

CSC3002

Introduction to Computer Science:
Programming Paradigms

Kinley Lam
SSE, CUHK(SZ)
Fall, 2022

Lecture 0: Introduction to CSC3002



Who? – The teachers

- Instructor: Kinley Lam (kinleylam@cuhk.edu.cn)
 - Office: DY 214 (Sept), CD411 (After Sept)
 - Hour: Fri, 11:00-12:00
- TAs:
 - WANG, Rulan (219019056@link.cuhk.edu.cn), RB205, Wednesday 15:00-16:00
 - SONG, Qi (221019037@link.cuhk.edu.cn), Zhiren502, Friday 15:30-17:00
 - CHEN, Jiwei (221019093@link.cuhk.edu.cn), Zhiren502, Friday 14:00-15:30
 - WANG, Wenyu (220019177@link.cuhk.edu.cn), LeTian101, Monday 20:00-21:00

Who? – The students

- For everyone who wants to take this course, you must understand the following:
 - Pre-requisites:
 - CSC1001 Introduction to Computer Science: Programming Methodology (and preferably also the lab course CSC1002)
 - Other courses that might help:
 - CSC3001 Discrete Math
 - EIE2050 Digital Logic and Systems
 - Other courses that might have an overlap with this one:
 - CSC3100 Data structures
 - CSC4120 Design and Analysis of Algorithms
 - Willingness to spend **a lot of time** in this course!

Who? – The students

- For applications from non-target students to add this course, I will take into consideration the chance of you taking this course in the future.
 - If this is your Major Required, try to follow the recommended study scheme;
 - If this is your Major Elective, consider your overall plan on MEs carefully;
 - Evaluate your workload together with all the other courses you choose;
 - I am very strict on the **late-drop** applications, so think again if you really want to add this course;
 - Once you decided, be honest and specific on the reasons why you want to add this course **at this particular term** in your add-and-drop applications.

When and where?

- Lectures
 - L1 - Mon/Wed, L2 - Tue/Thu, 15:30~16:50
 - Cheng Dao Bldg. 104
 - Zoom ID: 454 000 1002 (for mixed-mode teaching)
- Tutorials (starting from next week)
 - Mon, Tue, Wed or Thu, 18:00~18:50, 19:00~19:50, 20:00~20:50
 - CD-203. It's a computer lab, but feel free to bring your own laptop.
 - For online students, keep an eye on the notice from BB for Zoom ID (usually not the same as the above Zoom ID for lectures).
 - No tutorial this week.

Lectures – CD104

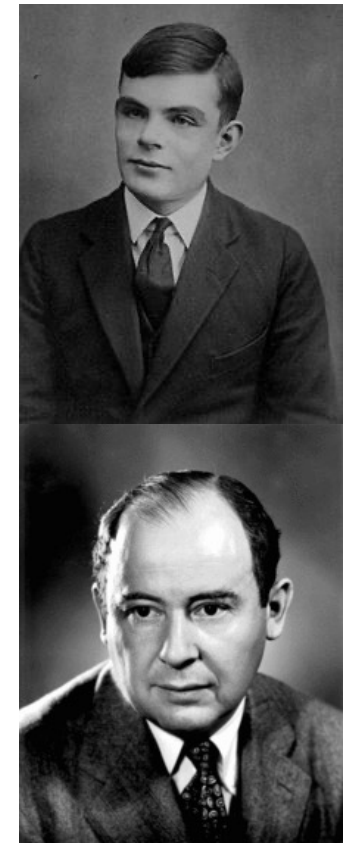
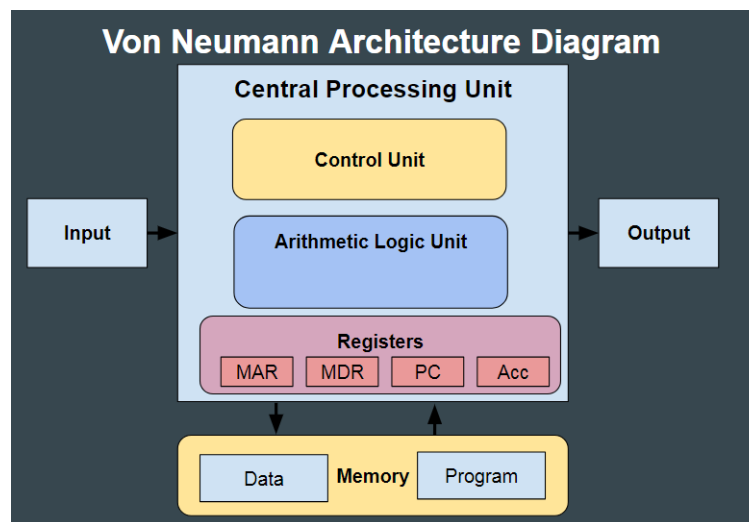
Time	Mon	Tue	Wed	Thu	Fri
15:30-16:50	✓	✓	✓	✓	

Tutorials – CD203

Time	Mon	Tue	Wed	Thu	Fri
18:00-18:50	✓	✓	✓	✓	
19:00-19:50	✓	✓	✓		
20:00-20:50			✓		

What (is a computer)?

- Alan Turing: Turing machine
 - A mathematical model of computation (abstract machine)
- John von Neumann: Von Neumann architecture
 - A practical computer architecture (**stored-program computer**)

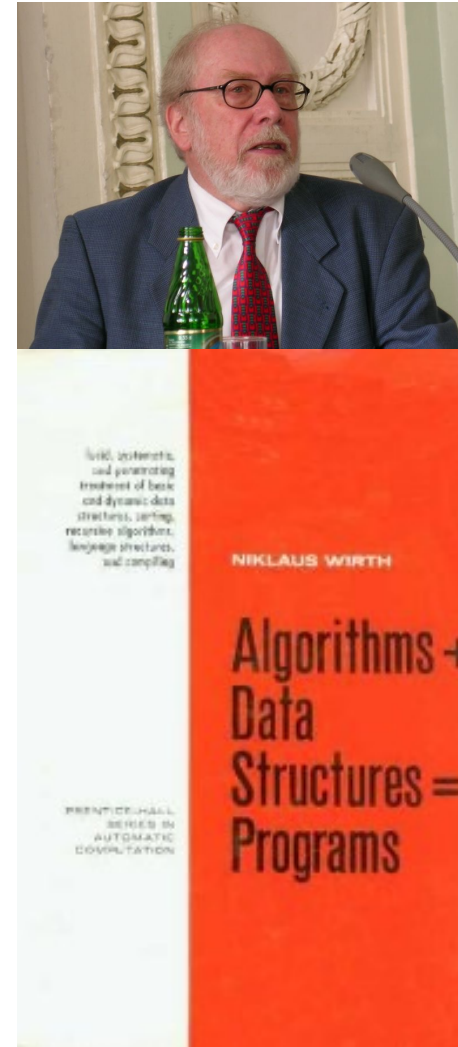


What (is a program)?

