# Python Lab Class-01

## Review on Git and GitHub

### What is Git

- Git is a popular version control system.

### It is used for

- Tracking code changes
- Tracking who made changes
- Coding collaboration

### What does Git do?

- Manage projects with **Repositories**
- **Clone** a project to work on a local copy
- Control and track changes with **Staging** and **Committing**
- **Branch** and **Merge** to allow for work on different parts and versions
- **Pull** the latest version of the project to a local copy
- **Push** local updates to the main project

### Working with Git

- Initialize Git on a folder, making it a **Repository**
- Git now creates a hidden folder to keep track of changes in that folder
- When a file is changed, added or deleted, it is considered **modified**
- You select the modified files you want to **Stage**
- The **Staged** files are **Committed**, which prompts Git to store a **permanent** snapshot of the files

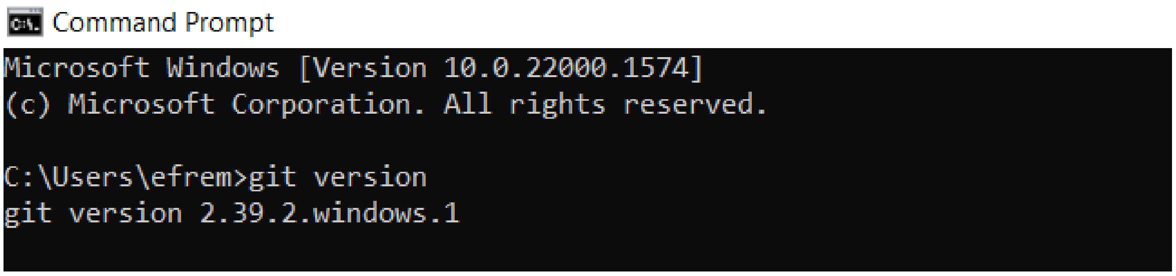
- Git allows you to see the full history of every commit.
- You can revert back to any previous commit.
- Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commit!

## Download and Install Git

- use this  website to download git : https://www.git-scm.com/

to check if Git is properly installed use the below command:

```
git --version
```

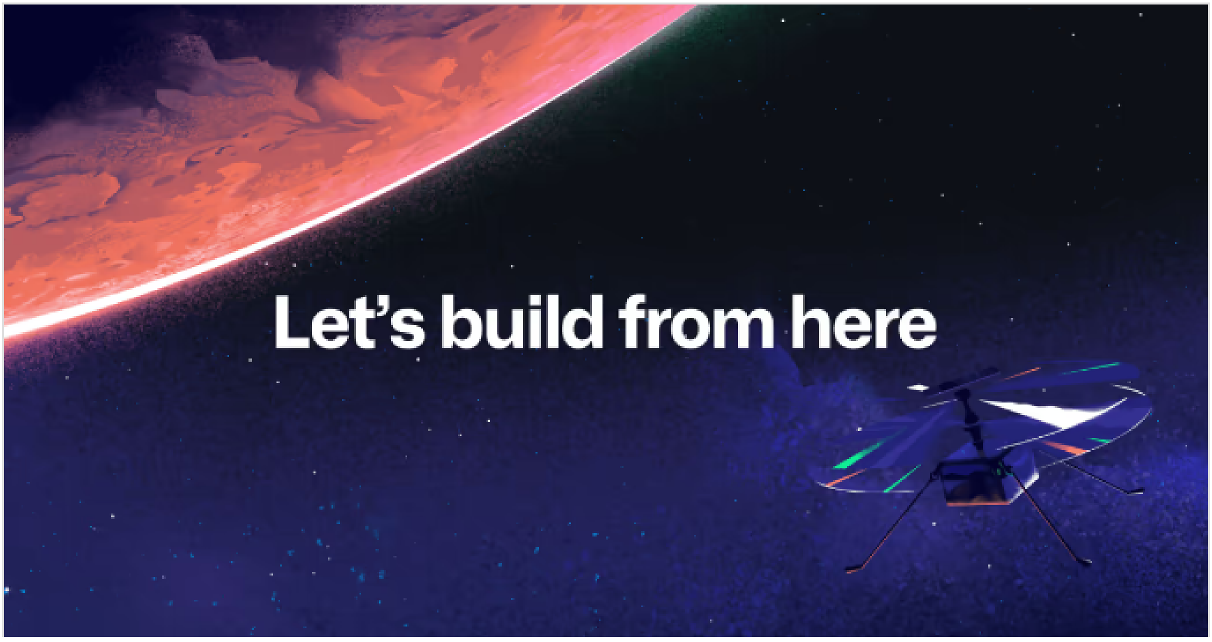


Command Prompt

```
Microsoft Windows [Version 10.0.22000.1574]
(c) Microsoft Corporation. All rights reserved.

C:\Users\efrem>git version
git version 2.39.2.windows.1
```

### Create GitHub Account

- GitHub makes tools that use Git.
- To use git first we should have to Create GitHub Account.



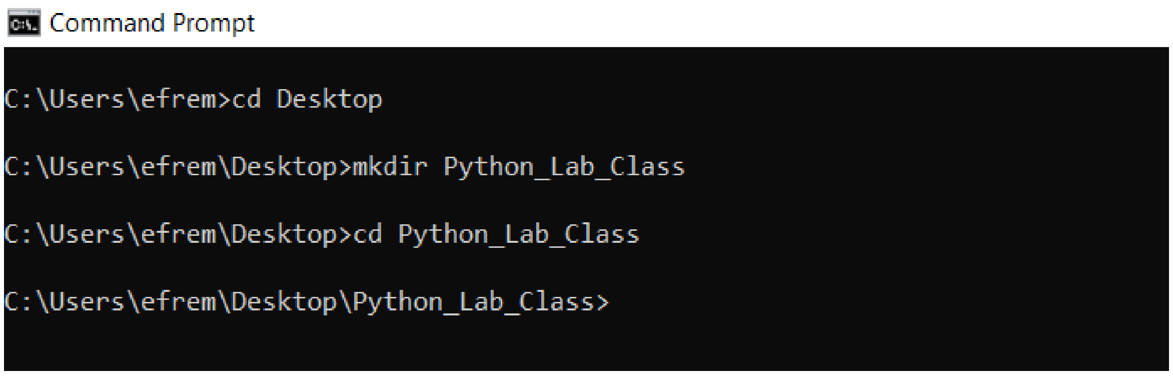
## Start work on git

- Configure Git
  - **Note:** Remember use the same username and e-mail address as GitHub account in the Git config.

```
git config --global user.name " "
```

