Introduction

Programming Language Theory

Contacts

- Communications
  - Jindae Kim (김진대), Mirae Hall(미래관) 331
  - Office Hour: You need to contact me to set a meeting.
  - E-mail: jindae.kim@seoultech.ac.kr
  - Post your questions in e-class, so that other classmates can discuss together.

Syllabus

- Programming Language Theory
- Pre-requisites
  - Good understanding of at least one programming language.
  - Basic understanding of C++ and Java, maybe Python too.

Syllabus

- Contents
  - Programming Language Design Principles
  - Programming Language Concepts
  - Programming Language Paradigm
  - A few programming languages in different paradigms.

Textbooks

- No official textbooks.
    - If you want to study further, we may check the following books as references.
    - Michael L. Scott, Programming Language Pragmatics, 4th Edition, Morgan Kaufmann
    - Maurizio Gabbrielli and Simone Martini, Programming Languages: Principles and Paradigms, Springer-Verlag London
    - Robert W. Sebesta, Concepts of Programming Languages, 11th edition, Pearson
- Please, don’t buy the books unless you’re really into it.
- We will use many other materials related to PL during the semester.

Course Organization

- We will first look at programming language design principles and concepts for the first half of the semester.
    - How can we design and implement PLs?
- For the next half, we will study various programming language paradigms with selected specific languages.
    - What kinds of programming languages are there?

Course Organization

- Every week consists of **two online lectures + online real-time practice**.
- **Lectures will be posted on e-class**.
  - It might be changed based on the university policy after the mid-term exam.
- **One practice session on Friday**: simple tasks related to the same week's lectures.
  - It means that ***you have to watch the lecture videos before the practice session***.

Evaluation

- Evaluation
  - Mid-term Exam 40%
  - Final Exam 40%
  - Assignments 20%

Assignments

- Basically assignments are to verify that you're finishing your tasks in practice sessions.
- You can submit your results to e-class during practice sessions.
  - i.e., an assignment will be posted at the beginning of each practice session, and its deadline will be around the end of the session.

Practice Sessions

- You need to attend practice sessions via Zoom.
- Every Friday 10:00AM~11:50AM (~2 hours). 😔
  - Schedule could be adjusted within the course schedule.
- Doing simple assignments - you can ask questions and discuss with your friends.
- If you finish your tasks and submit your results, you can leave early. 😏
- Assignments can only be submitted ***during practice sessions***.

Introduction to Programming Language Theory

Programming Language Theory

- So far, you're mostly 'using' programming languages for software development.
- How about 'making' programming languages?
- Programming Language (PL) theory is about **how to design good programming languages**, and build a basis for programming language development.

Scope of This Course

- The aim of this course is to ***understand PL concepts and paradigms***, and use that knowledge ***to help learning new programming languages***.
- Normally PL courses cover very serious theoretical stuff from the foundation.
- However, not everyone is interested in programming language development.
- Apologies to students who want to create their own programming languages; this course doesn't cover that much.

Scope of This Course

- Still, this is actually a theory course.
- We cannot avoid studies on theoretical foundation of programming languages.
- This will help you have more deep understanding in programming languages.
- Also, many of these contents will make you look very professional!

Why PL Theory?

- There have been so many different programming languages.
- Useful common concepts among these languages have been studied, evolved, and reflected on new programming languages.
- How can we include useful concepts in a new programming language?
    - While minimize accompanying drawbacks?

PL Concepts and Paradigms

- Many programming languages are different implementations of the similar concepts following the similar paradigms.
- For example, consider a sorting program.
- You may write many sorting programs in different languages implementing different algorithms.
- Still, they’re sorting programs which place something in order.

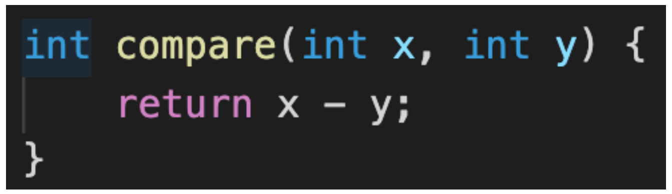
PL Concepts and Paradigms

- **PL concepts**: more like individual features.
- e.g.) data types, control flow, expression, statements, variables, functions, etc.
- **PL Paradigms**: principles and strategies which a PL follows.
- e.g.) Procedural, Imperative, Object oriented, Functional, Logic, etc.

PL Concepts and Paradigms

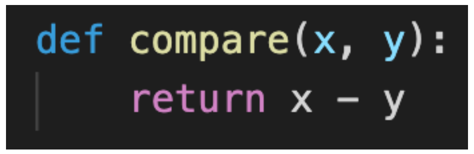
- PLs share common concepts and paradigms.
- Once you understand those concepts and paradigms, learning a PL is now learning how the PL implements them (e.g., syntax).

