Course Information

2023-2024
COMP2113B/C Programming Technologies / ENGG1340B/C Computer Programming II
Dr. Chenxiong Qian & Dr. T.W. Chim (E-mail: cqian@cs.hku.hk & twchim@cs.hku.hk)
Department of Computer Science, The University of Hong Kong

Teaching Team

	Team A	Team B
Instructors	Dr. Chenxiong Qian (cqian@cs.hku.hk)	Dr. T. W. Chim (twchim@cs.hku.hk)
Teaching Assistants	Mr. Kevin Y.K. Lam (yklam2@cs.hku.hk)	Mr. C.K. Lai (cklai@cs.hku.hk)
	Mr. Yuping Gu (ypgu@cs.hku.hk)	Mr. Sheng Wang (swang@cs.hku.hk)
	Mr. Xiaoru Dong (xrdong@cs.hku.hk)	Mr. Hongxiao Wang (hxwang@cs.hku.hk)

- More details can be found on Moodle.

Teaching Team

Student Teaching Assistants (sorted by surname):

- Cao Nuanyu, caorain@connect.hku.hk
  Cao Rui, caorui04@connect.hku.hk
- Chan Chi Wing, u3606528@connect.hku.hk
- Chan Hin Ting, jasoncht@connect.hku.hk
- Chau Ming Chun, oloollol@connect.hku.hk
- Chen Chi, Charliechen233@outlook.com
- Chiu Long Pak, pak123@connect.hku.hk
- Chow Pak Hang, u3578792@connect.hku.hk
- Chui Wai Chun Victor, victorc3@connect.hku.hk
- Hicks Daniel Lee, danielleehicks08@gmail.com
- Hu Yueyue, yy10086@connect.hku.hk
- Khan Ahmad, u3610067@connect.hku.hk
- Kwan Ho Hin, kkenkwan@connect.hku.hk
- Liu Hao Wei, u19liuh2@connect.hku.hk
- Lo Ian Leong Ting, iltlo@connect.hku.hk
- Ma Tin Yu, malun510@connect.hku.hk
- Nathalie Karina, karinath@connect.hku.hk
- Poon Ho Ting, u3580090@connect.hku.hk
- Qin Suyue, yy139515@connect.hku.hk
- Rajiv Arnav, u3570905@connect.hku.hk
- Shetty Siddhanth, sid2310@connect.hku.hk
- Sun Shengyao, fd2374@connect.hku.hk
- Wang Yam Yuk, u3606601@connect.hku.hk
- Xu Shuyang, xsy27@connect.hku.hk
- Yang Haozhe, yhz2004@connect.hku.hk
- Yang Mingtian, skylee03@connect.hku.hk
- Yiu Ka Chun, kcyiuac@connect.hku.hk
- Zhou Zihan, u3594784@connect.hku.hk
- Zhuang Keju, u3598820@connect.hku.hk
- Ziya Shaheer, shaheer@connect.hku.hk

What are you expected to know?

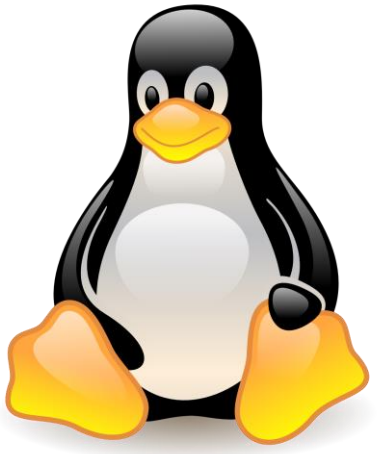
- **Programming basics in Python.**
  - Syntax, identifiers, control statements, functions, recursions and strings.
  - Programming style and program documentation
  - Lists, dictionaries, tuples and files

What this course is about?

- You'll learn about the various software development technologies and tools that a **competent computer science professional** should know and be good at.
- To prepare you with **a solid programming skill and background** to cope with the upcoming challenges in the fundamental and advanced courses in the Computer Science curriculum
- To get you start developing professional practice in coding

What will you learn?

- **Outcome 1. [Programming Environment and Technologies]**
  Able to work comfortably on the Linux platform and utilize its basic functionalities for program development.
  - Introduction to Linux and the Bash shell
  - Shell scripts



What will you learn?