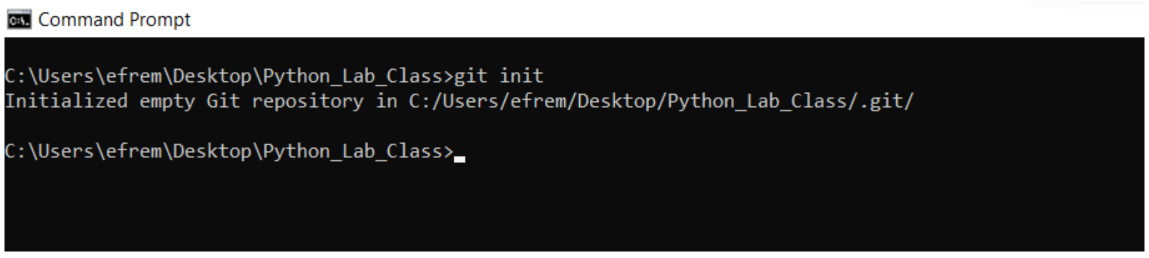
```
git config --global user.email " "
```

- Create project folder
  - mkdir Python_Lab_Class
  - cd Python_Lab_Class



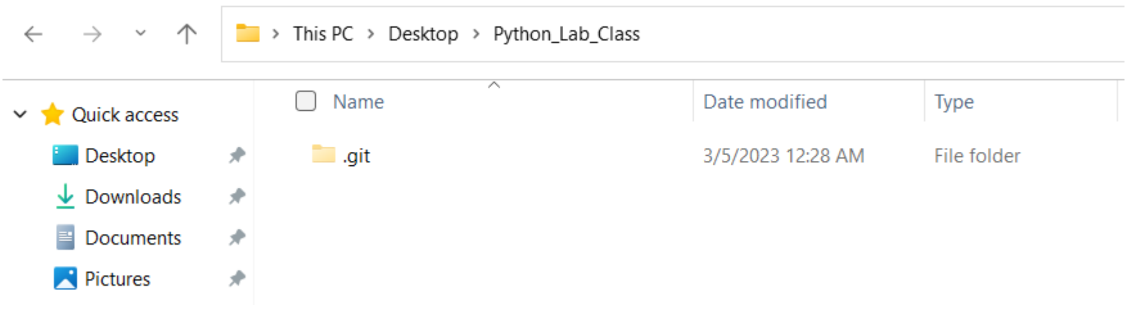
```
C:\Users\efrem>cd Desktop

C:\Users\efrem\Desktop>mkdir Python_Lab_Class

C:\Users\efrem\Desktop>cd Python_Lab_Class

C:\Users\efrem\Desktop\Python_Lab_Class>
```

- Initialize Git
  - git init



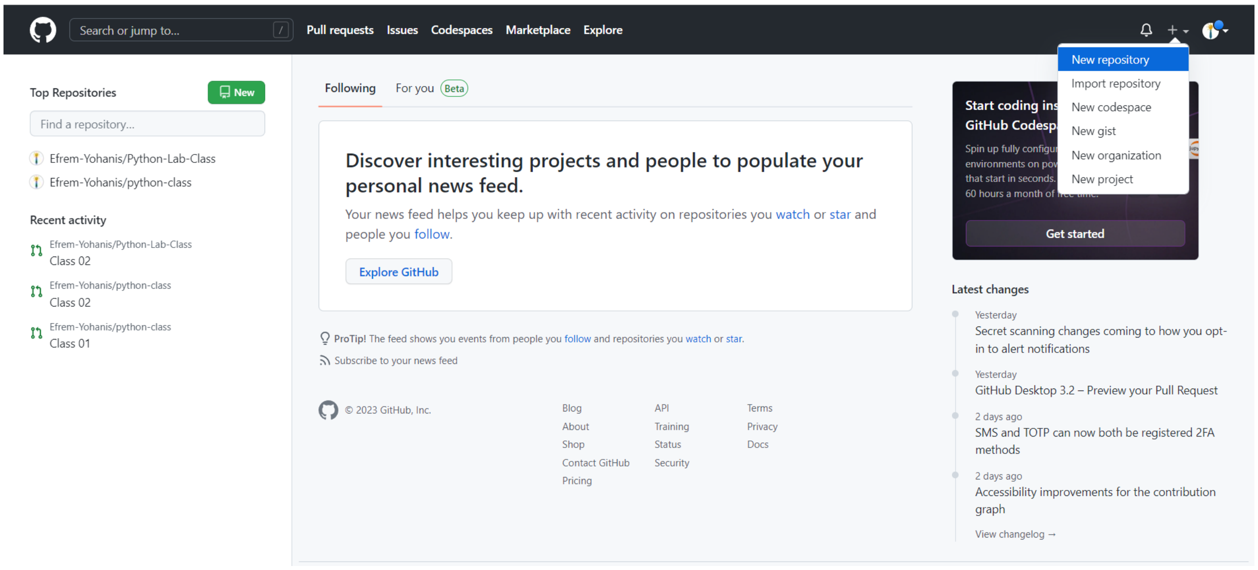
```
C:\Users\efrem\Desktop\Python_Lab_Class>git init
Initialized empty Git repository in C:/Users/efrem/Desktop/Python_Lab_Class/.git/

C:\Users\efrem\Desktop\Python_Lab_Class>_
```

Git creates a hidden folder to keep track of changes.



| Name | Date modified | Type |
|---|---|---|
| .git | 3/5/2023 12:28 AM | File folder |

- **Create a Repository on GitHub**



Top Repositories

Efrem-Yohanis/Python-Lab-Class
Efrem-Yohanis/python-class

Recent activity

Efrem-Yohanis/Python-Lab-Class
Class 02
Efrem-Yohanis/python-class
Class 02
Efrem-Yohanis/python-class
Class 01

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you watch or star and people you follow.

ProTip! The feed shows you events from people you follow and repositories you watch or star.

Subscribe to your news feed

© 2023 GitHub, Inc.

Start coding ins... GitHub Codesp...

Spin up fully configu... environments on pow... that start in seconds. 60 hours a month of free time.