C/C++



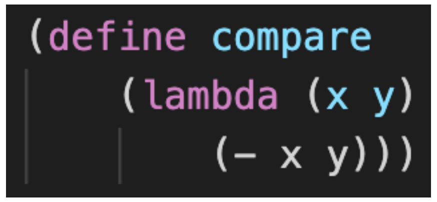
```
int compare(int x, int y) {
    return x - y;
}
```

Python



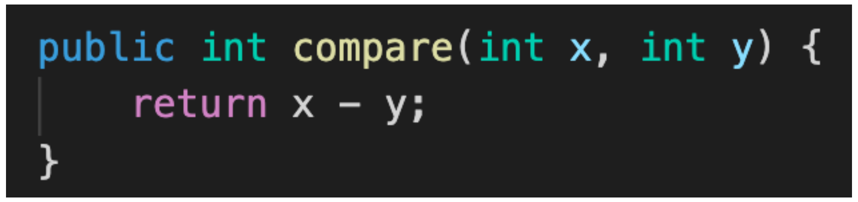
```
def compare(x, y):
    return x - y
```

Scheme



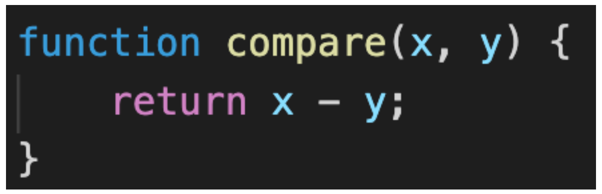
```
(define compare
    (lambda (x y)
        (- x y)))
```

Java



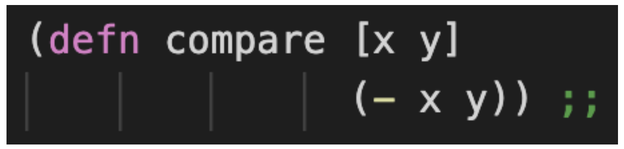
```
public int compare(int x, int y) {
    return x - y;
}
```

JavaScript



```
function compare(x, y) {
    return x - y;
}
```

Clojure



```
(defn compare [x y]
            (- x y)) ;;
```

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?

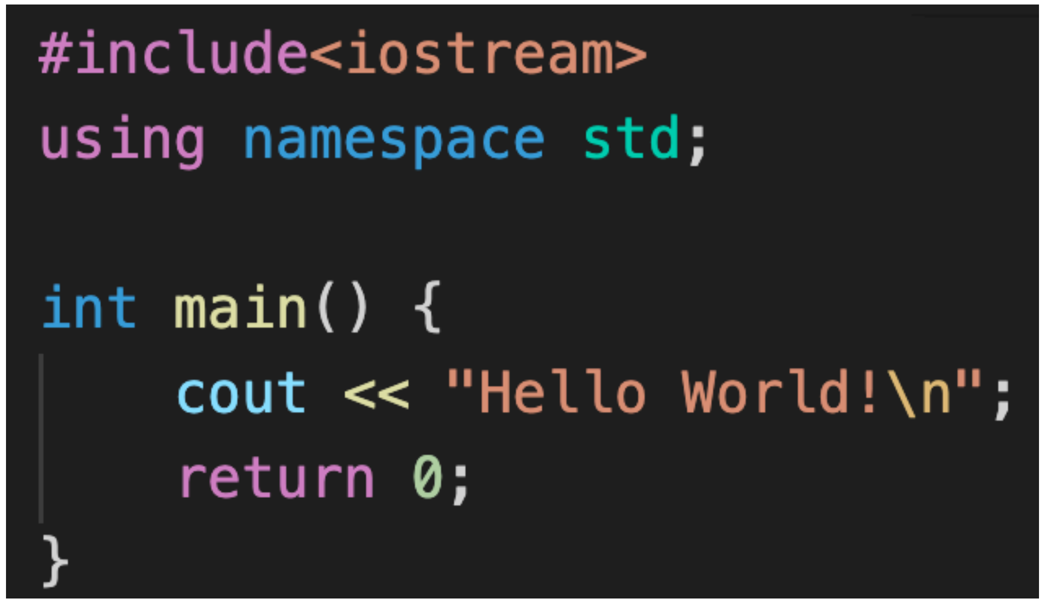
- Usually, it is setting up the environment to *write* and *build* code for your program, and also *execute* and *verify* the program.
- Mostly, it is done by installing compilers (or interpreters), and installing IDE and configuring it.
- There are other tasks such as source code management, issue tracking,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