- A computer program is a collection of instructions that can be executed by a computer to perform a specific task.

*Control structures,
subroutines, blocks...*

- Niklaus Emil Wirth:
 - Father of the programming language *Algo*
 - A small, efficient language intended to encourage good programming practices using **structured programming** and data structuring.
 - “Algorithms + Data Structures = Programs”





What (is the relation between CSC3002 and others)?

- Course placement: in programming related courses
 - Introduction to Computing
 - Programming Methodology } CSC1001/1002
 - Programming Abstractions
 - **Programming Paradigms** } **CSC3002**
 - Data Structures CSC3100
 - Algorithms CSC4120 } “Algorithms + Data Structures = Programs”
 - Software Engineering CSC4001
 - Distributed and Parallel Computing CSC4005
 - Mobile/Cloud/Web Computing CSC4150/4160/4190
 - Programming Languages CSC3120
 - Compilers CSC4180 } Strongly related, more theoretical aspects of programming

What (are we going to study)? Core concepts

- **Abstraction** Learn a variety of abstract data types that will prove to be enormously valuable as you write programs.
- **Recursion** An enormously powerful technique that enables you to solve complex problems that you would never be able to solve without it.
- **Representation and algorithmic efficiency** Different representations and algorithms can vary enormously in terms of the amount of time and space required to solve a particular problem.
- **Object-oriented programming** A widely used programming paradigm based on the concept of object, encapsulation, inheritance, polymorphism, etc.
- **Programming paradigm** A set of programming principles, following which makes a paradigm the best for a certain kind of problem.

What? Syllabus and (tentative) teaching plan

Week	Content	Activity
1	Course Introduction and an overview of C++	
2	Functions and libraries	Add and drop due
3	Strings and streams	Assignment 1 (4)
4	Collections	
5	Object-oriented programming I: classes	Assignment 1 due; Assignment 2 (4)
6	Pointers and arrays	
7	The C++ memory model	Assignment 2 due; Assignment 3 (2)
8	Recursion	
9	Algorithmic efficiency and representation	Assignment 3 due; Assignment 4 (4)
10	Linear structures: stacks, queues, vectors	No class on 04/05
11	Advanced structures I: maps, trees, sets	Assignment 4 due; Assignment 5 (3)
12	Advanced structures II: graphs	
13	Object-oriented programming II: inheritance	Assignment 5 due; Assignment 6 (2)
14	Iterators and programming paradigms	No class on 05/03
15		Assignment 6 due;

What (are we going to do)? Activities

- Six sets of programming assignments
 - 4 x 4-problem sets and 2 x 2-problem sets, ~20 problems in total.
 - Normally posted on Fridays, and due 2 weeks later on Sundays, so you will have 3 weekends.
- Written exam in the final exam week (no mid-term exam)

What (is the textbook)?

- Textbook:

Programming Abstractions in C++
Eric Roberts, *Pearson*.

- The author's course reader (**Autumn Quarter 2012** version):

<http://cs.stanford.edu/people/eroberts/courses/cs106b/materials/CS106BX-Reader.pdf>

- How to use the textbook:

- Text (**Try to read it!** It's only 1000 pages.)
- *Summary* (Must read and understand!)
- *Review questions* (what the final exam questions look like)
- *Exercises* (where the programming assignments come from)



Every semester someone gets the wrong version and only realizes it in the end. Let's see if this happens again.

What (about the course material)?

- All lecture notes and **sample code** used in class will be provided to the students via **Blackboard** (<https://bb.cuhk.edu.cn/>), normally uploaded on Fridays, with preview material for the next week.
- Preview slides are strictly for preview purposes only and may be subject to change. Make sure you get the final version **after class**.
- There are a lot of animations in the slides, so make sure you use the “**slideshow**” function in PowerPoint to see everything.
- It's your responsibility to remember to check BB or your email for official announcements!

What ... else?

- <http://www.cplusplus.com/>
- <http://www.cppreference.com/>
- <http://www.wikipedia.org/>
- <http://www.google.com/>
- For most of the "Why" questions about the design of C++, the first place to look is *The Design and Evolution of C++*, by Bjarne Stroustrup, though most of such questions are out of the scope of this course.

What (we **do not** learn in this course)?

- Final warning! This is not a C++ course, but a programming course using C++. We do not study:
 - ~~Every last detail of the C++ grammar~~
 - ~~The latest features of C++ 2022~~
 - ~~The 10/100/1000... craziest C++ skills~~
 - ~~Etc.~~

知乎

首页

发现

等你来答

粗暴发工作证大批员工离职

程序员

C (编程语言)

C++

性能优化

你见过哪些令你瞠目结舌的C/C++代码技巧?

update 2015/11/26

这个问题已经被引申为瞠目结舌的系列了，作为题主，我表示瞠目结舌。

X:

基友问题

[你见过哪些让你瞠目结舌的JAVA代码技巧? - Java](#)

[你见过哪些令你瞠目结舌的 JavaScript 代码技巧? - 程序员](#)

[你见过哪些令你瞠目结舌的前端设计? - 程序员](#)

[你见过哪些令你瞠目结舌的Python 代码技巧? - 编程](#)

[zhihu.com/question/3824...](https://www.zhihu.com/question/3824...)

Why (do we learn programming)?

- Former President Obama announces computer science for all initiative
“In the coming years, we should build on that progress, by ... offering every student the hands-on computer science and math classes that make them job-ready on day one.”

