Prepared by Julian García, based on material by
David Albrecht and María García de la Banda

Lecture 1
Introduction

FIT 1008
Introduction to Computer Science

Objectives of the unit

- Implement and modify **data types**.
- Compare and **evaluate** different **implementations of data types**.
- Design and implement **algorithms**.
- Calculate **complexity** of simple algorithms.
- Manually translate **high level code into assembly**.

What is this unit about

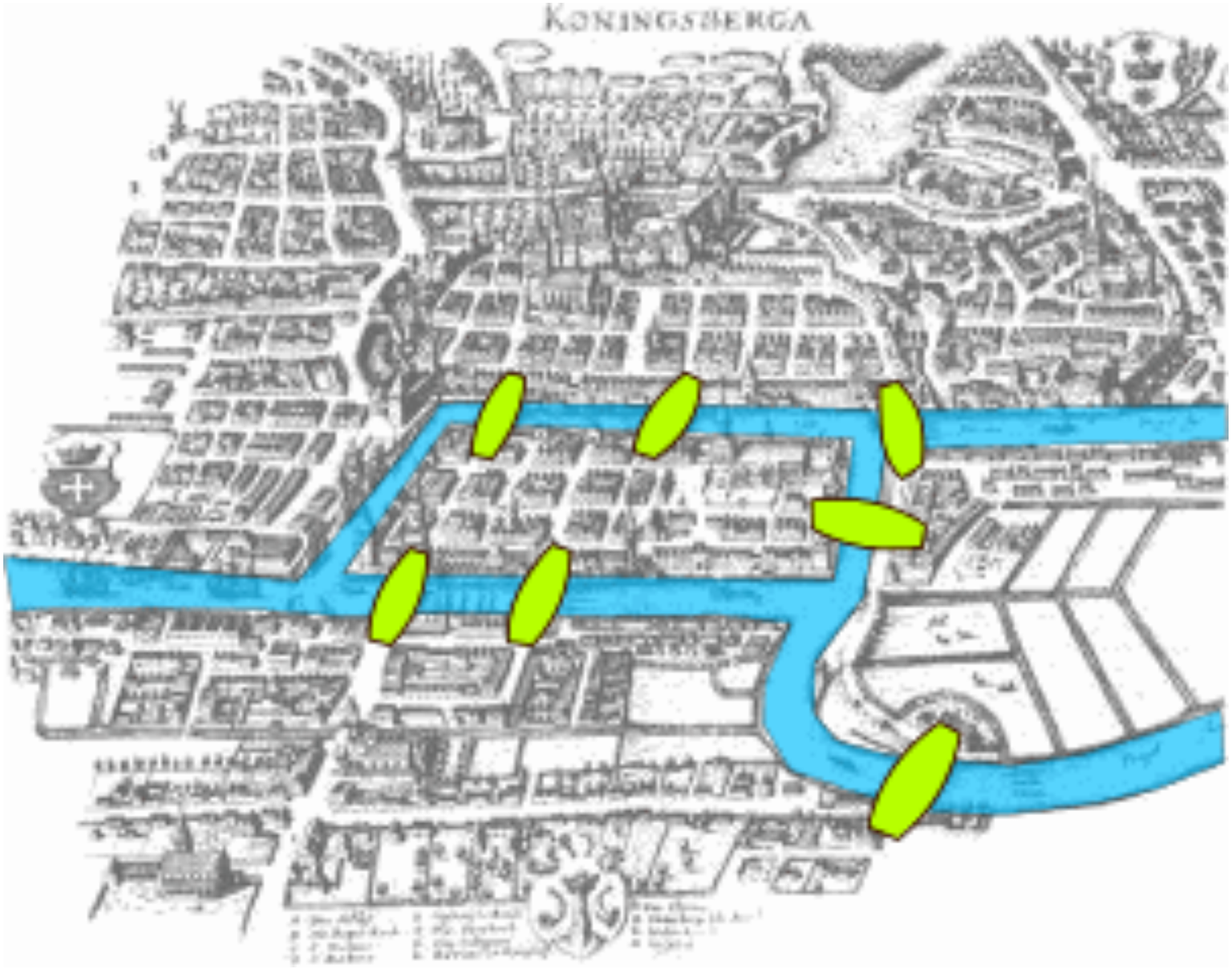
Rules of the house…

Some advice

Teaching team

Computer Science is **NOT** just about **Programming**

- Hardware
- Computer Systems Organisation
- Software and Data
- Theory of Computation
- Mathematics of Computing
- Analysis of computing methods
- Information Systems
- Computing Applications
- Computing Culture