Latest changes

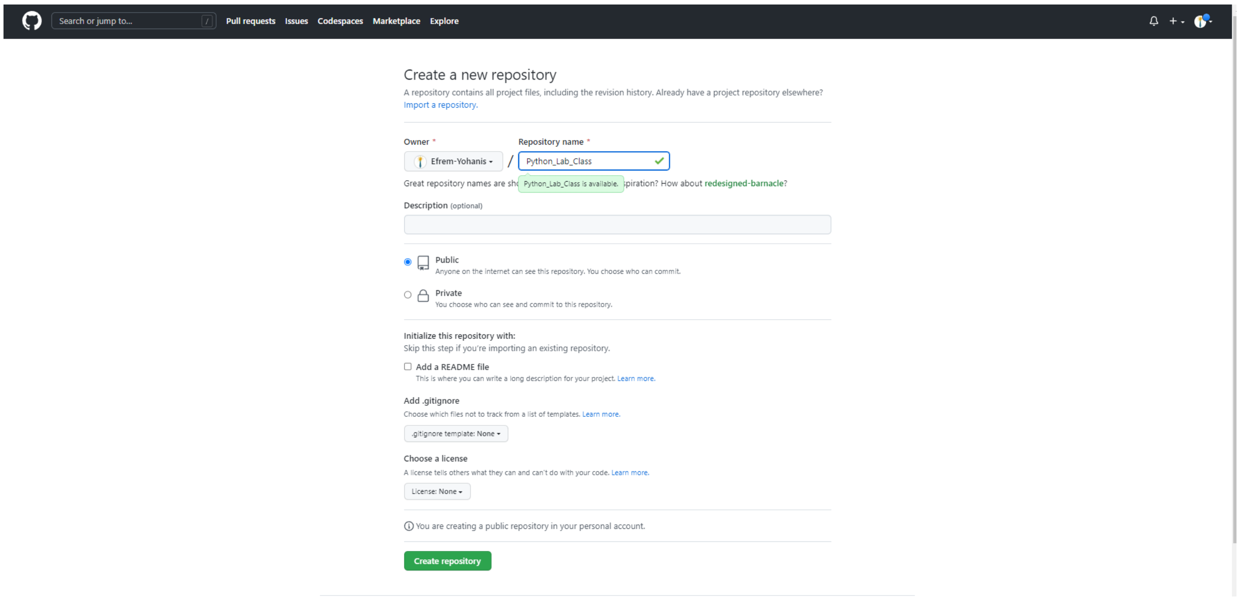
Yesterday
Secret scanning changes coming to how you opt-in to alert notifications

Yesterday
GitHub Desktop 3.2 – Preview your Pull Request

2 days ago
SMS and TOTP can now both be registered 2FA methods

2 days ago
Accessibility improvements for the contribution graph

View changelog →



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner *: Efrem-Yohanis / Repository name *: Python_Lab_Class

Great repository names are sh... Python_Lab_Class is available. ...spiration? How about redesigned-barnacle?

Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. Learn more.

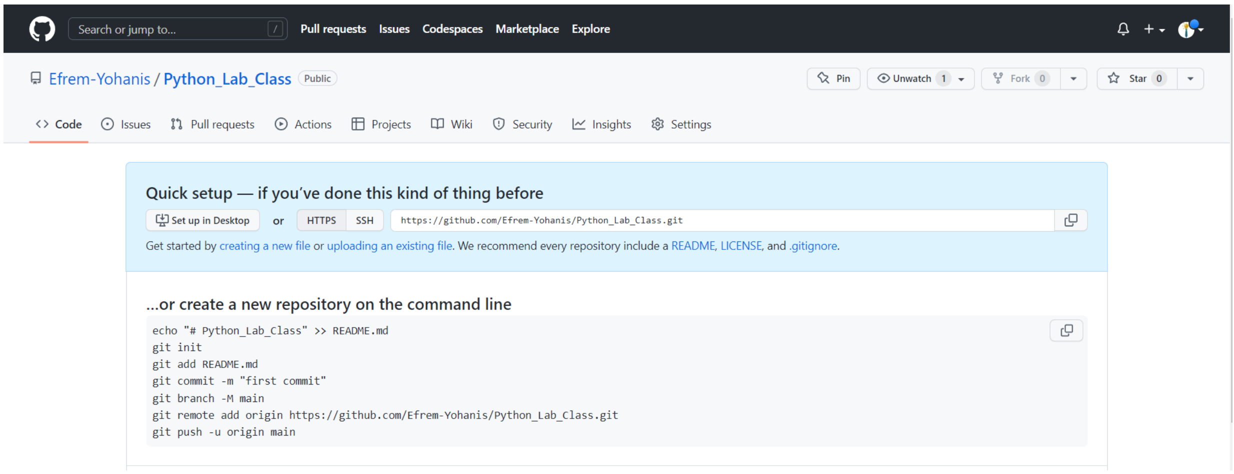
Add .gitignore
Choose which files not to track from a list of templates. Learn more.

.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.

License: None

You are creating a public repository in your personal account.



Efrem-Yohanis / Python_Lab_Class Public

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/Efrem-Yohanis/Python_Lab_Class.git

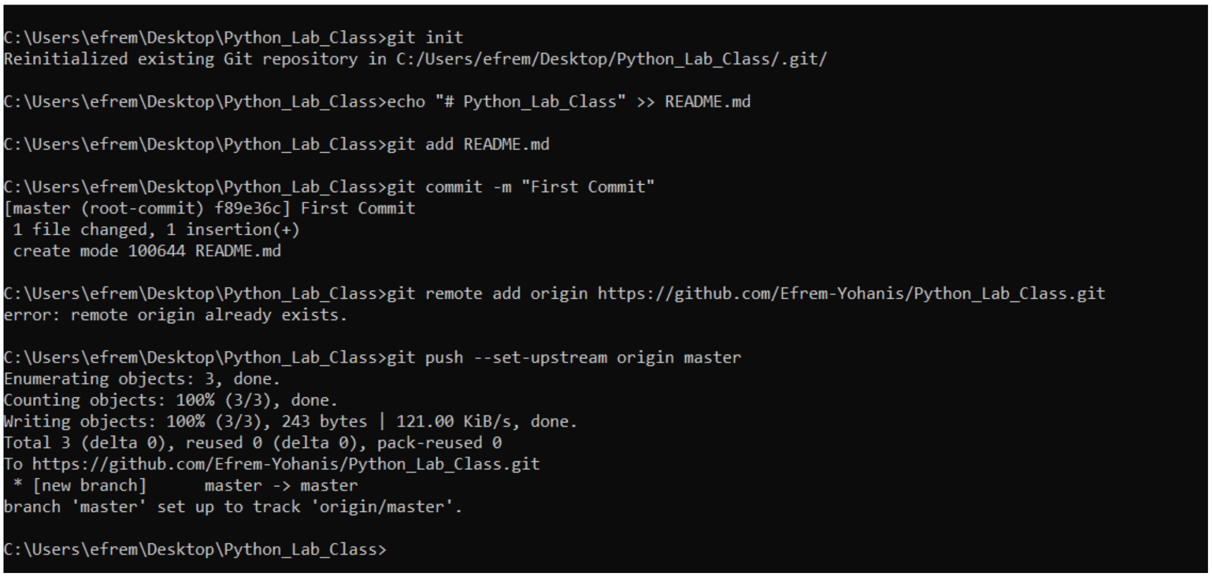
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