—President Obama, 2016 State of the Union Address

- Obama becomes first US president to write a computer program.



Why (do we learn programming)?

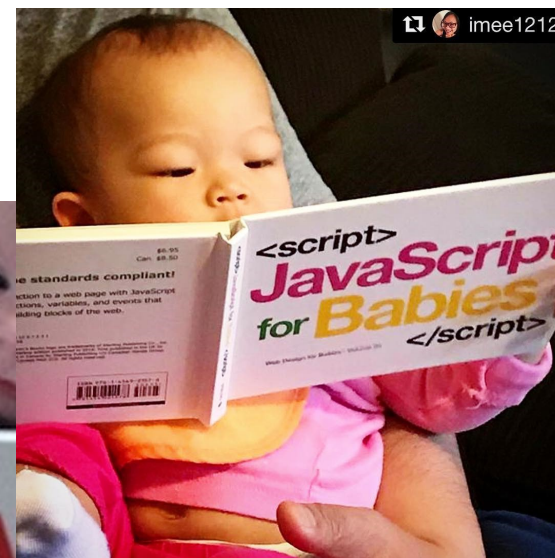
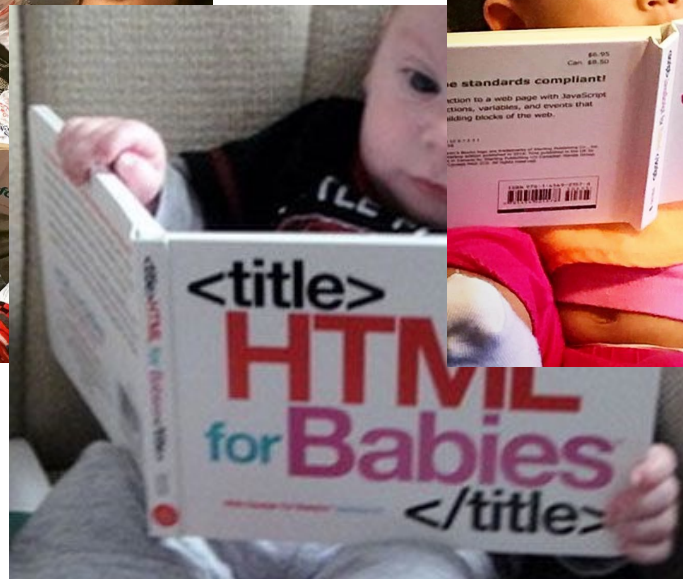
- In the eastern world...

#你知道吗# 现年65岁的现任新加坡总理李显龙，其实是一位C++程序员。之前曾经贴出其写的数独程序代码..... [🔗网页链接](#)

```
)  
e, Possibles, BitCount;  
Count = 100;  
  
< 81; T++) {  
Sequence[T];  
= Block[InBlock[Square]] & Row[InRow[Square]] & C  
= 0;  
ssibles) {  
les &= ~(Possibles & ~Possibles);  
nt++;  
  
unt < MinBitCount) {  
Count = BitCount;  
;  
}
```



Why (do we learn programming)?



Why (do we learn programming)?

- Programming is just generally awesome

-- Julie Zelenski@Stanford


- Learn relatively small set of fundamentals, but infinitely combinable to solve all sorts of problems
- Build impressive things that you can be proud of
- Nothing more satisfying than finding and fixing that ~~last~~ bug



Why (do we use C++ in this course)?

- Introduction to Computer Science: **Programming Paradigms**
 - C++ is a **multi-paradigm** programming language;
 - C++ and Python complement each other very well;
- C++ is a very useful programming language;
- C++ is challenging to learn.

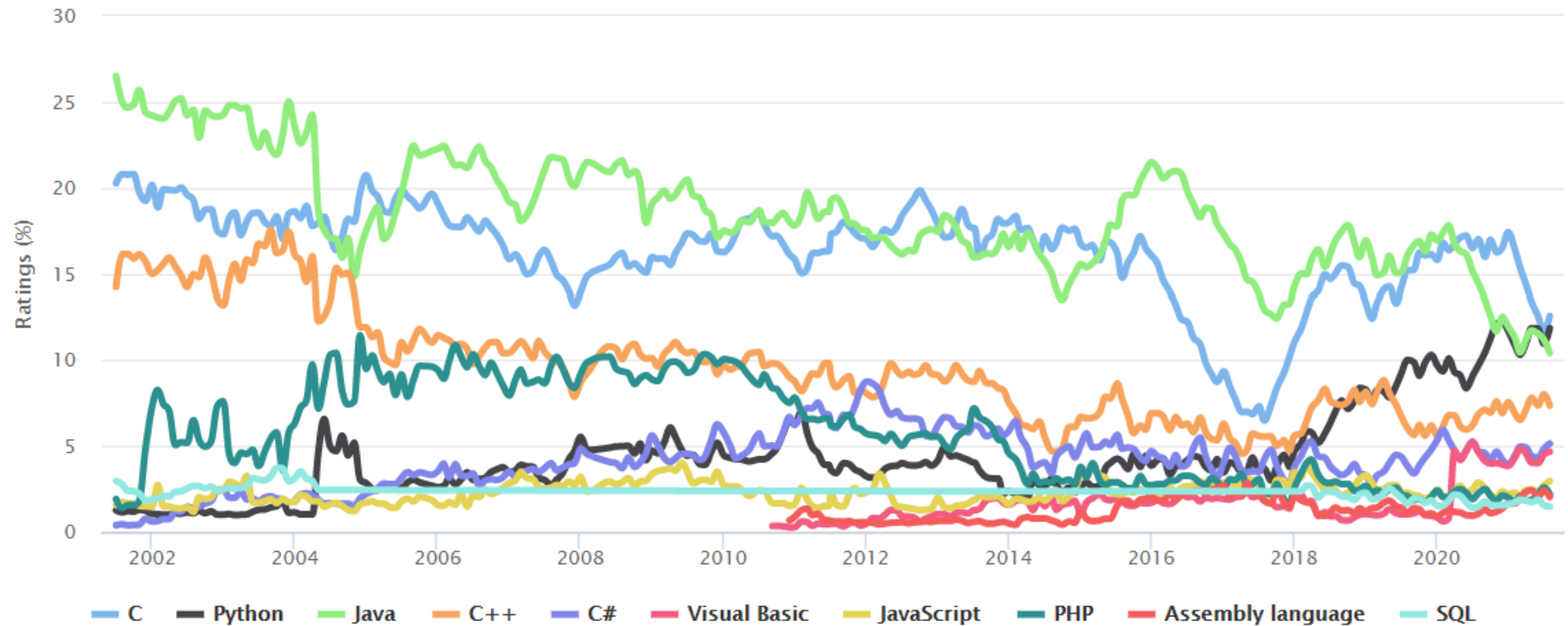
Why (do we use C++ at all)?