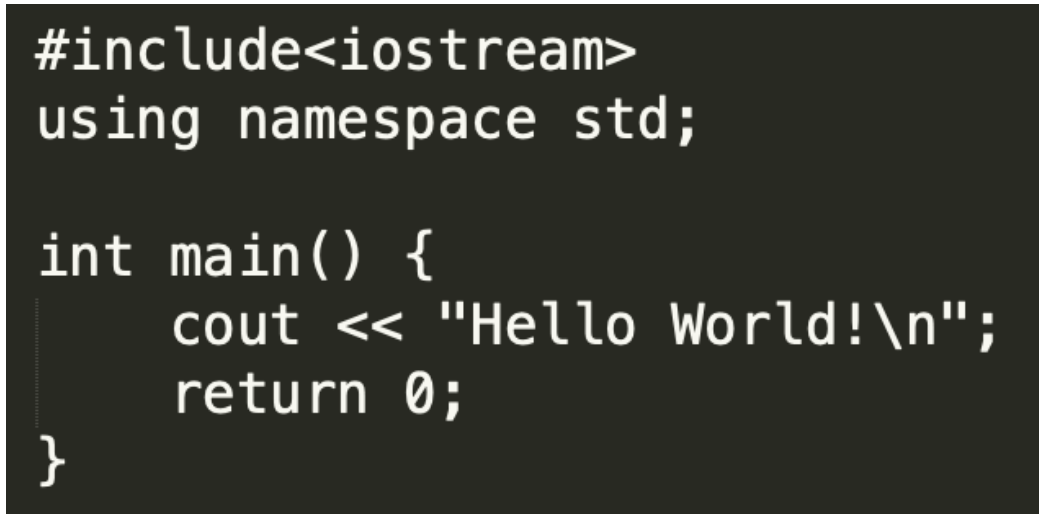
Syntax Highlight

- 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.



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

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

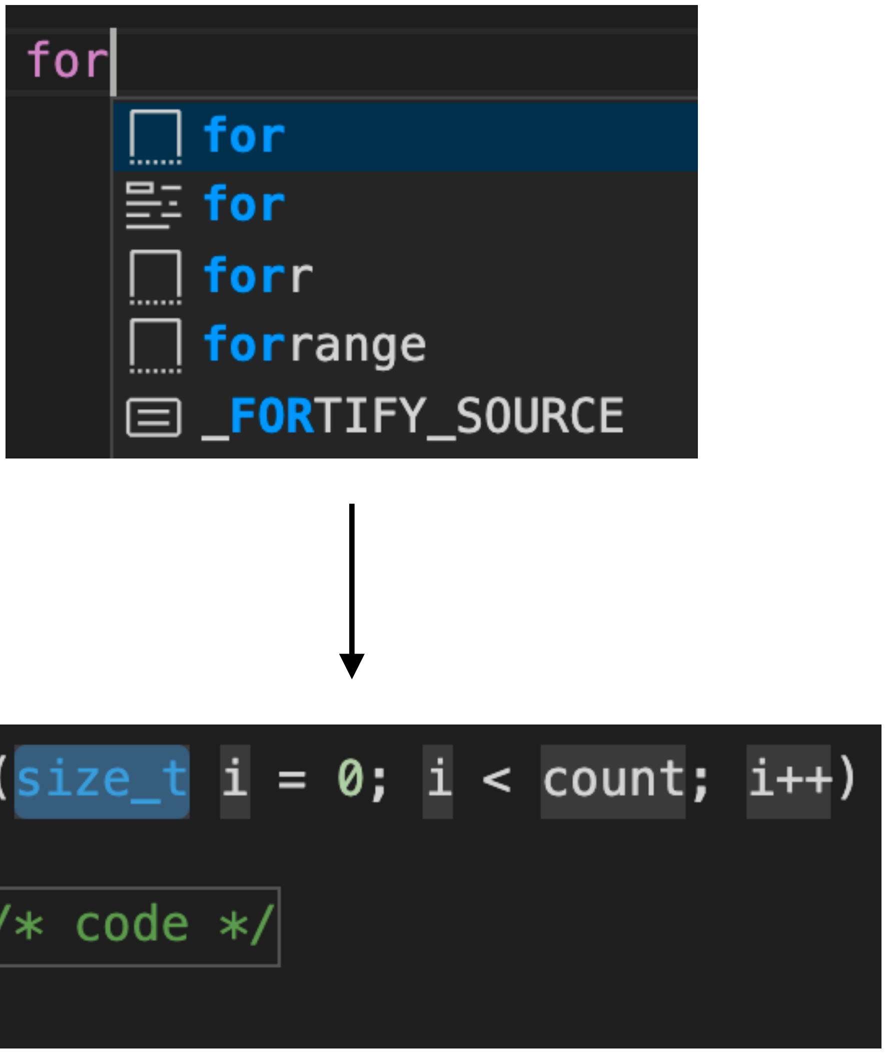


```cpp
#include<iostream>
using namespace std;

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.



