School of Computing & Communications | Lancaster University

SCC.111 Software Development – Lecture Lecture 21: Introduction to Lent Term

Adrian Friday, Hansi Hettiarachchi and Nigel Davies

Welcome back, we hope you had a good break!

Today

- A quick look back at how the week 10 quiz went
- An overview of what’s happening on 111 this term
- Introducing the teaching team (Hansi and I, plus the lab team)
- A reminder of the structure of the assessment and how to get help when you need it

Week 10 quiz, moderation and timeline

- Moderated the 111 marks
- An issue was found with one of the Q10 test cases (0 fixtures) – test case fix and all answers remarked
- Decided to externally rerun all tests as some code that compiles in the labs did not compile within coderunner, manually awarded any marks due
- Yet to do: those with special extensions (this week), plagiarism (next week)
- Should have marks released shortly thereafter (hopefully week 12)

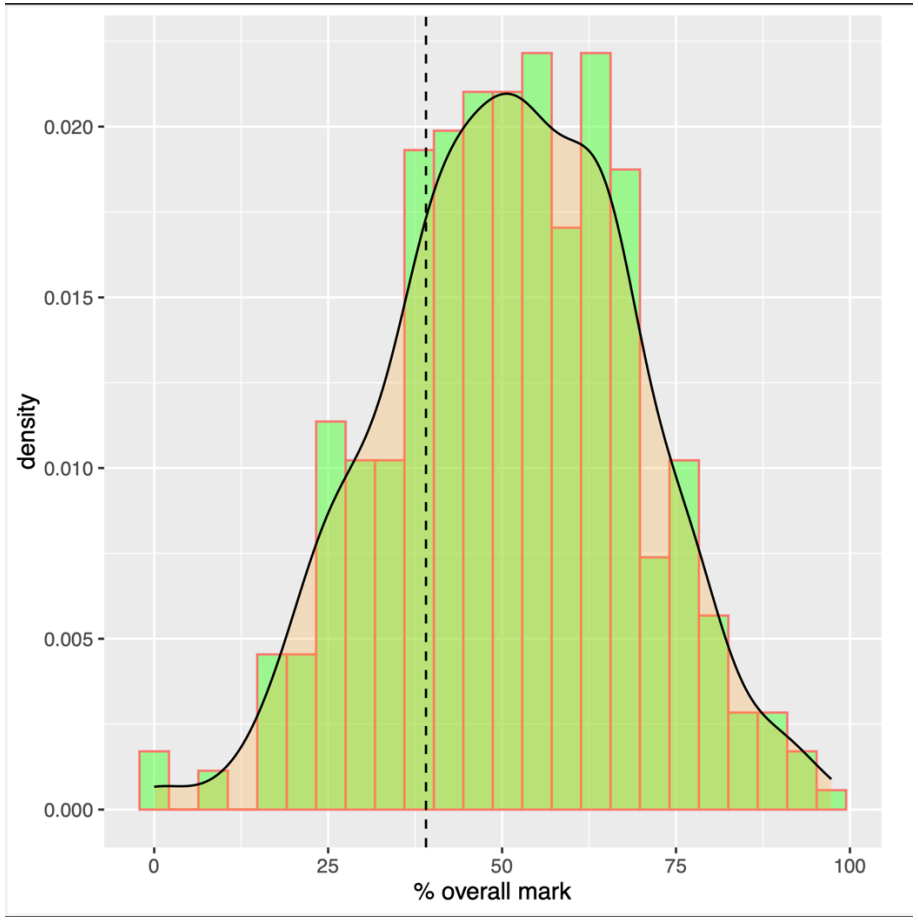
gcc –Wall – how coderunner is different with this option

Syntax Error(s)

```
__tester__.c: In function ‘process_cues’:
__tester__.c:66:9: error: unused variable ‘totalcues’ [-Werror=unused-variable]
   66 |     int totalcues = 0;
      |         ^~~~~~~~
cc1: all warnings being treated as errors
```