Rank	Language	Type	Score
1	Python	  	100.0
2	Java	  	95.4
3	C	  	94.7
4	C++	  	92.4
5	JavaScript		88.1
6	C#	   	82.4
7	R		81.7
8	Go	 	77.7
9	HTML		75.4
10	Swift	 	70.4

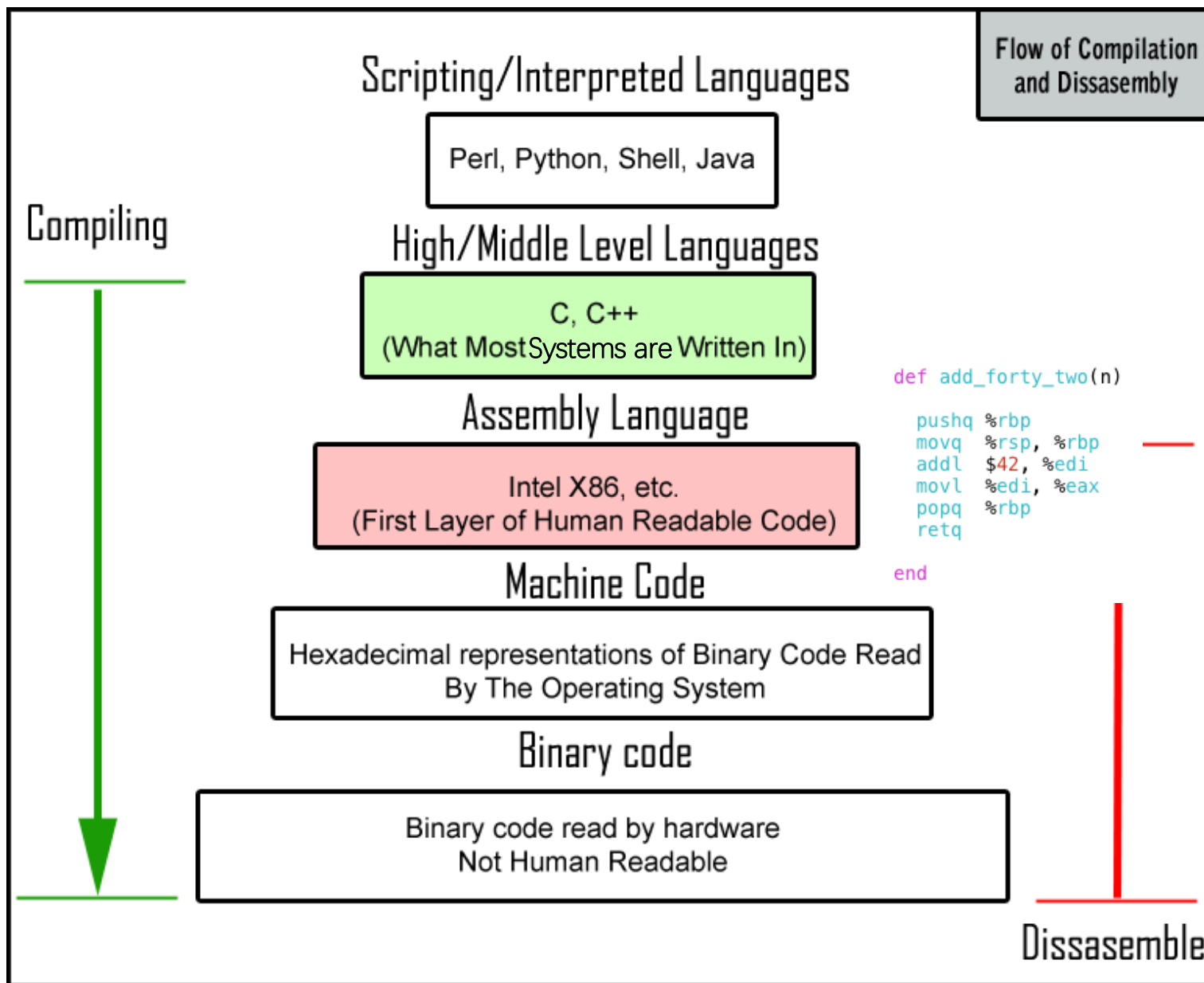
IEEE SPECTRUM

TIOBE Programming Community Index

Source: www.tiobe.com



The TIOBE Programming Community index is an indicator of the popularity of programming languages. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors.



Which language is best, C, C++, Python or Java?



Andrea Ferro

Knows a few programming languages....

Upvote • 1.1k upvotes by Paul Olaru, James...

If you are writing an operating system, I suggest you use C.

If you are writing a very complex application where execution speed is extremely important, I suggest you use C++.

If time to market is key, but execution speed is not important, I suggest you use python.

