Software Engineering Project 1
COMP10050

Dr. Liliana Pasquale

liliana.pasquale@ucd.ie

http://lili-pasquale.lero.ie

On the Menu Today

- **Why this module is important** for your professional development as a software engineer;
- **What will be the main topics** covered in this module;
- **How the module is organized**;
- **What you need to do to pass the module successfully**!

Why is COMP10050 Important?

Why did you decide to study computer science?

… Hopefully some of you want to be software engineers or programmers
in the future

… Hopefully some of you want to be software engineers or programmers
in the future

… **Are software engineering and programming the same thing?**

No Difference

I do not think there is a difference. Programming is just a form of Software Engineering.

Programming is just part of the software product lifecycle

Most of the times the term software development is used to replace software engineering

Development of Large Software Projects

…software engineering is concerned with the design of large software developments…

Software engineering is always a large scale endeavour which is maintained continuously…

A Systematic Process

A software engineer works in a more systematic way in order to develop a solution that will ultimately fill their needs

Software engineering is the artful way of creating a set of instructions

The study of a problem in order to come up with the most effective solution…

Brian Randell's Answer

The multi-version development of multi-version programs

Source: David Parnas, "Software Engineering: Multi-Person Development of Multi-version Programs"

Fred Brook's Answer

- Integration
- Productizing

Source: David Parnas, "Software Engineering: Multi-Person Development of Multi-version Programs"

At the 40th International Conference on Software Engineering (ICSE 2018)