```
for
```

```
for (size_t i = 0; i < count; i++)
{
        /* code */
}
```

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.

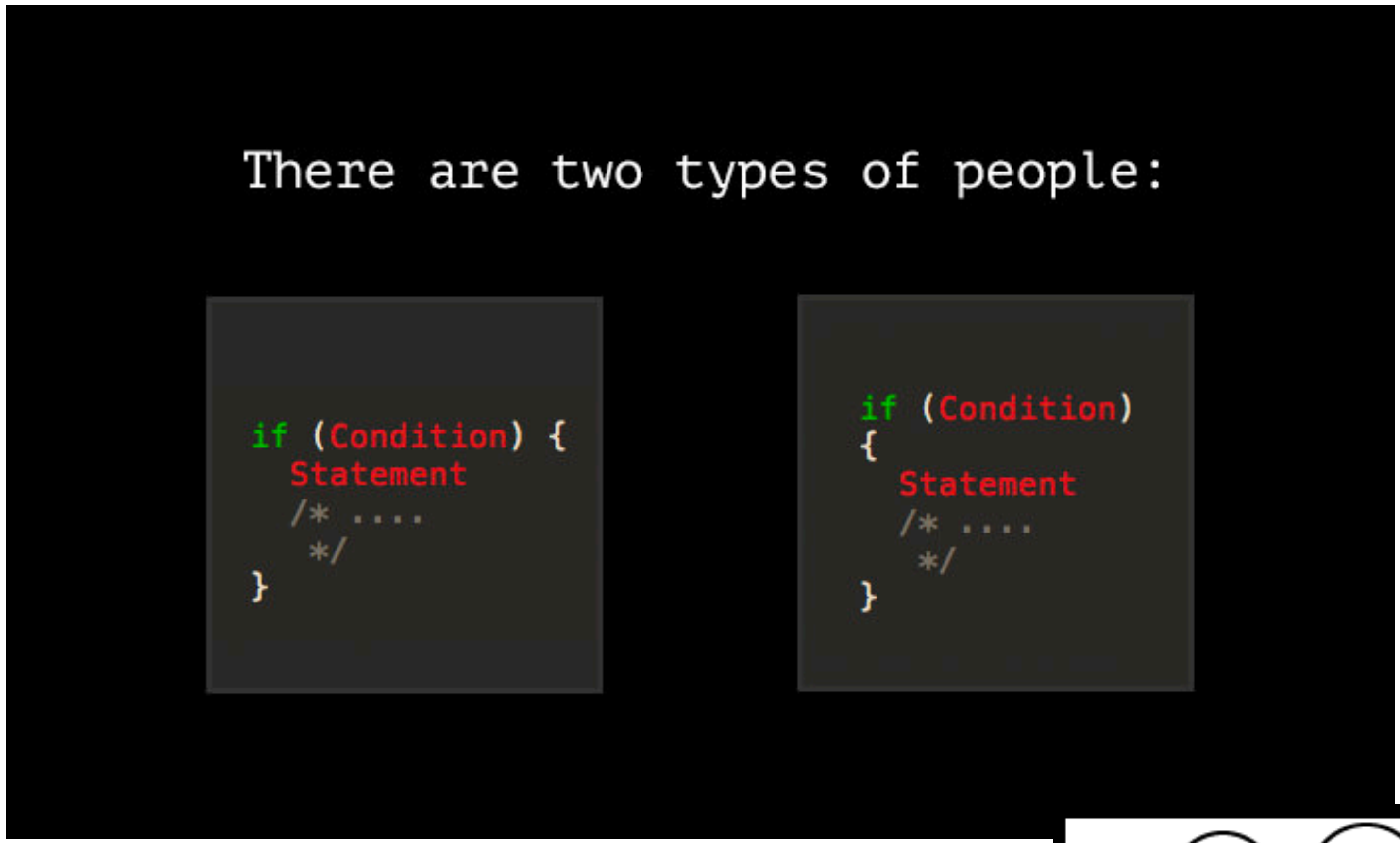
Debugging Support

- 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

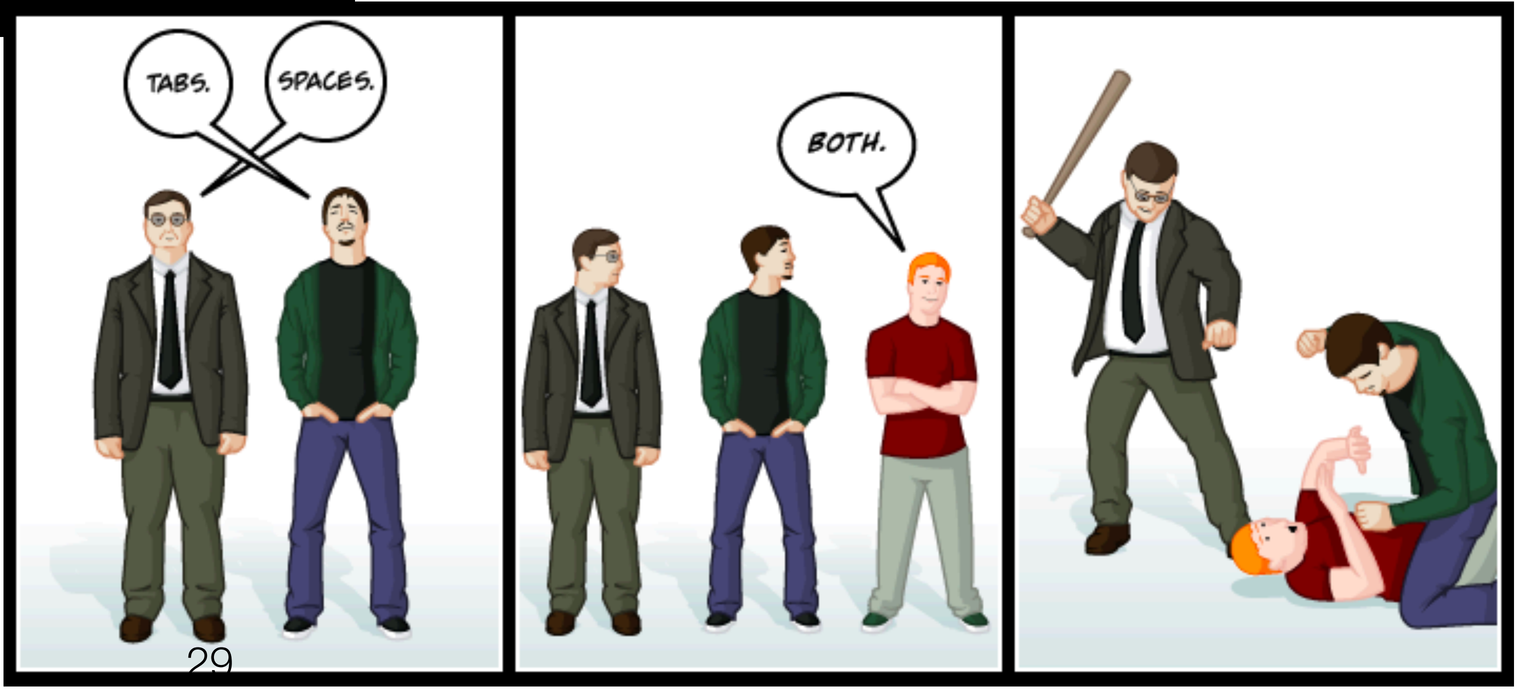


There are two types of people:

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

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

Curly Brackets:
Are you the left or the 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

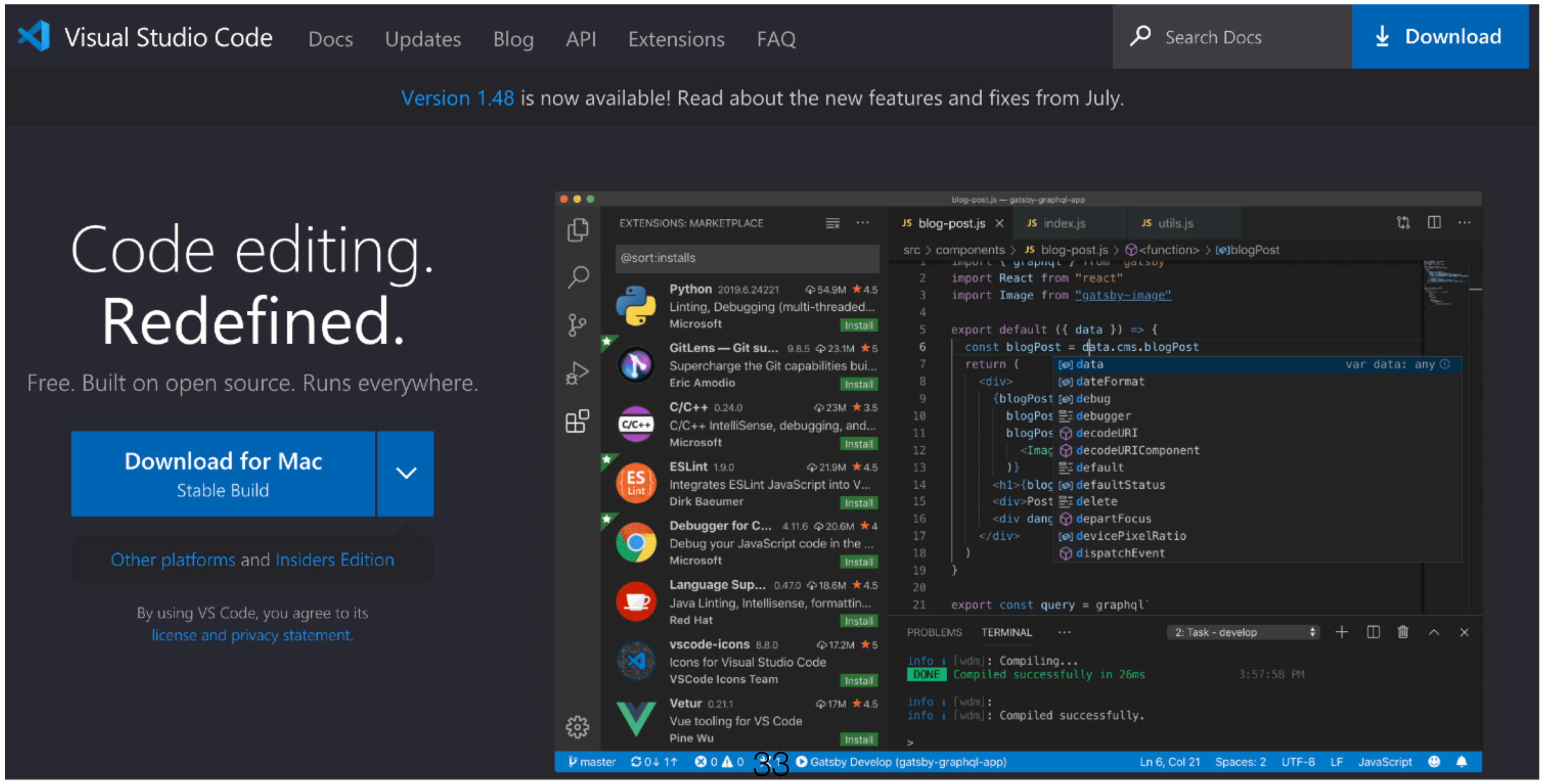
- Keep tracking modifications in code.
- When more than one people are involved in development, you can synchronize with the others and prevent conflicts.
- IDEs are often integrated with version control system.