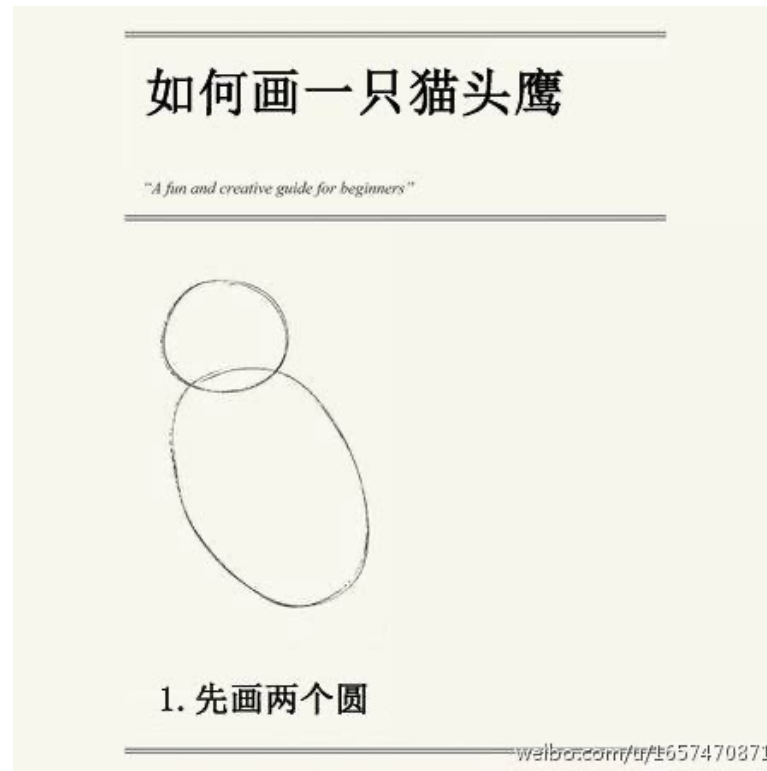
If your boss told you: "do it in Java or you are fired" I suggest you use Java and look for a better workplace.

*Disclaimer:
The opinions expressed in this post are those of the author and do not represent my own view in any way. Please do not show it to your CSC3100 instructor.*

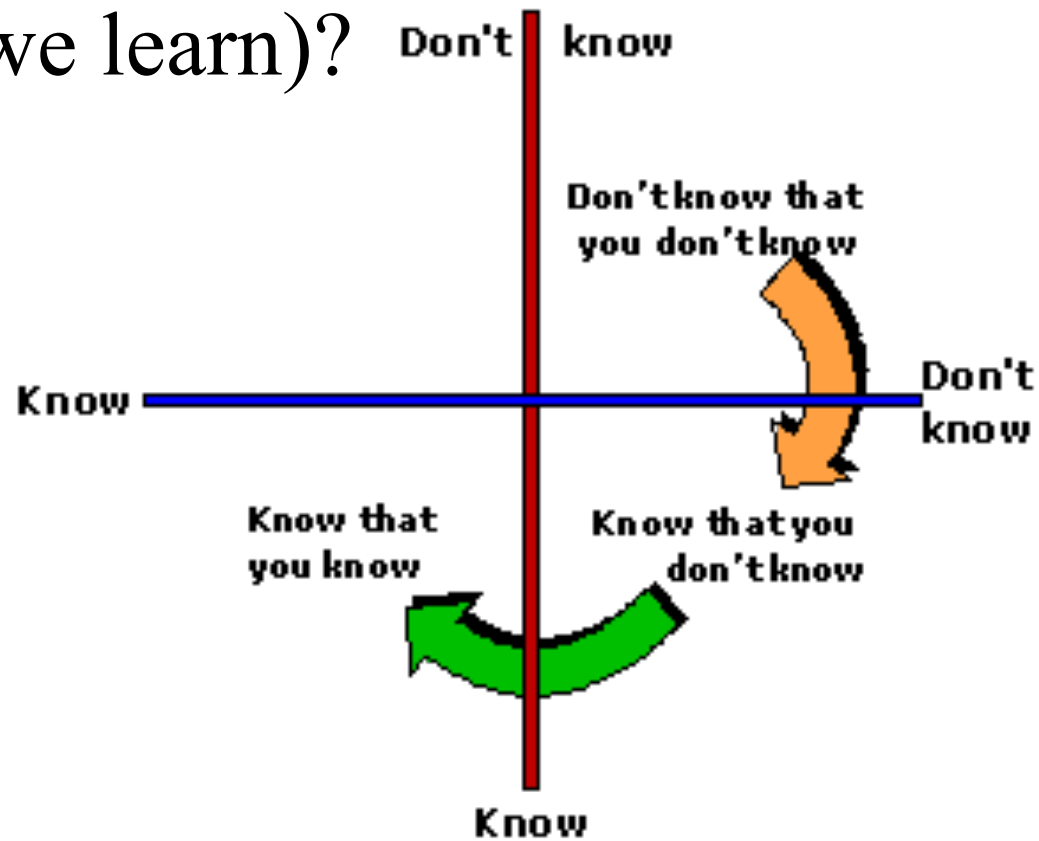
C++ vs. Python

	C++ (used in CSC3002)	Python (used in CSC1001)
Execution	Compiled and linked into an executable.	Interpreted and executed by an engine.
Types	Explicitly declared, bound to names, checked at compile time, and strict until they're not	Bound to values, checked at run time, and are not so easily subverted, an order of magnitude simpler.
Memory Management	Requires much more attention to bookkeeping and storage details	Nearly no attention to be paid to memory management. Supports garbage collection.
Language Complexity	Complex and full-featured. C++ tries to give you every language feature under the sun while at the same time never (forcibly) abstracting anything away that could potentially affect performance.	Simple. Python tries to give you only one or a few ways to do things, and those ways are designed to be simple, even at the cost of some language power or running efficiency.

How (do we learn)?



How (do we learn)?