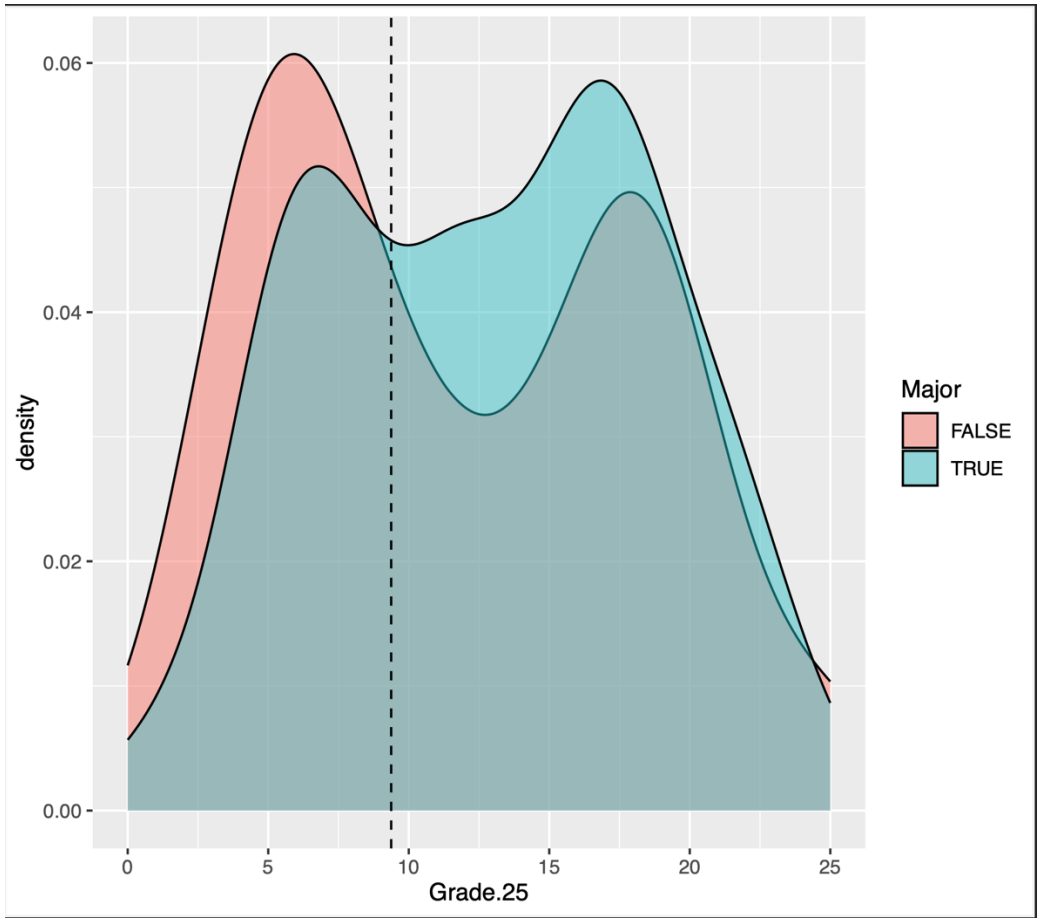
SCC.111 Preliminary Marks (%)

Median : 56%,
Mean : 54%



SCC.111 Questions only

Median/mean :
13 / 25 = ~52%



Please remember

- Once you see where you went wrong, you can go through it in your subsequent labs
- If you need/want to go to more than one lab per week you can do so
- The FAST hub is available each afternoon on B floor Infolab21, near the labs
- *If you found a better way to solve it than someone else, share your top tips and help others learn*!

Where it went well

- The quiz was composed of 11 questions from 4 question banks chosen at random
  - A 3 x 1 mark knowledge questions
  - B 3 x 1 mark applied knowledge questions
  - C 2 x 2 mark ‘dry run’ ability
  - D 3 x 5 mark code (ability to modify, debug and solve week 9)

Second most (44%)

Most non-attempts (~40%)

Block C, the most tricky question? (*variation*)

```
int main()
{
  srand(time(NULL));

  int greenBottles = 10,
      smashed = 0;

  for (; greenBottles < 10; greenBottles--) {
    if (rand() % 2 == 1)
      smashed++;
    printf("%d green bottles are hanging on the wall\n", greenBottles);
  }

  /* what is the value of greenBottles and smashed at this point */
  printf("Smashed = %d\n", smashed);
}
```

Block D, actually people did pretty well with programming!