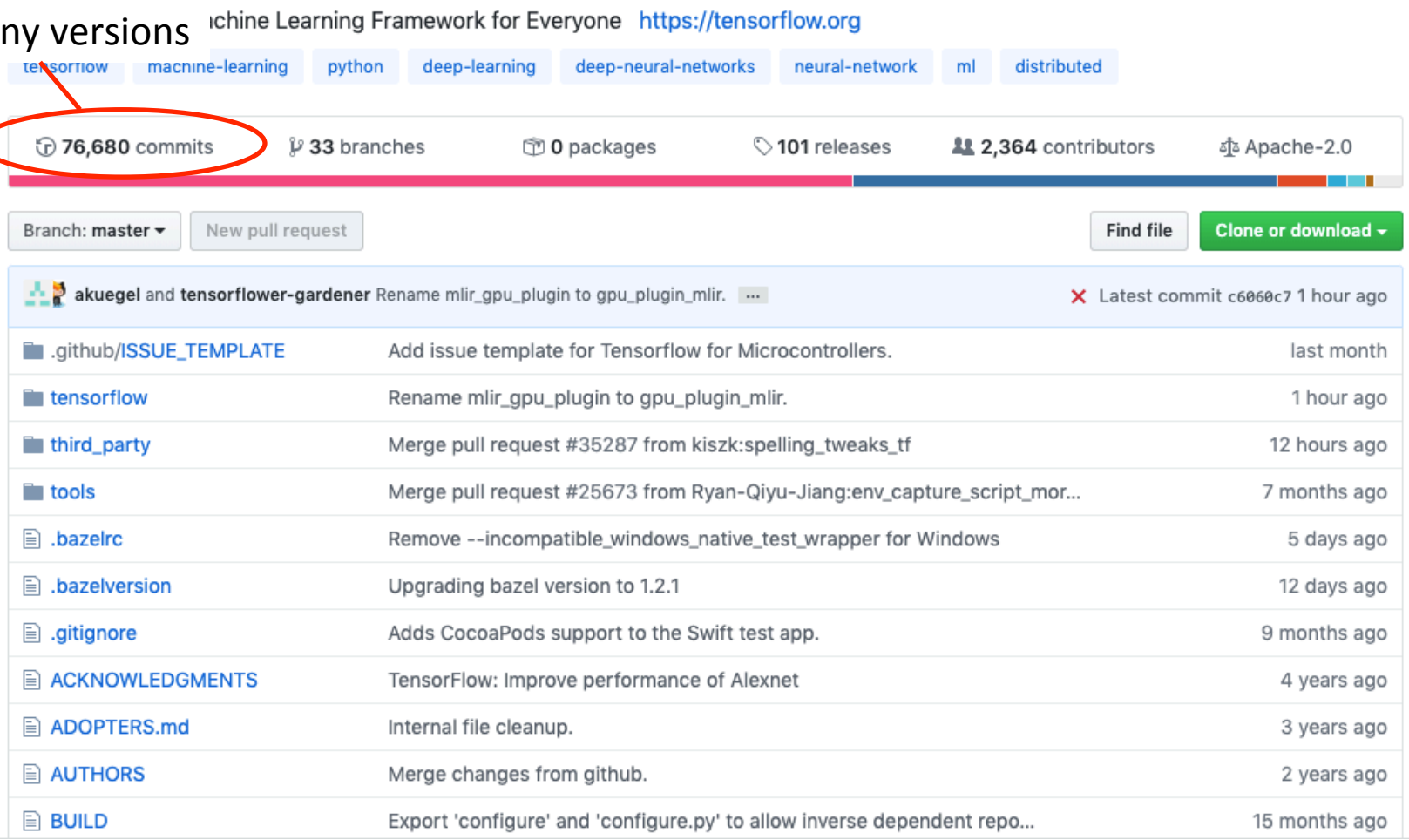
A Little Exercise

Go to this website

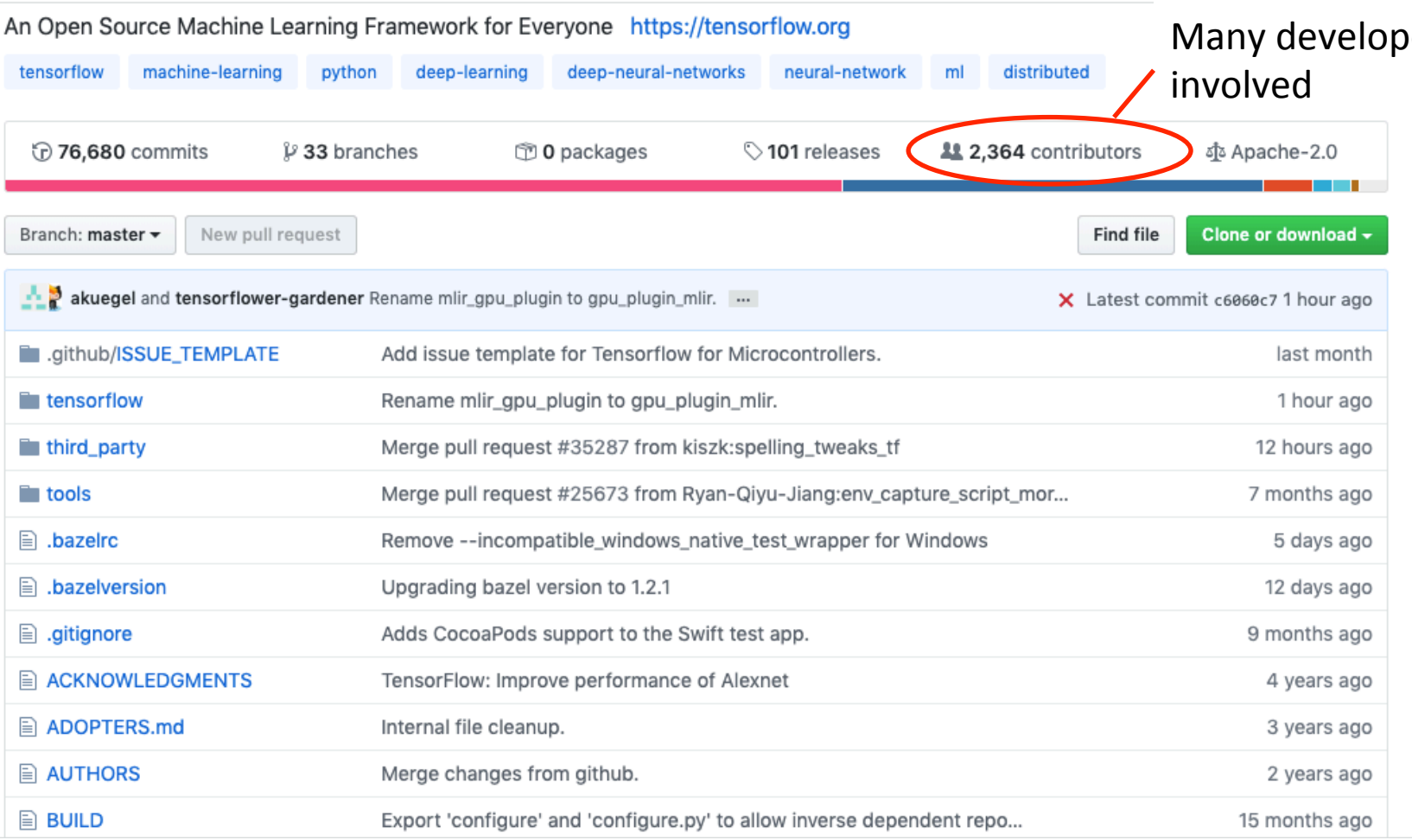
https://github.com/tensorflow/tensorflow

