

# Introduction

Programming Language Theory

# For Practices

- We will write some code in different languages.
- Hence you may need to install compilers and interpreters for several languages.
- Also, you need to setup your own software development environment.
- In this course, practices will be explained mainly with VSCode.
- However, you can use any tools which you're familiar with.

# Why We Setup Development Environment?

- This is the very first step for successful software development.
- Programming does not mean simply writing code.
- It also includes various tasks such as software design, verification and debugging.
- You cannot perform these tasks without good development environment.

# How to Setup Development Environment?

- Usually, it is setting up the environment to *write* and *build* code for your program, and also *execute* and *verify* the program.
- Mostly, it is done by installing compilers (or interpreters), and installing IDE and configuring it.
- There are other tasks such as source code management, issue tracking, documentations which you might need to consider.

# Integrated Development Environment

- IDE: a program supports various software development tasks (e.g., VSCode, Eclipse, IntelliJ, PyCharm, etc.).
- Major Features
  - Syntax Highlight
  - Auto Completion
  - Build
  - Debugging Support
  - Automatic Code Formatting
  - Refactoring
  - Version Control

# Syntax Highlight

- Highlight words in different syntactical positions.
- Readability of code is greatly increased, hence the productivity is increased too.
- Checking syntactic errors in pre-compile time in Code Editor.

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

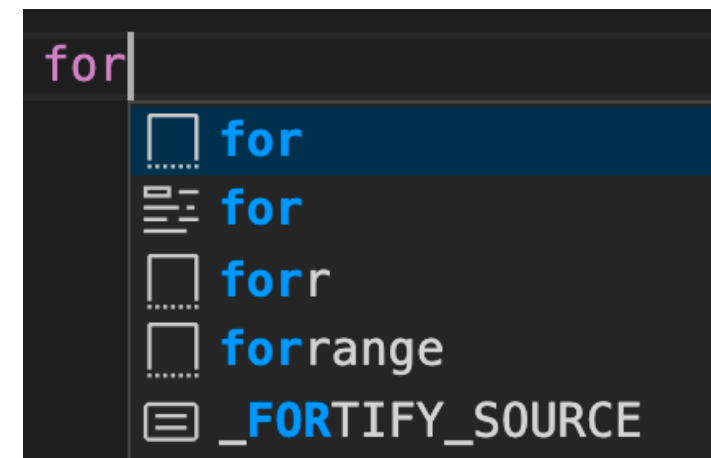
int main() {
    cout << "Hello World!\n";
    return 0;
}
```

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

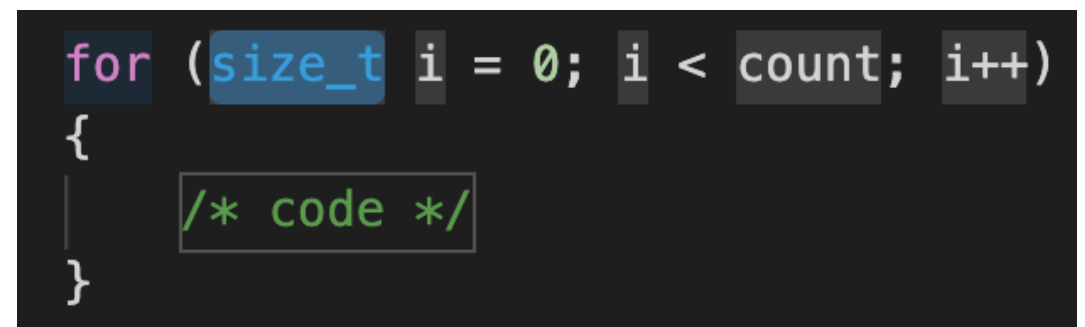
int main() {
    cout << "Hello World!\n";
    return 0;
}
```

# Auto Completion

- Automatically recommend or complete code after typing a few characters.
- One of the most great features of IDE.
- Significant influence on developers' productivity.
- So many research on more efficient, effective auto-completion.



A screenshot of an IDE's auto-completion feature. The word 'for' is typed in the editor, and a dropdown menu is displayed with several suggestions: 'for' (highlighted), 'for', 'forr', 'forrange', and '\_FORTIFY\_SOURCE'. Each suggestion is preceded by a small icon representing its syntax category.



