FIT1008/FIT2085
While I start, get to know the others

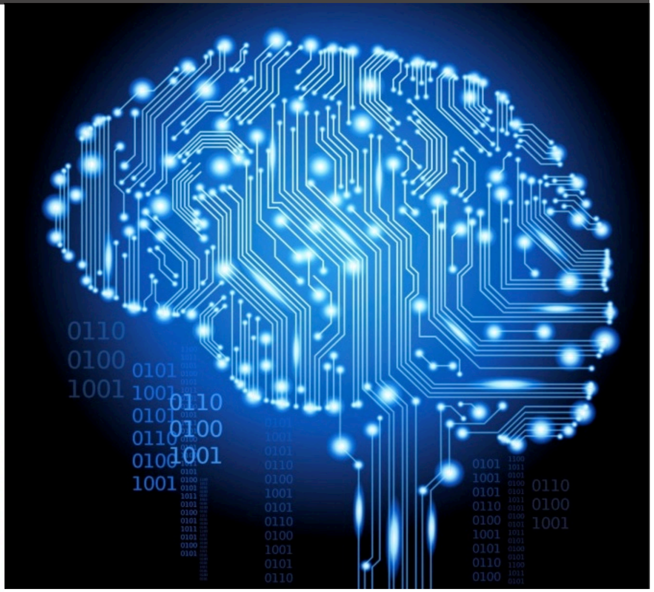
- **Fold the card in two and put your name in both sides**
- **Making friends is an important life skill**
- **Making friends at Uni will:**
  - Make you more likely to attend lectures, tutes and labs
  - Help clarify things that you did not understand
  - Remind you/make you aware of important events
  - Help you study
- **The friends you make at Uni may end up being life long friends**

MONASH University

Information Technology

FIT1008/FIT2085 Lecture 1 Introduction

Prepared by:
Maria Garcia de la Banda
Revised by A. Aleti, D. Albrecht, G. Farr, J. Garcia and P. Abramson



Learning objectives for this lecture

- **To briefly go over the unit basics:**
  - The teaching team
  - What it is the unit all about
  - Timetable and structure
  - Recommended reading
  - Assessment and hurdles
  - Cheating (please don’t)
  - Special consideration
  - Getting help
  - …
  - and much more

- **For the details, please look into the Unit Guide!**

Teaching Team at Clayton

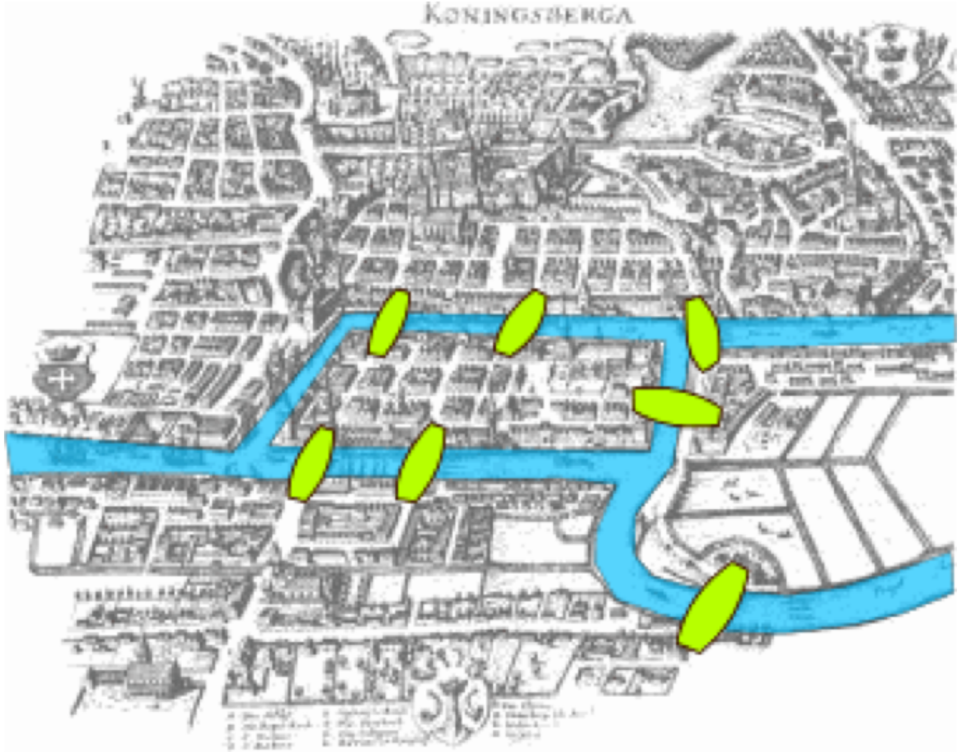
Chief Examiner

Lecturer **Lecturer** **Head Tutor**

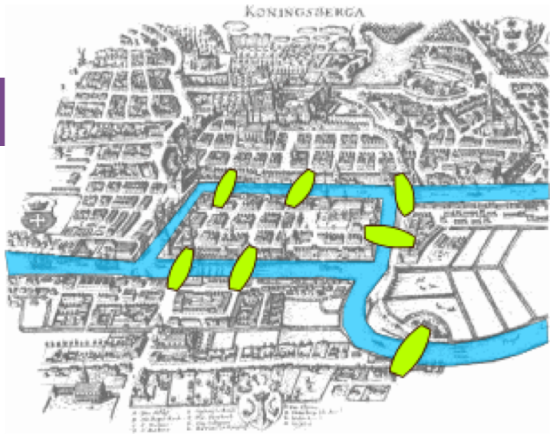
Maria Garcia de la Banda **Pierre Le Bodic** **Brendon Taylor**

What is this unit all about?

- It is about the fundamentals of CS and SE
- It all starts with a problem that needs to be solved
- Consider the famous Köningsberg problem:
  - Find a walk through the city that crosses each bridge once, and only once