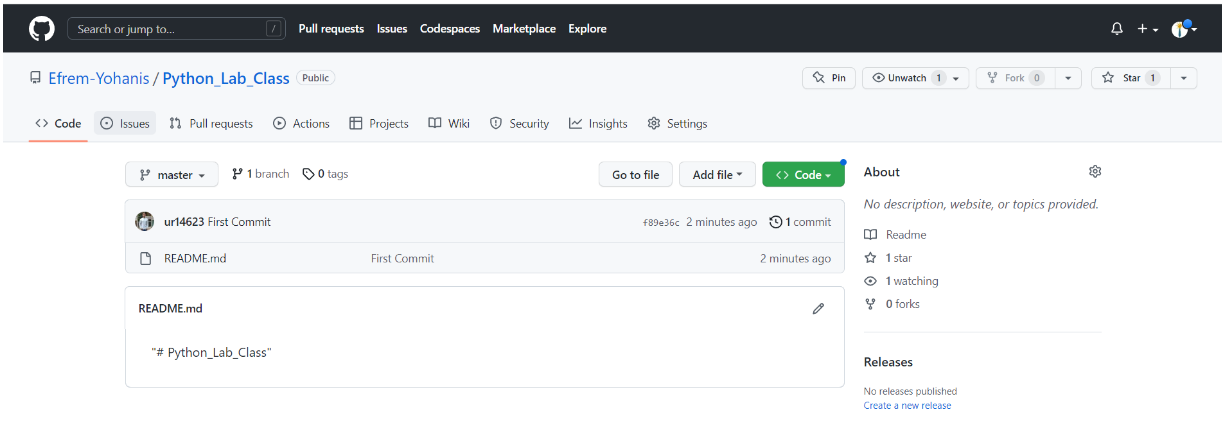
```
echo "# Python_Lab_Class" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Efrem-Yohanis/Python_Lab_Class.git
git push -u origin main
```

## To Push Local Repository to GitHub Run Following command

- echo "# Python_Lab_Class" >> README.md
- git add README.md
- git commit -m "first commit"
- git branch -M master
- git remote add origin https://github.com/Efrem-Yohanis/Python_Lab_Class.git
- git push --set-upstream origin master



```
C:\Users\efrem\Desktop\Python_Lab_Class>git init
Reinitialized existing Git repository in C:/Users/efrem/Desktop/Python_Lab_Class/.git/

C:\Users\efrem\Desktop\Python_Lab_Class>echo "# Python_Lab_Class" >> README.md

C:\Users\efrem\Desktop\Python_Lab_Class>git add README.md

C:\Users\efrem\Desktop\Python_Lab_Class>git commit -m "First Commit"
[master (root-commit) f89e36c] First Commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

C:\Users\efrem\Desktop\Python_Lab_Class>git remote add origin https://github.com/Efrem-Yohanis/Python_Lab_Class.git
error: remote origin already exists.

C:\Users\efrem\Desktop\Python_Lab_Class>git push --set-upstream origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 243 bytes | 121.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Efrem-Yohanis/Python_Lab_Class.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

C:\Users\efrem\Desktop\Python_Lab_Class>
```



Efrem-Yohanis / Python_Lab_Class Public

ur14623 First Commit   f89e36c 2 minutes ago   1 commit

README.md   First Commit   2 minutes ago

README.md

"# Python_Lab_Class"

About

No description, website, or topics provided.

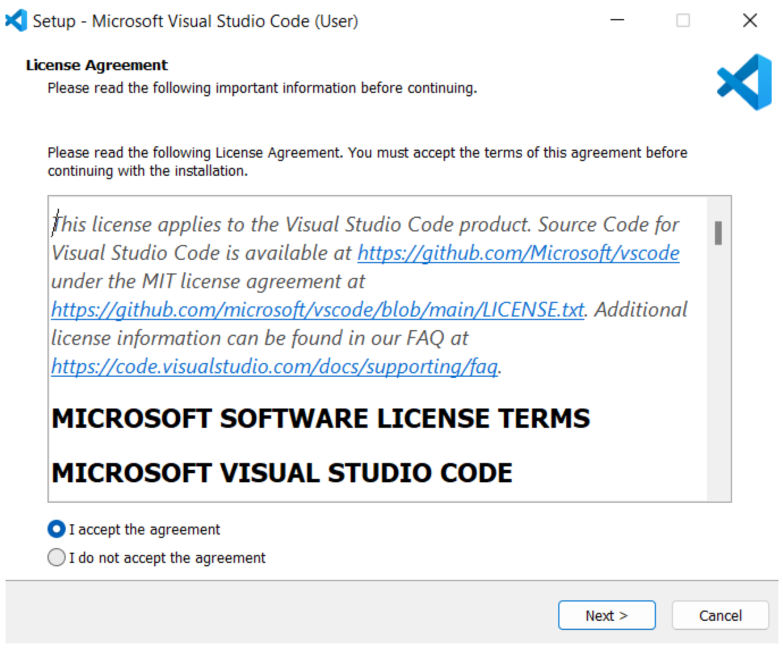
Readme
1 star
1 watching
0 forks

Releases

No releases published
Create a new release

## Install  Code Editor (Vscode)

- Use this link to download vscode https://code.visualstudio.com/docs/?dv=win
- Installation step are:



**License Agreement**

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

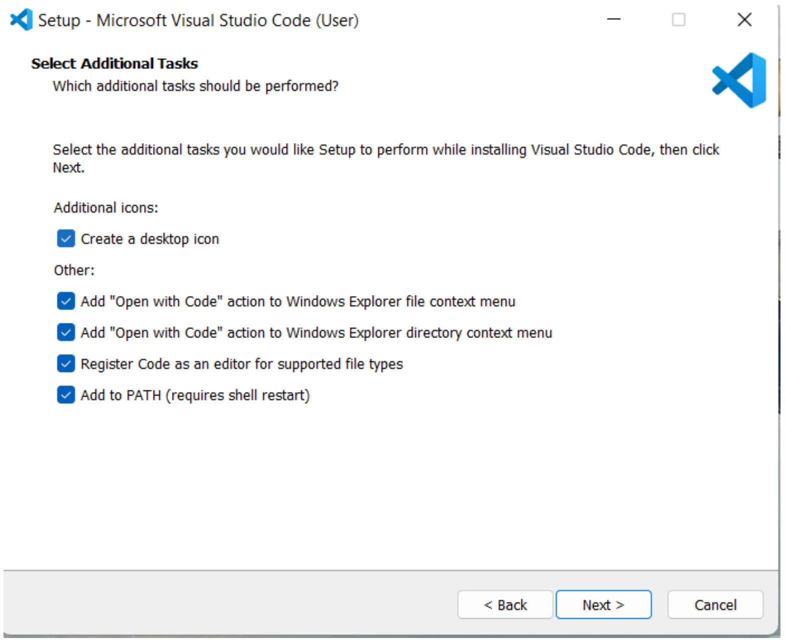
*This license applies to the Visual Studio Code product. Source Code for Visual Studio Code is available at https://github.com/Microsoft/vscode under the MIT license agreement at https://github.com/microsoft/vscode/blob/main/LICENSE.txt. Additional license information can be found in our FAQ at https://code.visualstudio.com/docs/supporting/faq.*