Which is great, as that’s the point!

Q9 The most common error?

The meta data section of the LAFF format has been extended with a content rating (PG, U, 18 etc.). See example input below. To accommodate this, the '`process_show_metadata`' function now takes an additional parameter '`ageRating`' and you'll need to extend your implementation to take this into account.

For instance:

```
BEGIN_SHOW_DATA
The Great SCC Panto
24.12.2024
PG
END_SHOW_DATA
```

You should not need to #include anything, nor write or include a main function.

Answer: (penalty regime: 0 %)

Reset answer

```
int process_show_metadata(FILE *fp, char *showName, char *
{
    // Your answer here
}
```

Q9 The second most common

```
SYNOPSIS
     #include <stdio.h>

     char *
     fgets(char * restrict str, int size, FILE * restrict stream);

     char *
     gets(char *str);

DESCRIPTION
     The fgets() function reads at most one less than the number of characters
     specified by size from the given stream and stores them in the string
     str.  Reading stops when a newline character is found, at end-of-file or
     error.  The newline, if any, is retained.  If any characters are read and
     there is no error, a '\0' character is appended to end the string.
```

- You may find you need to remove the `"\n"` character that `fgets()` will read in from the file. You can do this with the following line of code (assuming you have read the line into an array called line) `line[strcspn(line, "\n")] = '\0';`

Appendix A: A Sample I

```
BEGIN_SHOW_DATA
The Great SCC Panto
```

line

_	D	A	T	A	\n	\0

```c
if (fp) {
    char line[80];

    // Read first line only

    if (fgets(line, 80, fp)) {
        printf("Read <<%s>>\n", line);
    }

    fclose(fp);
}
```

```
Read <<BEGIN_SHOW_DATA
>>
```

Can ‘see’ hidden \n here if we printf it

Appendix A: A Sample LAFF File

```
BEGIN_SHOW_DATA
The Great SCC Panto
```

\n end of text line (newline)
\0 end of string (added by fgets to ‘mark end of string’)

line

_	D	A	T	A	\n	\0

```
Read <<BEGIN_SHOW_DATA
>>
```

Can ‘see’ hidden \n here if we printf it

Lesson 1: read before you write, test cases

Programs that *aren't tested* are worse than useless

Does it work for a simple representative set of inputs ?

Lesson 2: variable declarations

```c
int process_show_metadata(FILE *fp, char *showName, char *showDate, char *ageRating)
{
    char line[MAX_STRING_LEN];
    int counter, foundName, foundDate, foundRating = 0;
    int nameLine = 1;

    while(fgets(line, MAX_STRING_LEN, fp) != NULL){
      if(counter == nameLine){
        line[strcspn(line, "\n")] = '\0';
        strcpy(showName, line);
        foundName = 1;
      }
```

Join at menti.com | use code 1750 2414

Why did we get this?

Syntax Error(s)

```
__tester__.c: In function ‘process_cues’:
__tester__.c:107:1: error: control reaches end of non-void function [-Werror=return
  107 | }
      | ^
cc1: all warnings being treated as errors
```