- You can easily commit your changes to software repositories, resolve conflicts with IDE.

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.


Visual Studio Code
Docs
Updates
Blog
API
Extensions
FAQ
Search Docs
Download
Download for Mac
Stable Build
Other platforms and Insiders Edition
EXTENSIONS: MARKETPLACE
@sort:installs
Python 2019.6.24221 54.9M 4.5
Linting, Debugging (multi-threaded...
Microsoft
GitLens — Git su... 9.8.5 23.1M 5
Supercharge the Git capabilities bui...
Eric Amodio
C/C++ 0.24.0 23M 3.5
C/C++ IntelliSense, debugging, and...
Microsoft
ESLint 1.9.0 21.9M 4.5
Integrates ESLint JavaScript into V...
Dirk Baeumer
Debugger for C... 4.11.6 20.6M 4
Debug your JavaScript code in the ...
Microsoft
Language Sup... 0.47.0 18.6M 4.5
Java Linting, Intellisense, formattin...
Red Hat
vscode-icons 8.8.0 17.2M 5
Icons for Visual Studio Code
VSCode Icons Team
Vetur 0.21.1 17M 4.5
Vue tooling for VS Code
Pine Wu
Install
blog-post.js — gatsby-graphql-app
blog-post.js
index.js
utils.js
src > components > blog-post.js > <function> > blogPost
data
dateFormat
debug
debugger
decodeURI
decodeURIComponent
default
defaultStatus
delete
departFocus
devicePixelRatio
dispatchEvent
var data: any
PROBLEMS
TERMINAL
2: Task - develop
master
Gatsby Develop (gatsby-graphql-app)
Ln 6, Col 21
Spaces: 2
UTF-8
LF
JavaScript

Version 1.48 is now available! Read about the new features and fixes from July.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

By using VS Code, you agree to its license and privacy statement.