- **Outcome 2. [Problem Solving and Program Implementation]** Able to understand and analyze a problem and implement appropriate solutions correctly using C/C++. Able to utilize debuggers and techniques like separate compilation and make files to simplify and speed up the development.

Motivating question:
What if I am developing a large-scale program which consists of **20 classes**, and I just updated the code of one class only, do I need to recompile all my code?

What will you learn?

- **Outcome 3. [Advanced Programming Techniques]**
  Be able to understand and apply the principles of advanced programming techniques including recursion, C++ classes, STL, data structures and algorithms.

What will you learn?

- **Outcome 4. [Self-learning]**
  Be able to self-learn various programming techniques effectively.

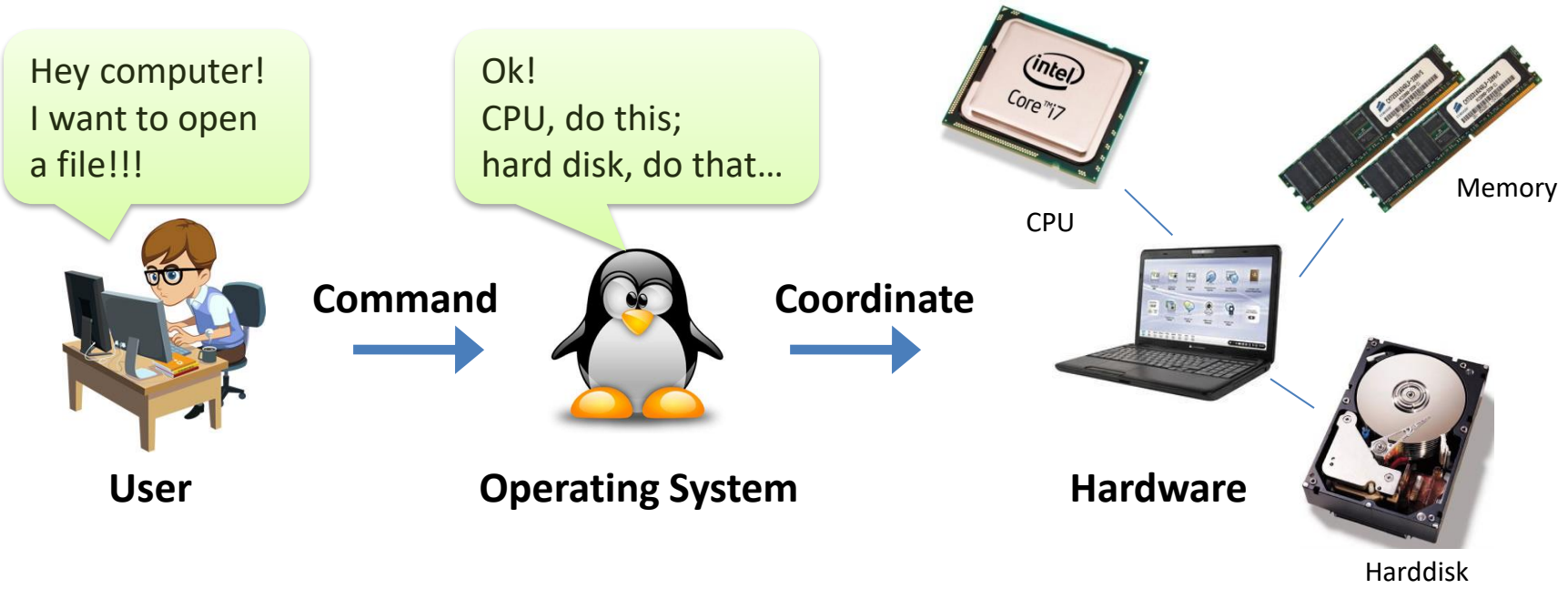
Module	Topic
0	Getting Started with Logging onto CS Servers
1	Linux Environment
2	Shell Script + Version Control
3	C/C++ Basics
4	Makefile, Programming Style, Basic Debugging, Python/C++ Comparison
5	Functions
6	Arrays & Strings
7	File I/O, Structures & Recursion
8	Pointers & Memory Management
9	C++ Standard Template Library
10	C Specifics and GDB Debugger

Why learning C?