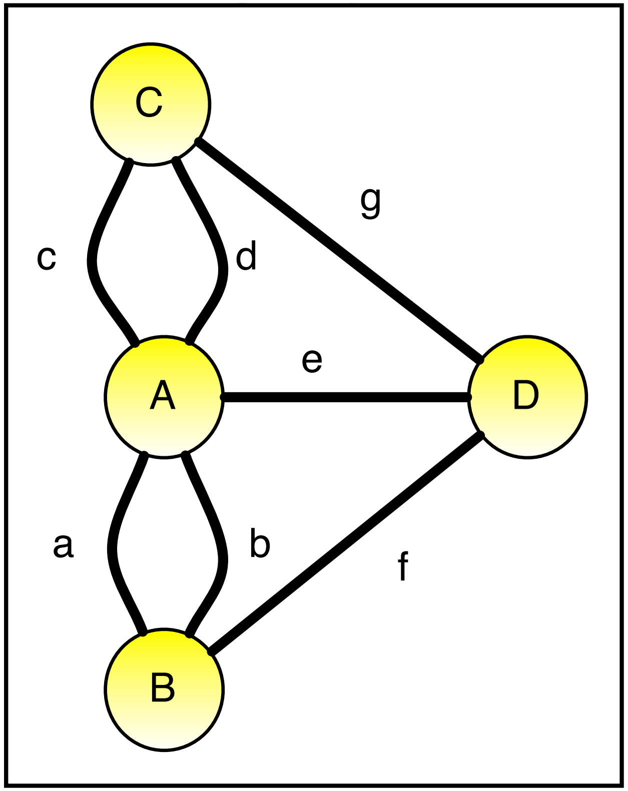


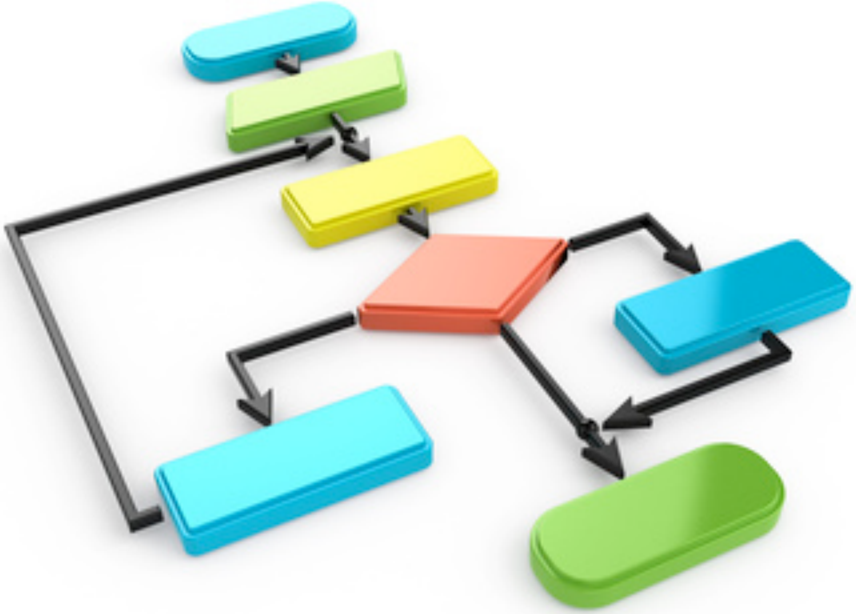
What is this unit all about?



- **To do this we often abstract it**
- **& develop a high-level algorithm**

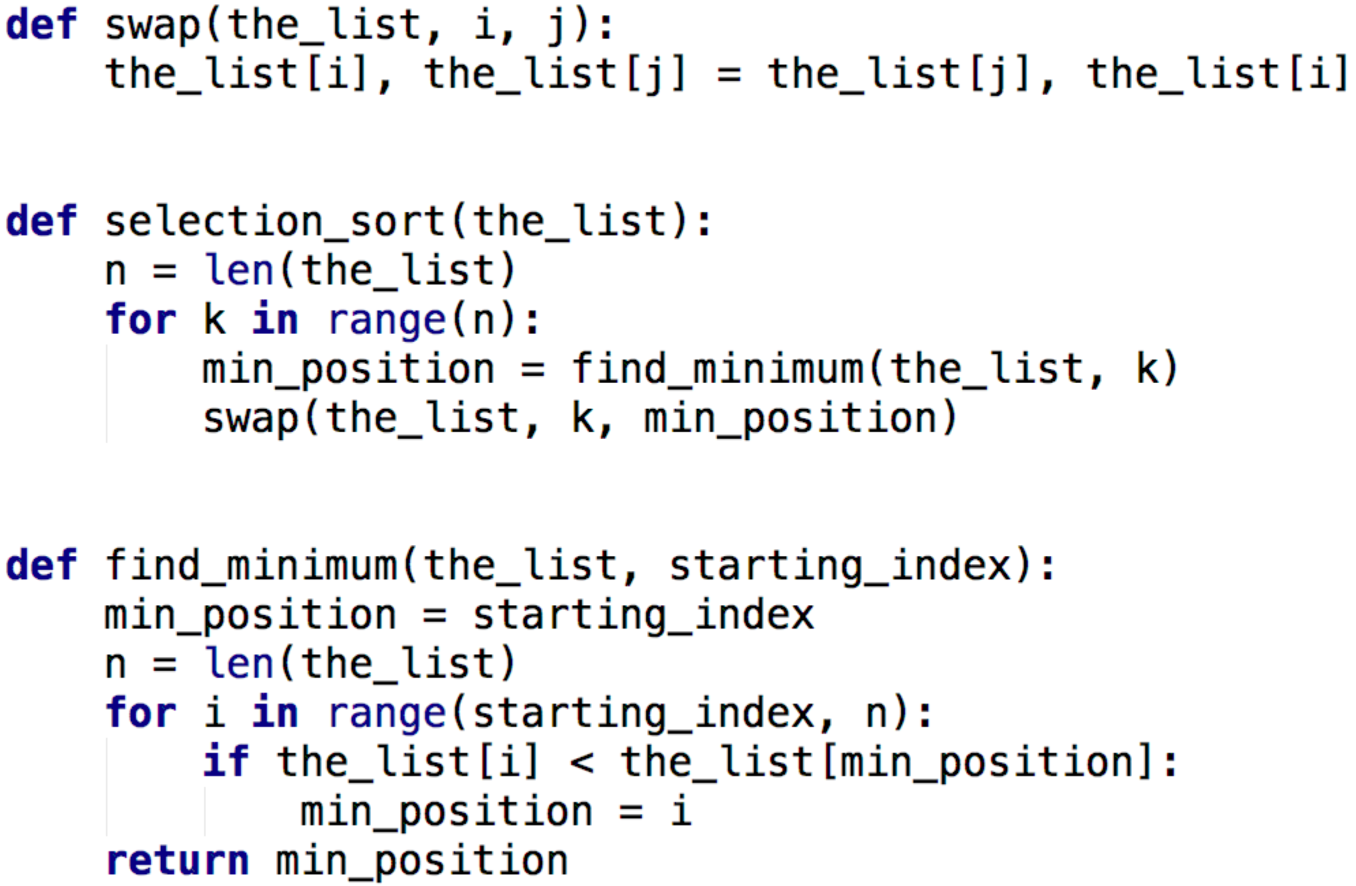
V = [A, B, C, D]





What is this unit all about?