A Closer Look to Tensorflow Project

Many versions

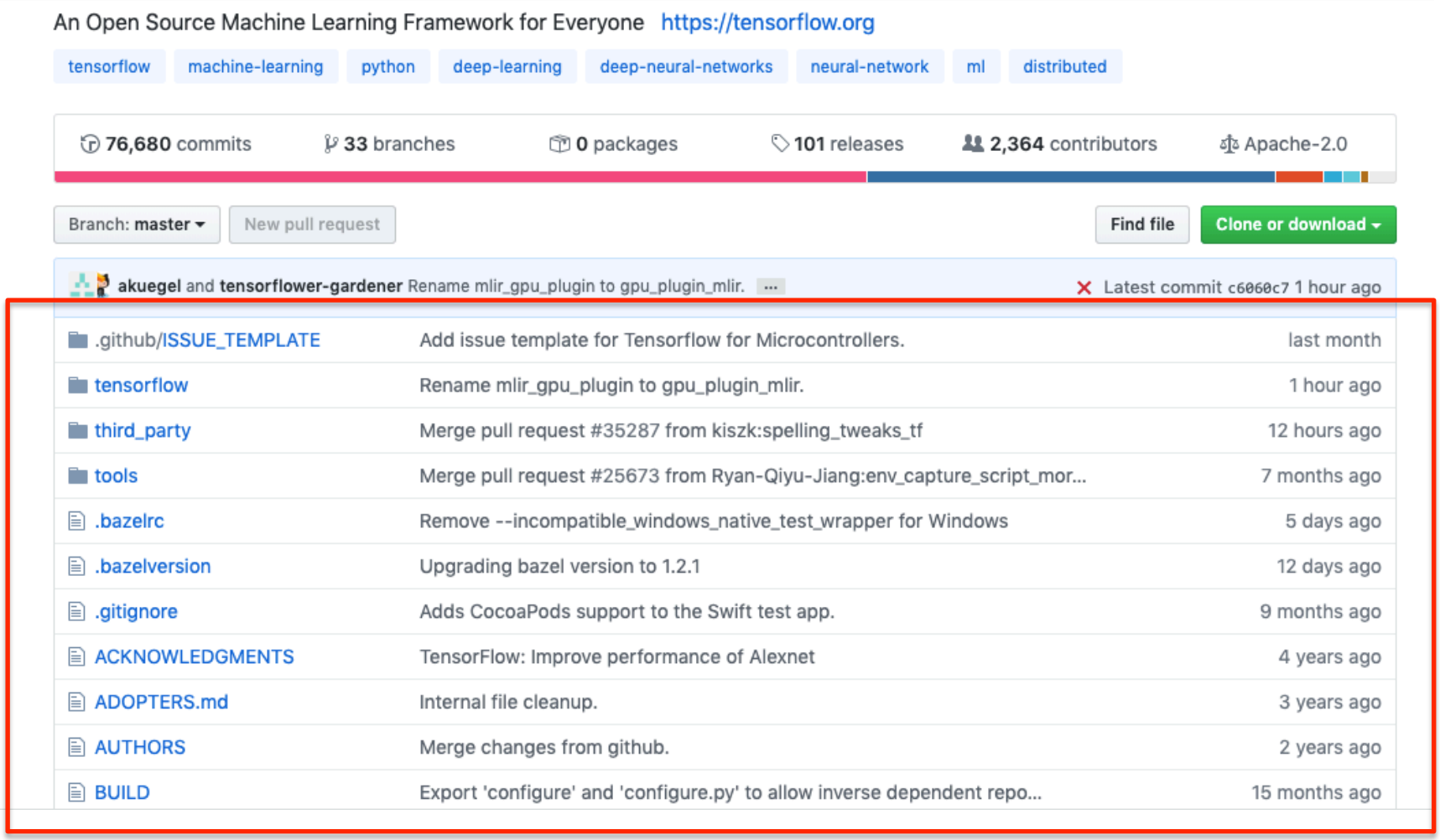


A Closer Look to Tensorflow Project



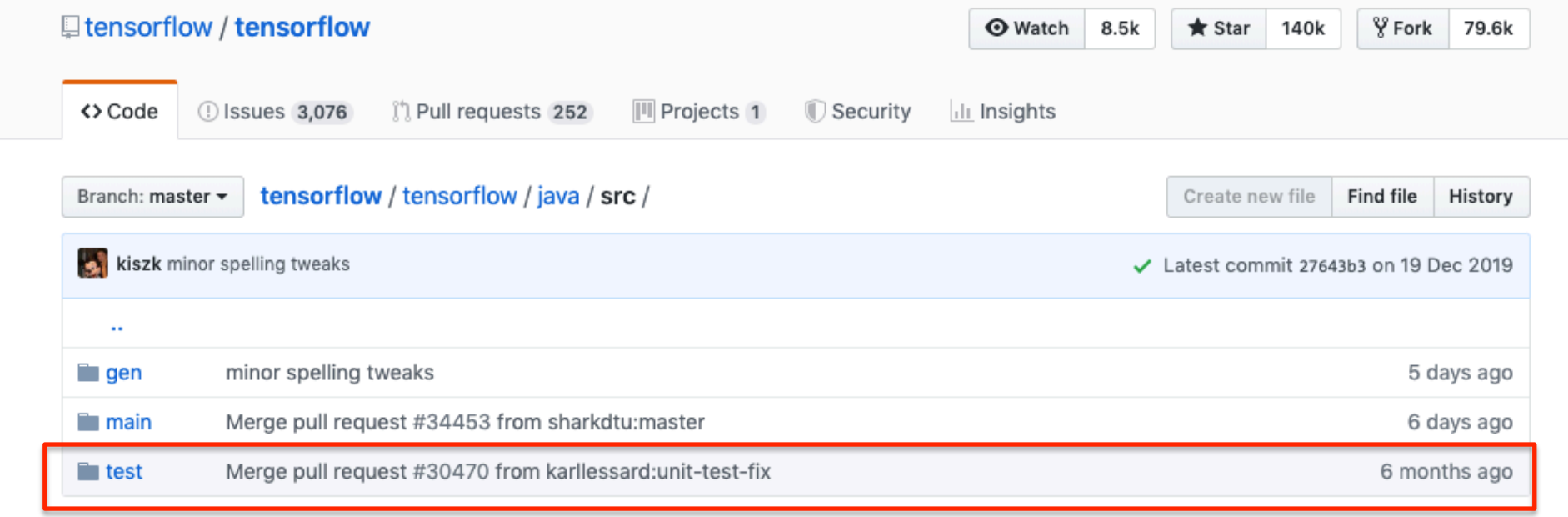
Many developers involved

A Closer Look to Tensorflow Project



Many files frequently updated

Go To tensorflow/java/src



All parts of the projects developed are also tested

Software Engineering

An engineering discipline that is concerned with all aspects of software production from the early stages of system specification to maintaining the system after it has going into use.

Software Engineering

*An **engineering discipline** that is concerned with all aspects of software production from the early stages of system specification to maintaining the system after it has going into use.*

- **Engineering discipline**
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints

Software Engineering

*An **engineering discipline** that is concerned with **all aspects of software production** from the early stages of system specification to maintaining the system after it has going into use.*

- **Engineering discipline**
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.
- **All aspects of software production**
  - Including project management: know how to work in teams, how to make schedules, set deadlines, estimate costs …

A bit of history…

- The notion of '*software engineering*' was first proposed at the NATO Software Engineering Conference in 1968 at Garmisch, Germany.
- The conference was held to discuss the 'software crisis'
  - difficulty in writing useful and efficient computer programs in the required time

… Afterwards a variety of software engineering techniques and methods were proposed

Software Engineering Activities

- Software Specification
- Software development
- Software verification & validation
- Software evolution

Software Engineering Activities

- **Software Specification**
  - Customers and engineers define the software that is to be produced and the constraints on its operation.
- **Software development**
- **Software verification & validation**
- **Software evolution**

Software Engineering Activities

- **Software Specification**