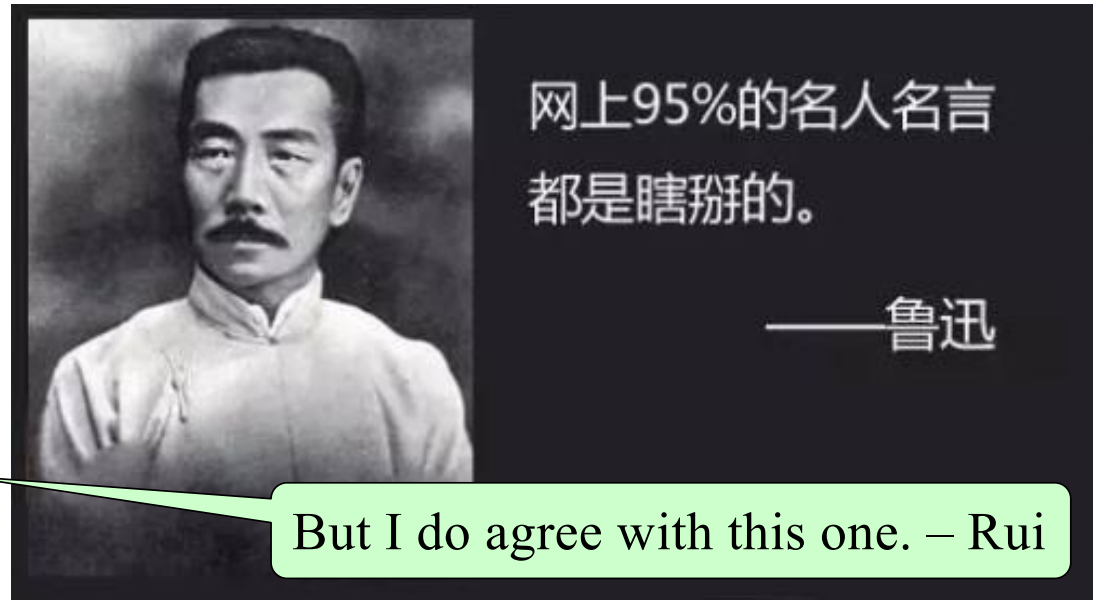


My job

Your job

How (do we learn)?

- “We Learn . . .
 - 10% of what we read
 - 20% of what we hear
 - 30% of what we see
 - 50% of what we both hear and see
 - 70% of what we discuss
 - 80% of what we experience
 - 95% of what we teach others.”



