Visual Studio Code and Google Colab

Panchatcharam M

Visual Studio Code

VS Code

Visual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft.

Lightweight yet powerful, it provides a rich programming environment that supports development in multiple languages, including JavaScript, Python, C++, Java, and more.

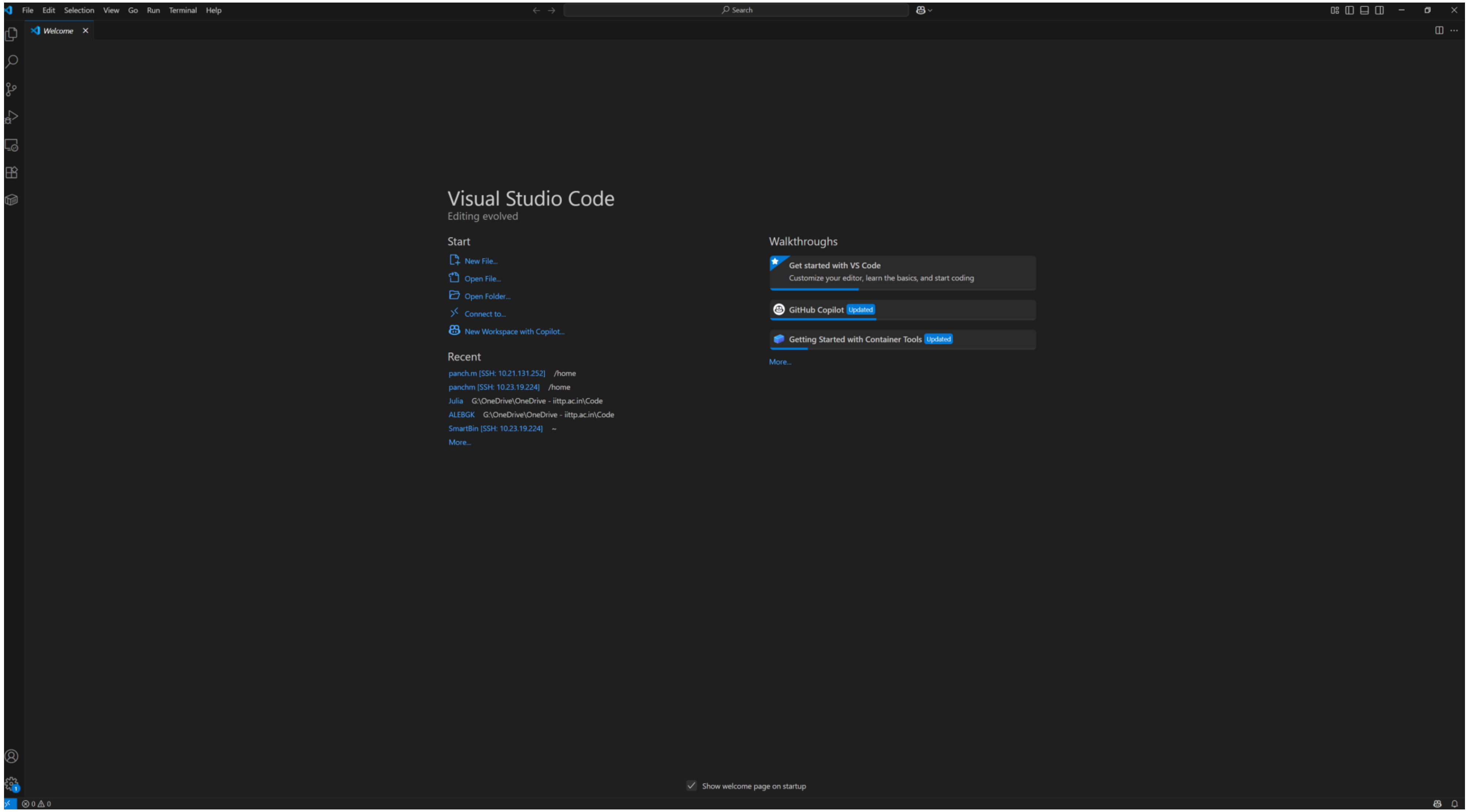
Vs code offers

VS Code offers intelligent code completion (IntelliSense), debugging tools, built-in Git integration, syntax highlighting, and code navigation features that streamline the development process.

Its vast marketplace allows users to install extensions for frameworks, languages, themes, and developer tools, making it highly customizable to fit various workflows

Designed for speed and efficiency, VS Code works across Windows, macOS, and Linux, making it a popular choice for web developers, software engineers, and data scientists worldwide.

VS CODE IDE



Visual Studio Code
Editing evolved

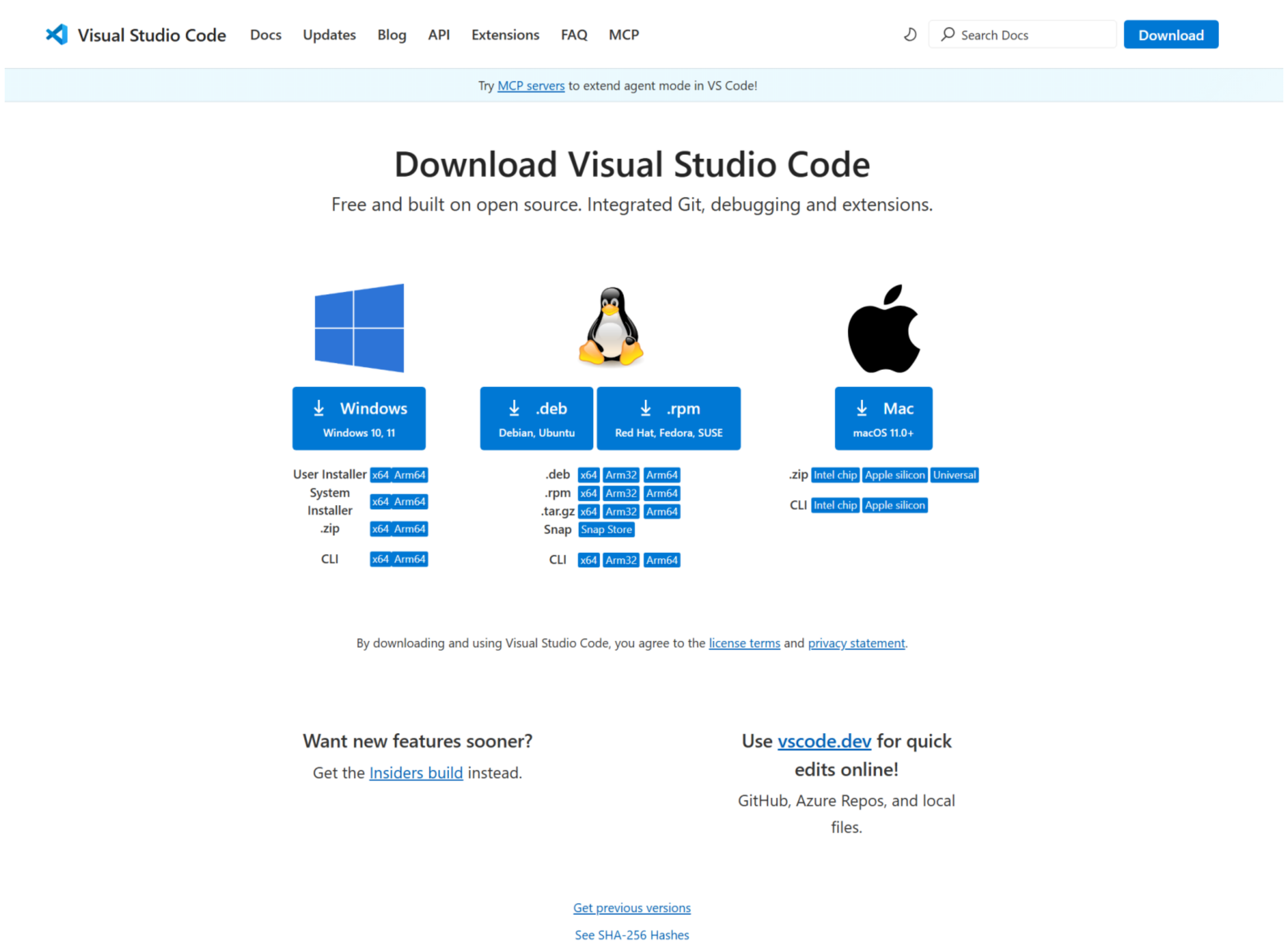
Start
- New File...
- Open File...
- Open Folder...
- Connect to...
- New Workspace with Copilot...

Recent
- panch.m [SSH: 10.21.131.252]   /home
- panchm [SSH: 10.23.19.224]   /home
- Julia   G:\OneDrive\OneDrive - iittp.ac.in\Code
- ALEBGK   G:\OneDrive\OneDrive - iittp.ac.in\Code
- SmartBin [SSH: 10.23.19.224]   ~
- More...

Walkthroughs
- Get started with VS Code
  Customize your editor, learn the basics, and start coding
- GitHub Copilot Updated
- Getting Started with Container Tools Updated
- More...

✓ Show welcome page on startup

DOWNLOAD



Visual Studio Code Docs Updates Blog API Extensions FAQ MCP

Search Docs Download

Try MCP servers to extend agent mode in VS Code!

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

User Installer	x64 Arm64
System Installer	x64 Arm64
.zip	x64 Arm64
CLI	x64 Arm64

.deb	x64 Arm32 Arm64
.rpm	x64 Arm32 Arm64
.tar.gz	x64 Arm32 Arm64
Snap	Snap Store
CLI	x64 Arm32 Arm64

.zip	Intel chip Apple silicon Universal
CLI	Intel chip Apple silicon

By downloading and using Visual Studio Code, you agree to the license terms and privacy statement.

Want new features sooner?

Get the Insiders build instead.

Use vscode.dev for quick edits online!

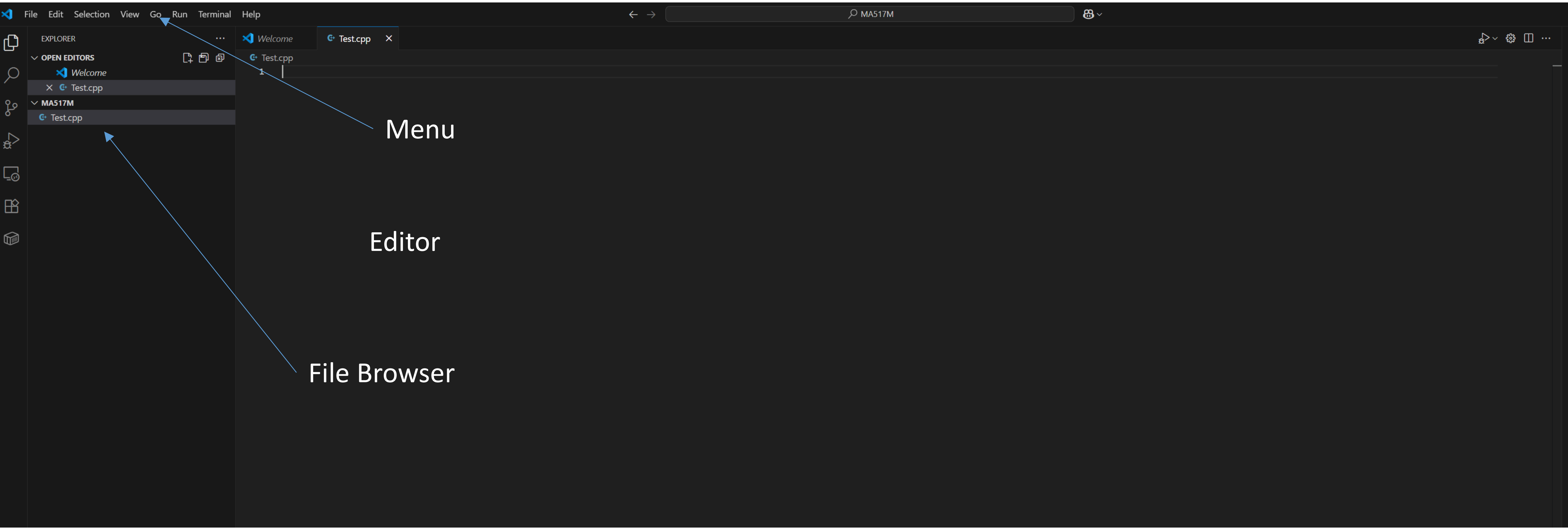
GitHub, Azure Repos, and local files.