KONINGSBERGA

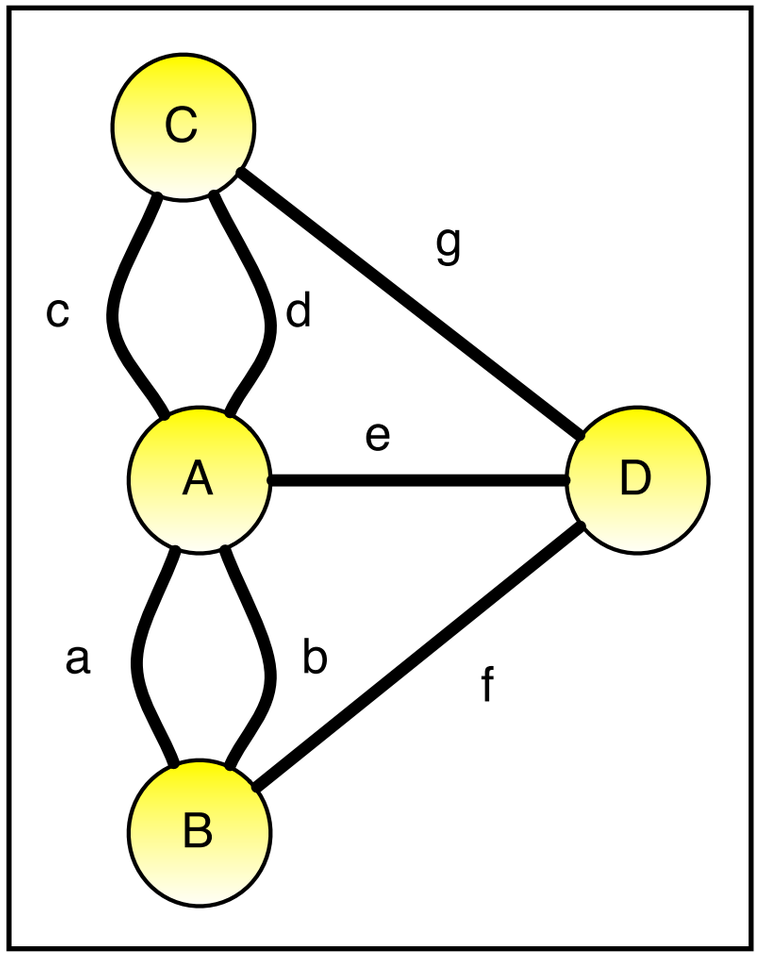
We put the vertices in a list…

$$V = [A, B, C, D]$$
$$\quad\ 0\ \ 1\ \ 2\ \ 3$$

Eulerian(V[0, n-1])