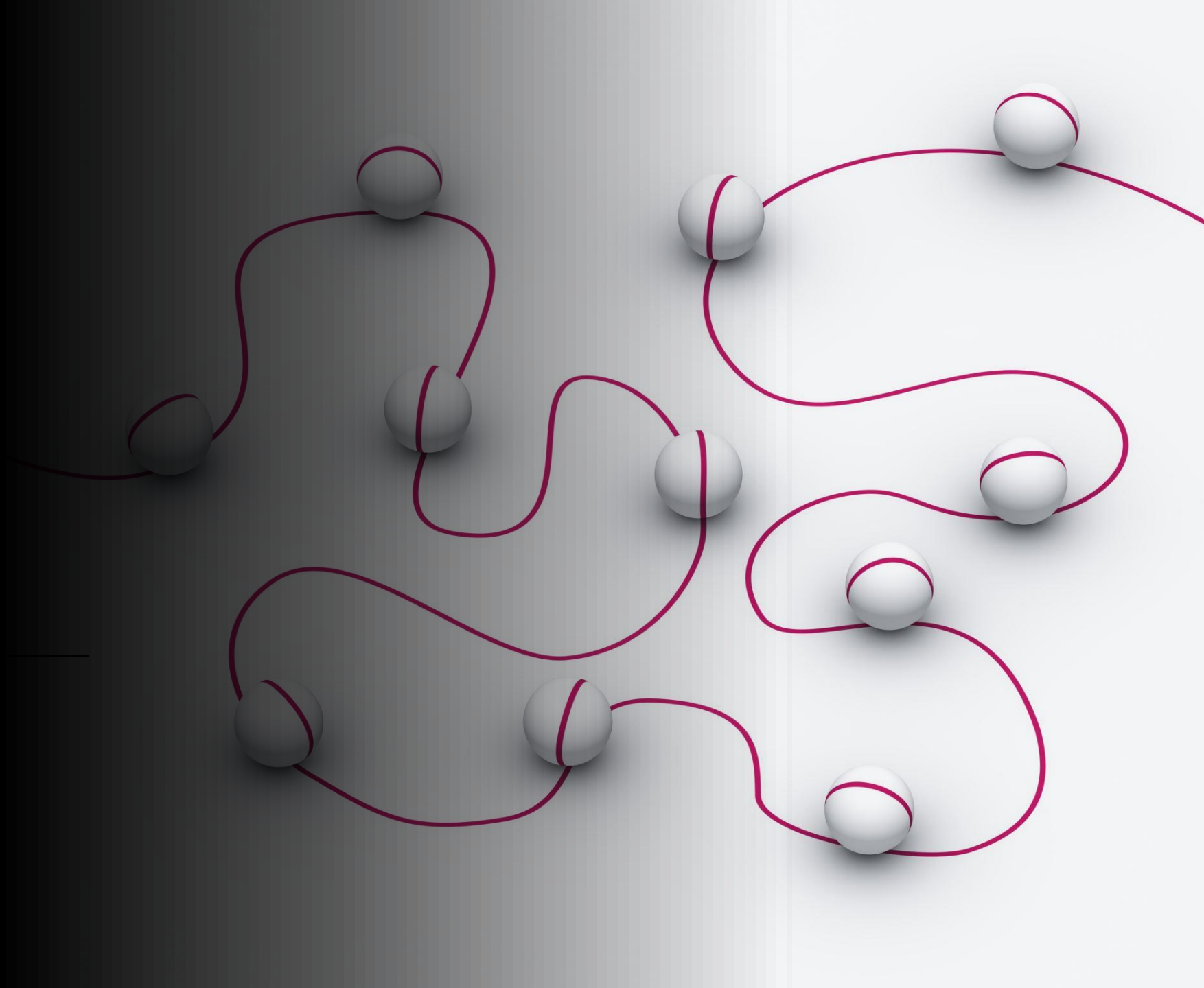
```c
#define MAX_STRING_LEN 80
#define MAX_CUES 10
#define MAX_STEPS 20

int process_cues(FILE *fp, cueType cues[MAX_CUES][MAX_STEPS])
{
  if (fp != NULL) {
    char buffer[MAX_STRING_LEN];

    while (fgets(buffer, MAX_STRING_LEN, fp) != NULL)
      line_counter++;

    return 1;
  }
}
```

Syntax Error(s)

```
__tester__.c: In function 'process_cues':
__tester__.c:107:1: error: control reaches end of non-void function [-Werror=return
  107 | }
      | ^
cc1: all warnings being treated as errors
```

Join at menti.com | use code 8227 9488

Walkthrough: return paths

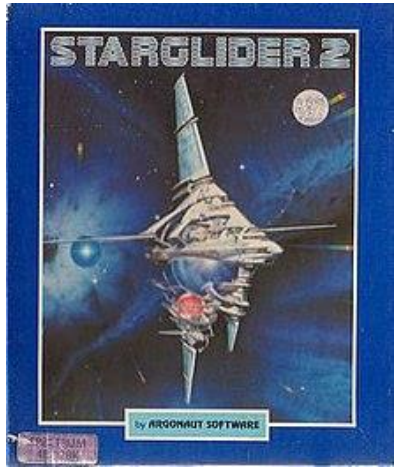


Part 2

The term ahead starts now…!