```
import { graphql } from "gatsby"
import React from "react"
import Image from "gatsby-image"

export default ({ data }) => {
  const blogPost = data.cms.blogPost
  return (
    <div>
      {blogPost
        blogPos
        blogPos
          <Imag
        )}
      <h1>{blog
      <div>Post
      <div dang
    </div>
  )
}

export const query = graphql`
```

```
info ℹ ｢wdm｣: Compiling...
 DONE  Compiled successfully in 26ms                    3:57:58 PM

info ℹ ｢wdm｣:
info ℹ ｢wdm｣: Compiled successfully.

>
```

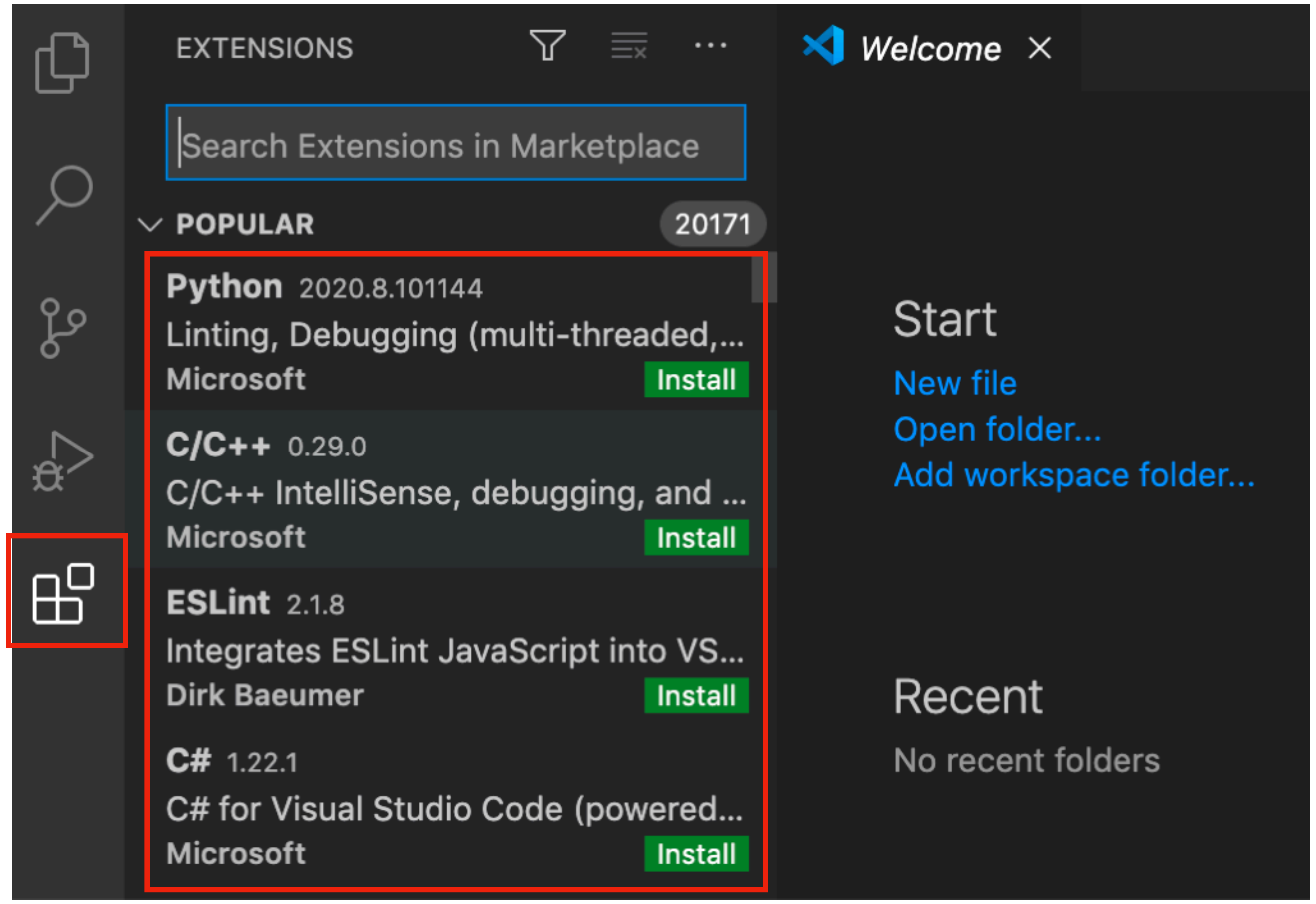
Extensions

- 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.



Python 2020.8.101144
Linting, Debugging (multi-threaded,...
Microsoft Install

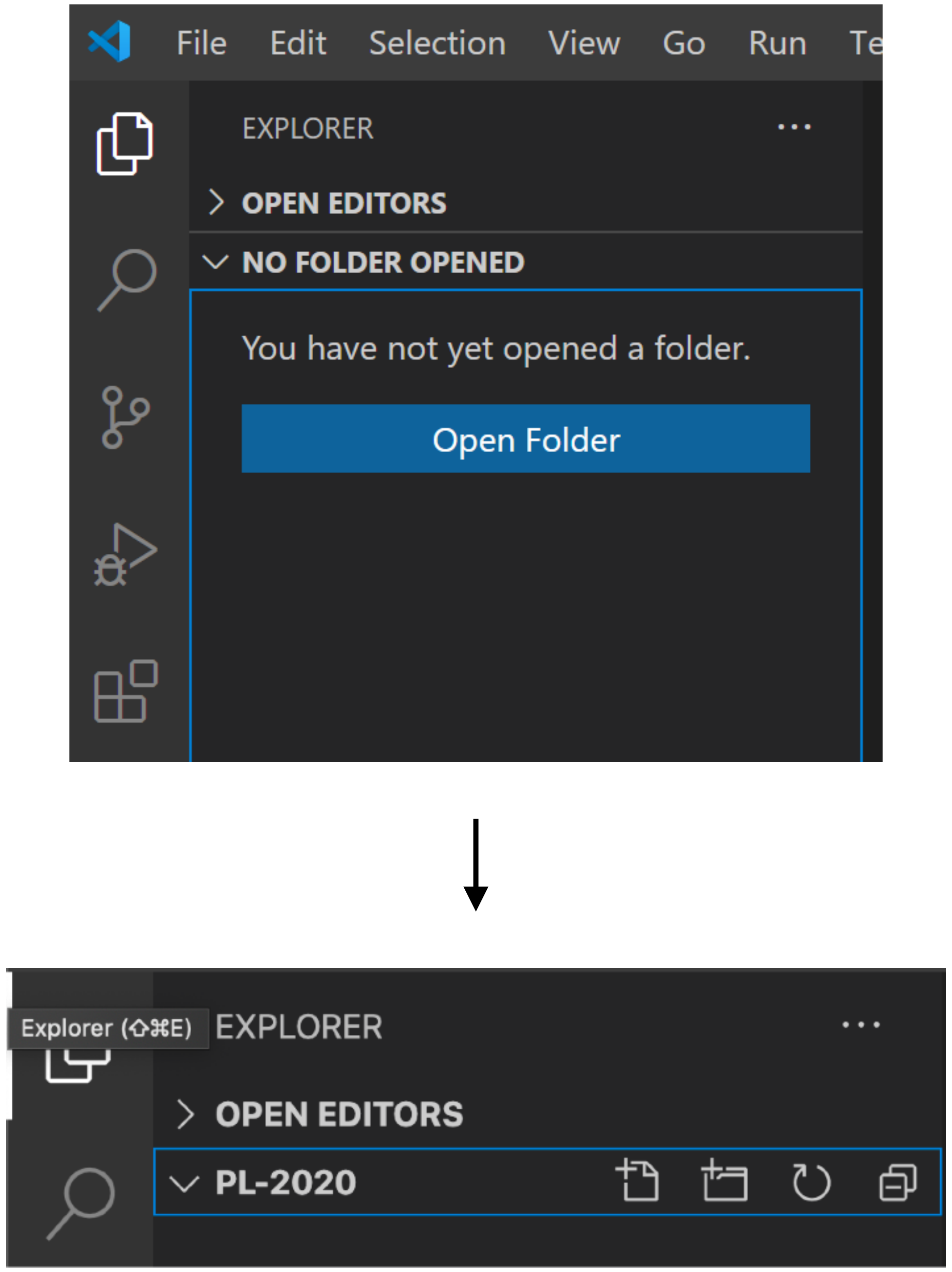