- The C language is a predecessor of C++.
- We learn this older language mainly for two reasons.
  - C is a more low-level language, and less functionality is provided. E.g., there is no classes or STL. In return, **the programs are usually faster** than their C++ counterpart.
  - A lot of existing systems like **Linux and Unix use C extensively**. For the operating systems and networking area, **many libraries are available in C only**.

What is Linux?

- Linux is an **operating system (OS)** like Microsoft Windows or Apple OS X.
- OS provides an **interface** for us to use the hardware components.



What is Linux?

- There are many different distributions of Linux. They deviate from each other mainly on the interfaces and tools provided.

- This course: Ubuntu (our departmental server is running Ubuntu)

Why learning Linux?

- Windows and OS X are probably more commonly known by the general public. However, Linux is an important operating system for several reasons.
  - Linux is **open-source** software. Hence, we can customize it for new or special hardware systems.

  - It provides a higher level of **security** because a company can inspect its source code.
  - Widely used for supercomputers. http://www.top500.org/lists/2012/11/

Titan (supercomputer)

Different ways of using Linux

- Making your computer dual-boot Windows & Linux
  - http://www.applegazette.com/mac/how-to-install-ubuntu-and-dual-boot-macos/
  - Drawback: Risky and troublesome!
- Running a virtual machine on Windows / MacOS and then run Linux inside the virtual machine
  - https://microchipdeveloper.com/linux:install-virtual-machine
  - Drawback: The virtual machine will share the same set of memory with your Windows / MacOS and so your computer will be slower.
- Connecting to our “academy11.cs.hku.hk” / “academy21.cs.hku.hk” Linux servers using “SSH” (text mode)
  - Please refer to Module 0 for details.
  - Drawback: Text-based only (but enough for our course!)
- Note: For the last approach, you need to use FTP software like “FileZilla” to transfer files to and from academy servers. For details, please refer to Module 0.

Course Format

- Self-learning course

This is the only lecture in the course!!!

- **Support sessions** (starting from Jan 22)
  - Monday 1230-1420 (ENGG1340B)
    - Venue: CPD-LG.34 / CBA
  - Tuesday 1030-1220 (COMP2113B)
    - Venue: CBA
  - Tuesday 1330-1520 (ENGG1340C)
    - Venue: CBA / KB223
  - Thursday 1630 -1820 (COMP2113C)
    - Venue: CBA
  - The sub-classes listed are arranged by the Faculty. You can join ANY session.
  - Optional, not compulsory but recommended if you find difficulties catching up or in tackling review exercises/checkpoint tasks/assignments. Students may also be asked to show up in case of unsatisfactory performance.

- Module-based with progress check

Why self-learning?

- CS experts should be able to **adapt to a fast-changing world of technology**.
- The **ability to self-learn** various programming languages efficiently is a necessary trait of our graduates.

Why self-learning?

- Is learning a new programming language difficult?

```cpp
// This is a sample C++ program
#include <iostream>
using namespace std;
int main() {
	cout << "Hello World!" << endl;
	return 0;
}
```

hello.cpp

```c
// This is a sample C program
#include <stdio.h>
main() {
	printf("Hello World!");
}
```

hello.c

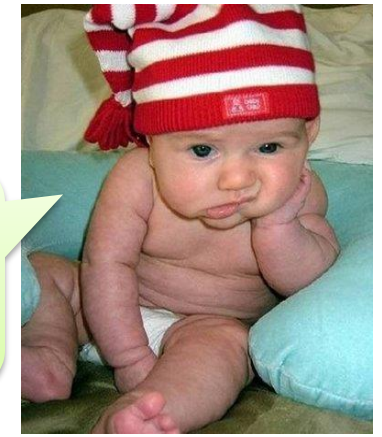
```sh
#!/bin/bash
# This is a sample shell script
echo "Hello, World!"
exit 0
```

hello.sh

```php
<?php
// This is a sample PHP program
echo "Hello, World!";
?>
```

hello.php

I don't need a lecture for learning these syntax because the lecturer just read out the code!