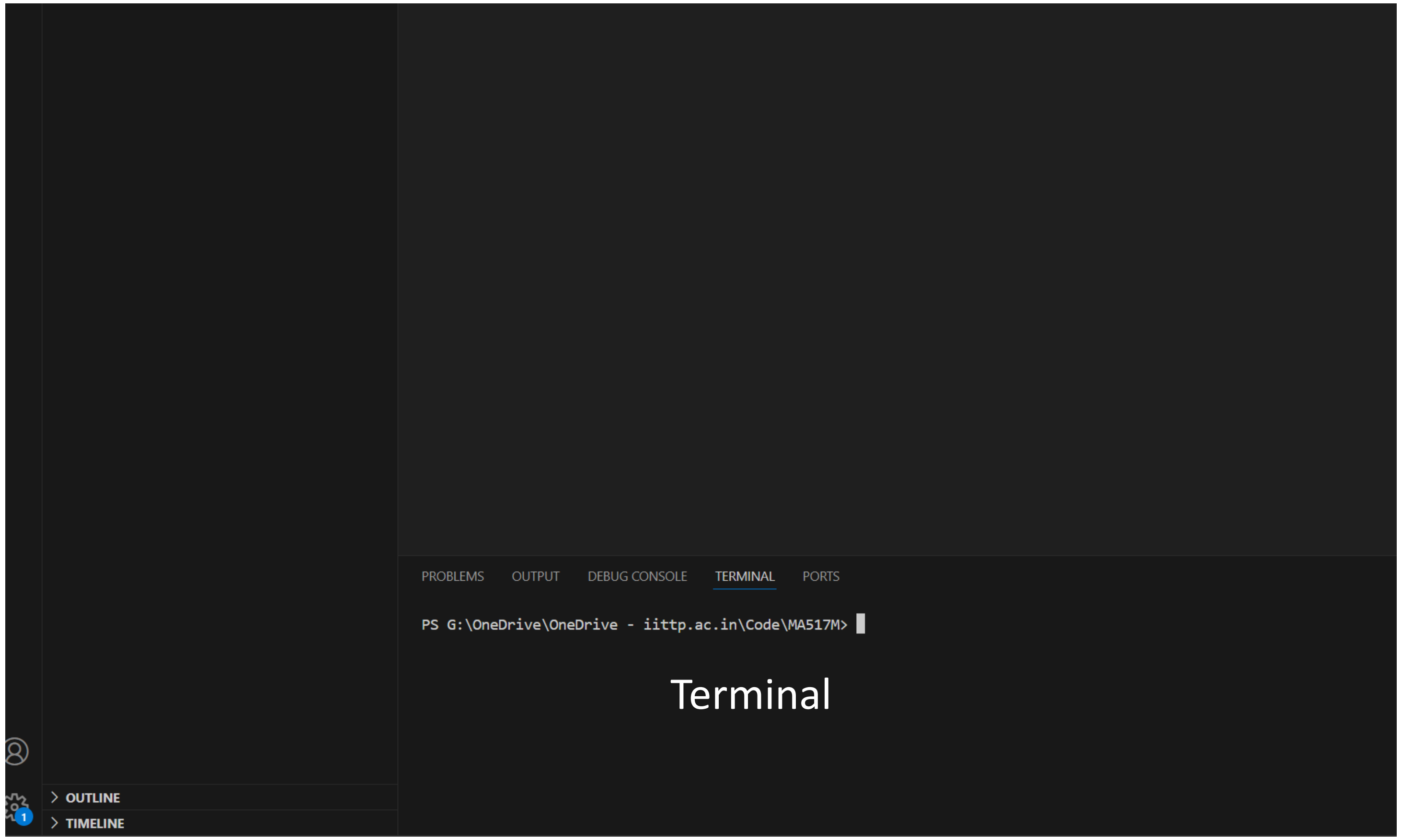
Get previous versions

See SHA-256 Hashes

https://code.visualstudio.com/download

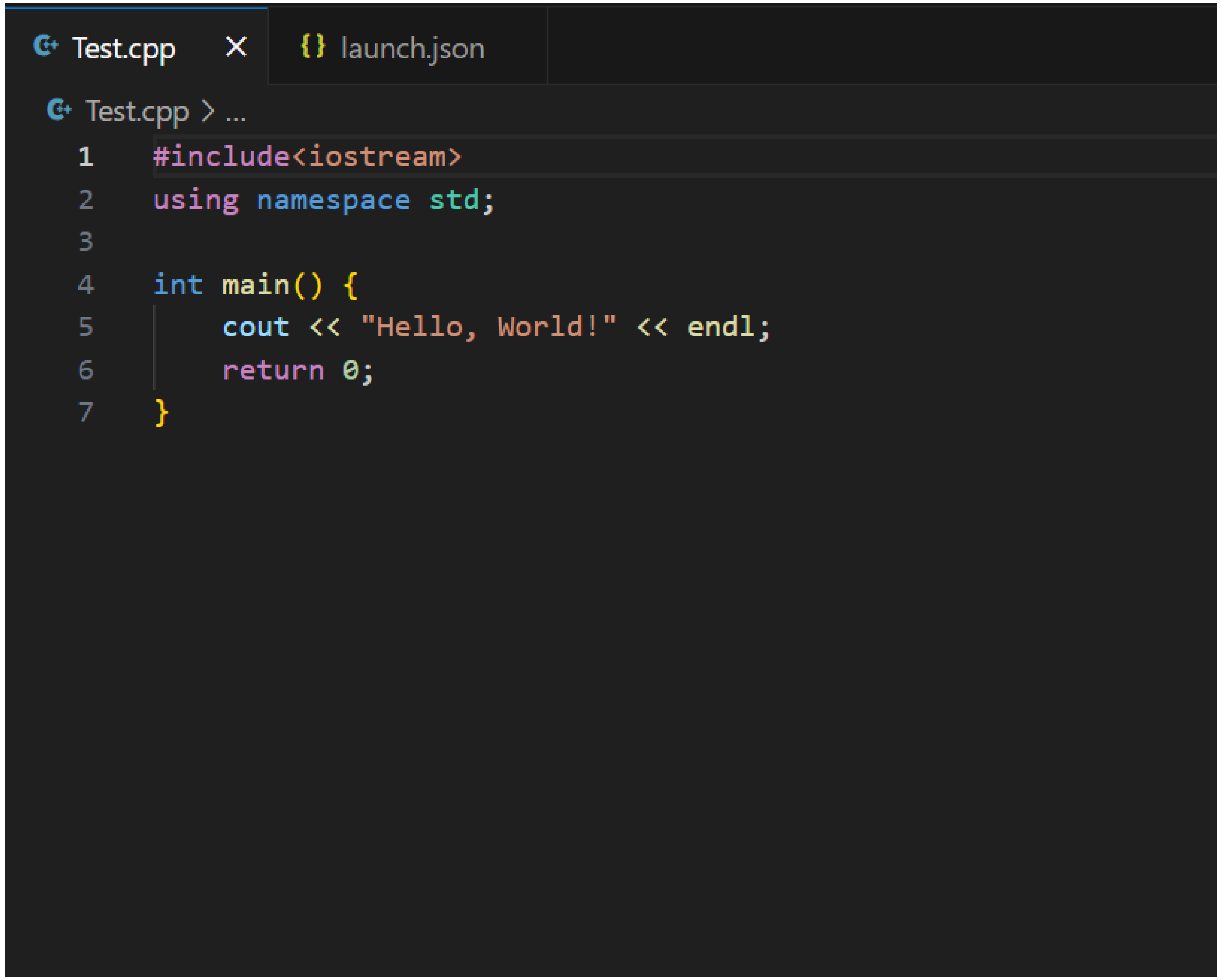
IDE


File
Edit
Selection
View
Go
Run
Terminal
Help
MA517M
EXPLORER
OPEN EDITORS
Welcome
Test.cpp
MA517M
Test.cpp
Welcome
Test.cpp
Test.cpp
Menu
Editor
File Browser



```
PS G:\OneDrive\OneDrive - iittp.ac.in\Code\MA517M>
```