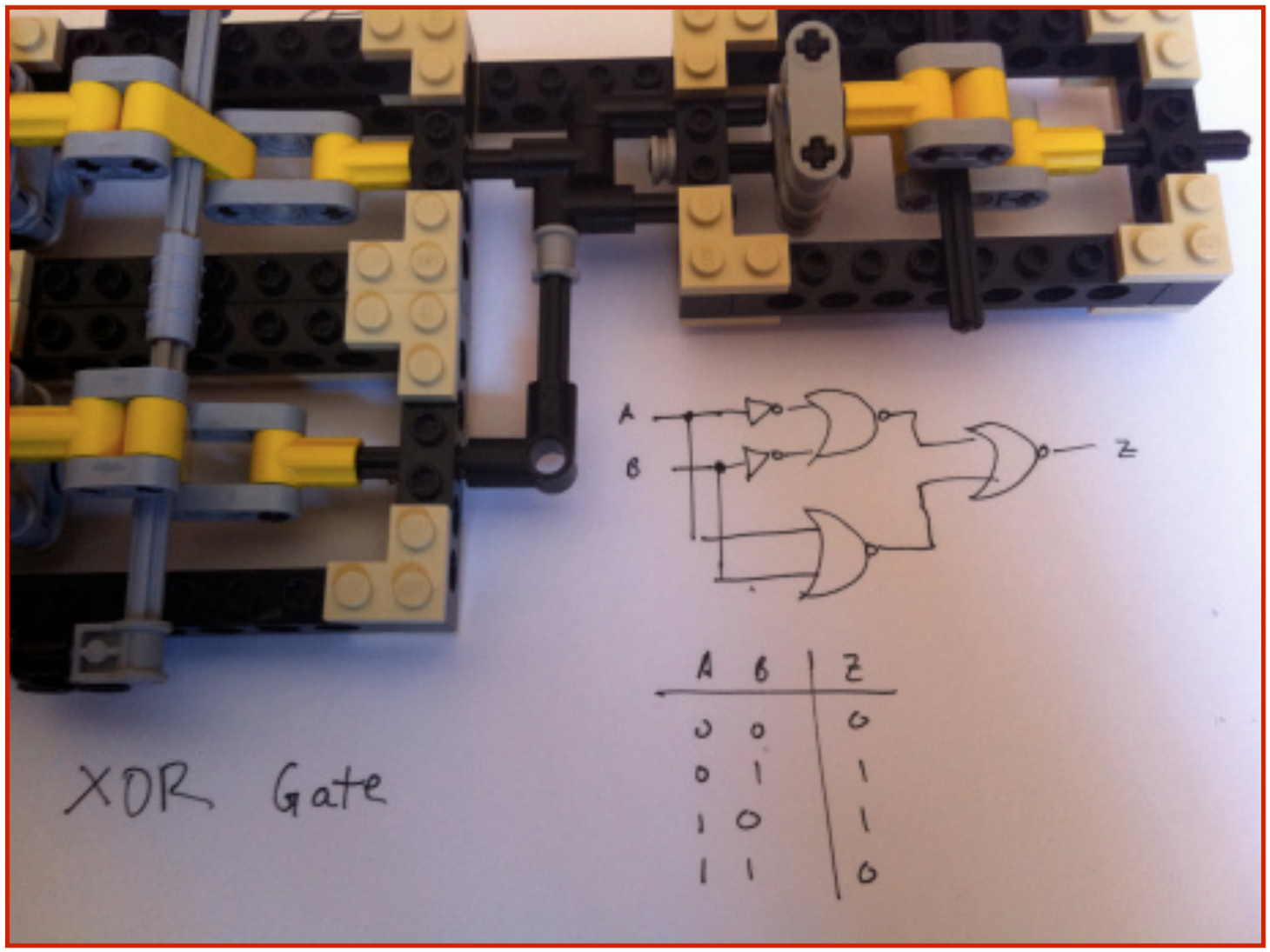
Input: A list of *n* vertices.