– William Glasser
(May 11, 1925 – August 23, 2013)
American psychiatrist

FROM CSC1002

VIA 9GAG.COM



Lawyer



Teacher



Nurse



Programmer

VIA 9GAG.COM



Musical



HSS

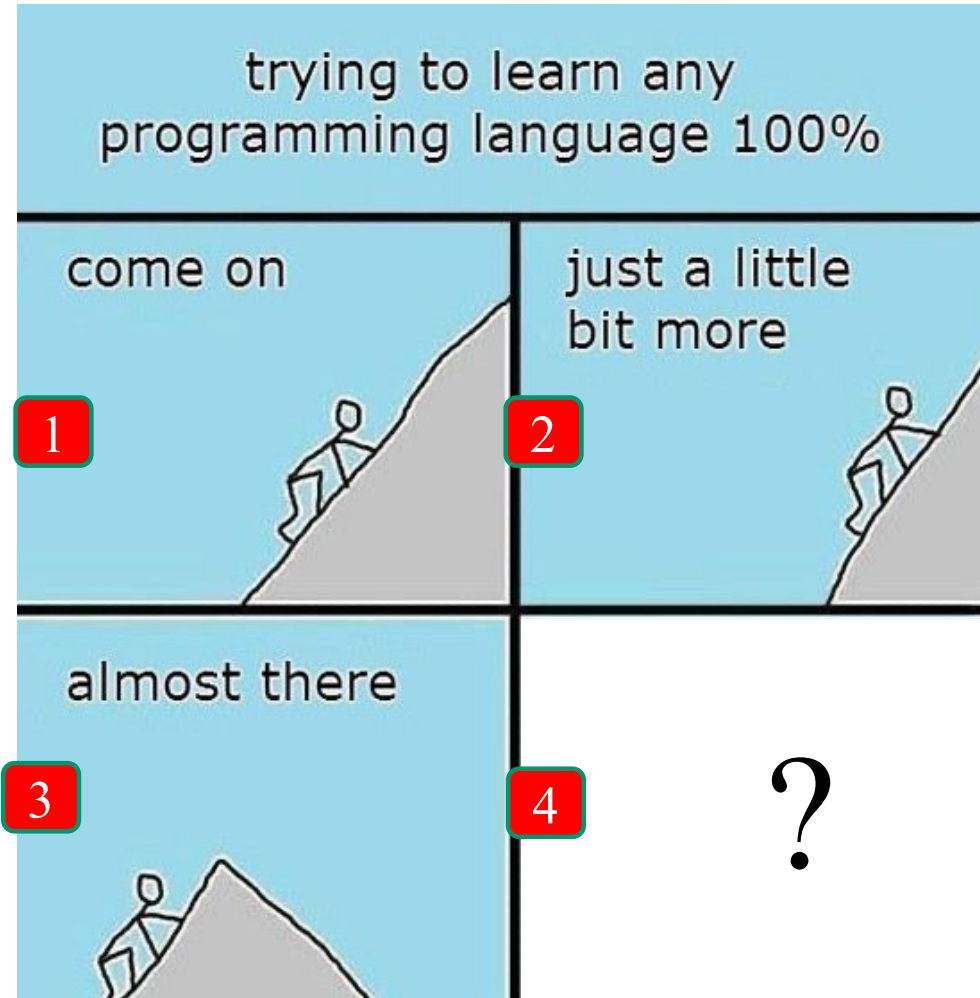


SME

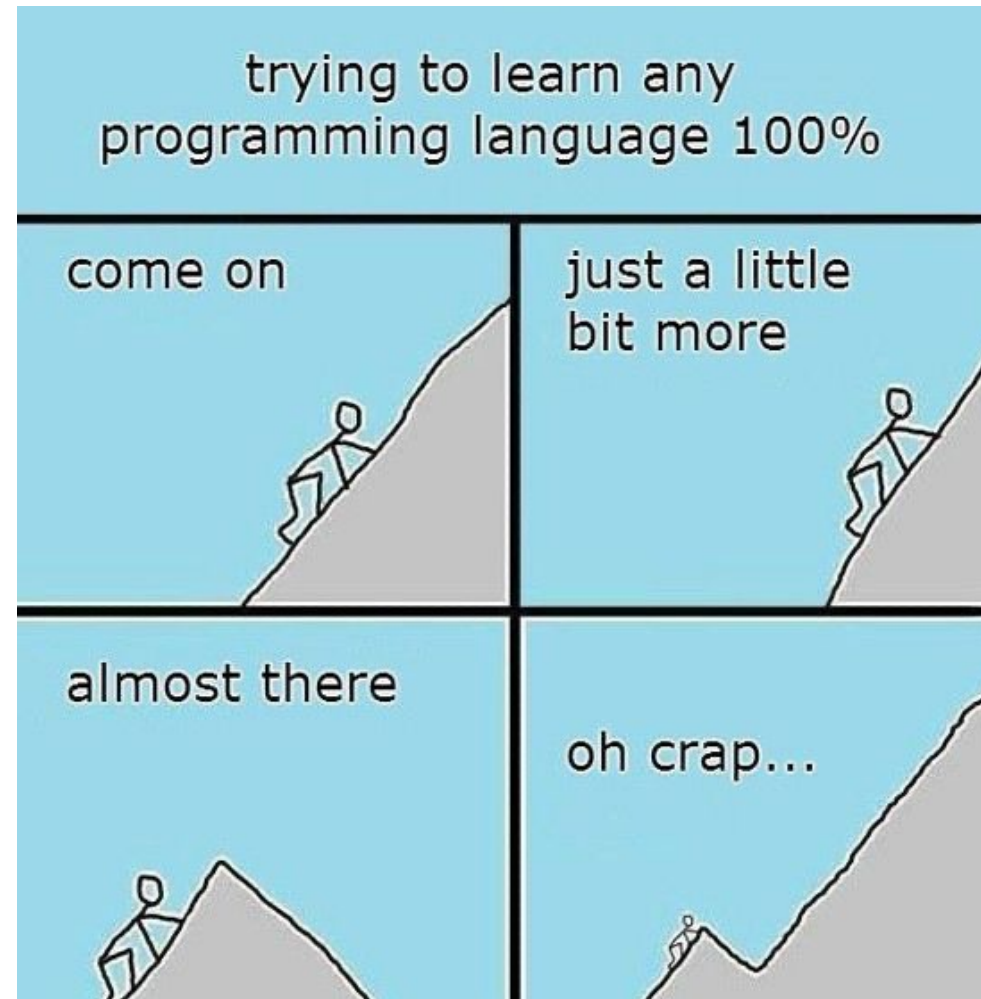


CSC3002

Learning programming



by Kinley Lam



by Kinley Lam

How (much time)?

- How much time do you need (at least)?

Activity	Hours/week
Lecture	3
Tutorial	1
Readings, assignments, project	6 (minimal)

- In the classroom, do not try to read every word on the slides and get distracted. Listen to me carefully. Read the textbook/slides beforehand or afterwards.

How (to grade)? Tentative Assessment Scheme

Component/method	% weight	Learning strategy
Programming assignments	50%	This is the basis of how you can pass this course.
Final exam	50%	This is where you rise and shine (“ A-/A ”). Read the textbook.

- Attendance is not required, especially with the mixed-mode teaching. But I still strongly encourage you to participate actively.

How (to grade)? Tentative Assessment Scheme

- Expecting roughly 15% A, 15% A-, 15~20% B+/B/B- each, but no guarantee.
- Missing the majority of any assessment component might directly lead to F.
- Plagiarism might directly lead to F.

How (to grade)? Academic Honest

- Declaration of originality
- Plagiarism detection: **VeriGuide**, **Moss**, etc.
- Everyone will probably be asked to explain to the teacher a randomly chosen block of code from his/her code sometime during the semester.
- You probably don't believe it, but every year, some students are plagiarizing, facing serious consequences.

I am submitting the assignment for:	
<input type="checkbox"/> an individual project or	
<input type="checkbox"/> a group project on behalf of all members of the group. It is hereby confirmed that the submission is authorized by all members of the group, and all members of the group are required to sign this declaration.	
<p>I/We declare that the assignment here submitted is original except for source material explicitly acknowledged, the piece of work, or a part of the piece of work has not been submitted for more than one purpose (i.e. to satisfy the requirements in two different courses) without declaration, and that the submitted soft copy with details listed in the <Submission Details> is identical to the hard copy(ies), if any, which has(have) been / is(are) going to be submitted. I/We also acknowledge that I am/we are aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the University website https://registry.cuhk.edu.cn/en/integrity. In the case of a group project, we are aware that each student is responsible and liable to disciplinary actions should there be any plagiarized contents in the group project, irrespective of whether he/she has signed the declaration and whether he/she has contributed directly or indirectly to the plagiarized contents.</p> <p>It is also understood that assignments without a properly signed declaration by the student concerned and in the case of a group project, by all members of the group concerned, will not be graded by the teacher(s).</p>	
Signature(s)	Date
Name(s)	Student ID(s)
Course code	Course title

How? Lessons from the previous runs

- Assignment late policy
 - *Hofstadter's Law*: it always takes longer than you think, even when you take Hofstadter's Law into account.
 - If something **unexpected** (better be **extraordinary**) happened, get an extension approved by the TAs and CCed to the instructor **before** the deadline.
- Correctness: test your program as thoroughly as possible
- Style matters!



How? Lessons from the previous runs

- The biggest complaint was always about the **workload**. Since the removal of the projects, the workload is not as heavy as before. But still, C++ is an extremely complex language. You'll have to spend a lot of time learning a lot of things **outside the classroom by yourself**.
- You WILL encounter a lot of problems **in the beginning**, with compilers, IDEs, libraries, etc., but do not give up because of the frustration. We are here to help, our TAs and USTFs are very experienced and really understand your pain.
- Learn from your classmates. Form a **study group**, even without the group projects. It is always very helpful to study together.

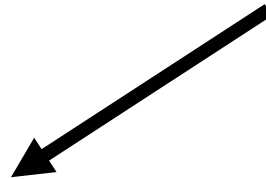
How? Lessons from the previous runs

- Communicate!
- Communicate!
- Communicate!

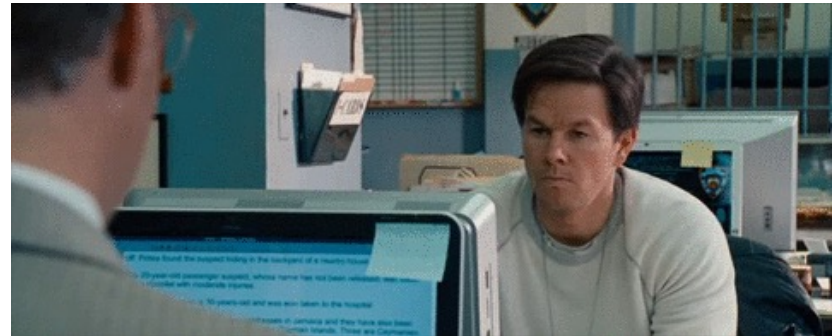
with your classmates, with the TAs, and with me.