C/C++ 0.29.0
C/C++ IntelliSense, debugging, and ...
Microsoft Install

ESLint 2.1.8
Integrates ESLint JavaScript into VS...
Dirk Baeumer Install

C# 1.22.1
C# for Visual Studio Code (powered...
Microsoft Install

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.

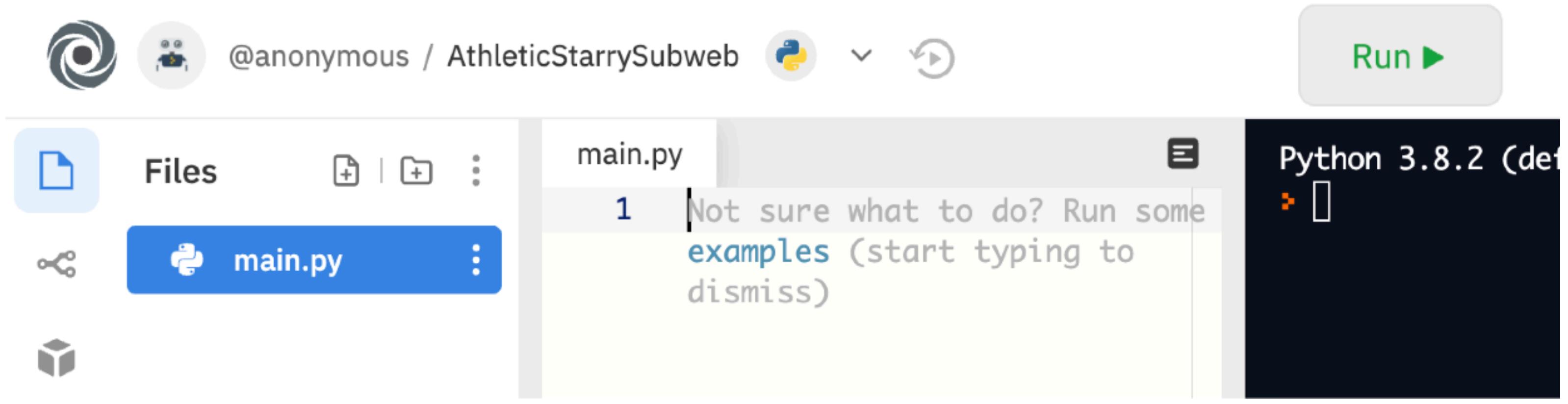


You have not yet opened a folder.

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



```
Not sure what to do? Run some examples (start typing to dismiss)
```

```
Python 3.8.2 (def
```

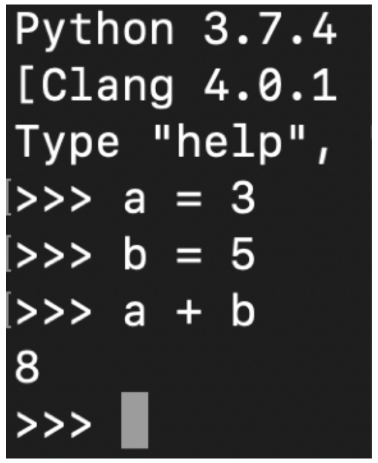
- Online IDE supports many languages (https://repl.it/).

- **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
>>>
```

Summary

- Syllabus
- Course Organization
- PL Concepts and Paradigms
- Using IDEs and REPL