A screenshot of an IDE showing the completed C++ code. The code is a for loop: `for (size_t i = 0; i < count; i++)`. The code is enclosed in curly braces, and a comment `/* code */` is visible inside the loop body. The code is color-coded: 'for' is pink, 'size\_t' is blue, 'i' is green, '0' is green, 'i' is green, '<' is green, 'count' is green, and 'i++' is green.

# Build

- Automatically compile necessary files to make an executable program.
- Dependency management, Packaging.
- Complex programs may have code on many files and complicated dependencies.
- Considering all these would be painful if you need to do that repeatedly.
- With IDE, you can simply build your program (or a project) by clicking a button.



# Program Execution

- You can also execute your program and check the output in IDE's console.
- When you modify your code, you can directly execute the program and verify the influence of modification.
- If your program requires complex inputs or configuration for execution, you can configure such requirements once, and use them repeatedly.

# Debugging Support

- You can execute your code line by line, and check how values in memory are changed.
- For instance, you can set a break point at line 10, then run your program in debug mode.
- The execution just stops at line 10, and waiting for your command.
- You can see the status of your variables and verify that they are as expected.
- Also, you can execute your program further from that point, to observe your program's execution in more details.

# Automatic Code Formatting

- It's very important to follow code style guidelines when many people working together.
- Consistent code style → Better Readability.
- Crude code style → Bad Handwriting.
- IDE provides various configurations to keep your code in appropriate style.

# Unresolved Debates

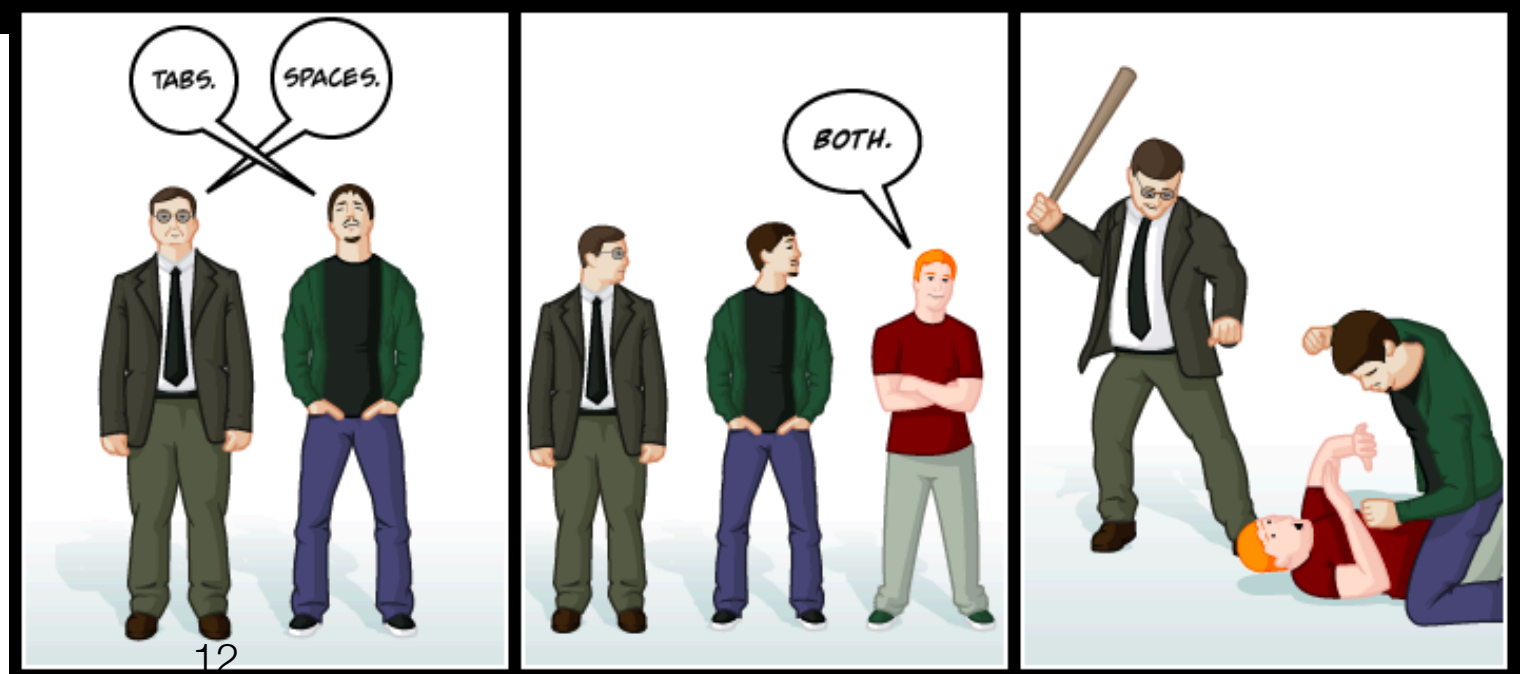
There are two types of people:

```
if (Condition) {  
  Statement  
  /* ....  
  */  
}
```

```
if (Condition)  
{  
  Statement  
  /* ....  
  */  
}
```