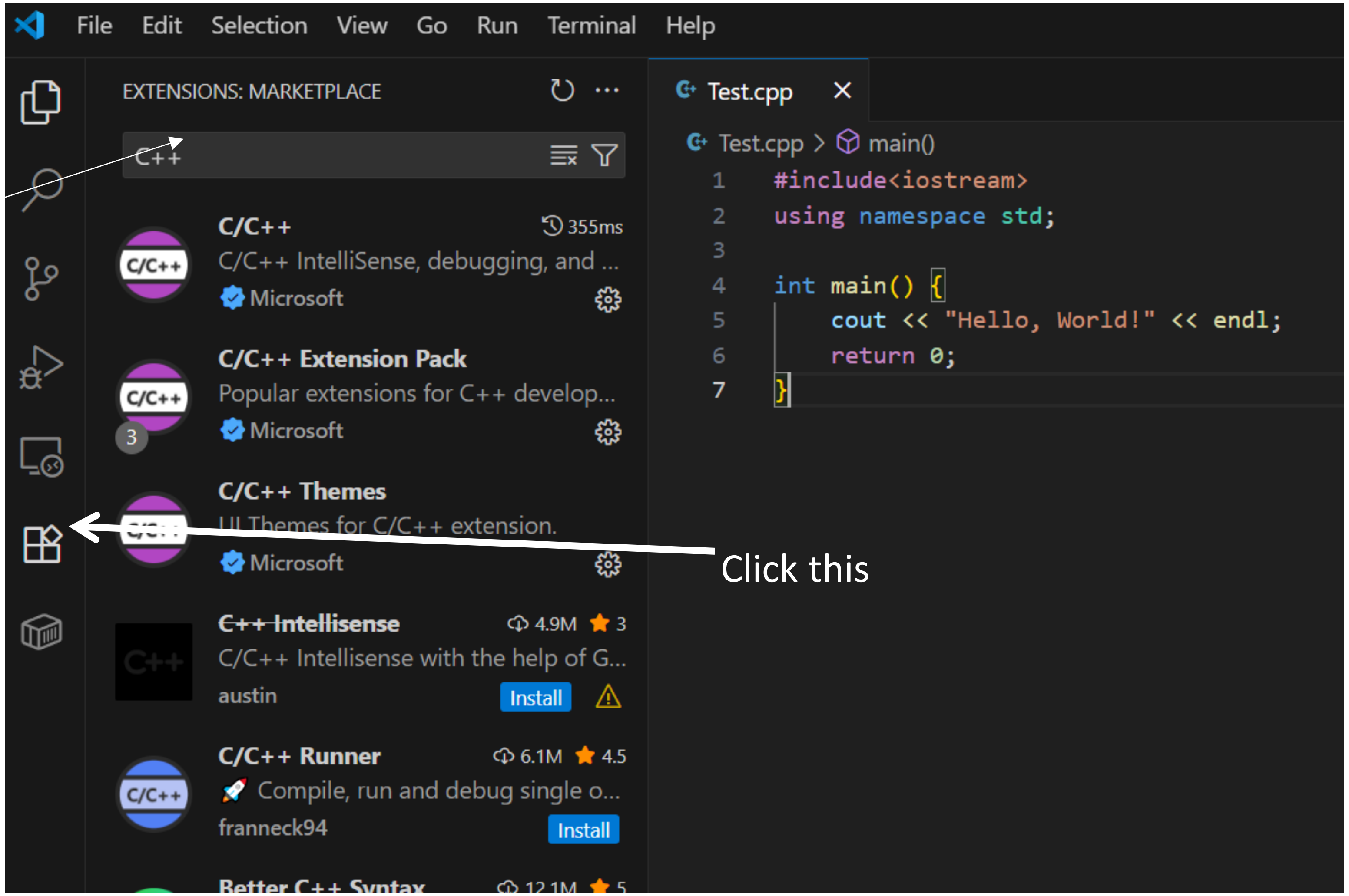
SIMPLE PROGRAM



```cpp
#include<iostream>
using namespace std;

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

INSTALL C++ EXTENSIONS

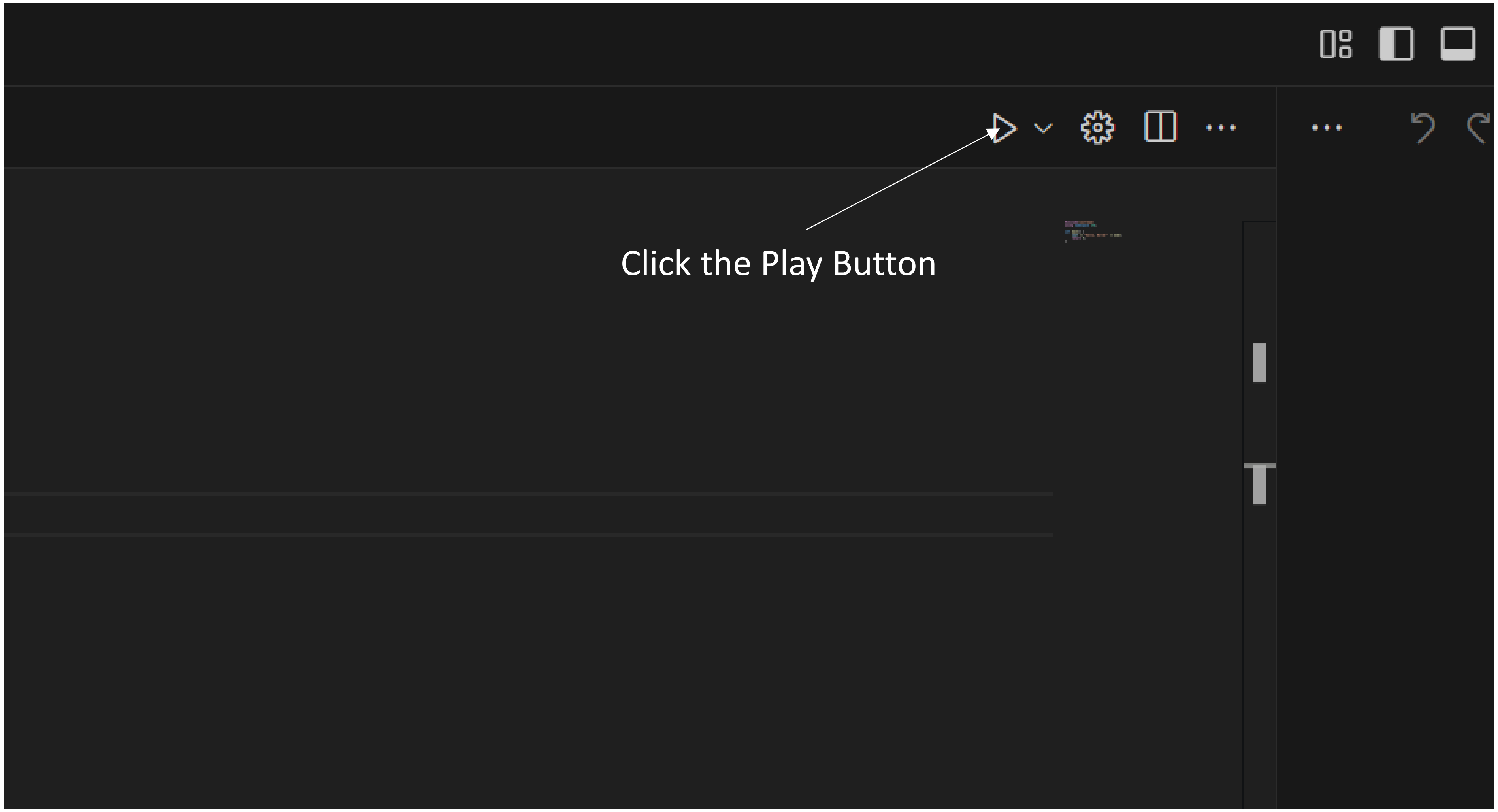
Type C++ here



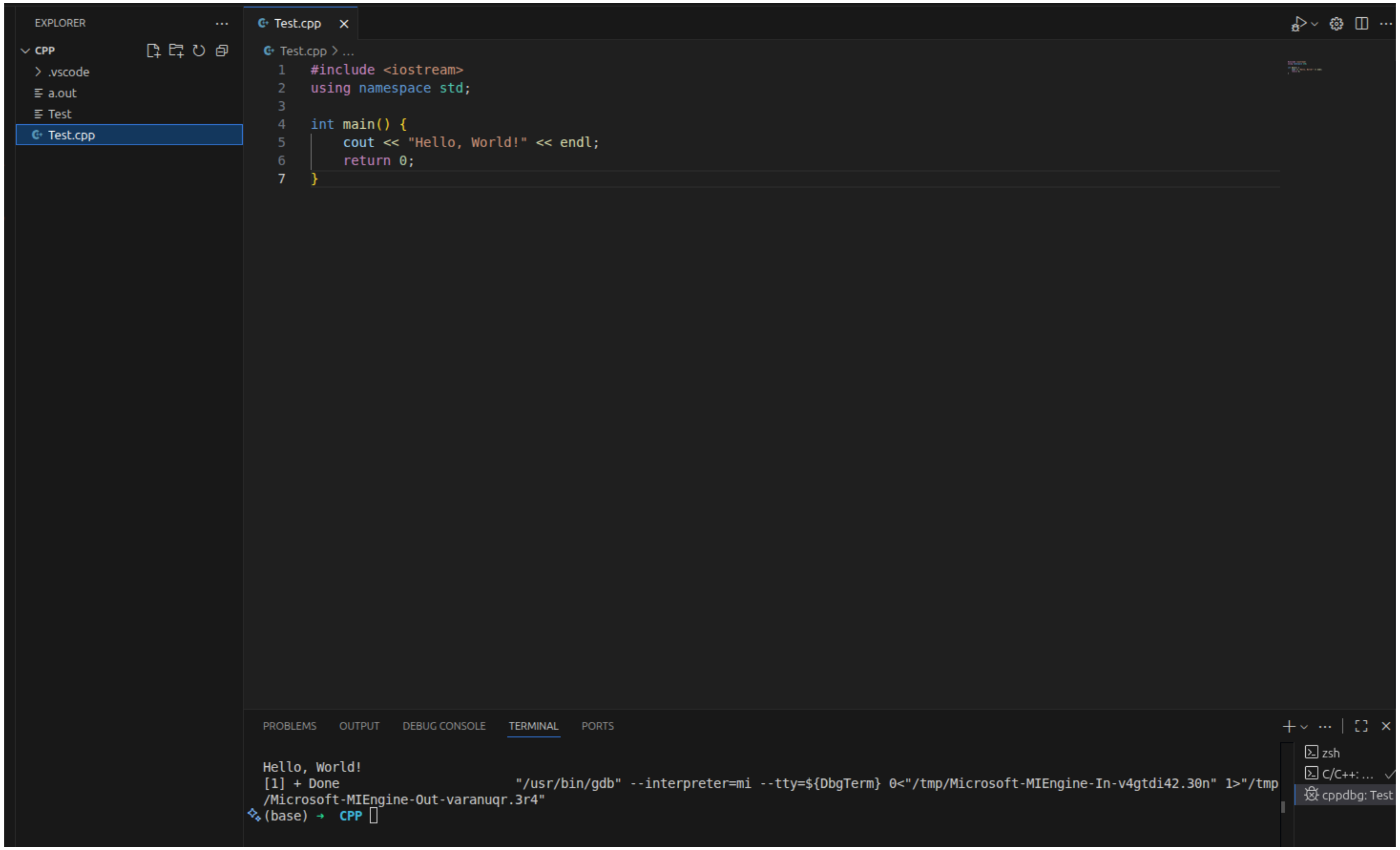
```cpp
#include<iostream>
using namespace std;

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

Click this

RUN AND EXECUTE



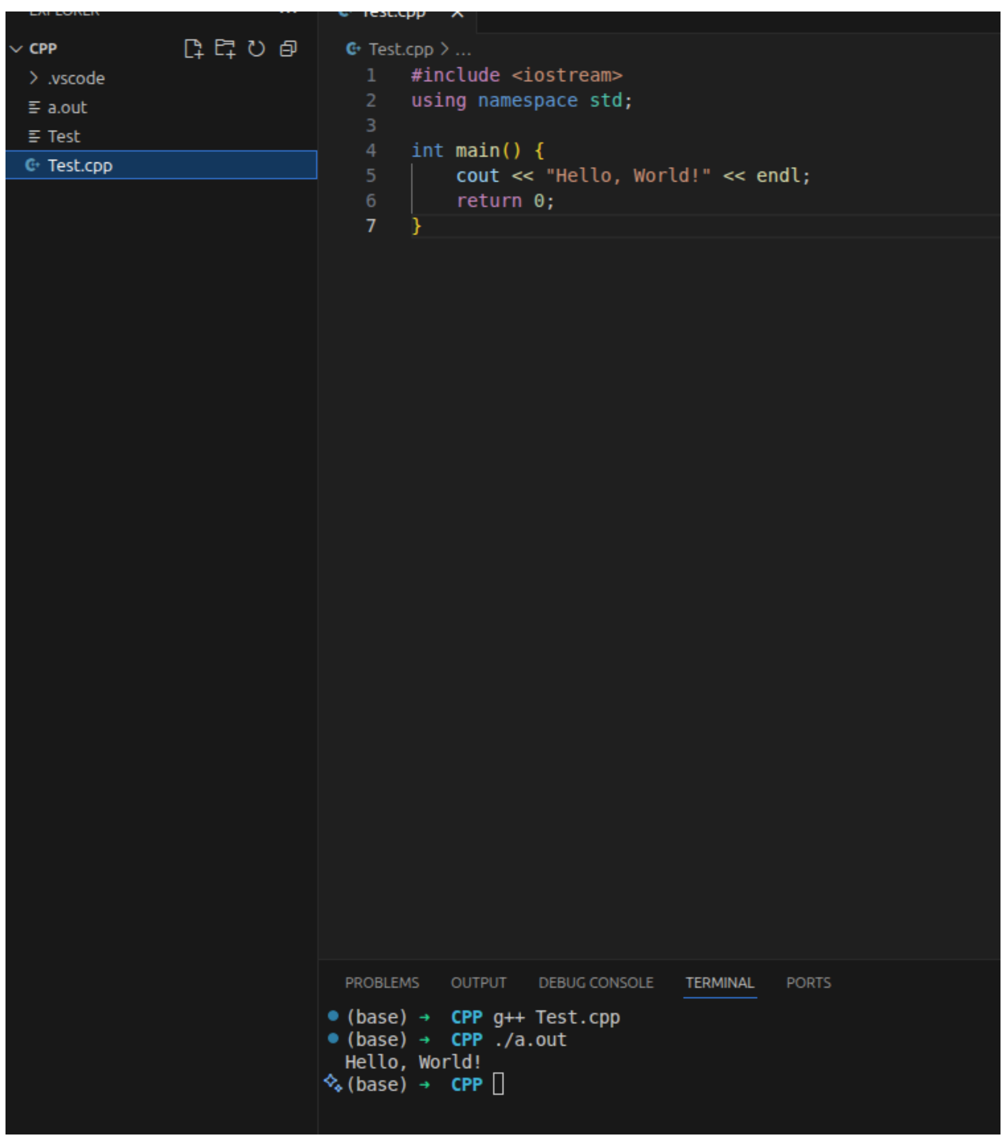
RUN AND EXECUTE



```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

```
Hello, World!
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-v4gtdi42.30n" 1>"/tmp
/Microsoft-MIEngine-Out-varanuqr.3r4"
(base) → CPP
```

RUN AND EXECUTE



```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

```
(base) → CPP g++ Test.cpp
(base) → CPP ./a.out
Hello, World!
(base) → CPP
```

RUN AND EXECUTE

```
g++ firstprogram.cpp
```

The command `g++ firstprogram.cpp` does the following:

- It loads the firstprogram.cpp file to the buffer memory
- It does all the 6 phases of compilation, preprocessing, lexical analyzing, syntax analyzing, semantic analyzing, intermediate code generation, code optimization and code generation.
- At the end of this phases, it will create an `a.out` file.
- This file is in machine language format.
- So, `g++` compiler successfully translated the high-level language to the machine-level language.