**MICROSOFT SOFTWARE LICENSE TERMS**

**MICROSOFT VISUAL STUDIO CODE**

[✓ I accept the agreement]
I do not accept the agreement

Next



**Select Additional Tasks**

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.

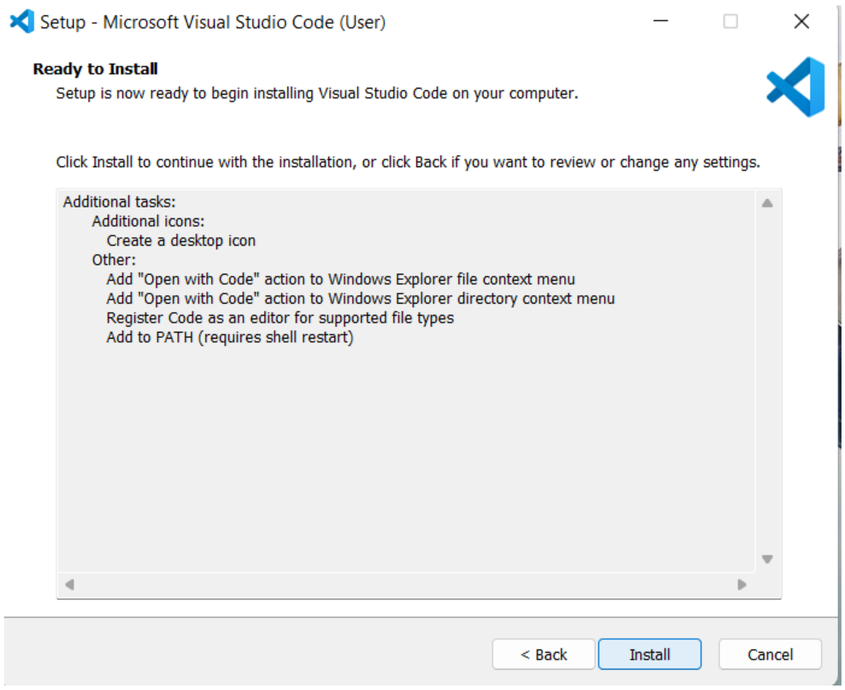
Additional icons:

- [✓] Create a desktop icon

Other:

- [✓] 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
- [✓] Add to PATH (requires shell restart)

Next


Setup - Microsoft Visual Studio Code (User)