**Curly Brackets:**  
**Are you left or right?**

**Tabs vs. Spaces:**  
**Are you a tab guy or a space guy?**



# Refactoring

- Refactoring is a task to improve the quality of code.
- It maintains the same functionalities of code, while modifying the structures of code.
- Often expect to improve Readability, Maintainability, and Reusability.
- IDE provides commands to automatically perform refactoring on your code.

# Version Control

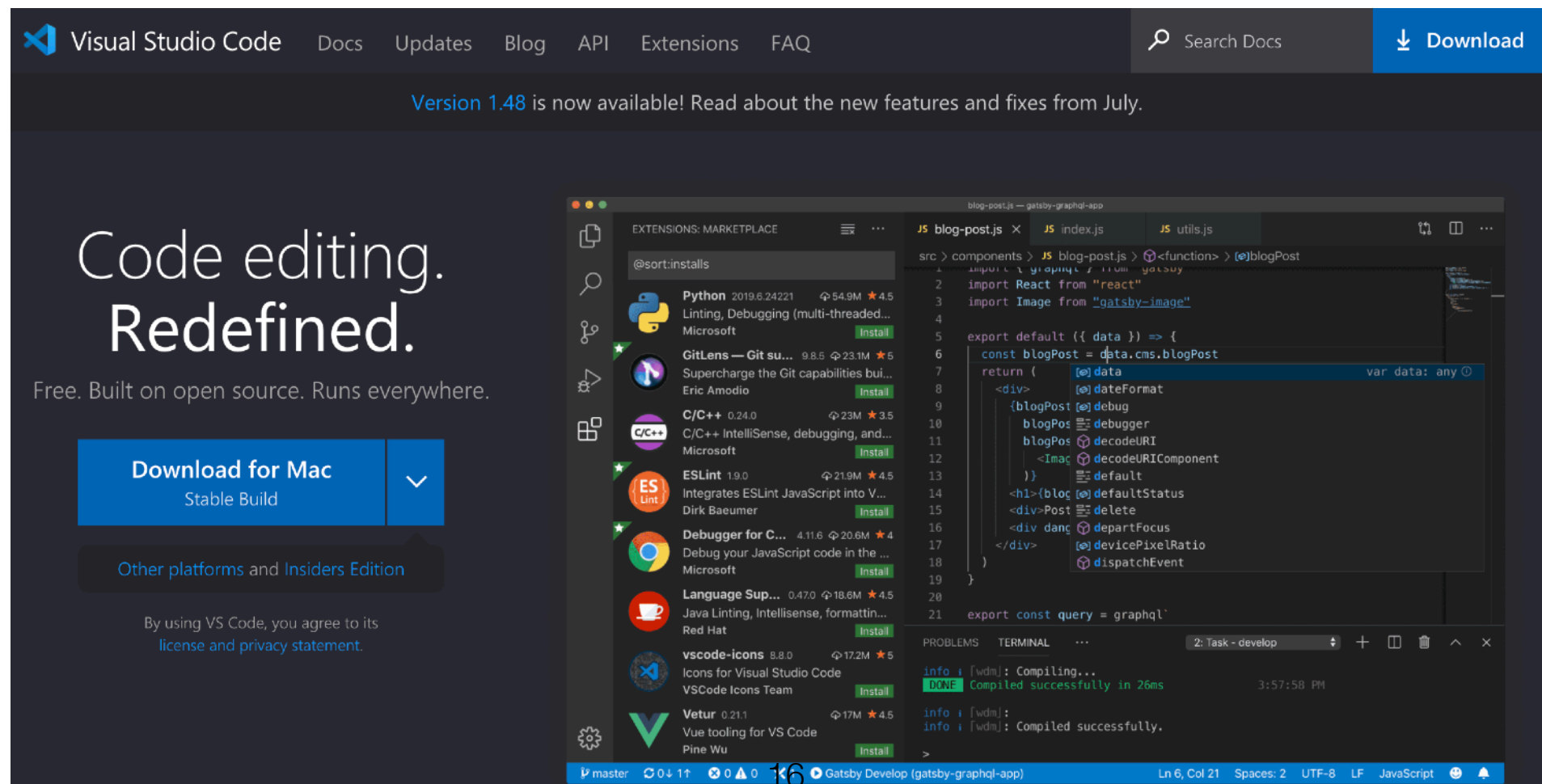
- Keep tracking modifications in code.
- When more than one people are involved in development, you can synchronize with the others and prevent conflicts.
- IDEs are often integrated with version control system.
- You can easily commit your changes to software repositories, resolve conflicts with IDE.