- **We then encode the algorithm in a programming language**



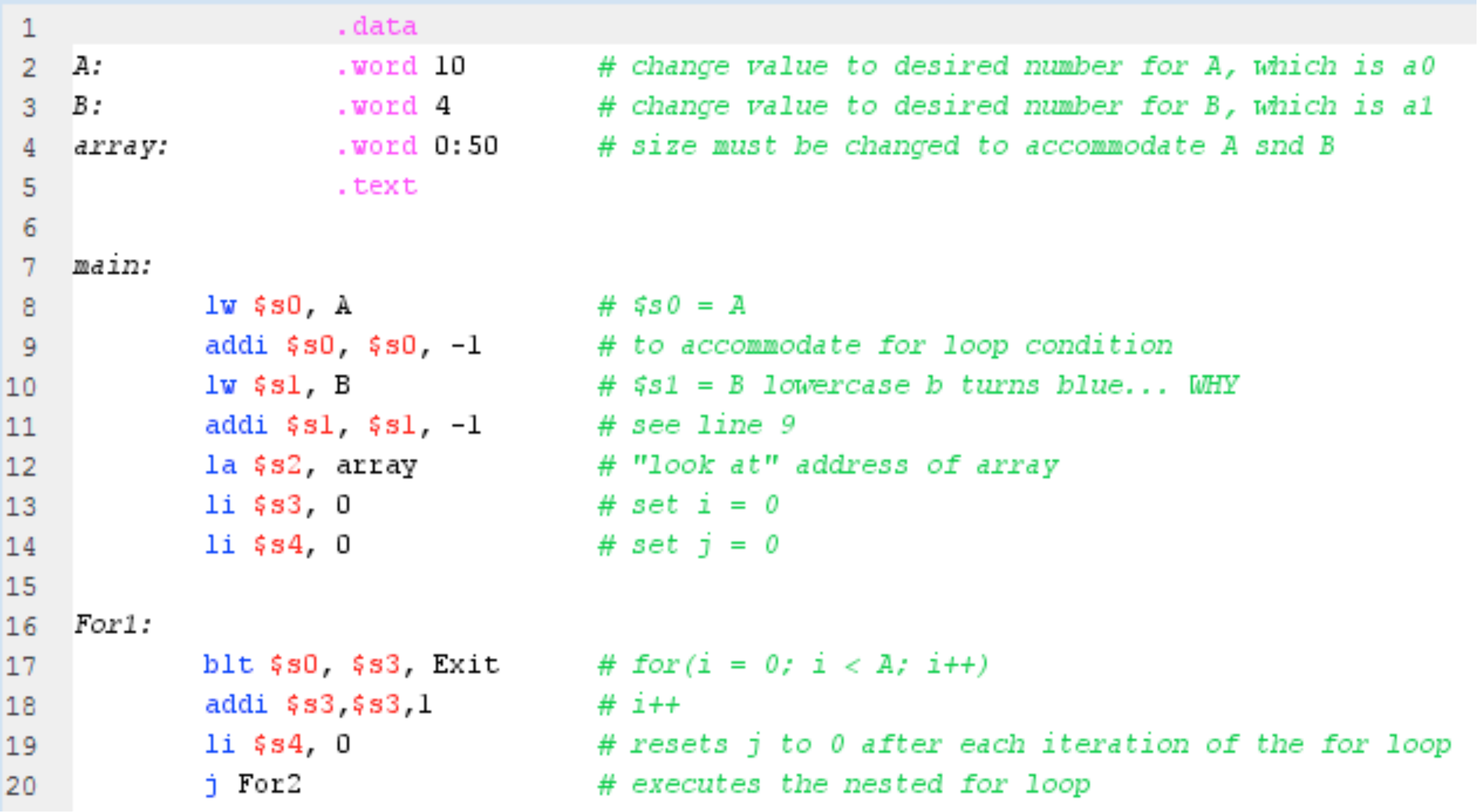
```python
def swap(the_list, i, j):
    the_list[i], the_list[j] = the_list[j], the_list[i]


def selection_sort(the_list):
    n = len(the_list)
    for k in range(n):
        min_position = find_minimum(the_list, k)
        swap(the_list, k, min_position)


def find_minimum(the_list, starting_index):
    min_position = starting_index
    n = len(the_list)
    for i in range(starting_index, n):
        if the_list[i] < the_list[min_position]:
            min_position = i
    return min_position
```

What is this unit all about?

- **Which is compiled/interpreted into assembly language**

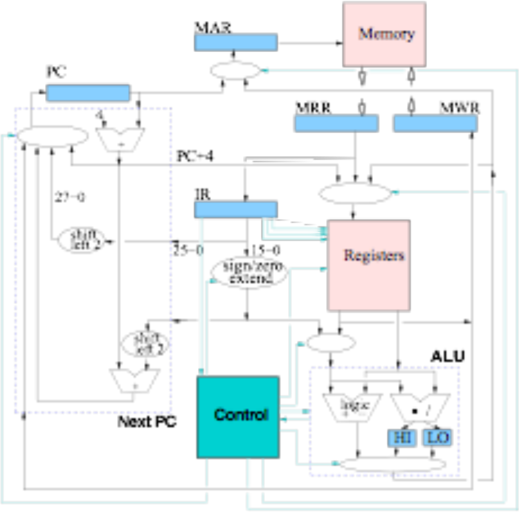


```
1                    .data
2   A:               .word 10       # change value to desired number for A, which is a0
3   B:               .word 4        # change value to desired number for B, which is a1
4   array:           .word 0:50     # size must be changed to accommodate A snd B
5                    .text
6
7   main:
8             lw $s0, A             # $s0 = A
9             addi $s0, $s0, -1     # to accommodate for loop condition
10            lw $s1, B             # $s1 = B lowercase b turns blue... WHY
11            addi $s1, $s1, -1     # see line 9
12            la $s2, array         # "look at" address of array
13            li $s3, 0             # set i = 0
14            li $s4, 0             # set j = 0
15
16  For1:
17            blt $s0, $s3, Exit    # for(i = 0; i < A; i++)
18            addi $s3,$s3,1        # i++
19            li $s4, 0             # resets j to 0 after each iteration of the for loop
20            j For2                # executes the nested for loop
```

What is this unit all about?

- **Then it is assembled and linked into machine language, and executed in a CPU**

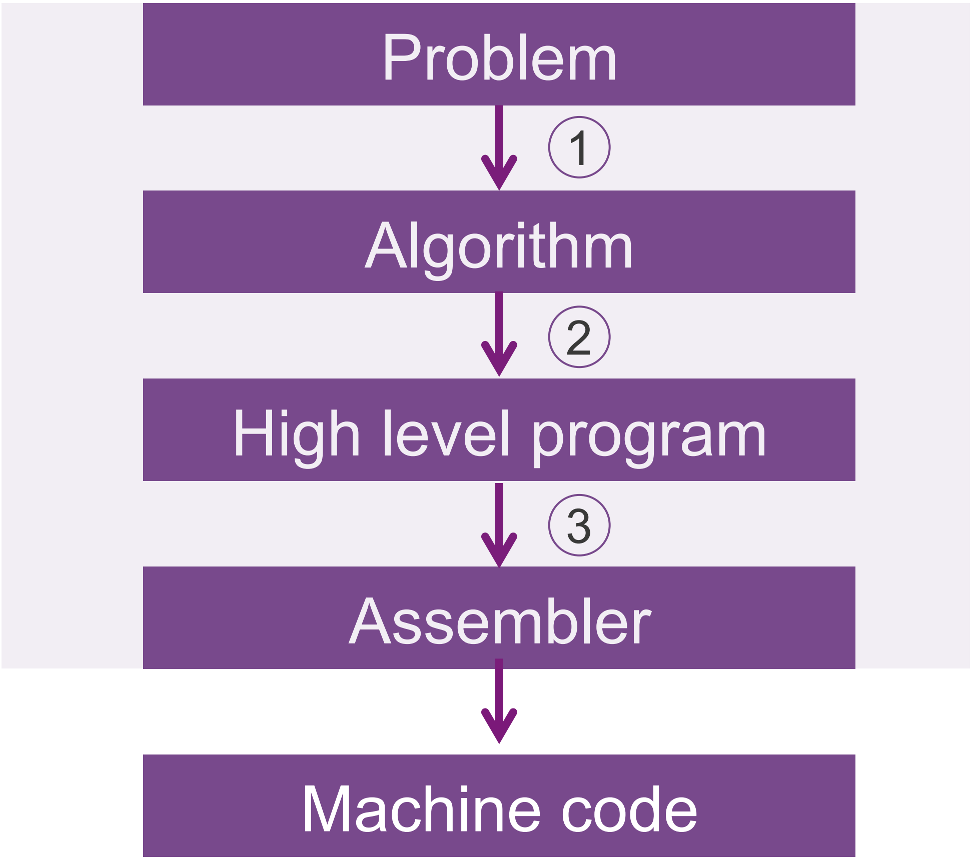
Machine language



CPU

What is this unit all about?