GOOGLE COLAB

GOOGLE COLAB

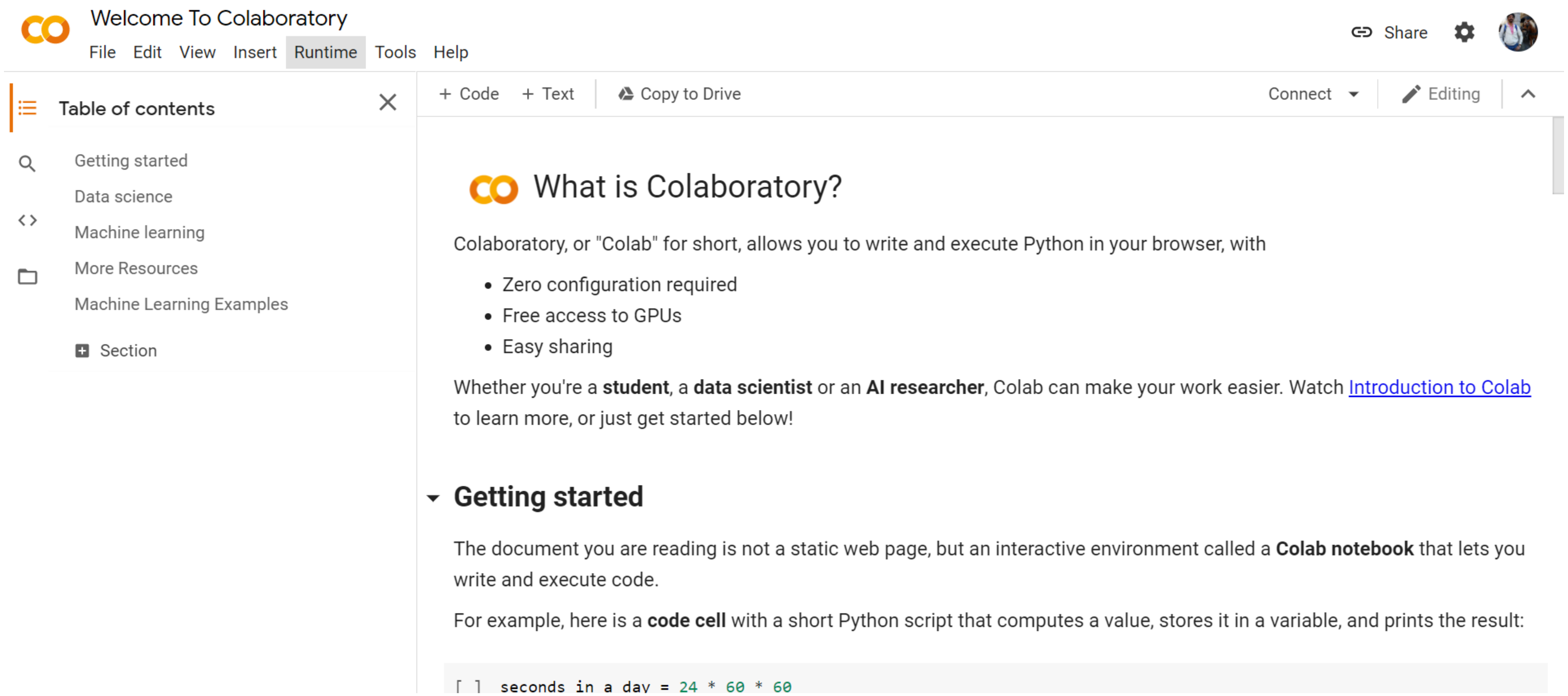
Visual Studio Code (VS Code) is a free, cloud-based platform by Google that allows you to write and run Python code in a Jupyter Notebook environment – right from the browser.

It supports machine learning, data analysis, and deep learning workflows, with free access to powerful GPUs and TPUs.

Colab is ideal for sharing code, collaborating with others, and running experiments without any setup or local installation.

GOOGLE COLAB

https://colab.research.google.com/notebooks/intro.ipynb#



What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

Getting started

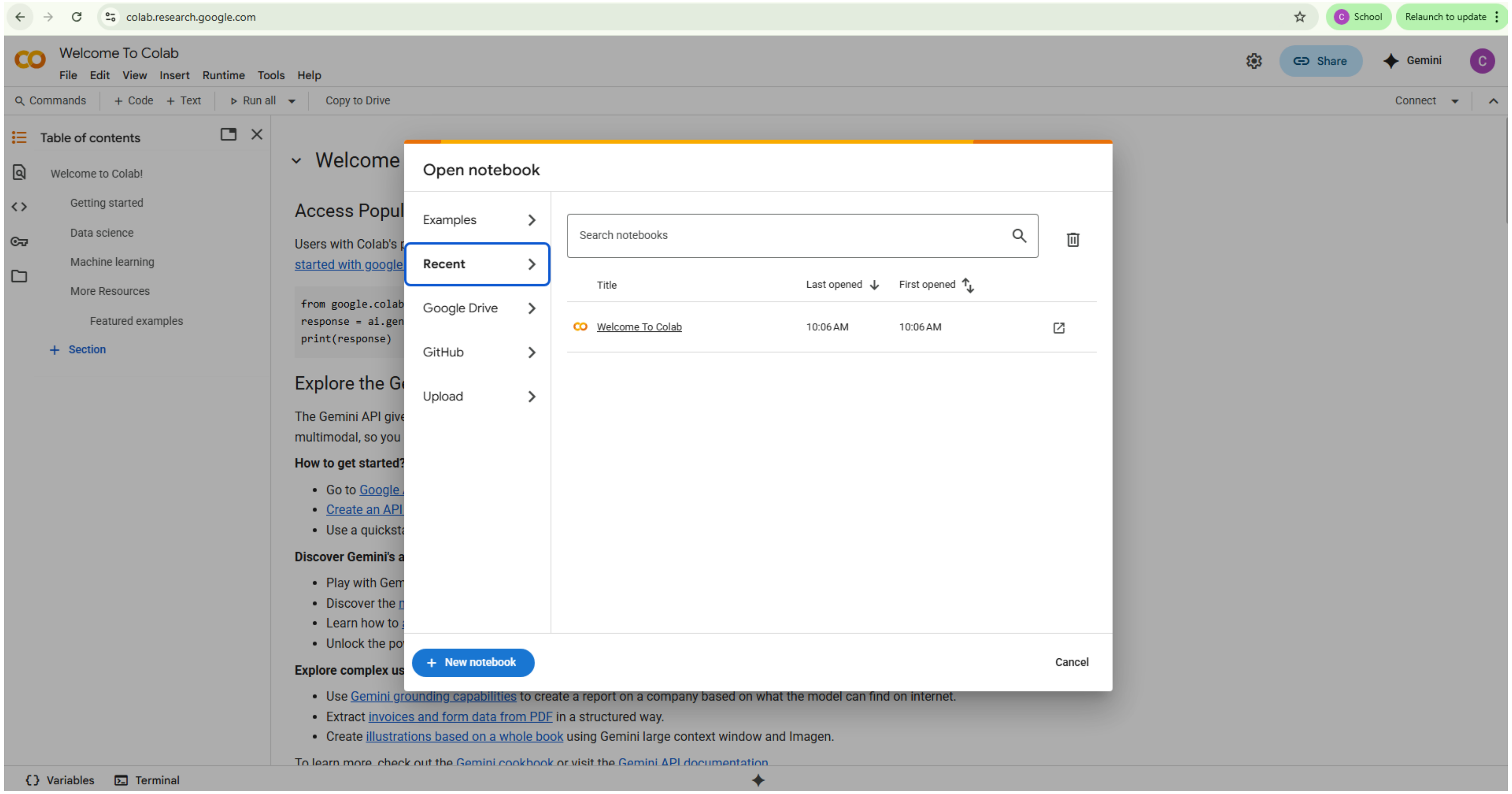
The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ]  seconds_in_a_day = 24 * 60 * 60
```

GOOGLE COLAB

https://colab.research.google.com/notebooks/intro.ipynb#



colab.research.google.com

Welcome To Colab

File Edit View Insert Runtime Tools Help

Table of contents

- Welcome to Colab!
  - Getting started
  - Data science
  - Machine learning
  - More Resources
    - Featured examples

Welcome

Access Popul

Users with Colab's p started with google

```
from google.colab
response = ai.gen
print(response)
```

Explore the G

The Gemini API give multimodal, so you

How to get started?

- Go to Google
- Create an API
- Use a quicksta

Discover Gemini's a

- Play with Gem
- Discover the r
- Learn how to
- Unlock the po

Explore complex us

- Use Gemini grounding capabilities to create a report on a company based on what the model can find on internet.
- Extract invoices and form data from PDF in a structured way.
- Create illustrations based on a whole book using Gemini large context window and Imagen.

To learn more, check out the Gemini cookbook or visit the Gemini API documentation.

Open notebook

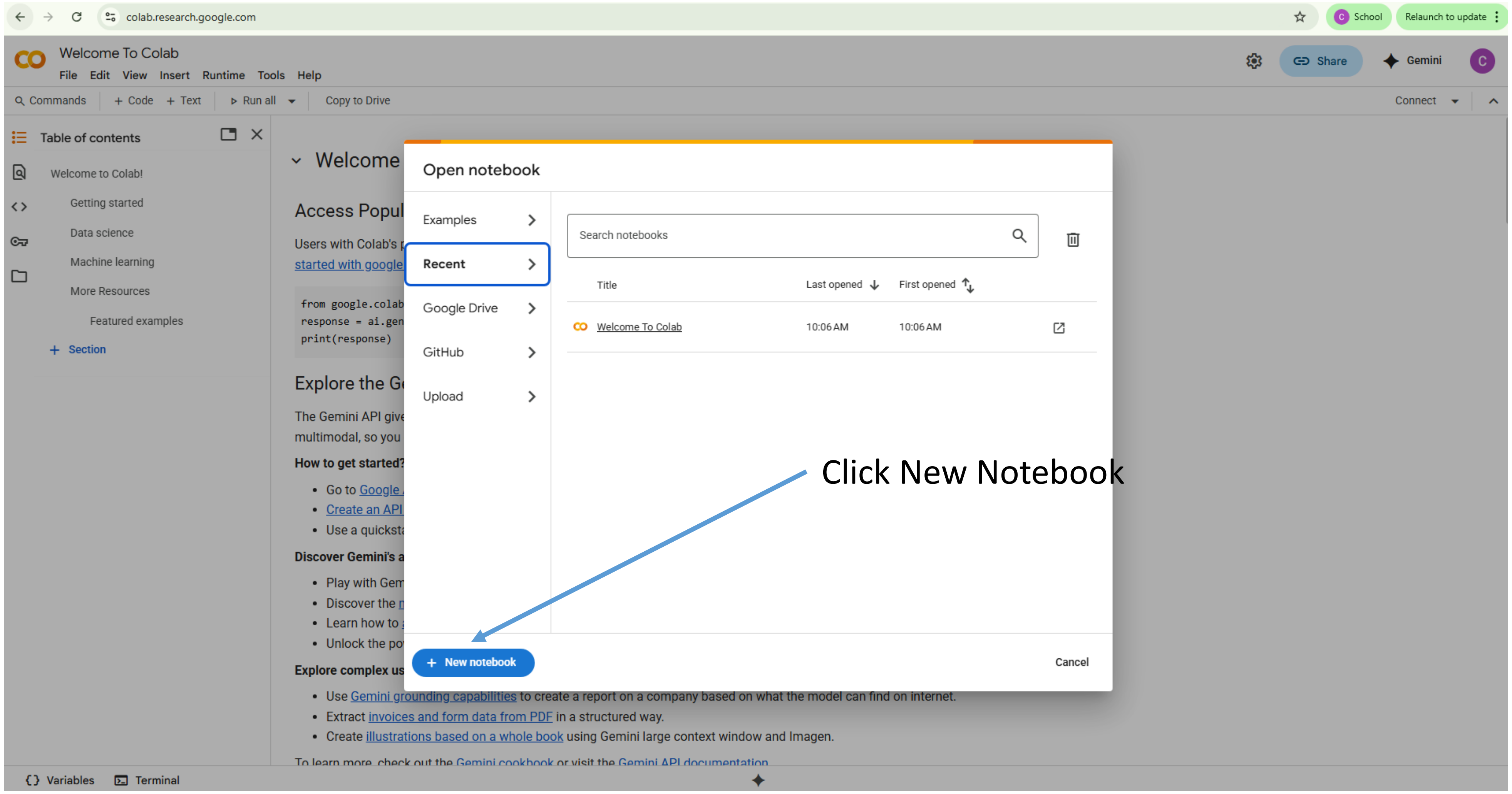
Examples | Recent | Google Drive | GitHub | Upload

Search notebooks

Title	Last opened	First opened
Welcome To Colab	10:06 AM	10:06 AM

GOOGLE COLAB

https://colab.research.google.com/notebooks/intro.ipynb#



Welcome

Access Popul

Users with Colab's p
started with google