Tentative schedule only, check Moodle announcement for official release and due dates

Team A Team B

Week	Date	Module	Assessment Task
1	Jan 15 – Jan 20	M0: Getting Started with Logging onto CS Servers (Due: Jan 27)	
2	Jan 22 – Jan 27	M1: Linux Environment (Due: Feb 3)	
3	Jan 29 – Feb 3	M2: Shell Script + Version Control (Due: Feb 10)	
4	Feb 5 – Feb 10	M3: C/C++ Basics (Due: Feb 24)	Release of A1 (Due: Feb 24)
5	Feb 19 – Feb 24	M4: Makefile, Programming Style, Basic Debugging, Python/C++ Comparison (Due: Mar 2)	
6	Feb 26 – Mar 2	M5: Functions (Due: Mar 9)	Release of A2 (Due: Mar 16) Release of Project (Due: Apr 27)
7	Mar 4 – Mar 9	Reading Week	
8	Mar 11 – Mar 16	M6: Arrays & Strings (Due: Mar 23)	
9	Mar 18 – Mar 23	M7: File I/O & Recursion (Due: Mar 30)	
10	Mar 25 – Mar 30	M8: Pointers & Dynamic Memory Management & Linked List (Due: Apr 6)	Release of A3 (Due: Apr 13)
11	Apr 1 – Apr 6		
12	Apr 8 – Apr 13	M9: Standard Template Library (Due: Apr 20)	
13	Apr 15 – Apr 20	M10: C Specifics and GDB debugger (Due: Apr 27)	Release of A4 (Due: May 4)
14	Apr 22 – Apr 27		

Modules

- 10 modules, each lasting for 1 or 2 weeks
- Learning materials with clear objectives will be provided
  - Notes
  - Readings & references
  - Checkpoint tasks (← need to complete and submit)
- You will need to complete module checkpoints tasks before a given deadline for each module
- Learning materials will be released on the Friday before the week and the checkpoint tasks will be due on the following Saturday.

Assessment

- Continuous Assessment (70%)
  - Checkpoint Task Submissions [10%]
  - 4 Programming Assignments [40%]
  - 1 Programming Project [20%]
- Final Written Examination (30%)

Assignments

- Programming assignments which you should **complete individually** and submit by the deadline
- About 2 weeks to complete (try not to be deadline fighters!!!)
- **Read assignment instructions carefully.**
- Deadlines are strictly enforced.
  - **Late Policy: If submit within 3 days after the deadline, 50% deduction. After that, no mark.**
- Discussions of materials and problems are encouraged. But make sure that your submission is your own work.

Assignment Grading

- No credit for a program that cannot compile/run in a given standardized environment (you will learn about it later).
- Plagiarism detection software will be used in the grading of every assignment.
- Your code will be auto-graded for technical correctness. Sample test cases are provided but additional test cases are used for grading.
- We have rebuttal mechanism. This is just to cater for possible technical errors, and you will have to do that within a specific time period. Stay tuned for every assignment.

Plagiarism Policy

- As defined in the University's Regulations Governing Conduct at Examinations, plagiarism is the **unacknowledged use, as one's own, of work of another person, whether or not such work has been published**. Or put it simply, plagiarism is **copying the work of another person without proper acknowledgement**. In case of queries on plagiarism, students are strongly advised to refer to "What is Plagiarism?".
- **First Attempt**: Students who admit committing plagiarism for the first time shall be warned in writing and receive a **zero mark for the component concerned**. For those who do not confess, the case would be referred to the Programme Director for consideration.
- **Subsequent Attempt**: If students commit plagiarism more than once during the course of studies, the case shall be referred to the Programme Director for consideration. The Programme Director will investigate the case and consider referring it to the University Disciplinary Committee, which may impose any of the following penalties: **a published reprimand, suspension of study for a period of time, fine, or expulsion from the University**.

Project

- You will work in a group of 2 – 5 students, but grading will be done on individual basis.
  - Individual contribution tracked by GitHub
  - Peer review
- You will work on the project for 6 to 7 weeks.
- You will need to come up with specific ideas about a given topic.
- At the end, you would have finished a complete project which incorporates the skills you learned from this course.
- More details to come.

What do we expect from you?

- Be responsible for your learning progress
  - **It is expected that you spend roughly 8 – 10 hours per week for this course.**
- Check Moodle frequently
- Follow course schedule
- **Read instructions carefully**
- Seek help whenever necessary
- Actively engaging in discussions (especially in discussion forums)
- Be eager to become a competent computer science professional

← This is the only official direct communication channel for the course

What kind of support do you have?