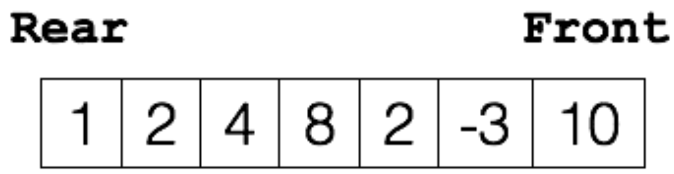
- **We will focus on basic knowledge that helps with the first three steps**



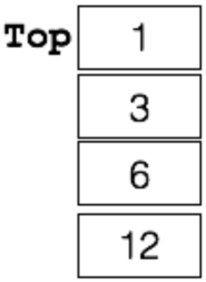
What you’ll get from FIT1008/FIT2085

- **Implement and modify many different data types**

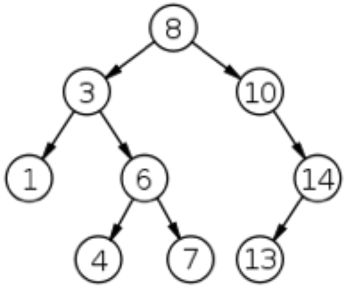
Some of the Data Types you will see



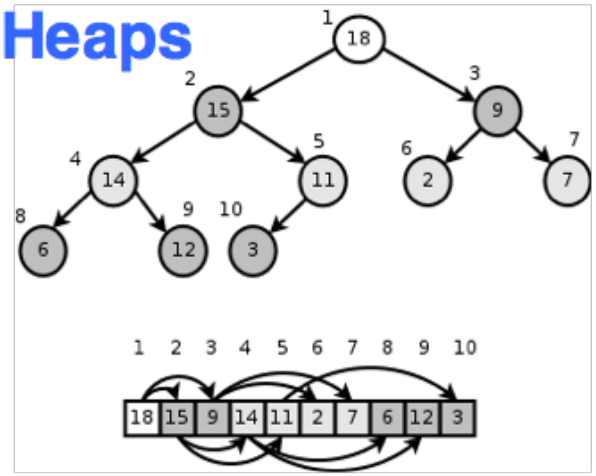
Queues



Stacks



Binary Search Trees



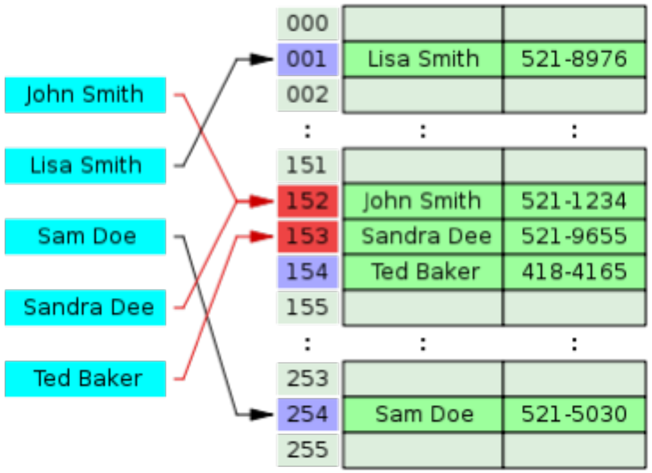
0. stop

1. pots

2. tops

Lists

Hash Tables



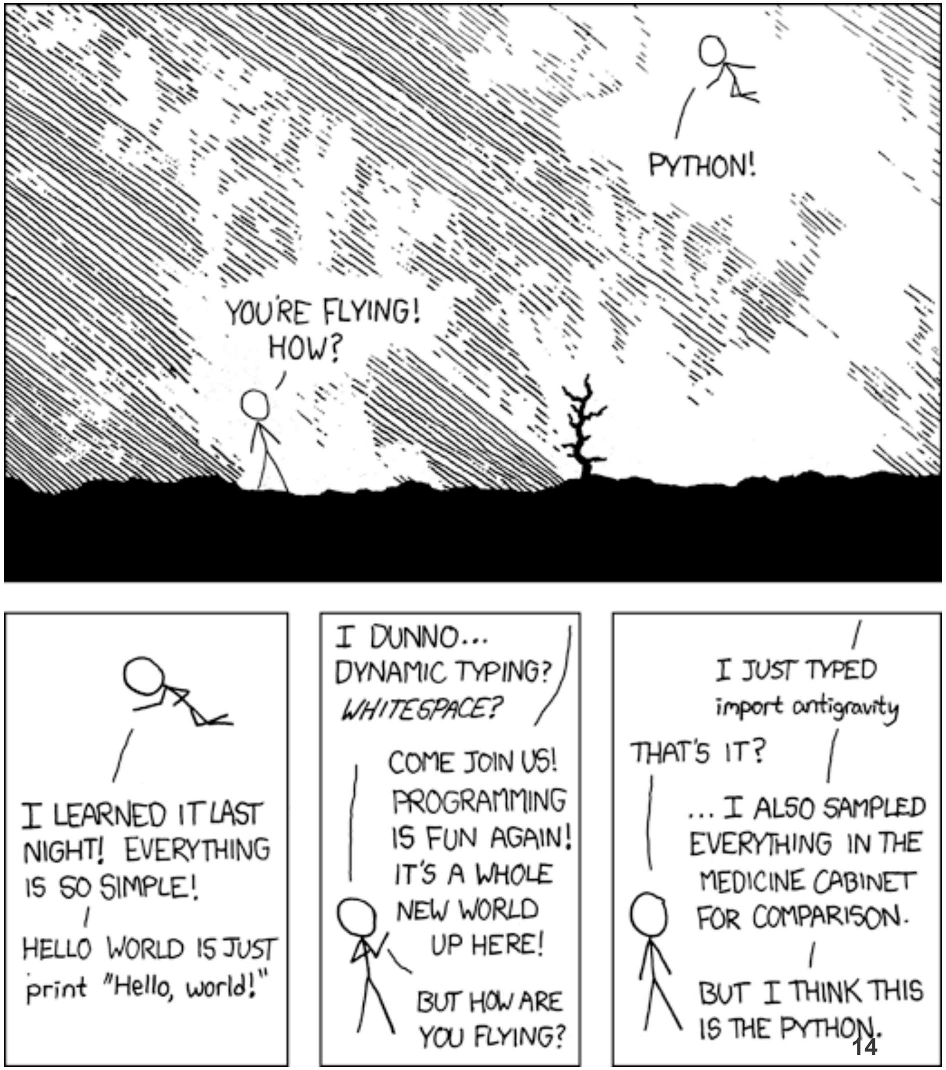
000		
001	Lisa Smith	521-8976
002		
:	:	:
151		
152	John Smith	521-1234
153	Sandra Dee	521-9655
154	Ted Baker	418-4165
155		
:	:	:
253		
254	Sam Doe	521-5030
255		

What you’ll get from FIT1008/FIT2085

- Implement and modify many different data types
- Evaluate/compare different implementations
- Design, implement and test algorithms

Programming Language to implement

- **We will use Python (3.3+) not JavaScript**
- **Why?**
  - General purpose (not web focused)
  - Simpler syntax, so better to learn basics
  - Has depth (mutli-paradigm)
  - Great libraries
  - Also very popular


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
...I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

xkcd.com

Important!