# Many Others

- Toggle Comment
- File Comparison
- Advanced Code Navigation
  - Go to Definition, Declaration, File, and Line.
- Advanced Code Search
  - Find all references of a variable.
- Fancy Fonts

# VSCode

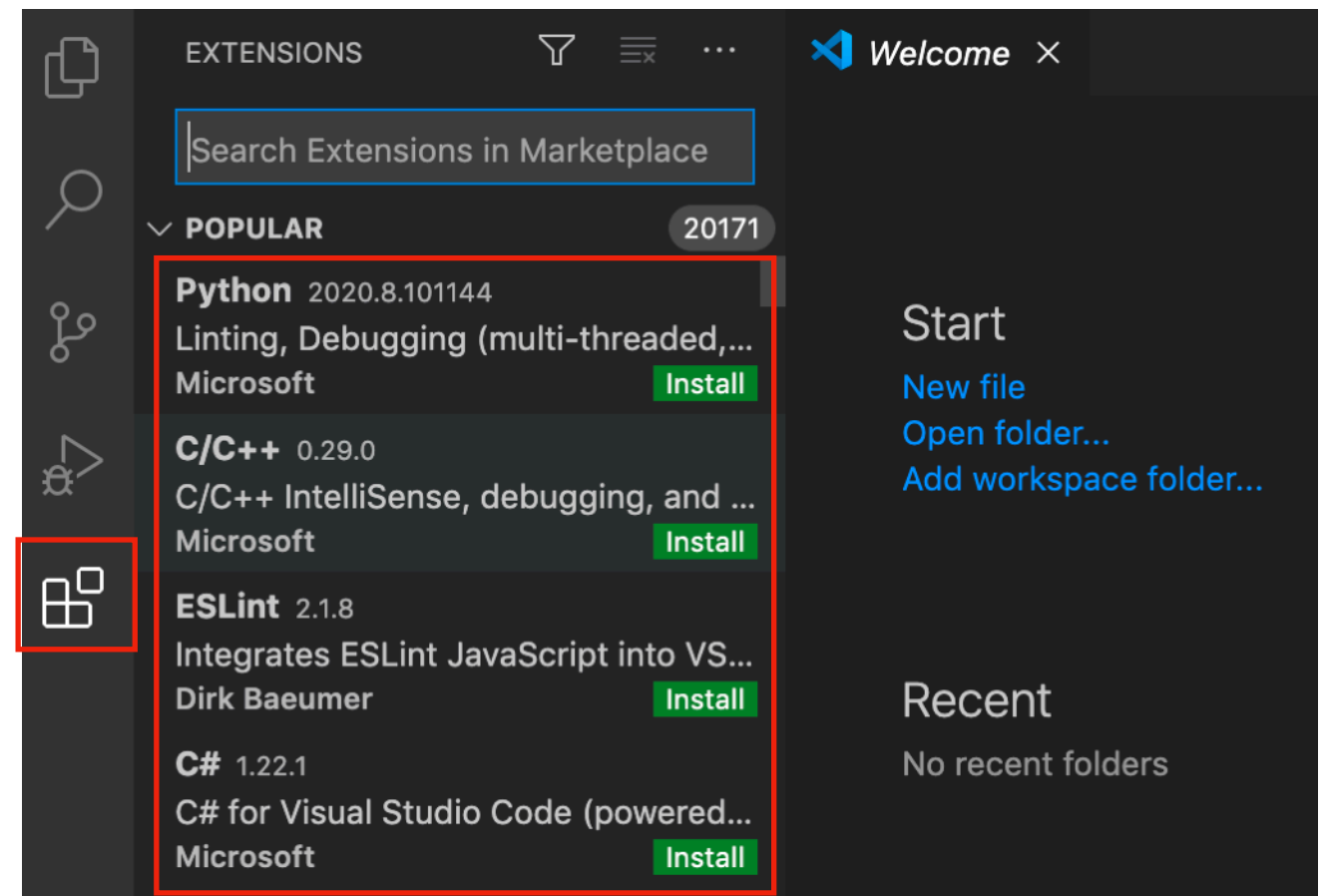
- Visual Studio Code: Free IDE developed by Microsoft.
- Support various OS - Windows, Mac, Linux
- Using Extensions to support various programming languages.