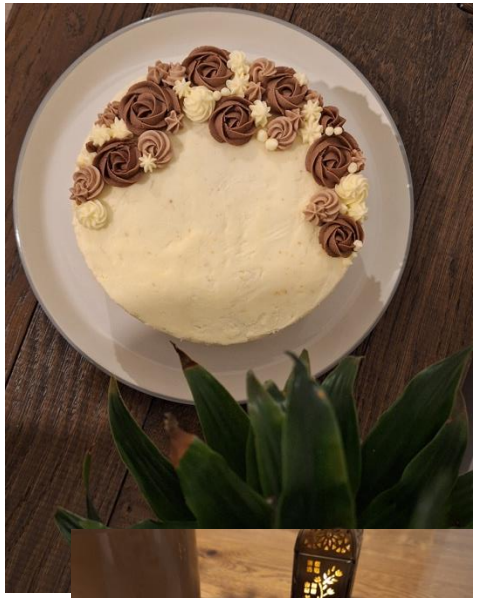
About Us…

About Us… Adrian



From mobile systems to sustainability; plus some other stuff

About Us… Hansi



I develop machine learning approaches for natural language processing (NLP) tasks.

SCC.111 The story so far…

- Primitive types: constants, variables, arrays
- Compound types: structs

- Principles of control flow: logical operators, loops and conditionals
- Principles of control flow: Functions, parameters and return values
- Principle of scope: variables
- Principle of indirection: pointers

- Principles of Software Engineering: testing, debugging, version control
- Principles of Systems Programming: files, dynamic memory…

- **Practiced these principles through the lens of the C programming language**

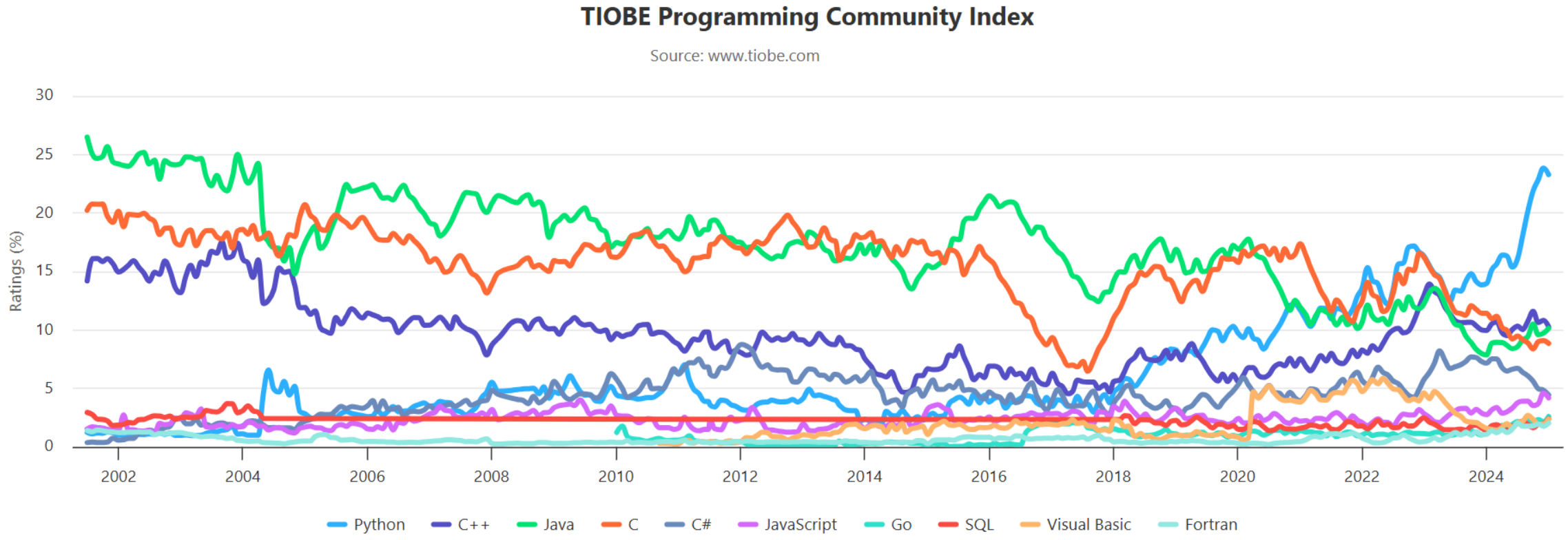
So are you done?