- Moodle online Q&A and discussions
- Student TA support sessions
- Consultation hours (by appointment)
- Additional TA / student TA support

Getting effective help

What **NOT** to do when seeking help

- Ask without doing some research if there are already answers to similar issues

  Be well-prepared before you ask
- Merely throw your code at instructor / TAs / STAs and hoping that they will debug for you
- Post your assignment code or answers to Moodle

Getting effective help

- Highly recommended to have online communications on Moodle so everyone can share and learn
- How to ask?
  - Be specific

    Bad example
    I don't understand.

    Good example
    Why isn't this variable effective inside this loop?
  - Provide the context / background

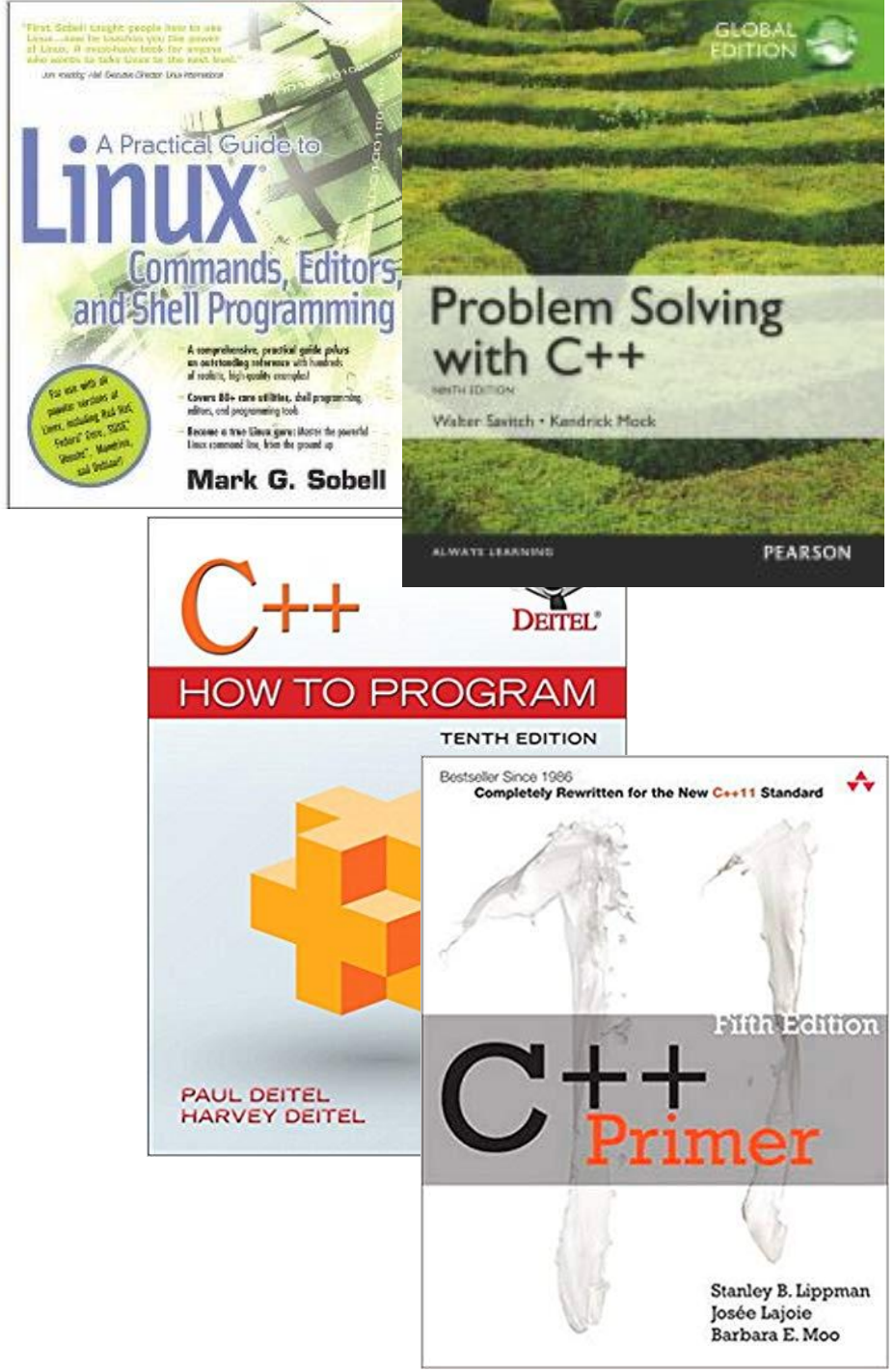
    Bad example
    It doesn't work.
    Why?