Output: **True** if the graph is Eulerian, **False** otherwise

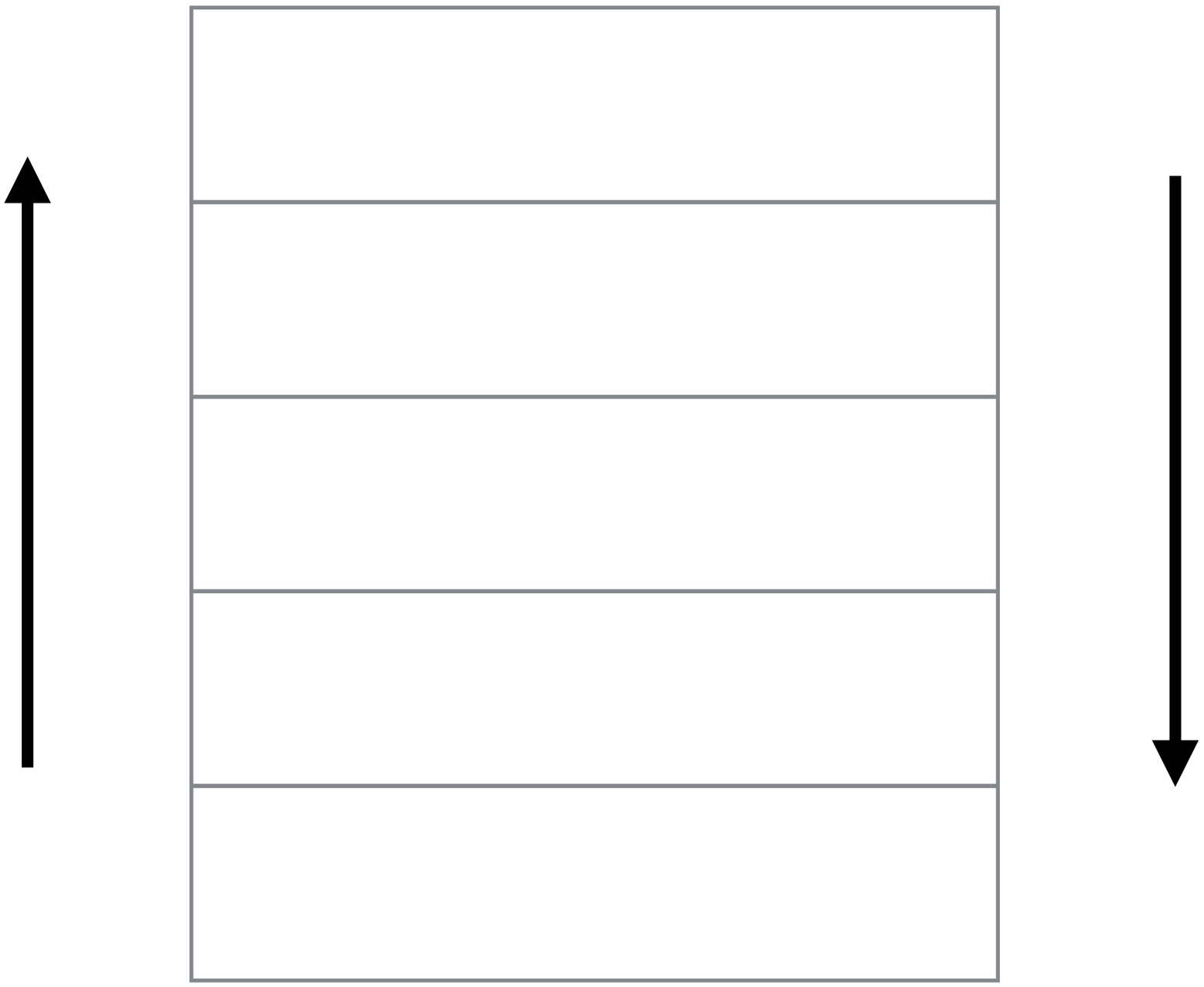



```
v←0
while(v < n)
{
    if( degree(L[v]) is odd)
    {
        return FALSE
    }
    v←v+1
}
return TRUE
```

"all computation done by large combinations of **on-and-off** switches, wired together in meaningful ways"



https://keshavsaharia.files.wordpress.com/2011/05/img_0270.jpg?w=460&h=343



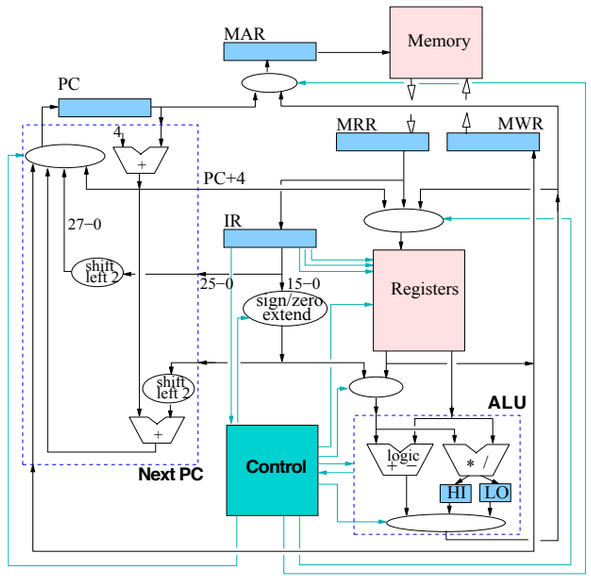
You

???

```
f = 1
n = int(input("Enter int:"))
while n > 0:
    f = f*n
    n = n-1
print(f)
```

Human-readable code

Machine language



CPU

High level programming language

Assembly Language Program

Machine language

```
def find_duplicates(a_list):
    n = len(a_list)
    k = 0
    while k < n:
        j = k + 1
        while j < n:
            if a_list[k] == a_list[j]:
                print(a_list[k])
            j += 1
        k += 1
```