  - Customers and engineers define the software that is to be produced and the constraints on its operation.
- **Software development**
  - Where the software is designed and programmed
- **Software verification & validation**

- **Software evolution**

Software Engineering Activities

- **Software Specification**
  - Customers and engineers define the software that is to be produced and the constraints on its operation.
- **Software development**
  - Where the software is designed and programmed.
- **Software verification & validation**
  - **Verificatio**n: the software implementation satisfies its specification.
  - **Validation**: the software delivers what the customer requires.
- **Software evolution**

Software Engineering Activities

- **Software Specification**
  - Customers and engineers define the software that is to be produced and the constraints on its operation.
- **Software development**
  - Where the software is designed and programmed
- **Software verification & validation**
  - **Verificatio**n: the software implementation satisfies its specification
  - **Validation:** the software delivers what the customer requires.
- **Software evolution**
  - The software is modified to reflect changing customer and market requirements.

What will software engineers do? [1]

- Elicit and specify the **requirements** of a software system.
- Assess **risks** of a software development process.
- **Design** the basic structure and the behaviour of a software system.
- Implement **understandable, well-documented** and **reusable** software.
- Perform software **testing**.
- **Maintain** and **refactor** existing software
- Teamwork, meeting deadlines.

What will software engineers do? [1]

- Elicit and specify the **requirements** of a software system.
- Assess **risks** of a software development process.
- **Design** the basic structure and the behaviour of a software system.
- Implement **understandable, well-documented** and **reusable** software.
- Perform software **testing**.
- **Maintain** and **refactor** existing software
- Teamwork, meeting deadlines.

Which Tools Do Software Engineers Use To Develop Code?