That set of principles makes you Turing complete... and then some.

Do you feel like a professional software developer yet?

Common Programming Languages in 2025…

- Programming languages allow us to abstract away from the detail of hardware, and focus upon describing the functionality of software…
- **So why do we have more than one programming language?**



In more detail…

Jan 2025	Jan 2024	Change	Programming Language	Ratings	Change
1	1		Python	23.28%	+9.32%
2	3	^	C++	10.29%	+0.33%
3	4	^	Java	10.15%	+2.28%
4	2	v	C	8.86%	-2.59%
5	5		C#	4.45%	-2.71%
6	6		JavaScript	4.20%	+1.43%
7	11	^^	Go	2.61%	+1.24%
8	9	^	SQL	2.41%	+0.95%
9	8	v	Visual Basic	2.37%	+0.77%
10	12	^	Fortran	2.04%	+0.94%

Why do we have so many languages?

- **There is strength in diversity…** Languages find their own niche…
  - C / C++
  - Java / C#
  - Python
  - JavaScript
  - **Choose the tool for the right job.**

"If the only tool you have is a hammer, you tend to see every problem as a nail."
[Abraham Maslow]

So…

- **We're going to teach you to embrace this diversity!**
  - C / C++ - We'll Introduce C++ principles in week 11-13.
  - Java / C# - We'll introduce Java in week 14-20
  - Python - We'll introduce Python in week 21-25
  - JavaScript - Take SCC.213 Internet Application next year ☺
  - **Then you CAN choose the tool for the right job…**

Good news: All these languages share common programming principles…

Classifying Languages

Languages are commonly classified by their **core paradigm**. But this isn't an exact science!

- **Imperative / Procedural**
  - Controlled flow of operations that effect a programs state.
  - e.g. C, assembler, [*]sh, Postscript, OpenGL, Logo, Scratch, Javascript, Python…
- **Object Oriented**
  - Imperative with implicit encapsulation of data and functions
  - e.g. C++, Java, C#, Javascript, Python…
- **(Pure) Functional**
  - Declarative languages with no explicit state
  - Haskell, Erlang, (Python ?)…
- **Declarative**
  - Describe logic of program, with minimal explicit control of the flow of execution.
  - SQL, HTML, Regex

The term ahead…

SCC111 Term 2 and 3	Lecture 1	Lecture 2
Week 11	Term 1 Feedback and Introduction	OO Fundamentals
Week 12	Encapsulation 1: Classes, variables and methods	runtime debugging
Week 13	Encapsulation 2: Constructors and Destructors	Case study: micro:bit
Week 14	(In)secure coding in C	Virtual Machine based languages
Week 15	Introducing Java	Pointers, references and Garbage collectors
Week 16	Collections	Class libraries
Week 17	code documentation	version control revisited
Week 18	Inheritance	Inheritance 2
Week 19	method overriding / overloading	Polymorphism
Week 20	Interfaces	recursion
Week 21	Term 2 Feedback and Introduction	Static vs Dynamic typing…
Week 22	Introducing Python	OO and Python
Week 23	lists and dictionaries	safety, performance and flexibility
Week 24	cross language integration	CI
Week 25	Revision	Revision