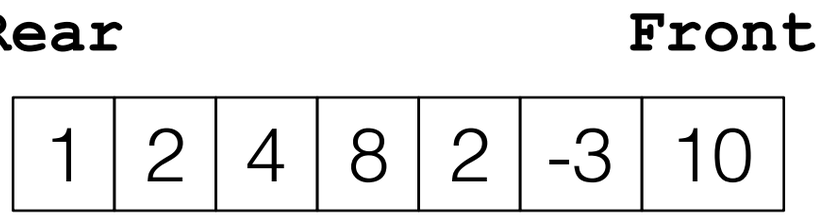
```
main:   # 1 * 4 = 4 bytes local.

        addi $fp, $sp, 0

        addi $sp, $sp, -4

        sw   $0, -4($fp) # n = 0

        addi $v0, $0, 5

        syscall

        sw $v0, -4($fp)  # n
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

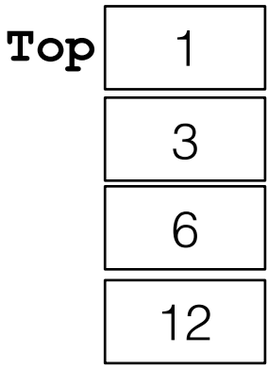
Programs = Algorithms + Data Structures

(Programming is still important)

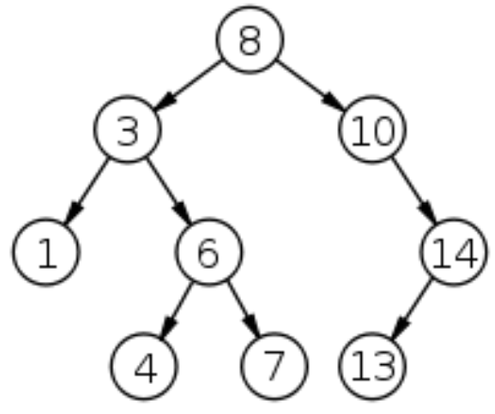
Data Structures



Queues

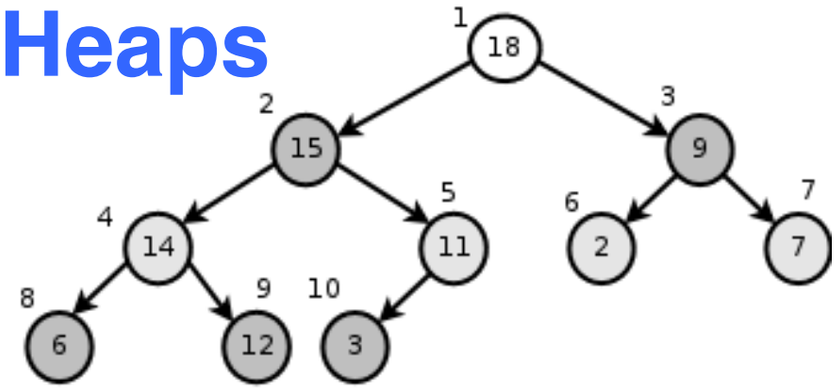


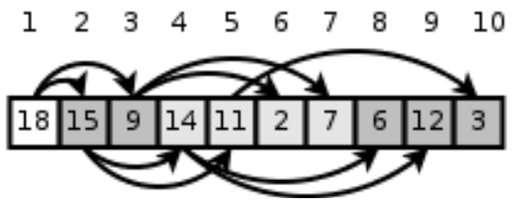
Stacks



Binary Search Trees

Heaps