- **This course is about learning/practicing the CS/SE fundamentals**

This is NOT a Python course

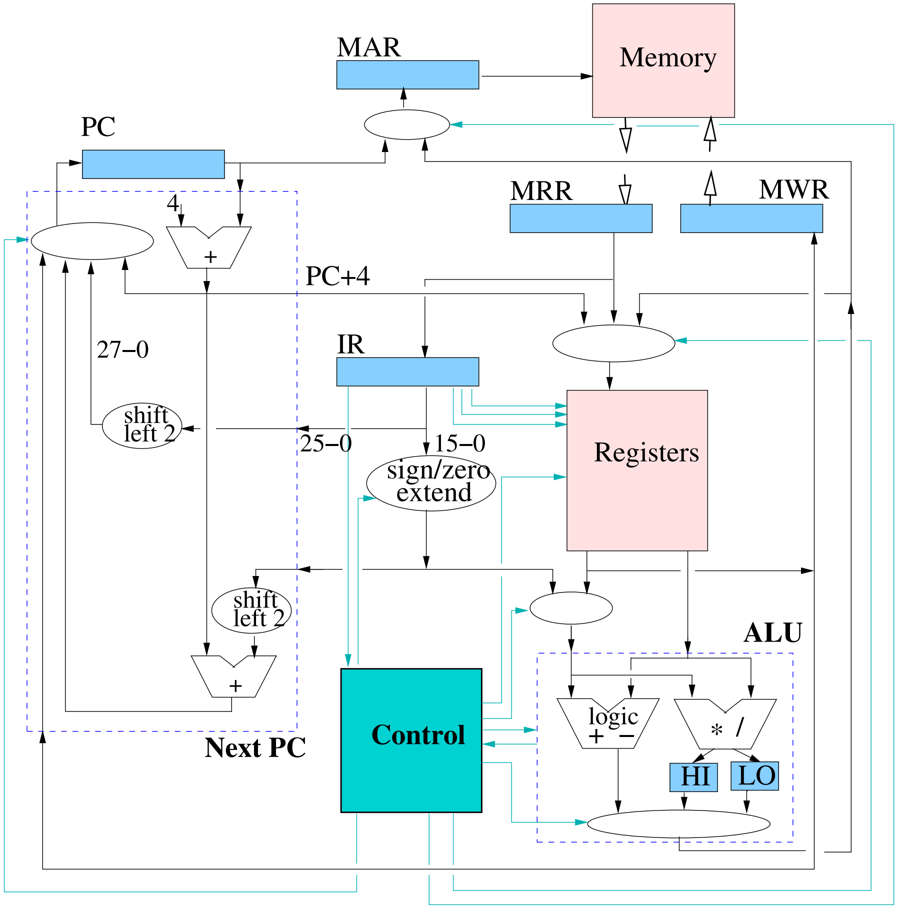
- **Python is only used as a tool to illustrate the concepts**
- **Some times we will have to bend it a bit…**
- **I am not a Python programmer expert!**

What you’ll get from FIT1008/FIT2085

- Implement and modify many different data types
- Evaluate/compare different implementations
- Design, implement and test algorithms
- Calculate the complexity of algorithms
- Translate high level code into assembly

Main aim: learn and/or practice the fundamentals

MIPS Architecture & assembly language



```
        .text
fact:   ori     $v0, $0, 1
        addi    $s0, $0, 1

loop:   slt     $t1, $s0, $t0

        bne     $t1, $s0, end

        mult    $v0, $t0
        mflo    $v0

        addi    $t0, $t0, -1

        j       loop

end:    jr      $ra
```

Things you should know about FIT2085

- **It used to be FIT1008 (taught in 1st year, 2nd semester)**
- **It has been taught to 2nd year Soft. Engs since S1 2017**
- **This has created some difficulties:**
  - Different background (know more and also less)
- **Bring up issues as soon as you see them**

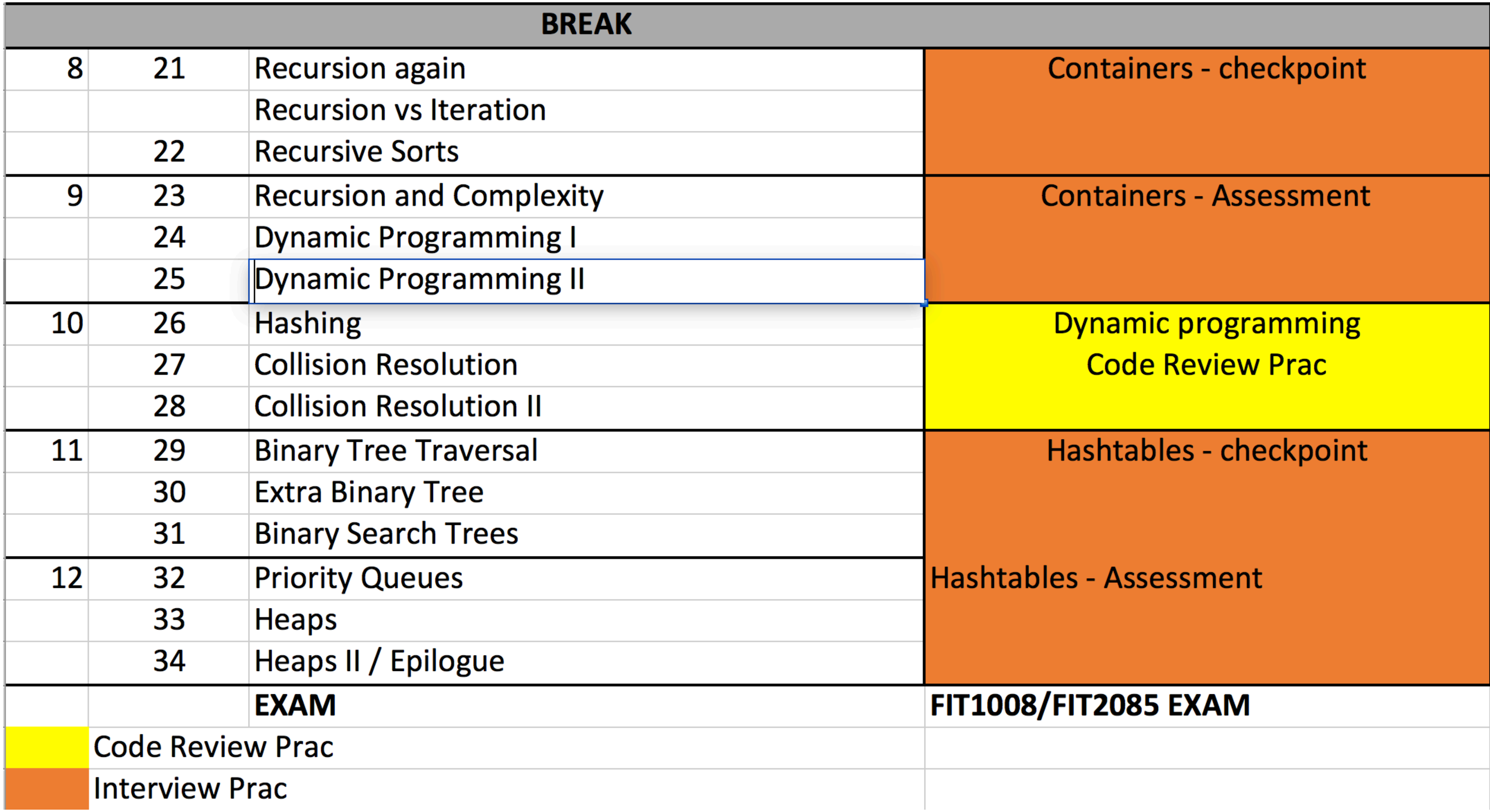
Weekly Timetable Synopsis

- **Lectures (3 hs): Mondays, Wednesdays and Thursdays**
- **Tutorial (1 h):**
  - Aim: discuss in group the concepts learned previous week
  - Prepare * questions or don’t come (affect others learning)
- **Pracs (3 s – need work before it, they are long!)**
  - Aim: individually practice the implementation of concepts
  - 3 fortnightly interview pracs:
    - **1st week**: reach the **checkpoint** and get feedback
    - **2nd week**: finish prac and get it **marked** (on site)
  - 5 weekly code review pracs:
    - **Code review** during the last hour, working in groups
  - **Must submit via Moodle** before leaving the lab (or no mark)