```
from google.colab
response = ai.gen
print(response)
```

Explore the G

The Gemini API give
multimodal, so you

How to get started?

- Go to Google
- Create an API
- Use a quicksta

Discover Gemini's a

- Play with Gem
- Discover the r
- Learn how to
- Unlock the po

Explore complex us

- Use Gemini grounding capabilities to create a report on a company based on what the model can find on internet.
- Extract invoices and form data from PDF in a structured way.
- Create illustrations based on a whole book using Gemini large context window and Imagen.

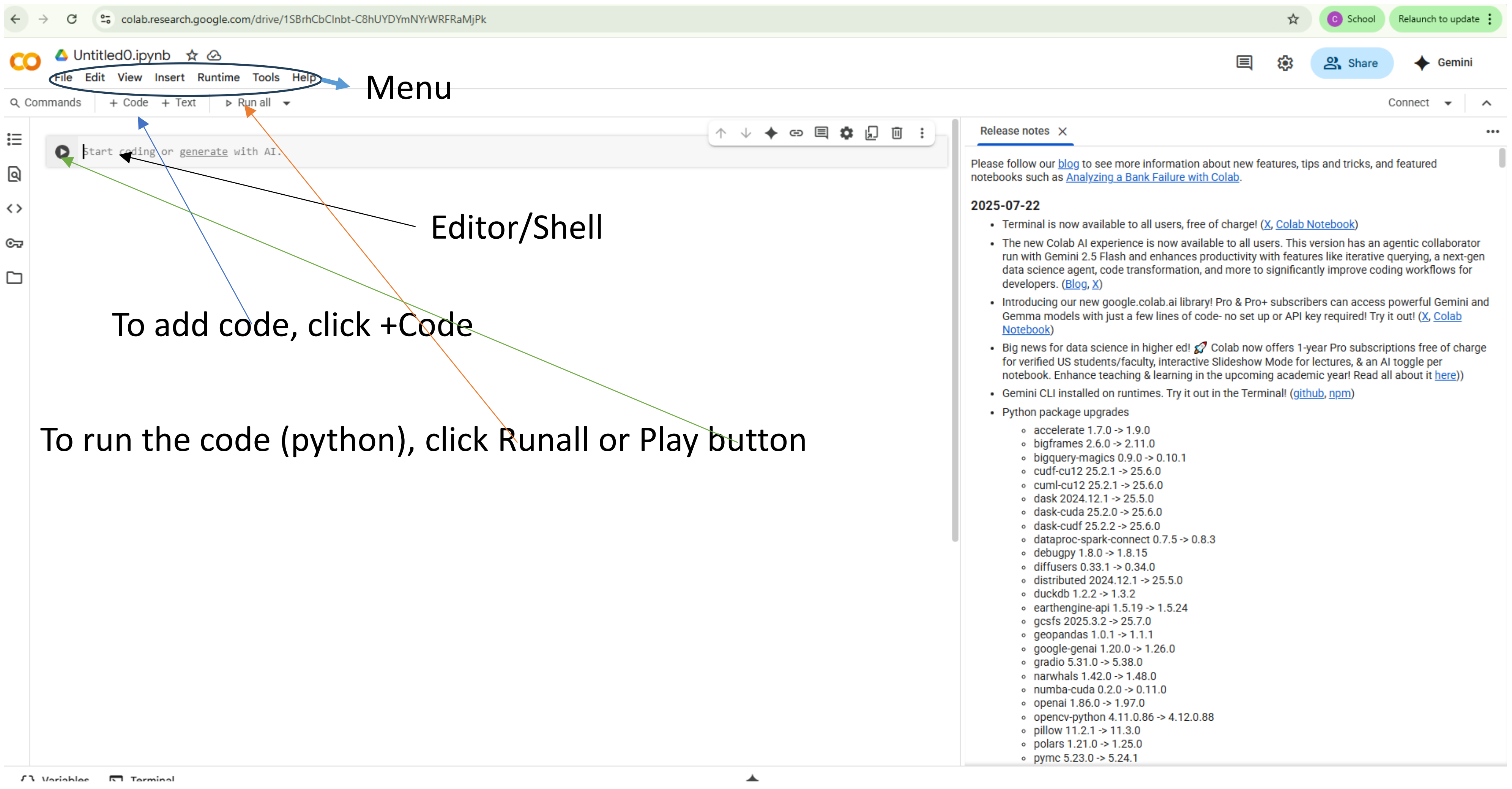
To learn more, check out the Gemini cookbook or visit the Gemini API documentation.

Title	Last opened	First opened
Welcome To Colab	10:06 AM	10:06 AM

Click New Notebook

GOOGLE COLAB

https://colab.research.google.com/notebooks/intro.ipynb#


colab.research.google.com/drive/1SBrhCbClnbt-C8hUYDYmNYrWRFRaMjPk
School
Relaunch to update
Untitled0.ipynb
File Edit View Insert Runtime Tools Help
Menu
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect
Start coding or generate with AI.
Editor/Shell
To add code, click +Code
To run the code (python), click Runall or Play button
Variables
Terminal
Release notes

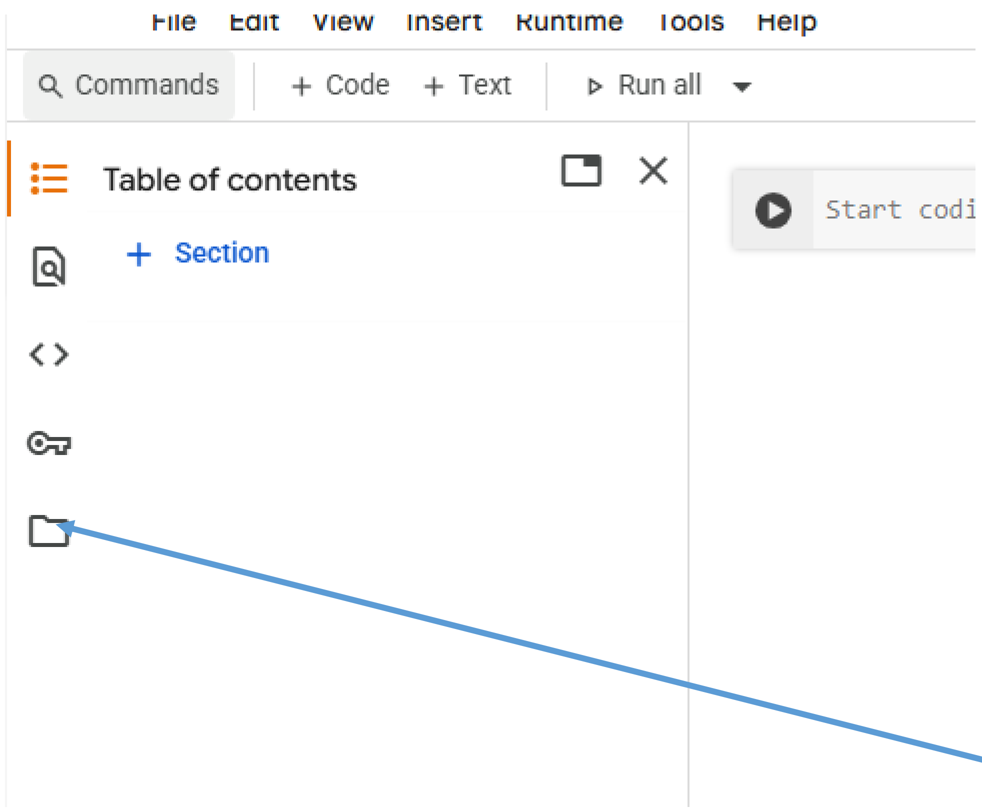
Please follow our blog to see more information about new features, tips and tricks, and featured notebooks such as Analyzing a Bank Failure with Colab.

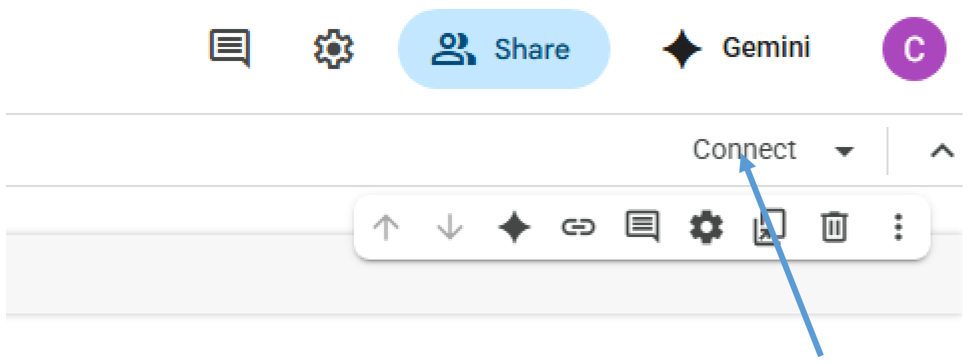
2025-07-22

- Terminal is now available to all users, free of charge! (X, Colab Notebook)
- The new Colab AI experience is now available to all users. This version has an agentic collaborator run with Gemini 2.5 Flash and enhances productivity with features like iterative querying, a next-gen data science agent, code transformation, and more to significantly improve coding workflows for developers. (Blog, X)
- Introducing our new google.colab.ai library! Pro & Pro+ subscribers can access powerful Gemini and Gemma models with just a few lines of code- no set up or API key required! Try it out! (X, Colab Notebook)
- Big news for data science in higher ed! 🚀 Colab now offers 1-year Pro subscriptions free of charge for verified US students/faculty, interactive Slideshow Mode for lectures, & an AI toggle per notebook. Enhance teaching & learning in the upcoming academic year! Read all about it here))
- Gemini CLI installed on runtimes. Try it out in the Terminal! (github, npm)
- Python package upgrades
  - accelerate 1.7.0 -> 1.9.0
  - bigframes 2.6.0 -> 2.11.0
  - bigquery-magics 0.9.0 -> 0.10.1
  - cudf-cu12 25.2.1 -> 25.6.0
  - cuml-cu12 25.2.1 -> 25.6.0
  - dask 2024.12.1 -> 25.5.0
  - dask-cuda 25.2.0 -> 25.6.0
  - dask-cudf 25.2.2 -> 25.6.0
  - dataproc-spark-connect 0.7.5 -> 0.8.3
  - debugpy 1.8.0 -> 1.8.15
  - diffusers 0.33.1 -> 0.34.0
  - distributed 2024.12.1 -> 25.5.0
  - duckdb 1.2.2 -> 1.3.2
  - earthengine-api 1.5.19 -> 1.5.24
  - gcsfs 2025.3.2 -> 25.7.0
  - geopandas 1.0.1 -> 1.1.1
  - google-genai 1.20.0 -> 1.26.0
  - gradio 5.31.0 -> 5.38.0
  - narwhals 1.42.0 -> 1.48.0
  - numba-cuda 0.2.0 -> 0.11.0
  - openai 1.86.0 -> 1.97.0
  - opencv-python 4.11.0.86 -> 4.12.0.88
  - pillow 11.2.1 -> 11.3.0
  - polars 1.21.0 -> 1.25.0
  - pymc 5.23.0 -> 5.24.1

GOOGLE COLAB

https://colab.research.google.com/notebooks/intro.ipynb#



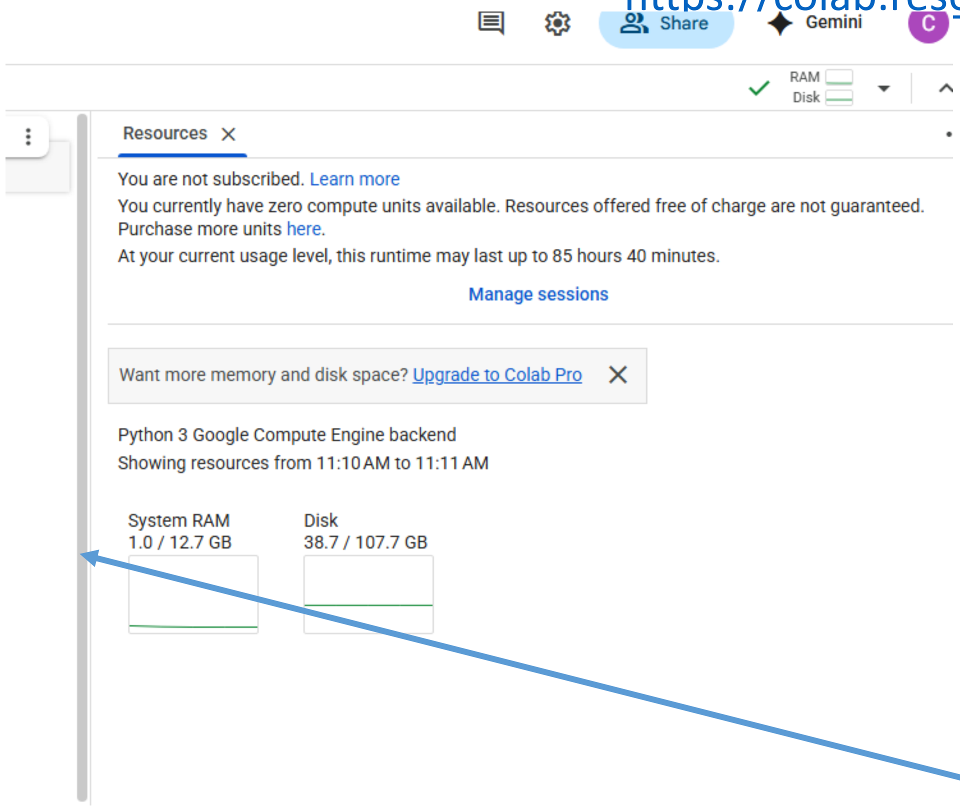


Click here to Connect

Click Folder to view your Folder

GOOGLE COLAB

https://colab.research.google.com/notebooks/intro.ipynb#



You are not subscribed. Learn more

You currently have zero compute units available. Resources offered free of charge are not guaranteed. Purchase more units here.

At your current usage level, this runtime may last up to 85 hours 40 minutes.

Manage sessions

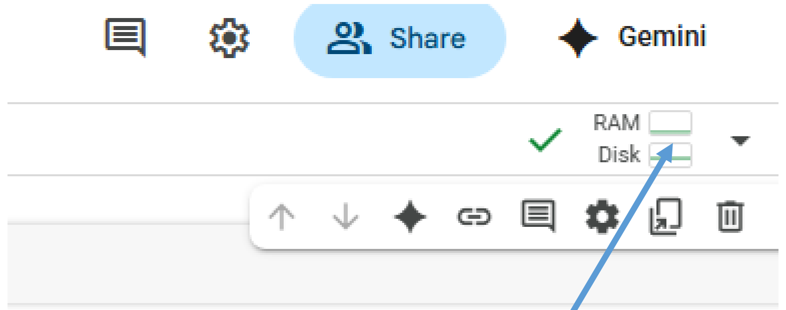
Want more memory and disk space? Upgrade to Colab Pro

Python 3 Google Compute Engine backend

Showing resources from 11:10 AM to 11:11 AM

System RAM 1.0 / 12.7 GB

Disk 38.7 / 107.7 GB

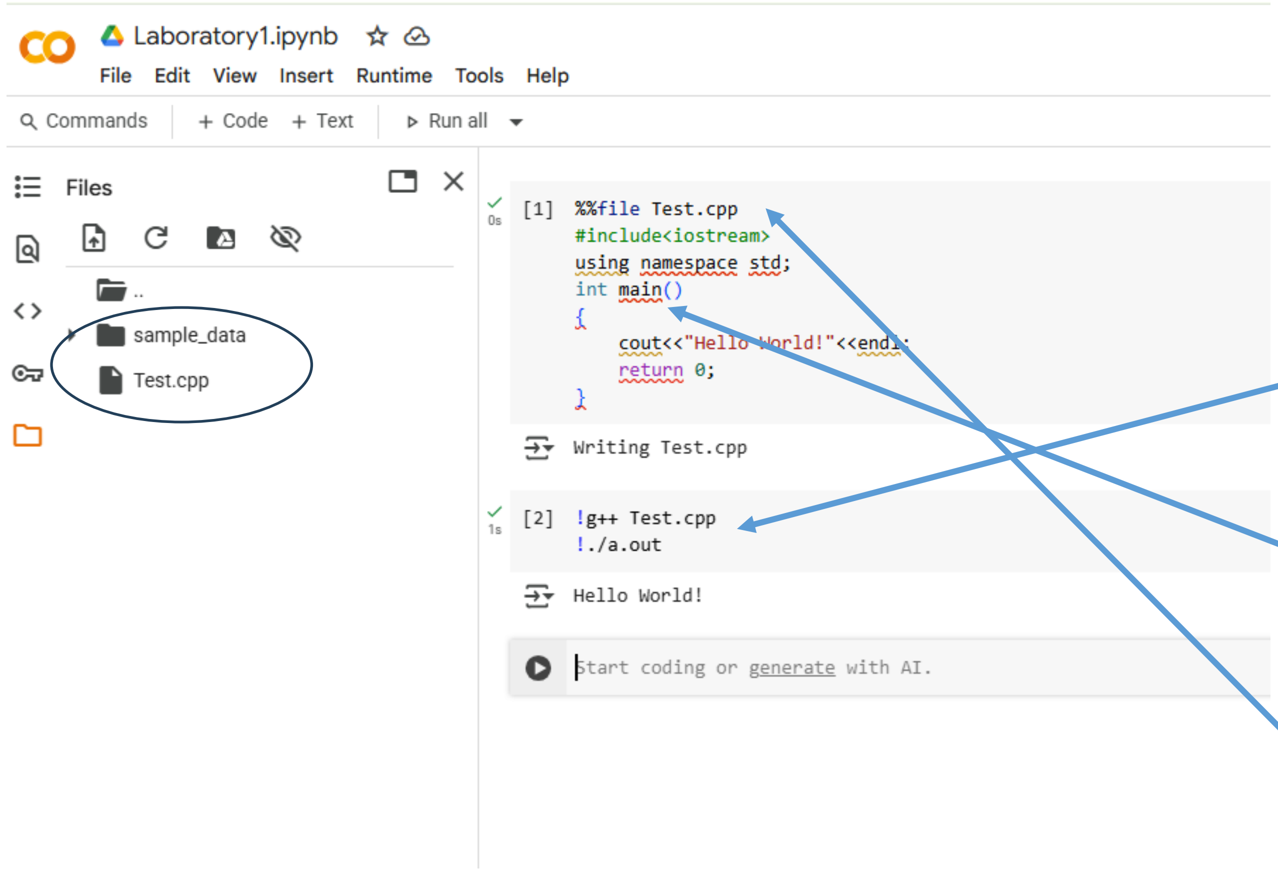


Connected

Resources Allotted

GOOGLE COLAB

https://colab.research.google.com/notebooks/intro.ipynb#



Laboratory1.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Files
..
sample_data
Test.cpp