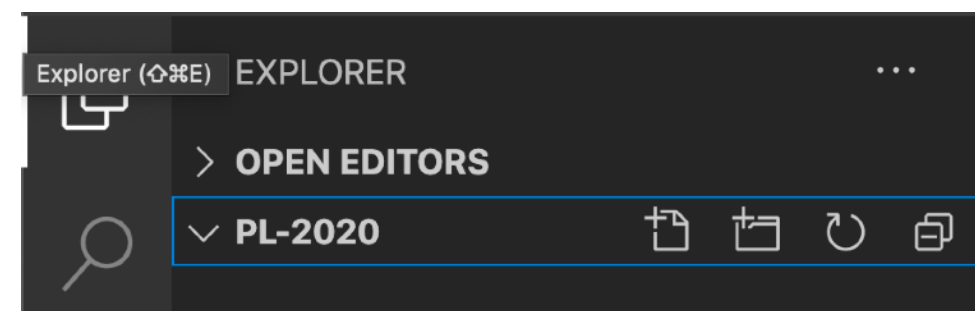
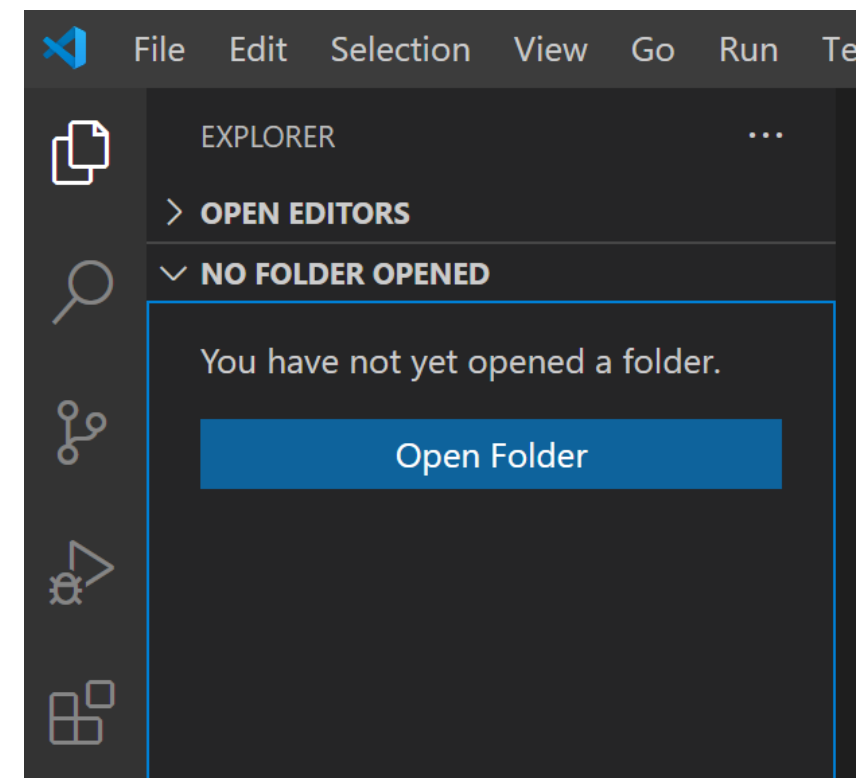
# Extensions

- Support for various programming languages via Extensions.
- To setup development environment for a new programming language,
  1. Install a compiler or an interpreter for the language.
  2. Install Extension and setup according to 1.