They use a tool called Integrated Development Environment
(IDE)

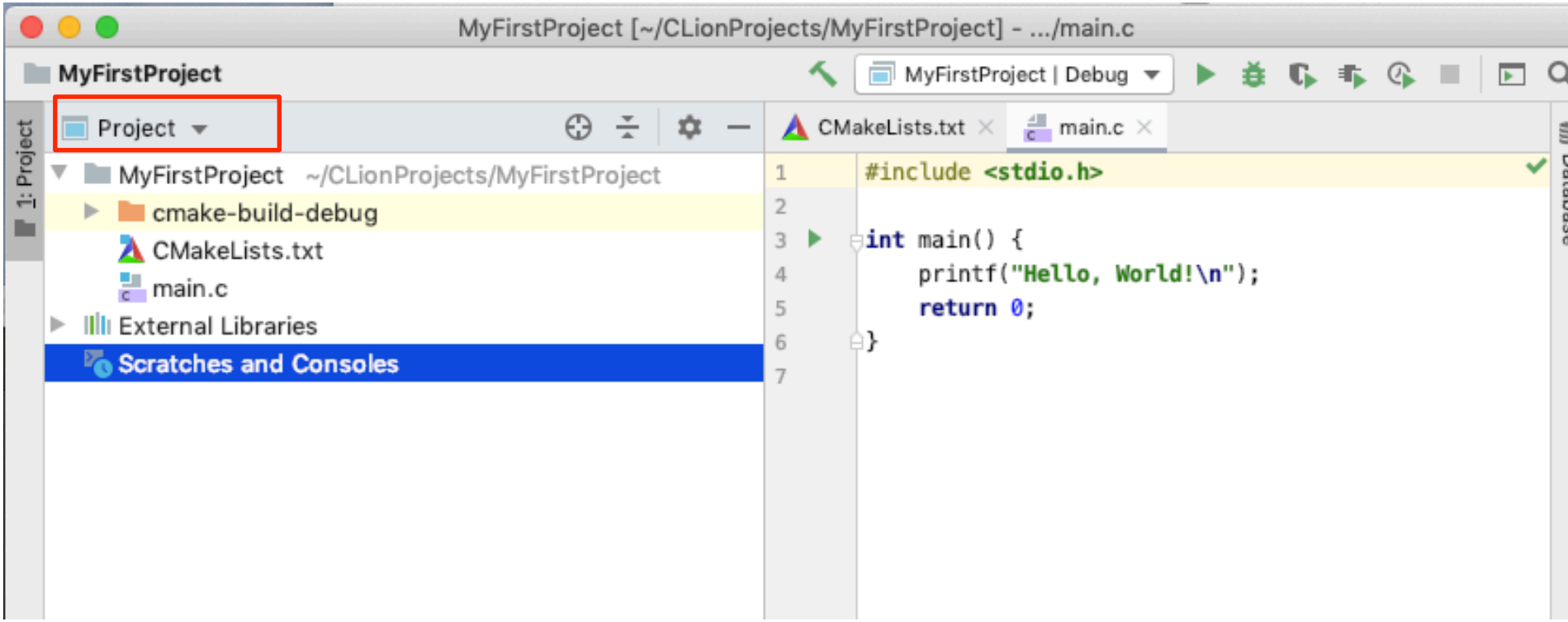
…Visual Studio, CLion, Eclipse, Netbeans, …

CLion

- An open-source software development environment comprising an IDE and a plug-in system to extend it.
- Used for building, deploying and managing software across its lifecycle.
- Product description available at:
  https://www.jetbrains.com/clion/
- Apply for a Free student pack to download it for free: https://www.jetbrains.com/community/education/