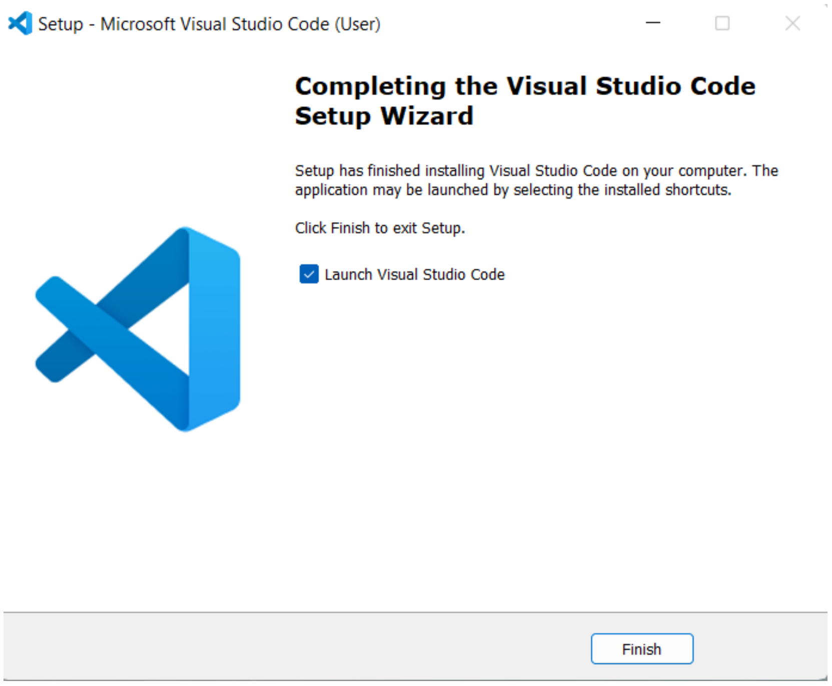
**Ready to Install**

Setup is now ready to begin installing Visual Studio Code on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Additional tasks:
    Additional icons:
      Create a desktop icon
    Other:
      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
      Add to PATH (requires shell restart)
```

Install


Setup - Microsoft Visual Studio Code (User)

**Completing the Visual Studio Code Setup Wizard**

Setup has finished installing Visual Studio Code on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

☑ Launch Visual Studio Code

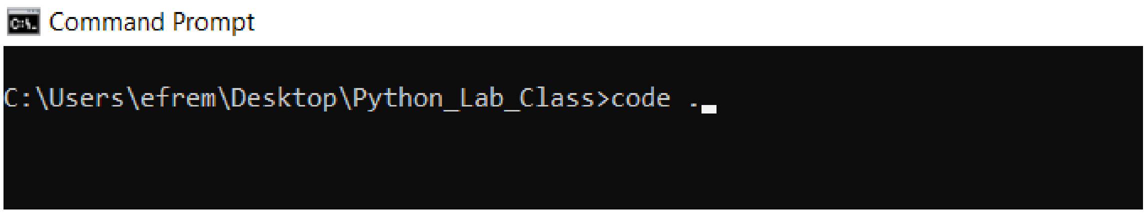
Finish

**We can now open our project file in Vscode after fully installing it.**

Two method to open project file in Vscode.

1.CMD

- Navigate in to project folder and type **code .**

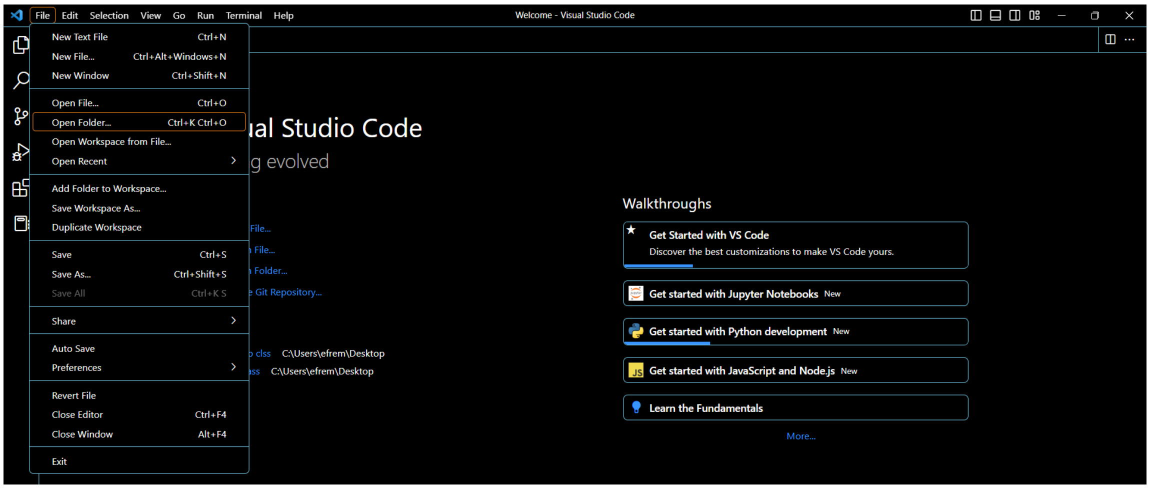


Command Prompt

```
C:\Users\efrem\Desktop\Python_Lab_Class>code .
```

2. Vscode

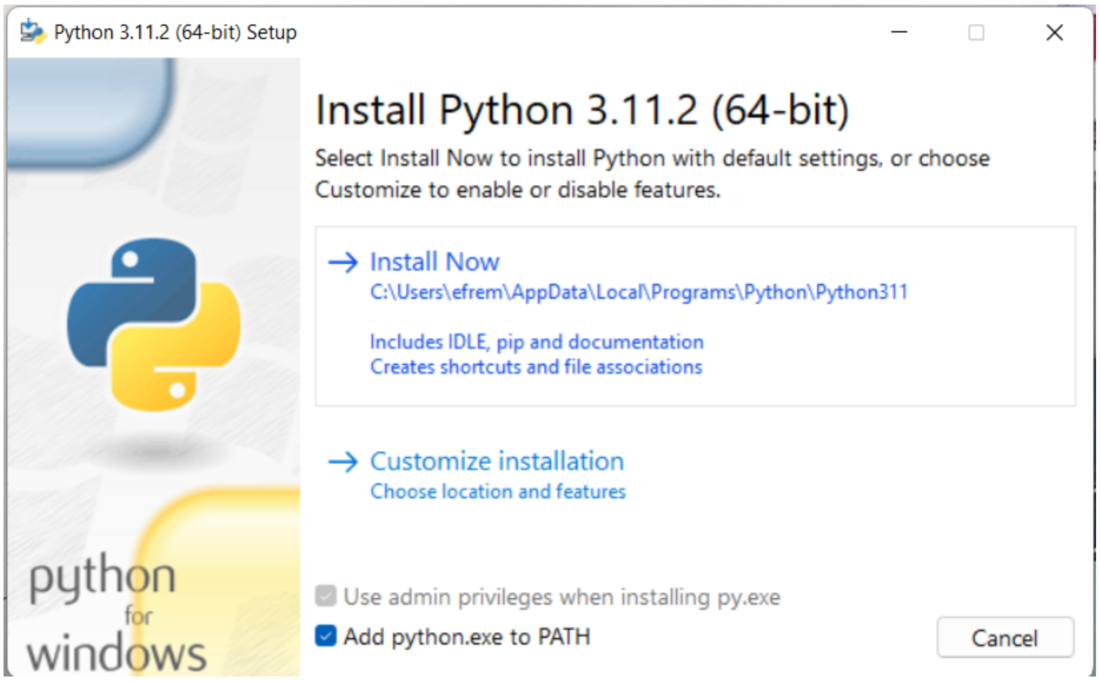
- open Vscode → File → Open File


File Edit Selection View Go Run Terminal Help
Welcome - Visual Studio Code
New Text File Ctrl+N
New File... Ctrl+Alt+Windows+N
New Window Ctrl+Shift+N
Open File... Ctrl+O
Open Folder... Ctrl+K Ctrl+O
Open Workspace from File...
Open Recent
Add Folder to Workspace...
Save Workspace As...
Duplicate Workspace
Save Ctrl+S
Save As... Ctrl+Shift+S
Save All Ctrl+K S
Share
Auto Save
Preferences
Revert File
Close Editor Ctrl+F4
Close Window Alt+F4
Exit
ual Studio Code
g evolved
File...
File...
Folder...
Git Repository...
clss C:\Users\efrem\Desktop
ss C:\Users\efrem\Desktop
Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Get started with Jupyter Notebooks New
Get started with Python development New
Get started with JavaScript and Node.js New
Learn the Fundamentals
More...

## Download and Install Python.

- Use This Link to download python [Download Python - Python.org]

- **Installation step are the following.**
    - check on add python.exe to PATH → Install Now



Python 3.11.2 (64-bit) Setup

Install Python 3.11.2 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

→ Install Now
C:\Users\efrem\AppData\Local\Programs\Python\Python311

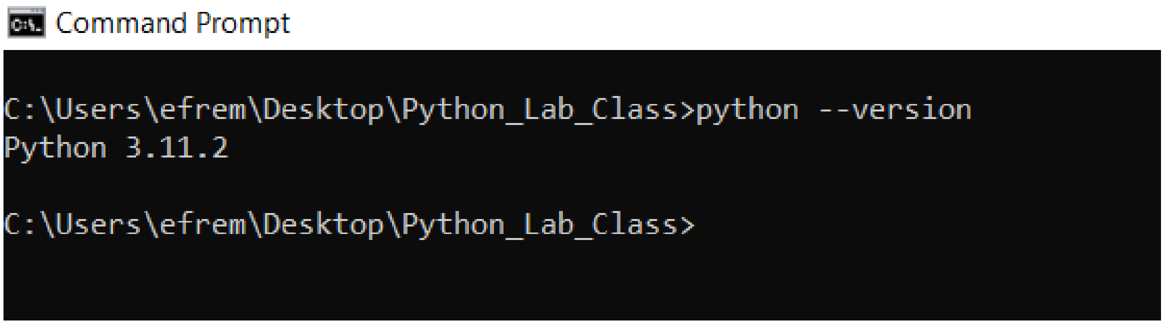
Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

[ ] Use admin privileges when installing py.exe
[✓] Add python.exe to PATH

- To check if Python is properly installed use the below command:

**python --version**

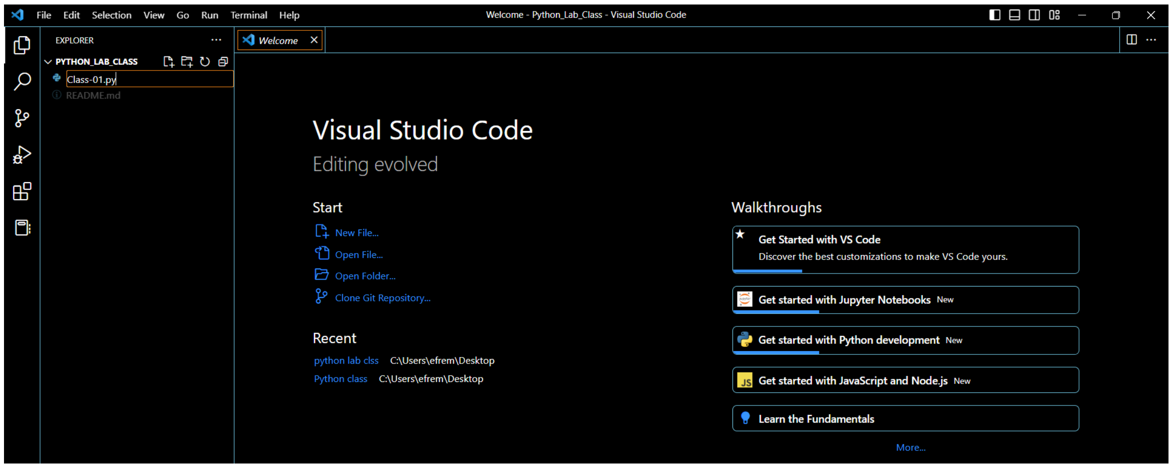


Command Prompt