***We may adapt this to improve coherence based on observations and feedback as we progress through the term**

Teaching and assessment…

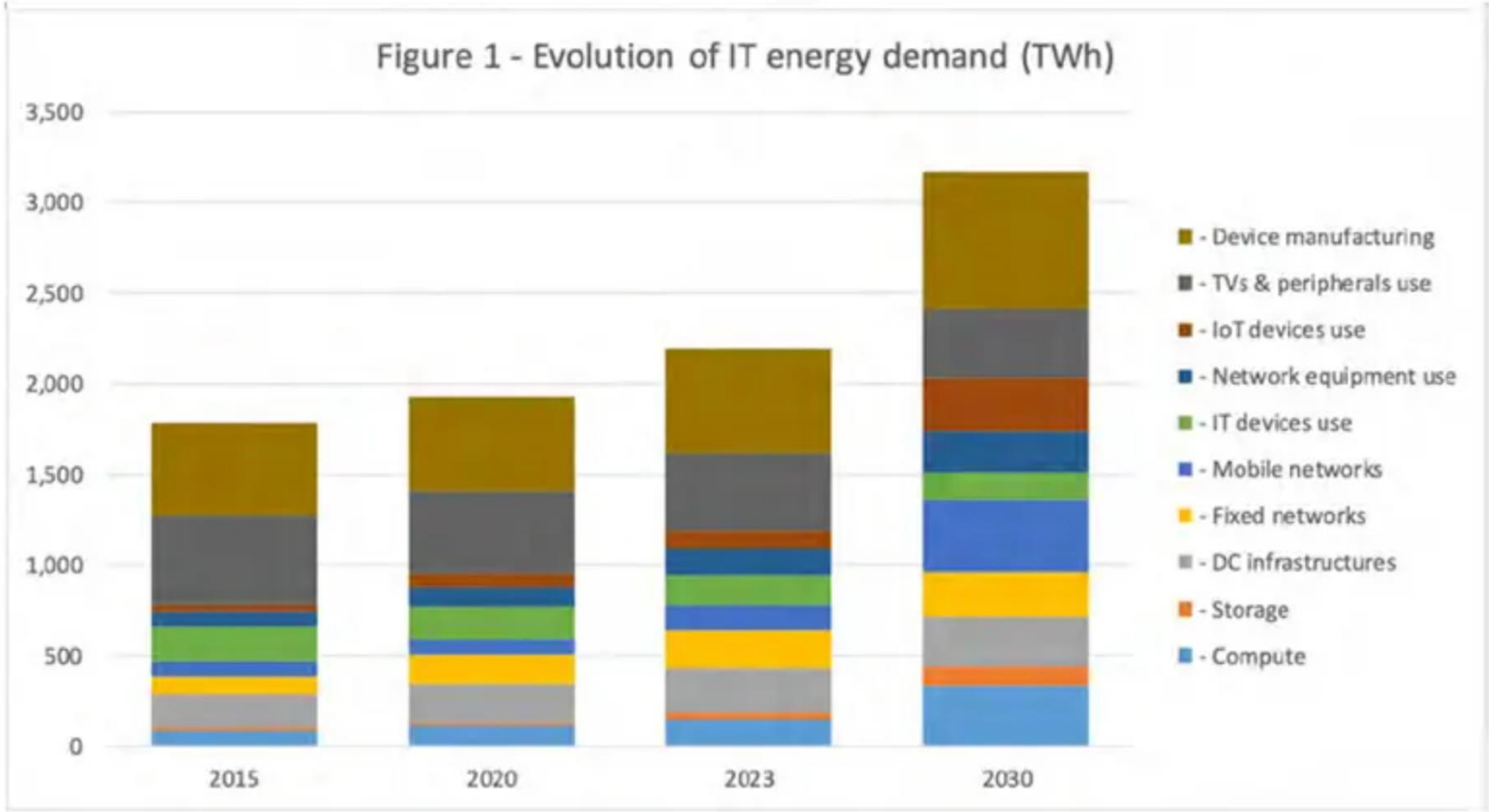
- **Weekly lab tasks set in your SCC1x1 lab**
  - We expect it to *take longer* than the lab session to complete
  - We will align with other courses where possible to support joined up teaching

  - This term will start to introduce new topics for many of you
  - **ASK** in your lab session about concept and techniques you don't fully understand

- **Assessment points**
  - Week 15 quiz
  - Week 20 coursework assignment
  - End of year exam