# Workspace

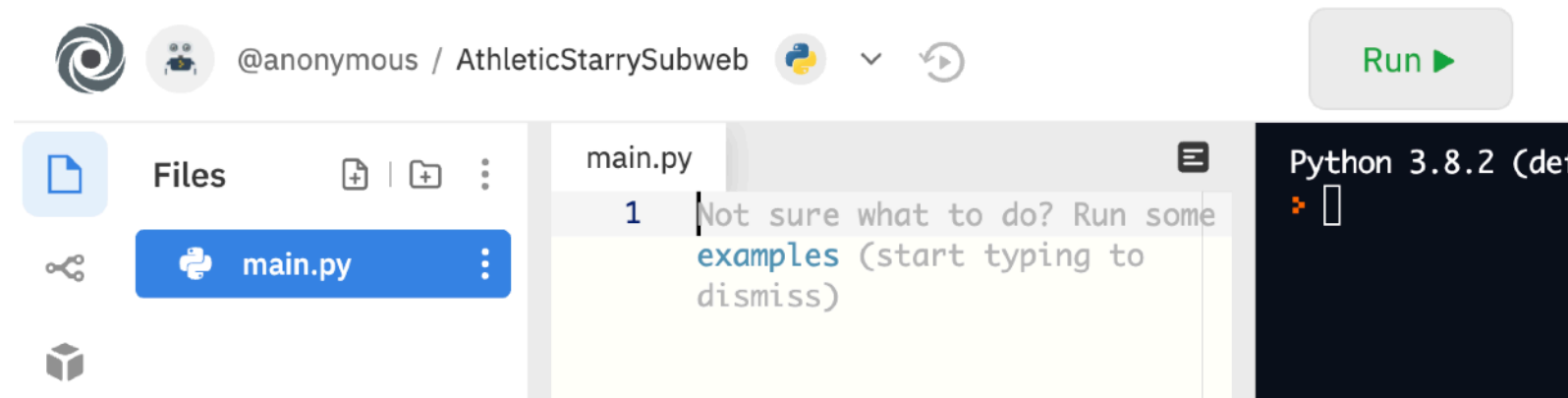
- Workspace for software development.
- Collection of all stuff for your program.
- Different configurations for different workspace.
- You can switch between workspaces when you need to work on different projects.



# Possible Scenario

- You're working on an assignment XXX class.
- You're getting tired of the assignment, and decide to fiddle with interesting PL course stuff.
- Then you just need to switch from XXX workspace to PL workspace.
- All the files and configurations will be switched and you can continue on what you're doing.
- Once you're prepared to go back to the boring stuff, you can switch back again to the previous workspace.

# Repl.it



- Online IDE supports many languages (<https://repl.it/>).
- **Pros:** Don't need to care about how to install and configure compilers and interpreters for various languages / Good practice for online coding exams or interviews.
- **Cons:** Lose an opportunity to learn how to setup development environment for various languages.

# REPL

- Read-Eval-Print Loop: or language shell.
- Read user input, Evaluate the input, and Print the result.
  - e.g.) Python
- Similar to Scripting languages.
- Do not require the whole compilable program.

```
Python 3.7.4  
[Clang 4.0.1  
Type "help",  
>>> a = 3  
>>> b = 5  
>>> a + b  
8  
>>> █
```

# Why IDE?

## This is PL Course!

- We're gonna use several different programming languages to write some programs.
- Although there exist several different ways, it would be better to perform these tasks in unified environment.
- Using IDE is a good options for this.
- Of course, it's totally up to you unless you get the things done!

# Summary

- What is IDE?
- Features supported by IDE
- VSCode, [repl.it](#)