```
C:\Users\efrem\Desktop\Python_Lab_Class>python --version
Python 3.11.2

C:\Users\efrem\Desktop\Python_Lab_Class>
```

## My First Python Code.

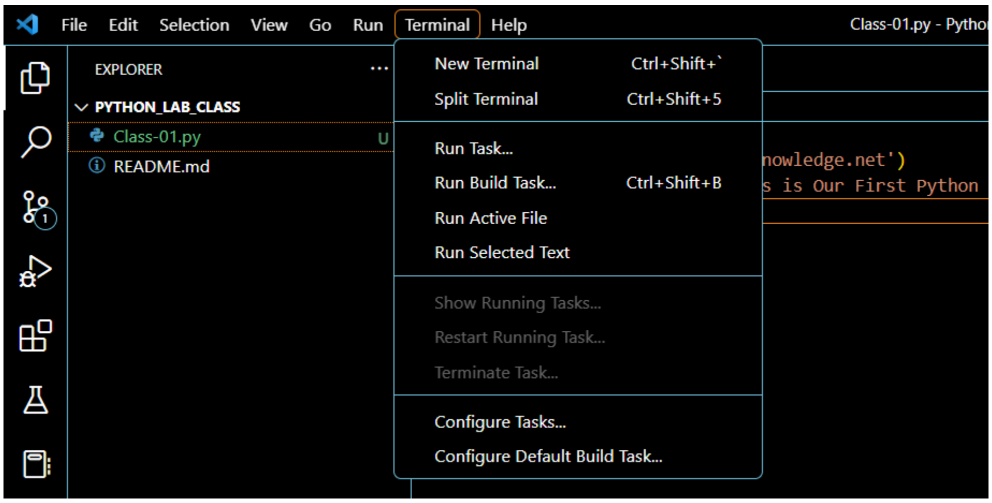
- Create python file with name Class-01.py in **Vscode**

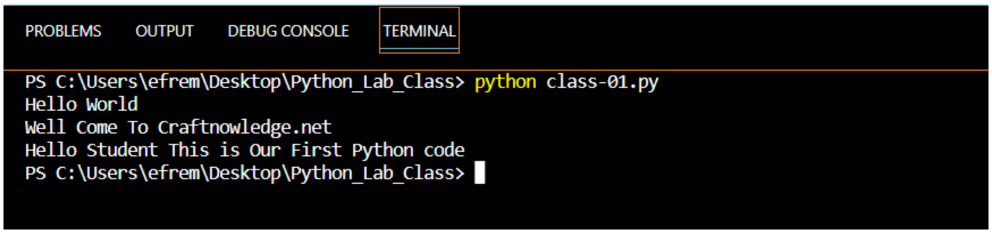

File Edit Selection View Go Run Terminal Help
Welcome - Python_Lab_Class - Visual Studio Code
EXPLORER
PYTHON_LAB_CLASS
Class-01.py
README.md
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Recent
python lab clss C:\Users\efrem\Desktop
Python class C:\Users\efrem\Desktop
Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Get started with Jupyter Notebooks New
Get started with Python development New
Get started with JavaScript and Node.js New
Learn the Fundamentals
More...

- Type the following code

```
print 'Hello World'
```

- To run ( execute) our code.
    1. using Terminal or CMD
        i. Terminal → New Terminal → python < file name >





```
PS C:\Users\efrem\Desktop\Python_Lab_Class> python class-01.py
Hello World
Well Come To Craftnowledge.net
Hello Student This is Our First Python code
PS C:\Users\efrem\Desktop\Python_Lab_Class>
```

## Python Comment

Comments are lines that exist in computer programs that are ignored by compilers and interpreters. Using comments in programs can make code more readable for humans, as it provides some information or explanation about what each part of a program is doing.

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.

**1.Single line comment Comments**

starts with #, and Python will ignore them:

Comments can be placed at the end of a line, and Python will ignore the rest of the line:

**2.Multiline Comments**

Python does not really have a syntax for multiline comments.

To add a multiline comment you could insert a # for each line:

```
#This is a comment

#written in

#more than just one line

print("Hello, World!")
```

Or, not quite as intended, you can use a multiline string. Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

Example

```
"""

  This is a comment

  written in

  more than just one line

"""

print("This is Multiline comment")
```

## Python Variables and Assignments

A **variable** is a string of characters and numbers associated with a piece of information.

In Python, variables need not be **declared** or defined in advance, as is the case in many other programming languages.

A Python variable is a symbolic name that is a reference or pointer to an object

The **assignment operator**, denoted by the “=” symbol, is the operator that is used to assign values to variables in Python.