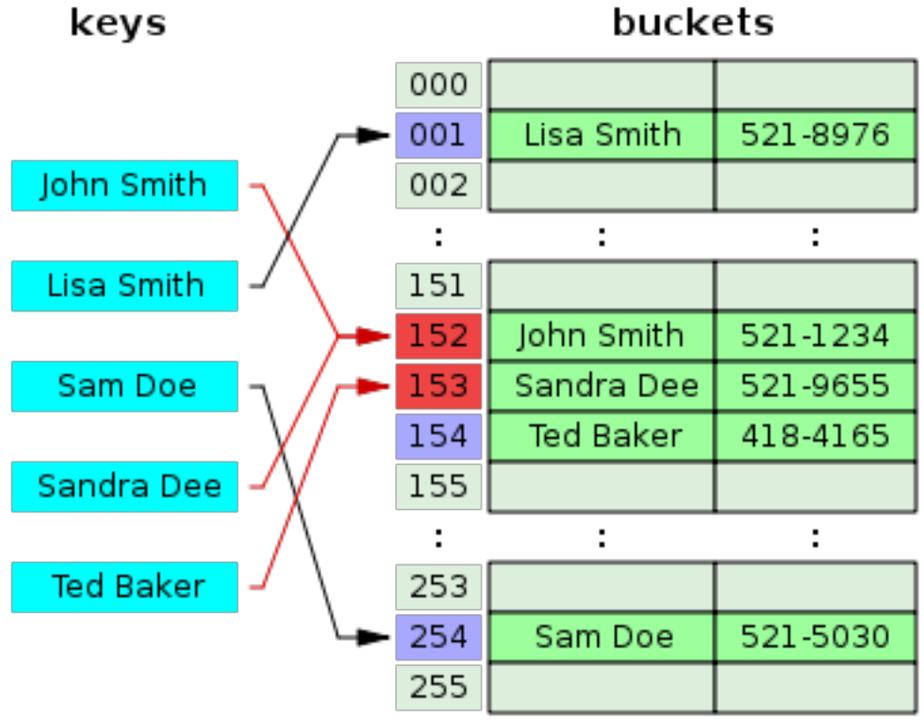


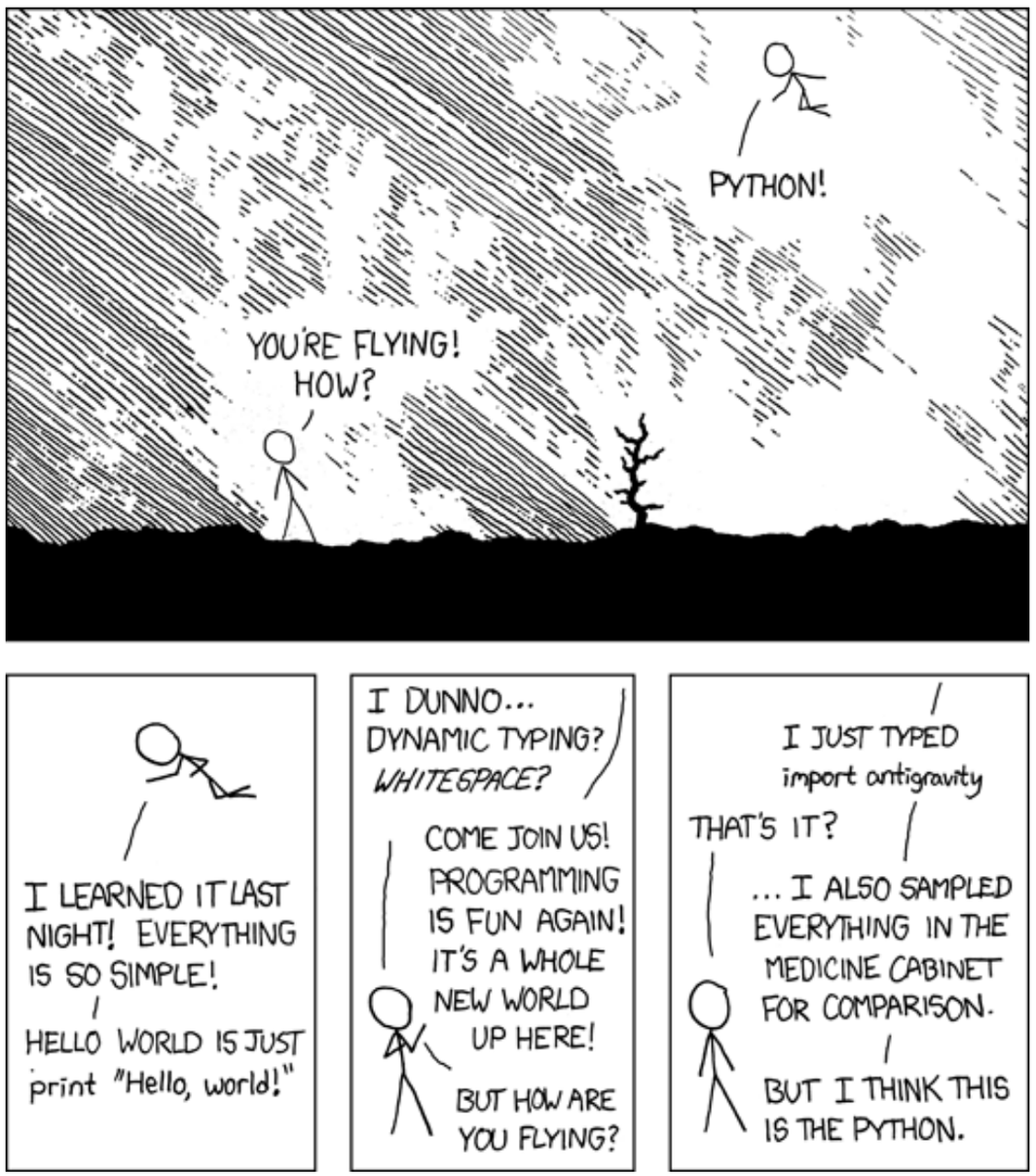


0. stop
1. pots
2. tops

Lists

Hash Tables





- We will use Python to **implement the algorithms.**
- We will use Python to **implement the data structures.**
- However this is **NOT a Python Course**. You do not need to know all the Python details.
- Disclaimer: I am **NOT** a Python expert.

Why Python

- **General** purpose
- Very popular with many libraries
- Very **easy** to program
- Has **depth** if you need it
  - Multi-paradigm: OO, functional, and imperative
  - Associated programming concepts: objects, higher order, etc.

