n << "x" << n << " matrix A row by row:" << endl;

    for (int i = 0; i < n; ++i)

    {

        cout << "Row " << i + 1 << ": ";

        for (int j = 0; j < n; ++j)

        {

            cin >> A[i][j];

        }

    }

    cout << "\nEnter the RHS vector b (" << n << " values): ";

    for (int i = 0; i < n; ++i)

    {

        cin >> b[i];

    }

    // Build augmented matrix: [A | I | b]

    vector<vector<double>> augmented(n, vector<double>(2 \* n + 1));

    for (int i = 0; i < n; ++i)

    {

        // Copy A

        for (int j = 0; j < n; ++j)

        {

            augmented[i][j] = A[i][j];

        }

        // Identity matrix

        for (int j = 0; j < n; ++j)

        {

            augmented[i][n + j] = (i == j) ? 1.0 : 0.0;

        }

        // Append b

        augmented[i][2 \* n] = b[i];

    }

    try

    {

        gaussJordan(augmented, n);

    }

    catch (runtime\_error &e)

    {

        cout << "Error: " << e.what() << endl;

        return 1;

    }

    // Output solution vector x

    cout << "\nSolution Vector x:\n";

    cout << fixed << setprecision(6);

    for (int i = 0; i < n; ++i)

    {

        cout << "x" << i + 1 << " = " << augmented[i][2 \* n] << endl;

    }

    // Output inverse of A

    cout << "\nInverse of A:\n";

    for (int i = 0; i < n; ++i)

    {

        for (int j = n; j < 2 \* n; ++j)

        {

            cout << setw(10) << augmented[i][j] << " ";

        }

        cout << endl;

    }

    return 0;

}

///  To run the program use the bellow comand

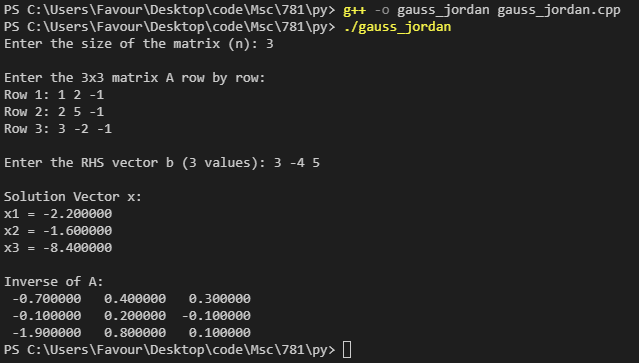
///  g++ -o gauss\_jordan gauss\_jordan.cpp

///  ./gauss\_jordan

**Explanation:**

* Uses row pivoting to avoid division by zero.
* Each row is scaled and used to eliminate corresponding entries in other rows.
* Final matrix yields the inverse and the solution vector.

**Output:**

****

**Conclusion**

This report documents how to install C++ and Python environments, explains basic input/output operations, and describes how to solve linear systems using Gauss-Jordan elimination in both programming languages.