Preliminary Timetable

Week	Lecture		
1	1	Introduction	Simple Python & Algorithmics Code Review Prac
	2	MIPS Architechture	
	3	MIPS Simple programs	
2	4	MIPS memory	MIPS/MARS Code Review Prac
	5	Decisions in MIPS	
	6	Decisions in MIPS	
3	7	Functions MIPS (Part 1 - Calling)	MIPS - Checkpoint MIPS - Interview
	8	Functions MIPS (Part 2 - Returning)	
	9	Arrays in MIPS	
4	10	Complexity: Searching, Sorting	
	11	Sorting and Complexity II	
	12	Assertions, Exceptions, Testing	
5	13	ADT/Classes and Objects	Complexity - Experimental Code Review Prac
	14	Objects, variables and Scoping in Python	
	15	List Array & Sorted List	
6	16	Stacks and Queues with Arrays	Classes & Objects Testing Code Review Prac
	17	Linked Structures & Linked Stacks	
	18	Linked Queues	
7	19	**Mid Semester Test**	No Pracs
	20	Linked Lists	
		Iterators	

Preliminary Timetable



BREAK			
8	21	Recursion again	Containers - checkpoint
		Recursion vs Iteration	
	22	Recursive Sorts	
9	23	Recursion and Complexity	Containers - Assessment
	24	Dynamic Programming I	
	25	Dynamic Programming II	
10	26	Hashing	Dynamic programming Code Review Prac
	27	Collision Resolution	
	28	Collision Resolution II	
11	29	Binary Tree Traversal	Hashtables - checkpoint
	30	Extra Binary Tree	
	31	Binary Search Trees	
12	32	Priority Queues	Hashtables - Assessment
	33	Heaps	
	34	Heaps II / Epilogue	
		EXAM	**FIT1008/FIT2085 EXAM**
🟨	Code Review Prac		
🟧	Interview Prac		

Time Requirements

- 3 hours lectures per week
- 1 hour tutorial per week
- 3 hour pracs per week

… plus preparation at home

About 5 hours more per week!

Assessment

But it’s unfair that my prac is on a Monday and other people have theirs on a Friday! They get so much more time than me!

- **Interview pracs (30%)**
- **Code review reports (5%) for code review pracs**
- **Mid Semester Test (5%) Week 7 during a lecture**
- **Exam (3 hours) (60%)**

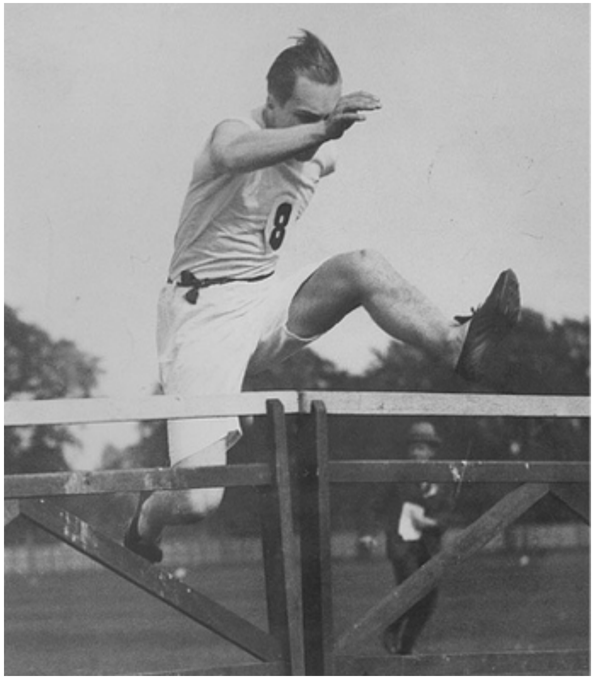
- **Note, weekly quizzes are available but not marked**

Everyone has the same amount of time

- **Every student has the same time to complete each prac**
  - From the day it finishes the previous prac
  - To the day it finishes the current prac
- **So, students with a Friday prac do not have more time than those with a Thursday prac, etc**
- **The same goes for the tutes**

Hurdles

- **Each student must obtain:**
  - At least 50% of the total in-semester assessment.
  - At least 50% of the exam marks.
  - An overall unit mark of 50% or more
- **If a student does not pass these hurdles then a mark of no greater than 49N will be recorded for the unit**



More on interview & code-review pracs

- **Demonstrator decisions are final**
  - Take up disagreements with me (I set the marking guide)
- **At the end of each one you must submit your solutions**
  - You must compress your source files and associated documentation in one zip file named as follows:

<STUDENTID>_CHECKPOINT_<N>.zip
<STUDENTID>_INTERVIEW_<N>.zip
<STUDENTID>_REVIEW_<N>.zip

eg. 123456789_CHEPOINT_1.zip.

- **Remember: START EARLY**
- **Keep a copy of tasks completed for your records.**

Missed Pracs and Tutes

- **If you miss a prac you will be marked ABSENT**
  - This means you will not be marked
- **If you had an illness or emergency AND you**
  - Obtain Medical Certificate or Police Accident Report
  - Apply for special consideration: https://goo.gl/forms/5Ti5AC6xvcLC8fL03
    Or email role account: FIT1008.Clayton-x@monash.edu
  - Get approval to the form/email
  - Then we will work out what to do next (submit later, etc)
- **No late submissions of pracs are allowed unless explicitly approved**

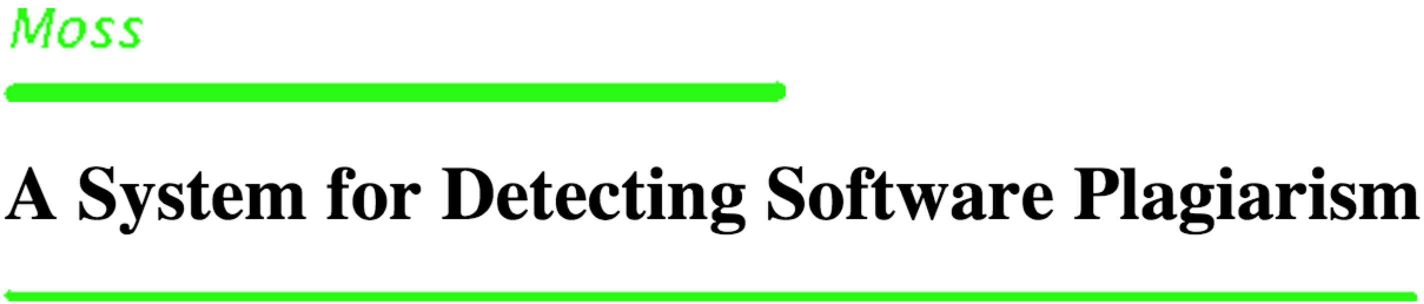
Cheating, Collusion, Plagiarism

- Cheating: Seeking to obtain an unfair advantage in an examination or in other written or practical work required to be submitted or completed for assessment.
- Collusion: Unauthorised collaboration on assessable work with another person or persons.
- Plagiarism: To take and use another person's ideas and or manner of expressing them and to pass them off as one's own by failing to give appropriate acknowledgement. This includes material from any source, staff, students or the Internet – published and un-published works.