    Good example
    I got this error when compiling my code with this Makefile

⇨ How do I ask a good question? (stackoverflow)

References

- A Practical Guide to Linux: Commands, Editors, and Shell Programming
- Online Linux tutorial (by Matt Welsh et al)
  - http://tldp.org/LDP/gs/node1.html
- Problem Solving with C++
- C++ How to Program
  - Electronic version available from HKU library https://proquestcombo-safaribooksonline-com.eproxy.lib.hku.hk/9780133378795
- C++ Primer
- A point to note: we will teach C++ and C
  - Strings (C-string vs. string class)
  - Standard I/O (printf/scanf vs. cin/cout)
  - file I/O (FILE * vs. fstream)
  - Memory management (malloc/free vs new/delete)



Programming Languages

Worldwide, Aug 2023 :

Rank	Change	Language	Share	1-year trend
1		Python	28.04 %	+0.3 %
2		Java	15.78 %	-1.3 %
3		JavaScript	9.27 %	-0.2 %
4		C#	6.77 %	-0.2 %
5		C/C++	6.59 %	+0.4 %
6		PHP	5.01 %	-0.4 %
7		R	4.35 %	+0.0 %
8		TypeScript	3.09 %	+0.3 %
9	↑↑	Swift	2.54 %	+0.5 %
10		Objective-C	2.15 %	+0.1 %
11	↑↑	Rust	2.14 %	+0.5 %
12	↓↓↓	Go	1.93 %	-0.2 %
13	↓	Kotlin	1.77 %	-0.0 %
14		Matlab	1.63 %	+0.1 %
15	↑↑↑↑	Ada	1.08 %	+0.3 %

Reference: http://pypl.github.io/PYPL.html

You have learned it in COMP1117 / ENGG1330!

You will learn in COMP2396!

Useful in webpages development

Useful in game development (e.g., Unity)

You will learn in this course!

Useful in server program development

Useful in iOS apps development (Xcode) [Getting popular]

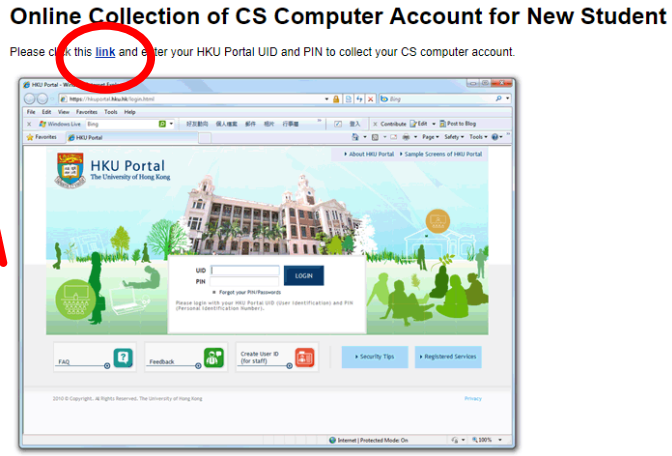
Useful in iOS apps development (Xcode) [More mature]

Useful in Android apps development (Android Studio) [Getting popular]

To learn Java, please take COMP2396 Object-oriented Programming and Java.
To learn Android apps development, please take COMP3330 Interactive Mobile Application Design and Programming.
To learn game development, please take COMP3329 Computer Game Design and Programming.
To learn web application development, please take COMP3322 Modern Technologies on World Wide Web.

CS Computer Account

- In order to login computers in our student labs, you need a CS computer account.
- To get one, please visit our CS homepage (http://www.cs.hku.hk).
- Click “Intranet”.
- Click “New Student”.
- Login using your HKU Portal account.
- You should see your CS username, password and a door PIN (for accessing non-public CS labs).



Let’s start !

C++

Class

Overloading

C

Bash command

grep

Shell script

STL

GIT

Python

Iterator

regex

PHP

Redirection

makefile

PIPE

Practice makes perfect!

End

2023-2024
COMP2113B/C Programming Technologies / ENGG1340B/C Computer Programming II
Dr. Chenxiong Qian & Dr. T.W. Chim (E-mail: cqian@cs.hku.hk & twchim@cs.hku.hk)
Department of Computer Science, The University of Hong Kong