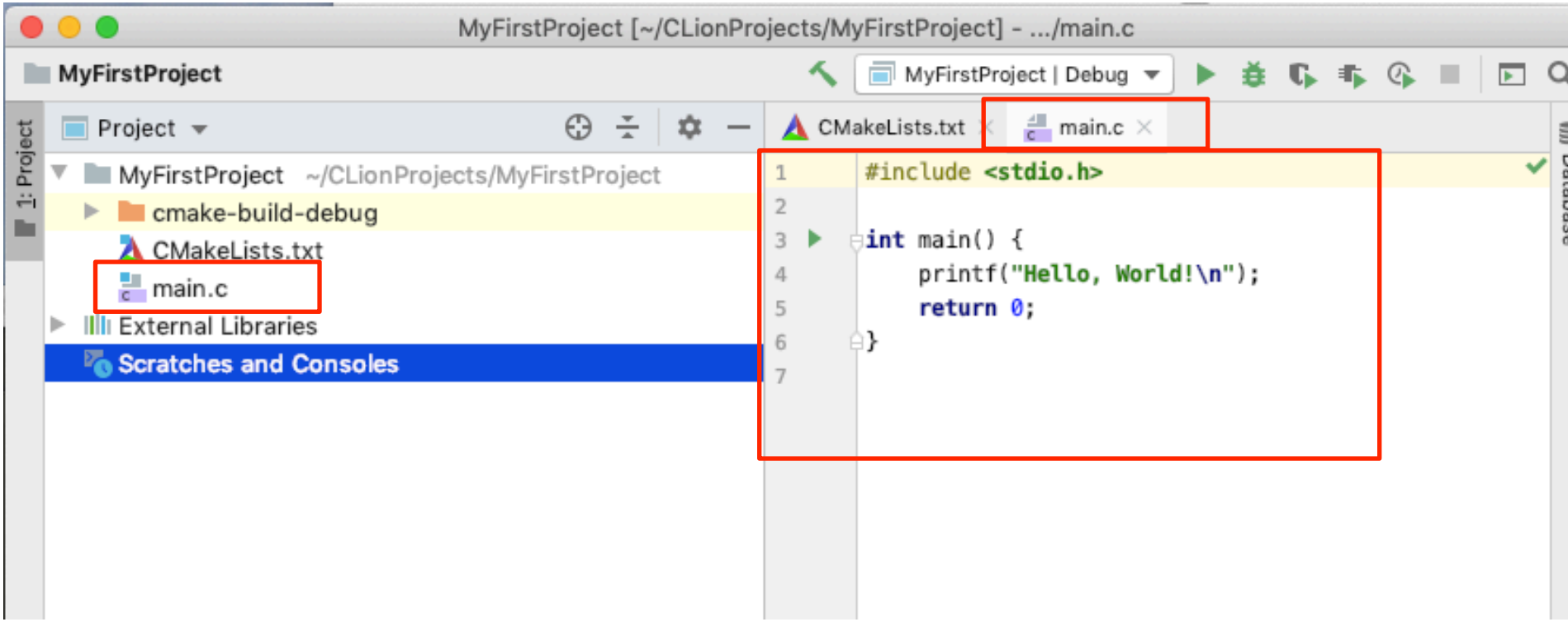
CLion- Basics

- **Project:** a collection of one or more **source** (as well as **header**) files



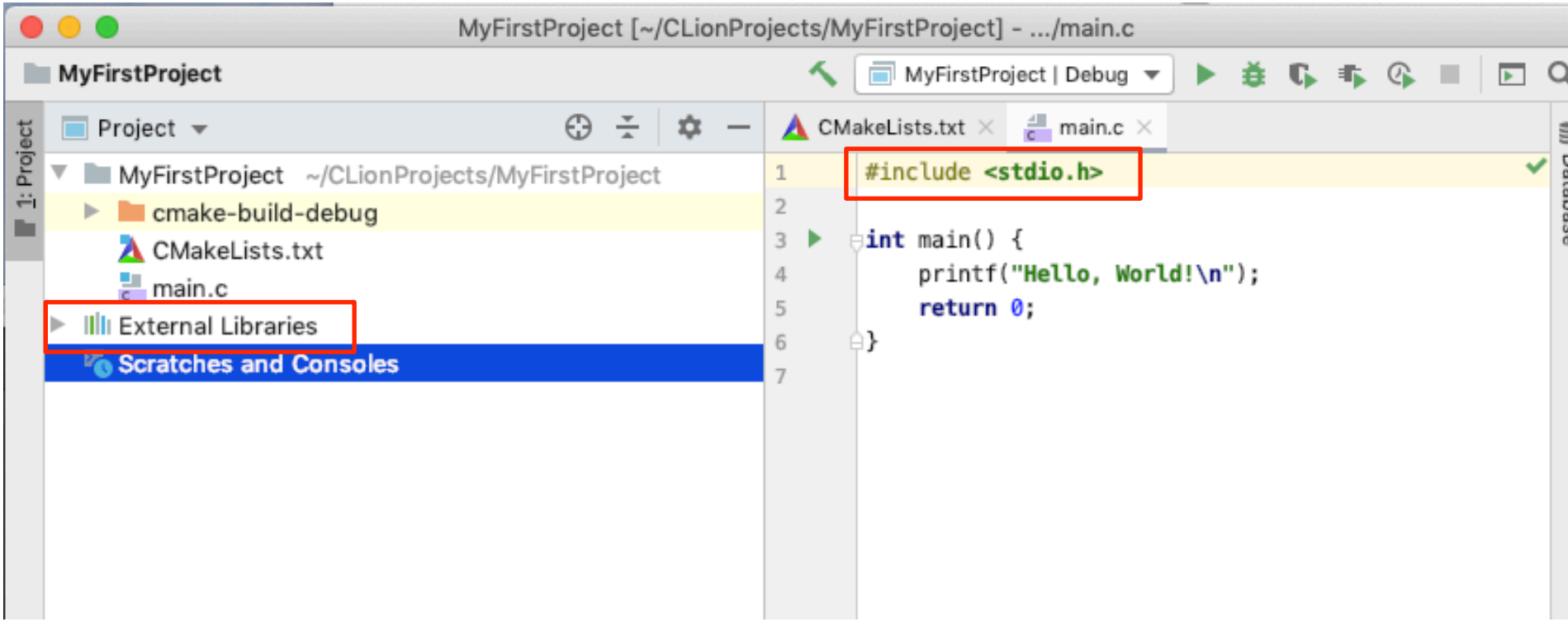
CLion- Basics

- **Source**: Contains the files with the source code of your program (.c file)