http://infotech.monash.edu.au/resources/student/assignments/policies.html

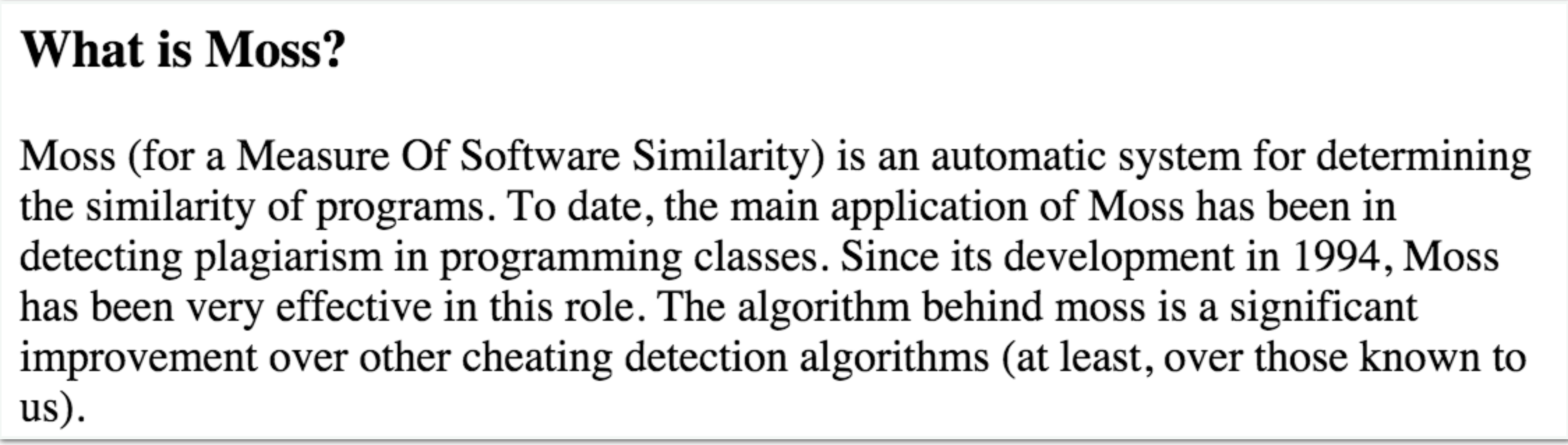
Cheating, Collusion, Plagiarism

- **Monash University takes these matters very seriously. There are severe penalties for them.**
  - http://www.monash.edu/students/academic/policies/academic-integrity
- **It is OK to work together discussing your pracs, but each student must write the entire assignment alone and be able to explain and modify it on request.**
- **This will be determined during the interview:**
  - The interview will determine whether your prac mark is multiplied by a 0, a 0.5, or 1 indicating no able, sometimes able but not all, and able, respectively



Moss

A System for Detecting Software Plagiarism



What is Moss?

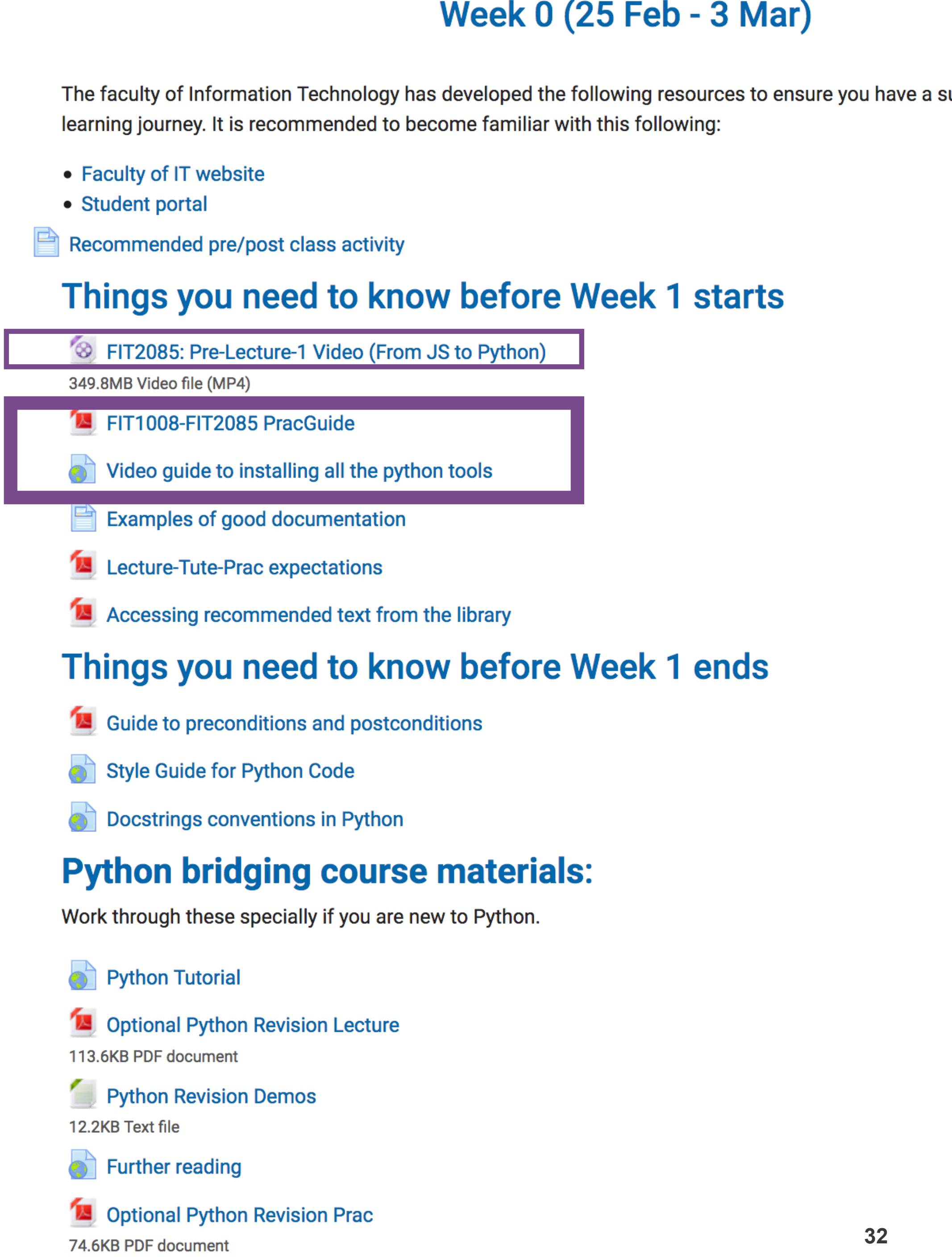
Moss (for a Measure Of Software Similarity) is an automatic system for determining the similarity of programs. To date, the main application of Moss has been in detecting plagiarism in programming classes. Since its development in 1994, Moss has been very effective in this role. The algorithm behind moss is a significant improvement over other cheating detection algorithms (at least, over those known to us).

https://theory.stanford.edu/~aiken/moss/

http://bit.ly/plagiarism_video

Week 0

- **Has important info**



Week 0 (25 Feb - 3 Mar)

The faculty of Information Technology has developed the following resources to ensure you have a su
learning journey. It is recommended to become familiar with this following:

- Faculty of IT website
- Student portal

Recommended pre/post class activity

Things you need to know before Week 1 starts

FIT2085: Pre-Lecture-1 Video (From JS to Python)
349.8MB Video file (MP4)

FIT1008-FIT2085 PracGuide

Video guide to installing all the python tools

Examples of good documentation

Lecture-Tute-Prac expectations

Accessing recommended text from the library

Things you need to know before Week 1 ends

Guide to preconditions and postconditions

Style Guide for Python Code

Docstrings conventions in Python

Python bridging course materials:

Work through these specially if you are new to Python.

Python Tutorial

Optional Python Revision Lecture
113.6KB PDF document

Python Revision Demos
12.2KB Text file

Further reading

Optional Python Revision Prac
74.6KB PDF document

Come to the lectures and participate!