HAPPINESS IS

This, not this.

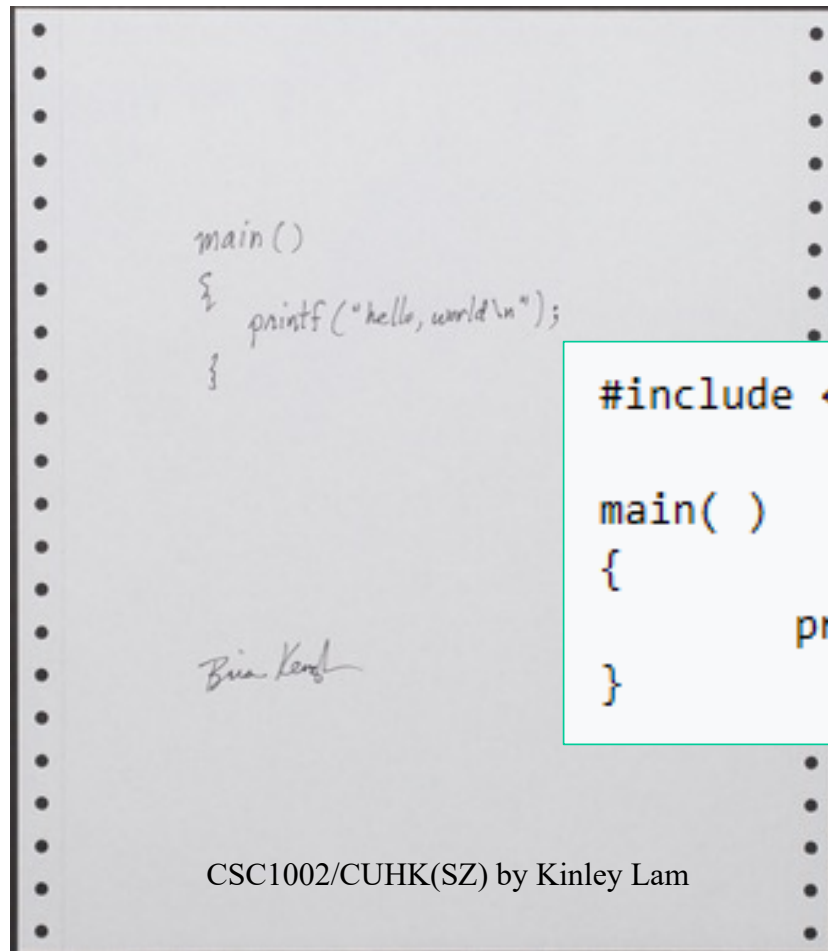
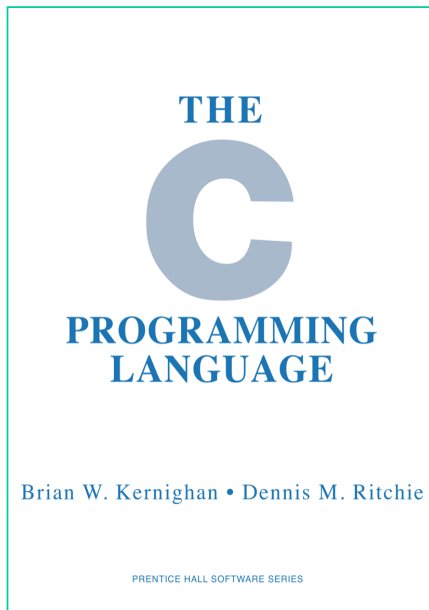


...programming
all day.



Questions?

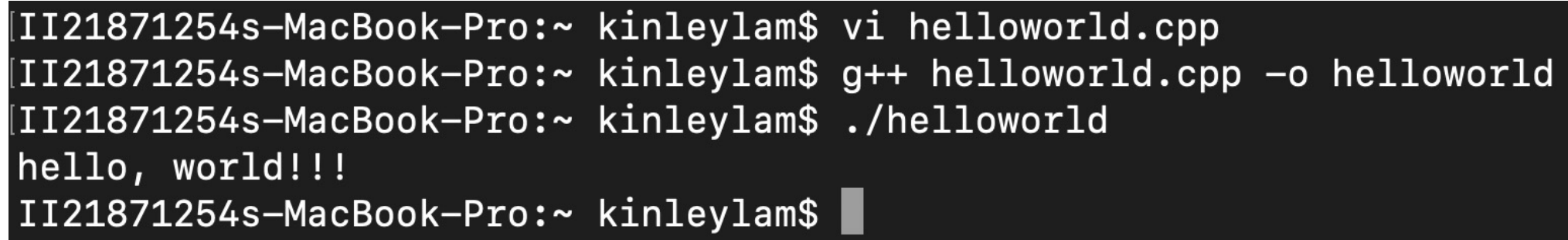
Brian Kernighan



QUICK DEMO (VI EDITOR)
HELLO, WORLD !!!

A terminal window with a dark background. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left, and a home icon followed by the text "kinleylam — vi helloworld.cpp — 80x24" on the right. The main area of the window displays the C++ source code for a program that prints "hello, world!!!". The code is numbered 1 through 5 on the left margin.

```
1 #include <iostream>
2
3 int main() {
4     std::cout << "hello, world!!!" << std::endl;
5 }
```

A terminal window with a dark background showing the steps to compile and run the C++ program. The prompt is "II21871254s-MacBook-Pro:~ kinleylam\$". The first command is "vi helloworld.cpp", the second is "g++ helloworld.cpp -o helloworld", and the third is "./helloworld". The output of the program is "hello, world!!!". The prompt is then shown again with a cursor.

```
II21871254s-MacBook-Pro:~ kinleylam$ vi helloworld.cpp
II21871254s-MacBook-Pro:~ kinleylam$ g++ helloworld.cpp -o helloworld
II21871254s-MacBook-Pro:~ kinleylam$ ./helloworld
hello, world!!!
II21871254s-MacBook-Pro:~ kinleylam$
```