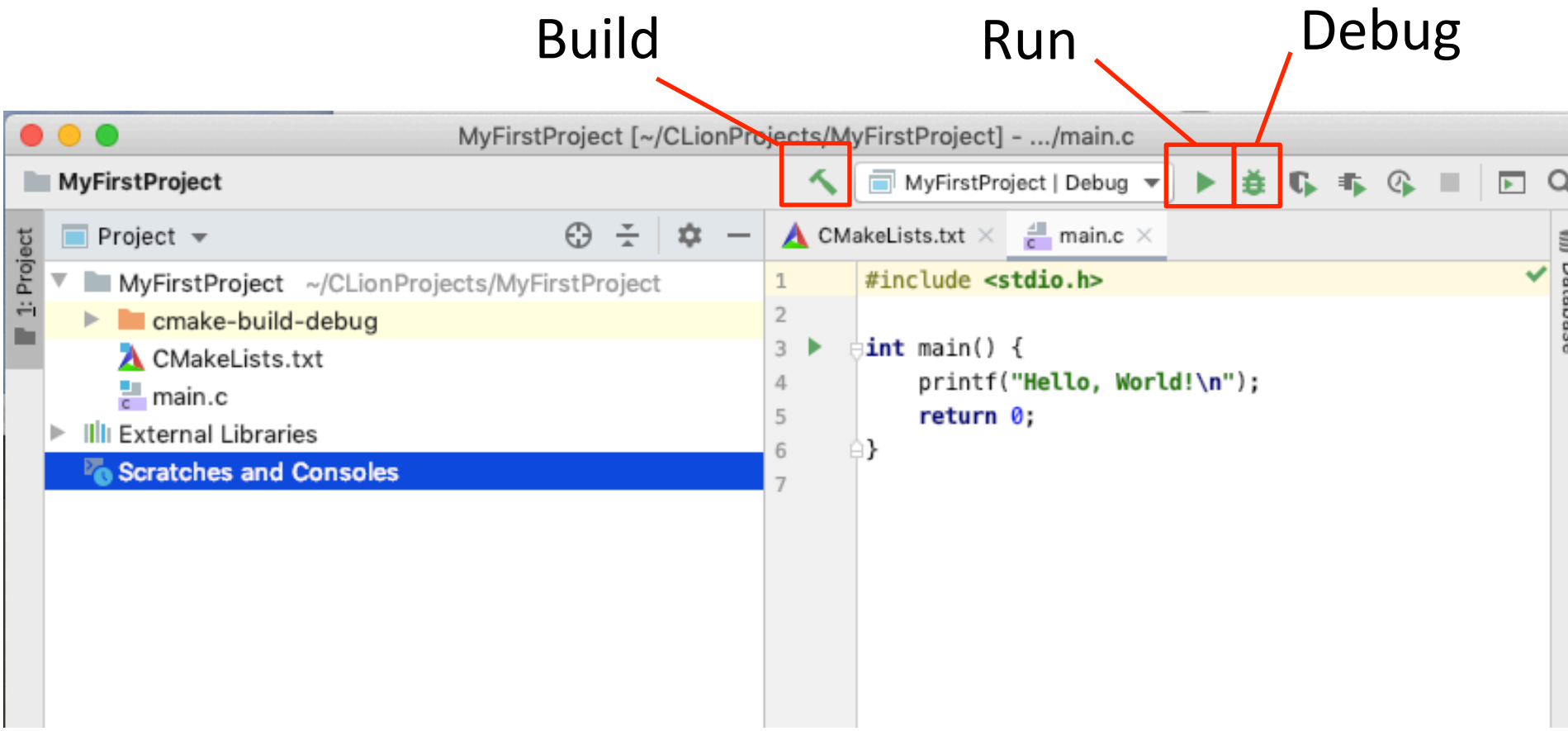


CLion- Basics

- **Header**: Contains library files (.h files). A library is a collection of functions that are called to perform specific tasks, such as doing math, etc.



Compiling and Running Projects



Software Engineers Also Write “Clean Code”

A Few Tips to Write Clean Code

Naming conventions

- Easy to understand what the variable represents.
- Easy to understand the variable scope.

Source: https://www.codeschool.com/blog/2015/09/29/10-ways-to-write-cleaner-code/

Naming Conventions Examples

```
char name[] = “George Orwell”
```

```
char fb_username [] = “George Orwell”
```

```
//constant values are in all caps and
//separated with underscores
  int I_WINDOW_SIZE =900;
```

Source: https://www.codeschool.com/blog/2015/09/29/10-ways-to-write-cleaner-code/

A Few Tips to Write Clean Code

Say What You Mean

```
int checkNum(int n)
{
    if(n<max){
       return -1;
    }
    else return 1;
}
```

```
int MIN_VALUE = 5;

int valueIsTooLow(int value)
{
    if(value < MIN_VALUE)
       return -1;
    else return 1;
}
```

Source: https://www.codeschool.com/blog/2015/09/29/10-ways-to-write-cleaner-code/

A Few Tips to Provide Comments

- Program comments are explanatory notes for the humans reading a program.
- Why so important?
  - Fundamental software engineering practice critical for efficient software development
  - Acts as a specification of behaviour for other engineers who might need to read, understand or modify your code.
- When should you add comments?
  - At the beginning of each C Source file
    - To explain the overall purpose of the program. 2-3 sentences will suffice.
  - Just before each method declaration
    - To explain its purpose and any method parameters
  - When your code is unusual and obscure
    - When something is important and not obvious, it merits a comment.