```
[1] %%file Test.cpp
    #include<iostream>
    using namespace std;
    int main()
    {
        cout<<"Hello World!"<<endl;
        return 0;
    }

Writing Test.cpp

[2] !g++ Test.cpp
    !./a.out

Hello World!

Start coding or generate with AI.
```

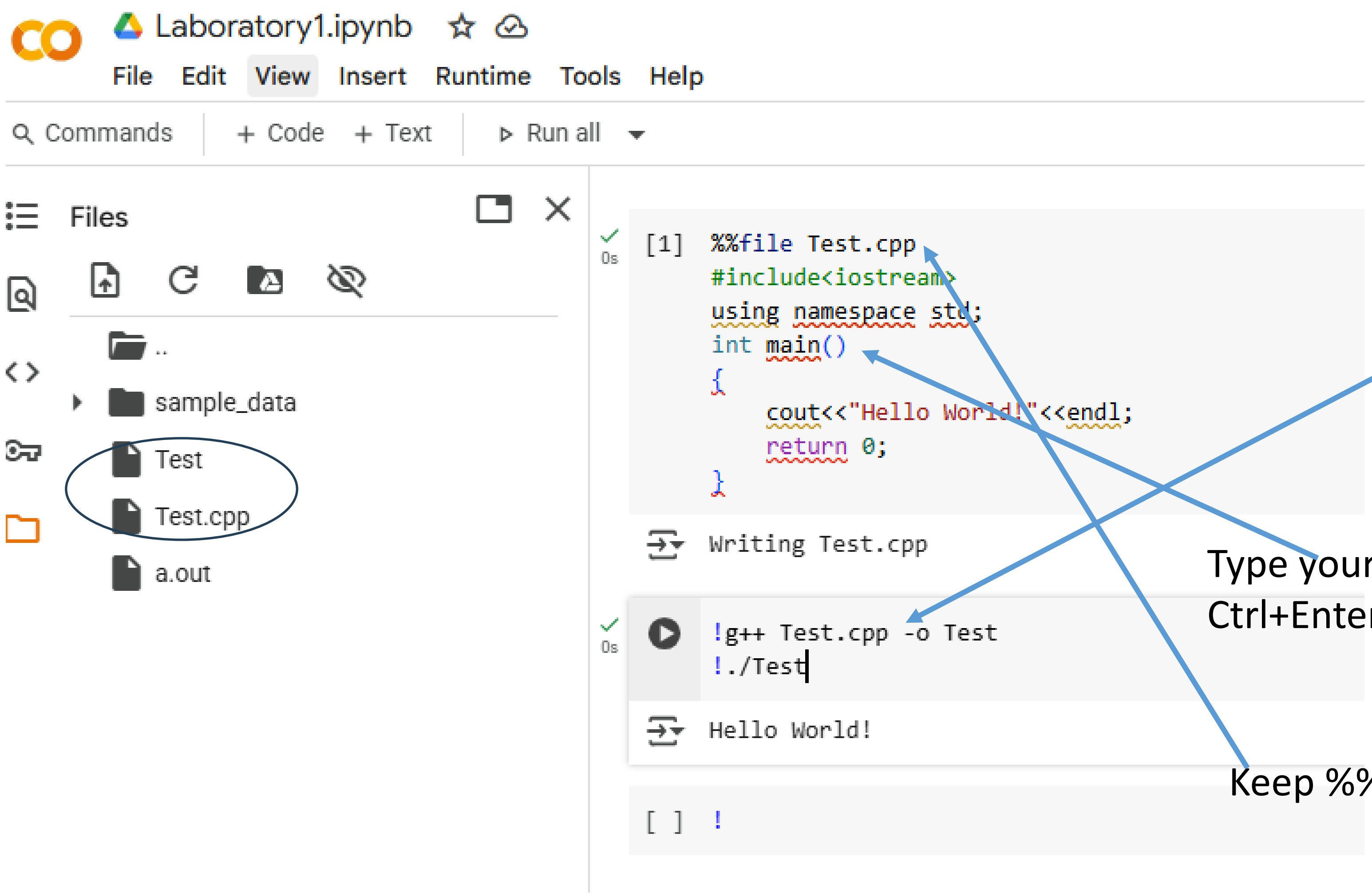
Run and Execute

Type your C++ Code and click the play button or Ctrl+Enter

Keep %%file FileName.cpp

GOOGLE COLAB

https://colab.research.google.com/notebooks/intro.ipynb#



```
[1] %%file Test.cpp
    #include<iostream>
    using namespace std;
    int main()
    {
        cout<<"Hello World!"<<endl;
        return 0;
    }
```

```
Writing Test.cpp
```

```
!g++ Test.cpp -o Test
!./Test
```

```
Hello World!
```

```
[ ] !
```

Run and Execute

Type your C++ Code and click the play button or Ctrl+Enter

Keep %%file FileName.cpp

SIMPLE PROGRAM ANATOMY

ANATOMY