Making it Real…

Making it Real… energy usage of technology



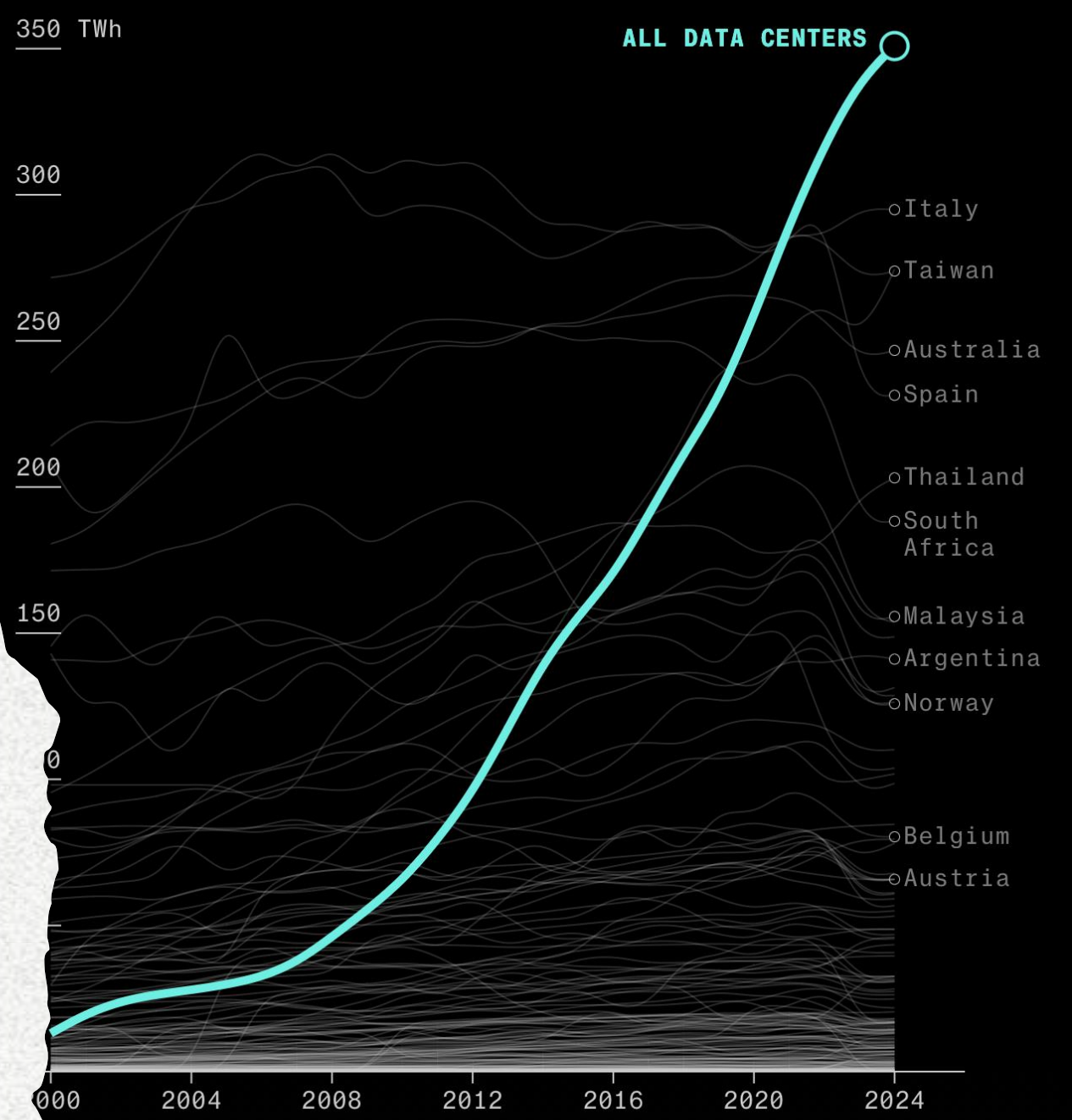
Schneider Electric estimates that IT sector electricity demand will grow by 50 percent by 2030, reaching 3,200TWh, equivalent to 5 percent Compound Annual Growth Rate (CAGR) over the next decade – evolution of IT energy demand in TWh

AI data centres are growing +33% annually

A google search with AI is 10x one without

Using ChatGPT emits ~7 metric tons of CO_2 per day

World data centres now total more energy demand than major nations!



ources: Bloomberg analysis of BloombergNEF and DC Byte data

Conclusions

Today we learned:

- **Diversity is important for tech too**
- **This term is going to be great, and there's lot of support available!**

- **Not everything is a nail. ☺**