Source: Previous year material by Prof. P Cunningham

Module Organization

Schedule

- **Lectures:**
  - One lecture per week. Tuesdays, 13:00-13:50
  - **Location:** H1.37

- **Labs:**
  - Fridays, 11:00-12:50
    - Room B108 (ART) → Surnames Aldabous to Davidov
    - Room B109 (ART) → Surnames Dempsey to Zhao

Objectives

- To introduce you to the field of **software engineering**

Objectives

- To introduce you to the field of **software engineering**

- Gain experience of some key software development practices
  - Good programming conventions
  - Software development (in C)

Objectives

- To introduce you to the field of **software engineering**

- Gain experience of some key software development practices
  - Good programming conventions
  - Software development (in C)
  - Using an IDE –> CLion

Objectives

- To introduce you to the field of **software engineering**

- Gain experience of some key software development practices
  - Good programming conventions
  - Software development (in C)
  - Using an IDE –> CLion
  - Distributed version repositories → Git

Objectives

- To introduce you to the field of **software engineering**

- Gain experience of some key software development practices
  - Good programming conventions
  - Software development (in C)
  - Using an IDE –> CLion
  - Distributed version repositories → Git
  - Testing
  - Teamwork

Outline of the Module

- **Part 1:** Arrays, Strings, I/O and Basic Algorithms
- **Part 2:** Git, Basic Structures, Pointers, Dynamic Data Structures
- **Part 3:** Testing

Outline of the Module

- **Part 1:** Arrays, Strings, I/O and Basic Algorithms
  - Individual Assignment (40%)
  - Deadline: Feb 28
- **Part 2:** Git, Basic Structures, Pointers, Dynamic Data Structures
  - Group Assignment (40%)
  - Deadline: Apr 21
- **Part 3:** Testing
  - Individual Assignment (15%)
  - Deadline: May 1

Enrollment on Brightspace

- You should be automatically enrolled in the module on Brightspace.
- Lecture notes will be posted immediately before or after each lecture
- Assignments and any other material will be posted as we progress through the module

Announcement will be made on Brightspace, so check the module page *frequently*!

A Few Advices…

- **Attend lectures and labs:**
  - LABs ARE MANDATORY and attendance contributes to 5% of the mark
- **Make sure your assignment does not have compilation errors**
  - Results in 0 marks for the assignment in question
- **Submit your assignment on time**

… And Some More Advices

- Don’t be shy: if there is something you don’t fully understand
  - Ask demonstrator of your table
  - Discuss with others at your table
  - Use the “Module Questions” forum on Brightspace
- Use Google
  - Get familiar with online resources (social coding)
    - Stackoverflow, cppreference, wikihow

Evaluation of the Module

For each assignment you are asked to

- Use the CLion Software Development Environment
- Write clean code
- Separate your code in various modules and files wherever possible
- Use a distributed version control system to work collaboratively
  - Only for assignment 2

Specific requirements for each assignment will be provided in due course.

Assessment Submission

- Deliverable deadlines are on the main module page
- Late submissions
  - NOT by e-mail
  - Brightspace assignments will remain open for submission after due dates to accommodate late submissions, but penalties will apply:
    - Up to 1 week late: 10% deduction
    - Up to 2 weeks late: 20% deduction
    - More than 2 weeks late: 50% deduction

Plagiarism & UCD Computer Science

- **Plagiarism is a serious academic offence**
  - [Student Code, sections 6.2 & 6.3] or [UCD Registry Plagiarism Policy] or [CS Plagiarism policy and procedures]
- Our staff/demonstrators are **proactive** in looking for possible plagiarism
- Suspected plagiarism is investigated by the CS Plagiarism subcommittee
  - Usually includes an interview with student(s) involved
  - 1st offence: **usually** 0 or NG in the affected components
  - 2nd offence: more serious consequences e.g. UCD Disciplinary process
- Student who *enables* plagiarism is *equally responsible* for it
- **Examples** of plagiarism:
  - Copying the files of another student and submitting them as your own work
  - Copying some/all of an assignment from the Internet/book/etc without referencing it
  - Sharing images of your work with another student (by e-mail, FB messenger, WhatsApp, ...)
  - A group of students working on a solution, then individually submitting the same work
  - Students collaborating at too detailed a level e.g. consulting each other after implementing a line/block/segment of code and sharing the results

What did we Learn Today?

- Difference between software engineering and software development
- What Software Engineers Do
- Integrated Development Environment (CLion)
- Writing clean code
- Module Organization