```
#include<iostream>
using namespace std;
int main()
{
   return 0;
}
```

- A header file is required to create the program.
- It has plenty of libraries regarding input, output other
- basic operations.
- For, example, it contains `cin`, `cout`.
- Without the inclusion of this file, the program won't run.
- This file is included in the main part of the program during the pre-processing stage.
- The `iostream` library in C++ is a part of the C++ Standard Library that provides functionality for input and output operations

ANATOMY

```
#include<iostream>
using namespace std;
int main()
{
   return 0;
}
```

- It is used to import the entire std namespace into the current namespace of the program

ANATOMY

```
#include<iostream>
using namespace std;
int main()
{
   return 0;
}
```

- It is the place where the main function begins the program execution
- Every C++ program must have a `main` function, and the behavior of the program is defined by the code written inside it.
- `int` denotes that the function should return an `integer` value to the operating system to denote the exit status of this program.
- If the function returns 0 (`return 0` for this case), then it is an indication that there are no errors after execution of this program.
- If an error occurs, a non-zero value is returned, often 1 or another positive integer.

ANATOMY

```
#include<iostream>
using namespace std;
int main()
{
  return 0;
}
```

- The starting curly brace { of the program says that the body of the main function begins here.
- It denotes the scope of the function. The end curly brace denotes the end of the main function.

ANATOMY

```
#include<iostream> //Header File for input and output. Contains cin, cout. It is a preprocessing stage
using namespace std; //We are going to use the std namespace, you will know what a namespace is later
int main() //The program starts from here
{ //This curly brace is important. It says where the program starts, between {}, the body of the main
program works
//Program compilation will read after this curly brace, till the end of the brace
   return 0; // This statement is returning the value 0 to indicate to the operating system that this
application has reached the exit status
}
```

Remarks

It is a bad practice to use using namespace std when you write a professional program.

However, for this laboratory purpose of this lab and for our convenience to write the program quickly, we will use this in all our C++ programming.

COMMENT LINES

SINGLE LINE COMMENT //

```
#include<iostream> //Header File for input and output. Contains cin, cout. It is a preprocessing stage
using namespace std; //We are going to use the std namespace, you will know what a namespace is later
int main() //The program starts from here
{ //This curly brace is important. It says where the program starts, between {}, the body of the main
program works
//Program compilation will read after this curly brace, till the end of the brace
   return 0; // This statement is returning the value 0 to indicate to the operating system that this
application has reached the exit status
}
```

MUTLILINE COMMENT LINE /* */

```
/* First Program
A demonstration program
Created by Panchatcharam
Date: 11-08-2023 */

#include<iostream>
using namespace std;
int main()
{
    return 0;
}
```

Escape sequence

The backslash textbackslash is an escape character

The escape sequence (\n) means newline

The escape sequence (\t) means horizontal tab

The escape sequence (\\) means insert a backslash in string

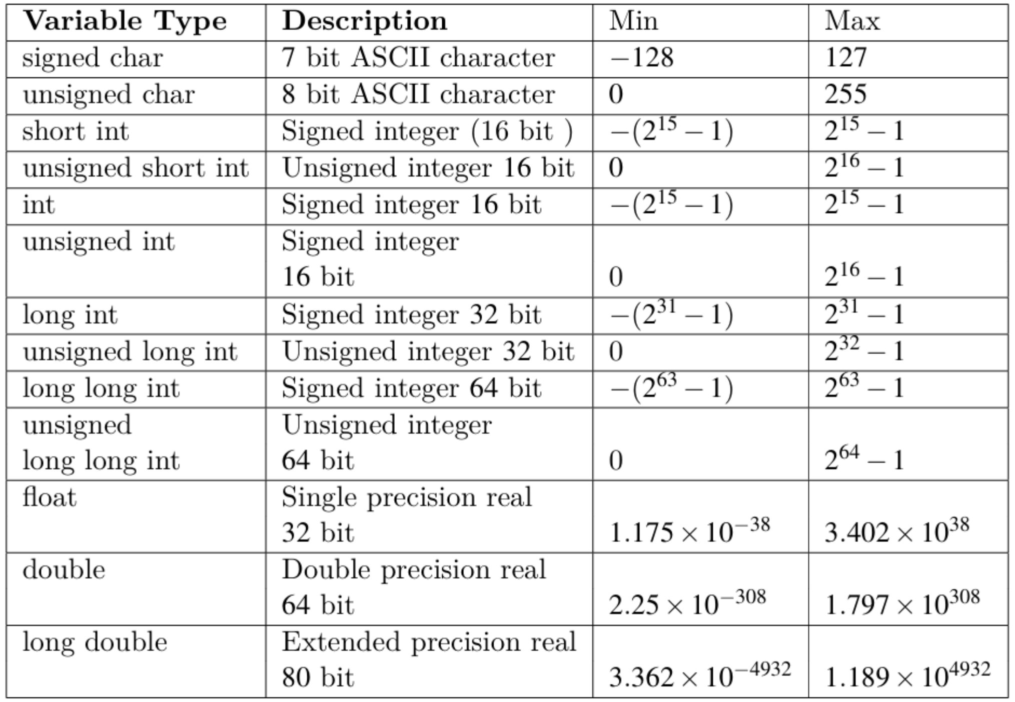
The escape sequence (\") means insert a double-quoter character in string

DATA TYPES

BASIC DATA TYPES

Variable Type	**Description**	Min	Max
signed char	7 bit ASCII character	-128	127
unsigned char	8 bit ASCII character	0	255
short int	Signed integer (16 bit)	$-(2^{15}-1)$	$2^{15}-1$
unsigned short int	Unsigned integer 16 bit	0	$2^{16}-1$
int	Signed integer 16 bit	$-(2^{15}-1)$	$2^{15}-1$
unsigned int	Signed integer 16 bit	0	$2^{16}-1$
long int	Signed integer 32 bit	$-(2^{31}-1)$	$2^{31}-1$
unsigned long int	Unsigned integer 32 bit	0	$2^{32}-1$
long long int	Signed integer 64 bit	$-(2^{63}-1)$	$2^{63}-1$
unsigned long long int	Unsigned integer 64 bit	0	$2^{64}-1$
float	Single precision real 32 bit	1.175×10^{-38}	3.402×10^{38}
double	Double precision real 64 bit	2.25×10^{-308}	1.797×10^{308}
long double	Extended precision real 80 bit	3.362×10^{-4932}	1.189×10^{4932}

BASIC DATA TYPES



Variable Type	Description	Min	Max
signed char	7 bit ASCII character	-128	127
unsigned char	8 bit ASCII character	0	255
short int	Signed integer (16 bit)	$-(2^{15}-1)$	$2^{15}-1$
unsigned short int	Unsigned integer 16 bit	0	$2^{16}-1$
int	Signed integer 16 bit	$-(2^{15}-1)$	$2^{15}-1$
unsigned int	Signed integer 16 bit	0	$2^{16}-1$
long int	Signed integer 32 bit	$-(2^{31}-1)$	$2^{31}-1$
unsigned long int	Unsigned integer 32 bit	0	$2^{32}-1$
long long int	Signed integer 64 bit	$-(2^{63}-1)$	$2^{63}-1$
unsigned long long int	Unsigned integer 64 bit	0	$2^{64}-1$
float	Single precision real 32 bit	1.175×10^{-38}	3.402×10^{38}
double	Double precision real 64 bit	2.25×10^{-308}	1.797×10^{308}
long double	Extended precision real 80 bit	3.362×10^{-4932}	1.189×10^{4932}

- Note that −127,−(215−1),−(231−1),−(263−1) are guaranteed in most of the C com pilers.
- However, some platform uses two's complement.
- Therefore, when you execute the next program in g++, you will get −128,−215,−231,−263.
- For more details, refer to the following links at page 22: ISO C and this wiki page.

open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf
C data types - Wikipedia

Basic Data Types

Variable Type	**Description**	Min	Max
signed char	7 bit ASCII character	-128	127
unsigned char	8 bit ASCII character	0	255
short int	Signed integer (16 bit)	$-(2^{15}-1)$	$2^{15}-1$
unsigned short int	Unsigned integer 16 bit	0	$2^{16}-1$
int	Signed integer 16 bit	$-(2^{15}-1)$	$2^{15}-1$
unsigned int	Signed integer 16 bit	0	$2^{16}-1$
long int	Signed integer 32 bit	$-(2^{31}-1)$	$2^{31}-1$
unsigned long int	Unsigned integer 32 bit	0	$2^{32}-1$
long long int	Signed integer 64 bit	$-(2^{63}-1)$	$2^{63}-1$
unsigned long long int	Unsigned integer 64 bit	0	$2^{64}-1$
float	Single precision real 32 bit	1.175×10^{-38}	3.402×10^{38}
double	Double precision real 64 bit	2.25×10^{-308}	1.797×10^{308}
long double	Extended precision real 80 bit	3.362×10^{-4932}	1.189×10^{4932}

- Used to store various types of data
- Must be defined before usage
- Must be declared
- May be initialized while declaring
- The syntax of declaring the variable is

```
const VariableType VariableName1[=Value1];
```

Variable Declaration

```
const VariableType VariableName1[=Value1];

VariableType VariableName1[=Value1]
[,VariableName2[=Value2],...VariableNameN[=ValueN]];
```

- ✓ The keyword const in front of the declaration makes it constant.
- ✓ It means, this variable can't be modified anywhere in the program after initialization.
- ✓ VariableType denotes the type of variable being declared.
- ✓ VariableNameJ denotes the Jth variable to declare.
- ✓ [ ] denotes optional.
- ✓ ValueJ denotes the values assigned to the Jth variable

FOUR BASIC DATA TYPES

char

- ✓ It is a keyword for character data types.
- ✓ 1 byte of memory space.
- ✓ Range: −128 to 127 or 0 to 255

int

- ✓ It is a keyword for integer data types.
- ✓ 4 bytes of memory space.

float

- ✓ It is a keyword to store single precision floating point or decimal value.
- ✓ Memory consumption: 4 bytes of memory

double

- ✓ It is a keyword to store double precision floating point or decimal value.
- ✓ Memory consumption: 4 bytes of memory

FOUR BASIC DATA TYPES

- ✓ Note: The memory consumption depends on the compilers.
- ✓ In order to find the correct memory consumption, use the following program and find its memory size.
- ✓ For more details, have a look at ISO C website

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Size of char: " << sizeof(char) << " byte" << endl;
    cout << "Size of int: " << sizeof(int) << " bytes" << endl;
    cout << "Size of float: " << sizeof(float) << " bytes" << endl;
    cout << "Size of double: " << sizeof(double) << " bytes" << endl;
    cout << "Size of long double: " << sizeof(long double) << " bytes" <<
    endl;
    cout<<"Size of wchar_t: " << sizeof(wchar_t) << "bytes" <<endl;
        return 0;
}
```

Four basic data types

- Use meaningful variable names to improve code readability.
- Initialize variables before using them to avoid undefined behavior.
- Be mindful of memory usage and the range of values that can be stored in each data type.
- Use appropriate data types based on the requirements of your pro gram to balance precision, memory usage, and performance

FOUR BASIC DATA TYPES

```
#include <iostream>
#include <climits>

using namespace std;
int main(void)
{
    cout<<"CHAR_MIN\t\t\t= "<< CHAR_MIN<<endl;
    cout<<"CHAR_MAX    = "<< CHAR_MAX<<endl;
    cout<<"Signed CHAR_MIN   = "<< SCHAR_MIN<<endl;
    cout<<"Signed CHAR_MAX   = "<< SCHAR_MAX<<endl;
    cout<<"Unsigned CHAR_MAX   = "<<  UCHAR_MAX<<endl;
    cout<<endl;

    cout<<"Short INT_MIN    = "<< SHRT_MIN<<endl;
    cout<<"Short INT_MAX    = "<< SHRT_MAX<<endl;
    cout<<"Unsigned Short INT_MAX   = "<<  USHRT_MAX<<endl;
    cout<<endl;

    cout<<"INT_MIN     = "<< INT_MIN<<endl;
    cout<<"INT_MAX     = "<< INT_MAX<<endl;
    cout<<"Unsigned INT_MAX   = "<<  UINT_MAX<<endl;
    cout<<endl;

    cout<<"LONG INT_MIN   = "<< LONG_MIN<<endl;
    cout<<"LONG INT_MAX   = "<< LONG_MAX<<endl;
    cout<<"Unsigned LONG INT_MAX  = "<<  ULONG_MAX<<endl;
    cout<<endl;

    cout<<"LONG LONG INT_MIN   = "<< LLONG_MIN<<endl;
    cout<<"LONG LONG INT_MAX   = "<< LLONG_MAX<<endl;
    cout<<"Unsigned LONG LONG INT_MAX = "<<  ULLONG_MAX<<endl;
    cout<<endl;

    cout<<"FLOAT_MIN      ="<< FLT_MIN<<endl;
    cout<<"FLOAT_MAX      ="<< FLT_MAX<<endl;
    cout<<endl;

    cout<<"DOUBLE_MIN      = "<< DBL_MIN<<endl;
    cout<<"DOUBLE_MAX      = "<< DBL_MAX<<endl;
    cout<<endl;

    cout<<"LONG DOUBLE_MIN      = "<< LDBL_MIN<<endl;
    cout<<"LONG DOUBLE_MAX      = "<< LDBL_MAX<<endl;
    cout<<endl;
    return 0;
}
```