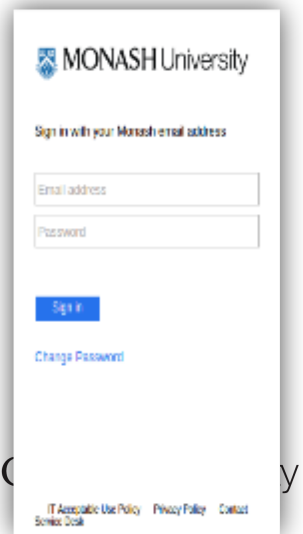
- Pracs are only 30% of your final mark
- So, do not miss lectures or tutes to finish a prac
- Most who came to lectures passed (with a high mark)
- So, get interested, engaged, read, think, discuss, and have fun!
- In the process you will learn … a lot!

Recommended Reading

- **Slides are not enough: read the recommended texts!**
- **For MIPS:**
  - R.L. Britton - MIPS Assembly Language Programming
- **For data structure and algorithms:**
  - B. Miller and D. Ranum - Problem solving with algorithms and data structures using Python https://interactivepython.org/runestone/static/pythonds/index.html
  - M.T. Goodrich, R. Tamassia, M.H. Goldwasser - Data Structures and Algorithms in Python
- **For Python:**
  - http://docs.python.org/3/tutorial/
  - http://openbookproject.net/thinkcs/python/english3e/ chapters 1-7

Using MARS for answering questions

1. Visit http://mars.mu on your phone, tablet or laptop
2. Log in using your Authcate details
3. Touch the + symbol
4. Enter the code for your unit: XVKLMH
5. Answer questions when they pop up
6. Give yourself a Display Name &manage your feed subscriptions



MONASH University

Sign in with your Monash email address

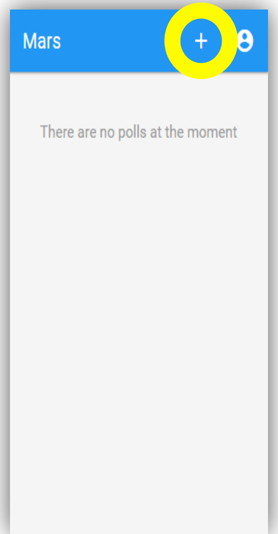
Email address

Password

Sign in

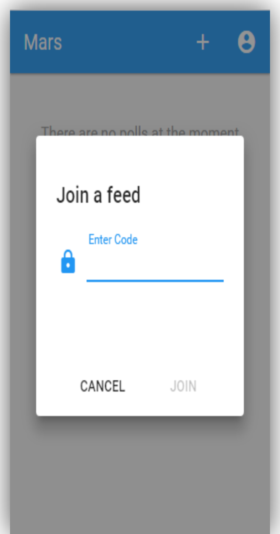
Change Password

IT Acceptable Use Policy Privacy Policy Contact Service Desk



Mars

There are no polls at the moment



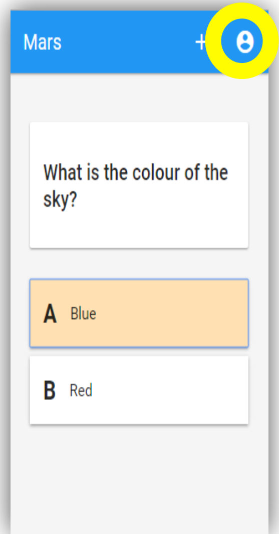
Mars

There are no polls at the moment

Join a feed

Enter Code

CANCEL JOIN



Mars

What is the colour of the sky?

A Blue

B Red

Noticeboards

- **Everything in the unit is handled through Moodle**

You will receive e-mails from Moodle
Pay them attention!

Communication

- **For non-urgent matters:**
  - Consultation and forums
- **For urgent matters: use e-mail**

All e-mails must start with [FIT1008/FIT2085] in their “Subject”. Failure to do this risks the e-mail being discarded!

- **For issues with the unit:**
  - Start with your Tutor (if it relates to a tute/prac)
  - Continue with your Lecturer (me)
  - If unresolved, contact Course Director

Getting Help at Clayton

- **Consultations: soon to be announced**
  - Waiting for all enrolments to decide best times
- **For FIT General Enquires:**
  - Go to the General Office (Building 63)
- **For Software Engineering General Enquires**
  - Talk to the Course Director (Yuan-Fang Li)
- **Can also talk to your student representative**
  - They will participate in the Staff/Student meetings

For contact details, go to Monash website

Disability Support Services

Do you have a disability, medical or mental health condition that may impact on your study?

Disability Support Services provides a range of services for registered students including:

- Note takers and Auslan interpreters
- Readings in alternative formats
- Adaptive equipment and software
- Alternative arrangements for exam and class tests

Disability Support Services also support students who are carers of a person with a disability, medical or mental health condition, or who is aged and frail.

For further information and details about how to register:

T: 03 9905 5704
E: disabilitysupportservices@monash.edu
monash.edu/disability

Summary

- **You now know the big picture regarding the unit:**
  - Teaching team
  - What it is the unit all about
  - Timetable and structure
  - Recommended reading
  - Assessment and hurdles
  - Communication
  - Use of MARS
  - Special consideration
  - Cheating, collusion, plagiarism
  - Getting help
  - Disability support
  - And much more!

The following is important info for your pracs

Programming Expectations for Pracs

- **Good code layout (Style Guide for Python in Week 0)**
- **Consistent and meaningful variable names**
- **Accurate and informative comments**
  - At the beginning of the file/class/module
  - At the beginning of every method/function
  - Brief inline comments if needed (for clarifying the control)
- **Clear logic in loops and if-then-elses**
- **No redundant or unnecessarily repeated code (modularise and reuse!)**
- **No overcomplicated code (as simple as it can be)**
- **Appropriate tests for every method/function**

Minimal file comments

- Name of file
- Brief description of what it does
- Programmer’ s name
- Start date of coding (and status)
- List of variable and function names with description
- How to run the testing code
- List of modifications, dated and signed
- Known bugs (faults)

Commenting your methods:

- **Precondition: properties of data that must be true before a piece of code (method, loop, etc) is executed**
  - If violated (not true), the result of the computation is undefined (unknown)
- **Postcondition: any change in data or I/O, that results from the code being executed**
  - If violated, there is a bug in the code
- **Pre/Post guide in Moodle (under Week 0)**
- **Every function/method should include:**
  - Preconditions and postconditions (if any)
  - Brief explanation of how the postcondition is achieved
  - Complexity in Big O notation (once we study it)

Testing your methods

- **Each method A should be accompanied by another method (say, testA) which tests A**
- **Method testA will use assertions (see Lecture 1) to:**
  - call A with a series of inputs (or “test cases”)
  - print a message if the answer is not the expected
- **The number of possible test cases is often too big**
- **Idea: use the smallest number of test cases that maximises the chances of finding a bug**
- **How to select those ones?**

Testing your methods (continued)

- **There are many approaches. We would like you to try:**
  - Equivalence testing: divide them into equivalent classes
  - Boundary analysis: test the boundaries of each class
- **Example: check whether a list is empty**
- **Two classes:**
  - Inputs that yield True: List of 0 elements
  - Inputs that yield False: List of 1 or more elements
- **Boundaries:**
  - For true: list with 1 element (no list with -1)
  - For false: list with 0 and 2 elements
- **This gives us the following four test cases:**
  - 0, 1 and 2 elements
  - Many elements (say 5)

the first one is expected to return True, the others False

Testing your functions/methods (cont)

- **We will not treat test case creation as an exact science**
- **In other words, there are often many correct answers:**
  - one can use different classes and different boundaries
- **Properly testing every function will initially be too much (it takes time!), but you should try some**
- **We will expect a reasonable number of varied set of cases by early-mid semester**
- **We will let you know when your mark depends on it**