```
 n = 300

Name= 'Icraft'

Age= 14

price= 200.34

is_active= True


print Name

print Age

print price

print is_Paid
```

### Variable Names

variable names in Python can be any length and can consist of

- uppercase and lowercase letters (A-Z, a-z),
- digits (0-9), and the
- underscore character (_).

**Note** The first character of a variable name cannot be a digit.

## Data Type

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

| | |
|---|---|
| Text Type: | str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple |
| Mapping Type: | dict |
| Boolean Type: | True, False |

### 1. **Python String Data Type**

- Python Strings are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:] ) with
- indexes starting at 0 in the beginning of the string, and working their way from -1 at the end.
- Plus (+) sign is the string concatenation operator
- Asterisk (*) is the repetition operator in Python.

Example

```
str = 'Hello World!'

print (str)          # Prints complete string

print (str[0])       # Prints first character of the string

print (str[2:5])     # Prints characters starting from 3rd to 5th

print (str[2:])      # Prints string starting from 3rd character

print (str * 2)      # Prints string two times

print (str + "TEST") # Prints concatenated string
```

**Note** we'll learn more about string in coming class.

## 2.**Python Numeric Data Type**

Python supports three different numerical types –

1.**int** :- integers are zero, positive or negative whole numbers without a fractional part.

Example.

```
int_1 = 0

int_2 = 100

int_3=-10

int_4= 1234567890
```

2.**float** (floating point real values) : floating point numbers (float) are positive and negative real numbers with a fractional part denoted by the decimal symbol .

Example.

```
float_1=1234.56

float_2=3.142

float_3=- 1.55

float_4=0.23
```

3.**complex** (complex numbers) : A complex number is a number with real and imaginary components.

For example,

```
complex__num = 5 + 6j
```

5 is the real component and 6 multiplied by j is an imaginary component.

## 3.**Python Sequence Data Type**

1. **Python List** Data type is a **mutable** sequence type. A list object contains one or more items of different data types in the square brackets [] separated by a comma.

Example:

```
Student_List=  "Abebe", "Kebed", "Chala"

print Student_List 0

print Student_List 1

print Student_List 2

print Student_List
```

**Note** we'll learn more about List in coming class.

2. **Python -Tuple** is an immutable (unchangeable) collection of elements of different data types. It is an ordered collection, so it preserves the order of elements in which they were defined.

Tuples are defined by enclosing elements in parentheses (), separated by a comma. The following declares a tuple type variable.

```
names_tuple =  'Abebe'  'Kebed'  'Chala'  # string tuple

print names_tuple 0

print names_tuple 1

print names_tuple 2

print names_tuple
```

**Note**:- The primary difference between tuples and lists is that **tuples are immutable as opposed to lists which are mutable.**

Therefore, it is possible to change a list but not a tuple. The contents of a tuple cannot change once they have been created in Python due to the immutability of tuples.

**Note** we'll learn more about Tuple in coming class.

## 4. **Python Mapping Data Type**

**Python - Dictionary**:-The dictionary is an unordered collection that contains key : value pairs separated by commas inside curly brackets.

```
capitals =  "USA" "Washington D.C."  "France" "Paris"  "India" "New Delhi"

print capitals 'USA'

print capitals 'France'

print capitals 'India'

print capitals
```

The left side of : is a key, and the right side is a value. The key should be unique and an immutable object. A number, string or tuple can be used as key.

**Note** we'll learn more about Dictionary in coming class.

## 5. **Python Boolean Data Type**

Booleans represent one of two values: True or False.

In programming you often need to know if an expression is True or False.

Example.

```
    print 10 > 9

    print 10 == 9

    print 10 < 9
```

## Python Operators

### Arithmetic Operators

Arithmetic operators perform the common mathematical operation on the numeric operands.

The following table lists all the arithmetic operators in Python:

| Operator | Name | Example |
|---|---|---|
| + | Addition | x + y |
| - | Subtraction | x - y |
| * | Multiplication | x * y |
| / | Division | x / y |
| % | Modulus | x % y |
| ** | Exponentiation | x ** y |
| // | Floor division | x // y |

**1. Addition Operators** :- Adds values on either side of the operator.

Example.

```
x=5  y=10

z=x + y

print z

import operator

z=operator.add(x, y)

print z
```

**2. Subtraction Operators**:- Subtracts right hand operand from left hand operand.

Example.

```
x=5  y=10

z=x - y

print z

import operator

z=operator.sub(x, y)

print z
```

**3. Multiplication  Operators:-** Multiplies values on either side of the operator.

Example.

```
x=5  y=10

z=x * y

print z

import operator

z=operator.mul(x, y)

print z
```

**4. Division Operators:-** Divides left hand operand by right hand operand.

Example.

```
x=5  y=10

z=x / y

print z

import operator

z=operator.truediv(x, y)

print z
```

5. **Modulus Operators:-** Divides left hand operand by right hand operand and returns remainder.

Example.

```
x=5  y=10

z=x % y

print z

import operator

z=operator.mod(x, y)

print z
```

**6. Exponentiation Operators** :- Performs exponential (power) calculation on operators.

Example.

```
x=5  y=10

z=x ** y

print z

import operator

z=operator.pow(x, y)

print z
```

**7. Floor division Operators** The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) −

Example.

```
x=5  y=10

z=x // y

print z

import operator

z=operator.floordiv(x, y)

print z
```

**Division vs Floor division — division vs floor division**

The result of regular division is **always** a float, whereas if one of the operands is a float in *floor* division, then the output will be a float.

## Operator Precedence

The order of operation for arithmetic operators follows **PEMDAS** rule.

- Parenthesis,
- Exponentiation
- Multiplication
- Division
- Addition
- Subtraction

Associativity rule will be followed for operators with the same precedence.

All the operators, except exponentiation(**) follow the left to right associativity.

Multiplication and Division the have same precedence

Addition and Subtraction the have same precedence

**Example** calculate the following exertions.

(43+13−9/3∗7)

solution

Interpreter encounters the parenthesis (. Hence it will be evaluated first).

Precedence of (/,∗) > Precedence of (+,−)

In this case, the precedence of multiplication and division is equal, but further, they will be evaluated according to the left to right associativity.

Associativity rule will be followed for operators with the same precedence.

This expression will be evaluated from left to right, 9/3∗7= 3∗7= 21

Now, our expression has become 43+13−21

Left to right Associativity rule will be followed again. So, our final value of expression will be, 43+13−21 = 56−21= 35