FOUR BASIC DATA TYPES

3.4.3 Float

- 1 bit - **bin**ary digi**t**
- 8 bits = 1 byte
- float has 4 bytes = 32 bits
- Single Precision, bindary32, decimal32
- The storage format of the float is

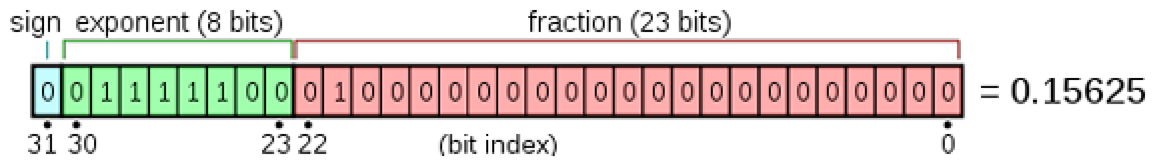
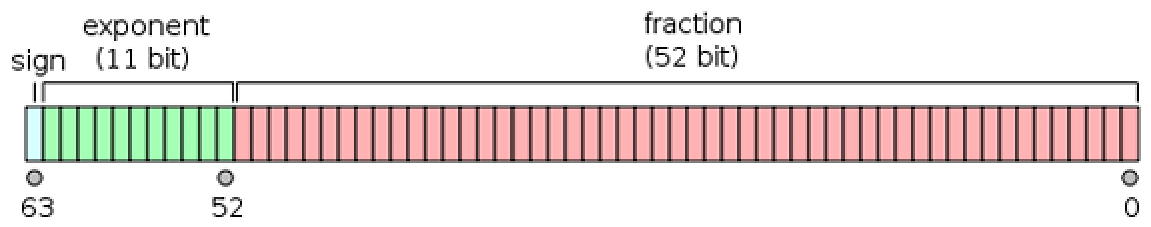


Figure 3.2: (Image Source:Wikpedia)

3.4.4 Double

- double has 8 bytes = 64 bits
- Double Precision, bindary64, decimal64
- You can use suffixes to explicitly indicate the type of a floating-point literal.
- For double, you can use d or D (e.g., 3.14d or 1.23D).
- The storage format of the double is



OUTPUT REDIRECTION

OUTPUT ON SCREEN

Output in C++ can be done in multiple ways such using `cout`, output to a file using `fstream`. We can also output to a file using the terminal command

```
cout << VariableName;
```

cout

- cout is defined in header file
- iostream- input output stream
- It is an object in ostream class
- Displays the output to the standard output device, that is monitor
- Associated with stdout stream
- c-refers character, out refers output
- cout- character output
- This object works with the insertion operator (<<) to display stream of characters

- cout << VariableName;

COUT

```
cout << VariableName;
```

- `cout` is defined in header file
- `iostream`– input output stream
- It is an object in ostream class
- Displays the output to the standard output device, that is monitor
- Associated with stdout stream
- c-refers character, out refers output
- cout- character output
- This object works with the insertion operator (<<) to display stream of characters

```
#include <iostream>
using namespace std;
int main() {
    int a = 5;
    float b = 7.5;
    double c = 8.9;
    char d = 'a';
    cout << a << "\t" << b <<"\t" << c <<"\t" << d <<endl;
    cout << a <<endl;
    cout << b <<endl;
    cout << c <<endl;
    cout << d <<endl;
    return 0;
}
```

OUTPUT TO A FILE

```
outfile << VariableName;
```

In order to send the output to a file, we use `fstream` which stands for file stream

- `ofstream`– output file stream, used to create files, write information to files
- `ifstream`– input file stream, used to read information from files
- `fstream`– file stream, includes the features of both `ofstream` and `ifstream`, it can create files, read from file, write to files.

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
    int a = 5, b = 7;
    ofstream myfile;
    myfile.open("Calculator.txt");
    myfile<<a<<" + "<<b<<" = "<<a+b<<endl;
    myfile<<a<<" - "<<b<<" = "<<a-b<<endl;
    myfile<<a<<" * "<<b<<" = "<<a*b<<endl;
    myfile<<a<<" / "<<b<<" = "<<a/b<<endl;
    myfile<<a<<" % "<<b<<" = "<<a%b<<endl;
    myfile.close();
    return 0;
}
```

FORMATTED OUTPUT

FORMATTED OUTPUT

`setw()` and `setfill()`

- If you would like to get a formatted output with alignments, you can use `setw()` and `setfill()` function with `cout`.
- The `setw()` function sets the width of the next output field.
- `setfill()`- A C stream function to fill character
- These functions are available under `iomanip` header files

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    cout<<setfill('0')<<setw(2)<<2<<" x "
<<setfill('0')<<setw(2)<<3<<" = 
"<<setfill('0')<<setw(2)<<6<<endl;
    cout<<setfill('y')<<setw(10)<<456<<endl;
    return 0;
}
```

FORMATTED OUTPUT

```
g++ -std=c++20 Program.cpp
```

```cpp
#include <iostream>
#include <format>

int main() {
    double pi = 3.14159265;
    std::cout << std::format("Pi rounded to 3 decimal places: {:.3f}\n", pi);
    int x = 42, y = 1000;
    std::cout << std::format("x = {:06}, y = {:>10}\n", x, y);
    return 0;
}
```

INPUT

INPUT FROM SCREEN

- ✓ `cin` is defined in header file
- ✓ It is an object in `iostream` class
- ✓ Reads the input from keyboard
- ✓ Associated with `stdin` stream
- ✓ c-refers character, in refers input
- ✓ `cin`– character input
- ✓ Thisobject works with the extraction operator (>>) to receive stream of characters. The general syntax is

```
cin >> VariableName;
```

INPUT FROM SCREEN

- ✓ `cin` is defined in header file
- ✓ It is an object in `iostream` class
- ✓ Reads the input from keyboard
- ✓ Associated with `stdin` stream
- ✓ c-refers character, in refers input
- ✓ `cin`– character input
- ✓ Thisobject works with the extraction operator (>>) to receive stream of characters.

The general syntax is

```
cin >> VariableName;
```

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
    int m,g,h,PE;
    cin>>m>>g>>h;
    PE=m*g*h;
    cout<<PE<<endl;
    return 0;
}
```

INPUT FROM A FILE

```
infile >> VariableName;
```

`fstream` library provides classes for handling I/O operations.
It has classes like `ifstream`, `ofstream`
`ifstream` class is used for reading data from files
`ofstream` class is used for writing data to files

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
    int a,b,c,disc;
    ifstream myfile;
    myfile.open("TutorialInput.txt");
    myfile>>a>>b>>c;
    disc=b*b-4*a*c;
    myfile.close();
    cout<<disc<<endl;
    return 0;
}
```

Variable

WHAT IS A VARIABLE?

A variable is a named storage location. It stores a value of a particular data type.

- Programs process data
- A variable stores a piece of data for processing
- Variable because it can change the value stored
- It is used to store and manipulate data within a program
- Before you can use a variable, you need to declare it
- Initializing a variable assigns an initial value to it at the time of declaration

```
dataType VariableName;
```

```
int m;
float g;
double h;
char name;
```

Variable Example

```
#include<iostream>
using namespace std;
int main()
{
    int a=4,b=5,c=1,disc;
    double root1,root2;
    disc=b*b-4*a*c;
    root1=(-b+sqrt(disc))/(2*a);
    root2=(-b-sqrt(disc))/(2*a);
    cout<<"Root 1="<<root1<<endl;
    cout<<"Root 2="<<root2<<endl;
    return 0;
}
```

OPERATORS

Arithmetic operators

An arithmetic operator is a symbol that performs a simple mathematical operation.

These operators allow you to perform addition, subtraction, multiplication, division, and more.

Operators are operated in the following precedence.

If more than one operator has the same precedence, then they are evaluated from left to right.

Be aware of the precedence of arithmetic operators

ARITHMETIC OPERATORS

Operator	Description	Example	Precedence	Associativity
$()$	Parentheses	$(a+b)*c$	First	Left to Right
$*$	Multiplies	$a*b$	Third	Left to Right
$/$	Division	a/b	Third	Left to Right
$\%$	Remainder after Division Modulo Division	$a\%b$ $5\%3 = 2$	Third	Left to Right
$+$	Addition	$a+b$	Fourth	Left to Right
$-$	Subtraction	$a-b$	Fourth	Left to Right
$=$	Assignment operator	$a=b$	14th	Right to Left
$++$	Increment by 1	$x++$	First	Left to Right
$--$	Decrement by 1	$x--$	First	Left to Right
$++$	Increment by 1	$++x$	Second	Right to Left
$--$	Decrement by 1	$--x$	Second	Right to Left

Arithmetic Operators

```
#include<iostream>
using namespace std;
//This program shows different arithmetic operations for integer variables including increment and decrement operators
int main()
{
    int a=5,b=6,sum,diff,mul,div,moddiv,postinc,postdec,preinc,predec;
    sum=a+b;
    diff=a-b;
    mul=a*b;
    div=a/b;
    moddiv=b/a;
    cout<<a<<" + "<<b<<" = "<<sum<<endl;
    cout<<a<<" - "<<b<<" = "<<diff<<endl;
    cout<<a<<" * "<<b<<" = "<<mul<<endl;
    cout<<a<<" / "<<b<<" = "<<div<<endl;
    cout<<a<<" % "<<b<<" = "<<moddiv<<endl;
    postinc=a++; //a = 5, a++ = 6, but postinc = 5
    cout<<"postinc = "<<postinc<<endl;
    cout<<"a = "<<a<<endl;
    preinc=++b;
    cout<<"preinc = "<<preinc<<endl;
    cout<<"b = "<<b<<endl;
    postdec=a--;
    cout<<"postdec = "<<postdec<<endl;
    cout<<"a = "<<a<<endl;
    predec=--b;
    cout<<"predec = "<<predec<<endl;
    cout<<"b = "<<b<<endl;
    return 0;
}
```

ARITHMETIC OPERATORS

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
    float a=5.5,b=1.5,sum,diff,mul,div,moddiv;
    sum=a+b;
    diff=a-b;
    mul=a*b;
    div=a/b;
    moddiv=fmod(b,a);
    cout<<a<<" + "<<b<<" = "<<sum<<endl;
    cout<<a<<" - "<<b<<" = "<<diff<<endl;
    cout<<a<<" * "<<b<<" = "<<mul<<endl;
    cout<<a<<" / "<<b<<" = "<<div<<endl;
    cout<<a<<" % "<<b<<" = "<<moddiv<<endl;
    return 0;
}
```

Arithmetic operators

When using arithmetic operators with different data types, C++ performs implicit type conversions based on type promotion rules.

Be aware of potential loss of precision or unexpected results when mixing different data types.

Is it possible to use increment and decrement operators for float variable?

Arithmetic operators

Divide by zero is undefined and causes fatal error and makes the program to terminate.

1. Choose a meaningful variable and appropriate variable type
2. Leave a space after comma(,) to make programs readable
3. Do proper indentation (at least 2 spaces and at most 4 spaces) between {} and contents
4. Place spaces between variable and operators
5. Use .0 to distinguish float or double from integers.
6. Example: float a=2.0; int b=2