```
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
```

Tentative Timetable

Week	Lecture		
1	1	Intro and House Rules, A simple python program	Simple Python & Algorithmics Workshop
	2	MIPS Architechture	
	3	MIPS Simple programs	
2	4	Decisions in MIPS	MIPS/MARS Workshop
	5	Iteration	
	6	Arrays in MIPS	
3	7	MIPS Memory	MIPS Prac - Checkpoint
	8	Functions MIPS (Part 1 - Calling)	
	9	Functions MIPS (Part 2 - Returning)	
4	10	Complexity: Searching, Sorting	MIPS Prac - Assessment
	11	Sorting and Complexity II	
	12	Assertions, Exceptions, Testing	
5	13	ADT/Classes and Objects	No Pracs
	14	Objects, variables and Scoping in Python	
	15	**Mid-semester test**	
BREAK			

Tentative Timetable

6	16	List Array & Sorted List	Complexity Workshop - Experimental
	17	Stacks and Queues with Arrays	
	18	Linked Structures & Linked Stacks	
7	19	Linked Queues	Classes & Objects Workshop / Testing
	20	Linked Lists	
		Iterators	
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 workshop
	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	

What is this unit about

Rules of the house…

Some advice

Teaching team

Timetable Synopsis

Lectures

- **Monday** 11:00, 19 Anc **G31**.
- **Tuesday** 8:00, 19 Anc **G31**.
- **Wednesday** 9:00, 19 Anc **G31**.

Pracs: Once a week.

- **3 Interview Pracs:**
  - Each assessed Prac runs over <u>two weeks.</u>
  - <u>First week</u>: aim to reach the **checkpoint** (**<u>hurdle</u>**) and get feedback.
  - <u>Second week</u>: Whole prac will be marked.
  - Weeks 3 and 4, 8 and 9, 11 and 12.
- **5 Workshops:** All other weeks (except week 5)
  - Prepare before the workshop.
  - **<u>Code review</u>** at the end of the workshop. Working in groups. Ends with group presentation.
    <u>Must happen at the end of the prac, must be present.</u>

Tutes:

Once a week (1 hour) / Must show to demonstrator worked out answers or attempts **before start**.

Assessment

- Assessed practical sessions (20%)
- Code review reports (5%) - during workshops
- Student participation — via **quizzes** (5%) + tutorial prep hurdle
- Mid Semester Test (10%) — Week 7 during a lecture.
- Exam (3 hours) (60%)

Pracs

At Clayton, BYOD.

Revision

Revision material

Introduction to algorithms

New material that you should start looking into

Guide to preconditions and postconditions

Python Tutorial

Style Guide for Python Code

Docstrings conventions in Python

Things you need to know

FIT1008 PracGuide

Video guide to installing all the python tools

Examples of good documentation

Lecture-Tute-Prac expectations

Accessing recommended text from the library

Python Tutorial

Python bridging course materials: work through these specially if you are new to Python.

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

Submitting Pracs

- At the end of **each prac** you must submit your solutions (for both assessed & non-assessed pracs).

- You must compress your source files and associated documentation in one **zip file**.

- You **must** name your zip file with the following format:

<STUDENTID>_PRAC<N>.zip

- eg. 123456789_Prac1.zip.

- **START EARLY**

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

http://bit.ly/plagiarism_video

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/

Special consideration

http://www.monash.edu.au/exams/special-consideration.html

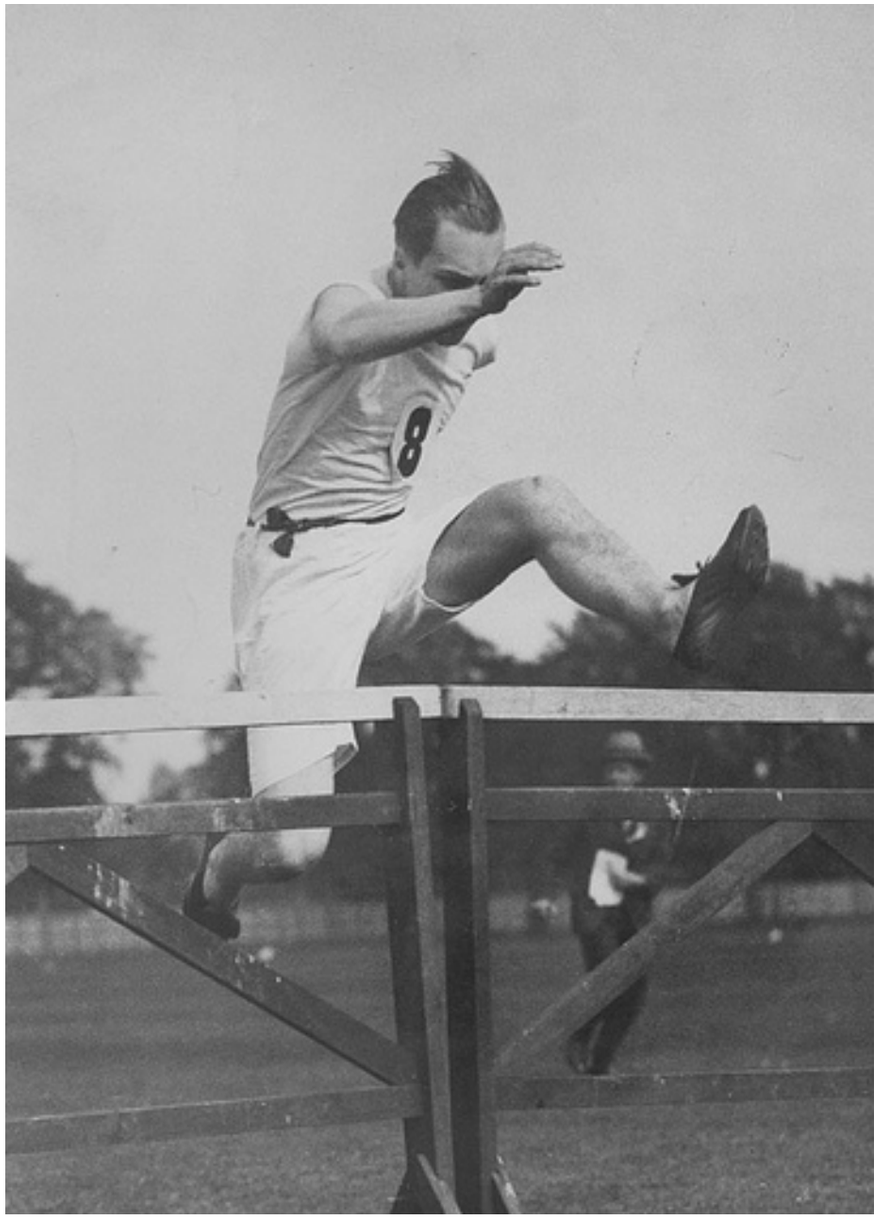
Send scanned form + support to brendon.taylor@monash.edu

No exception

Hurdles

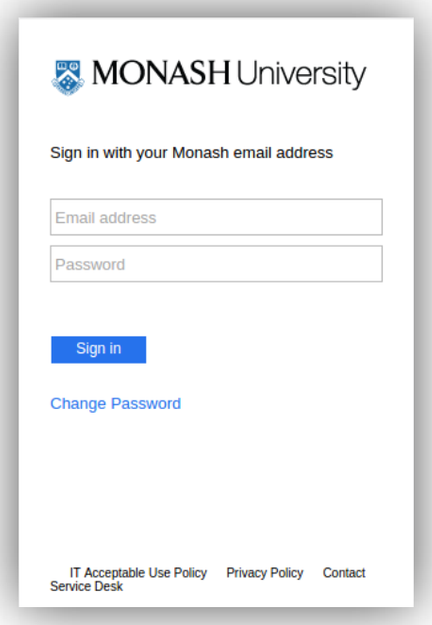
- To pass this unit a student must obtain:
  - ➡ **At least 40% of the total within semester assessment.**
  - ➡ **At least 40% of the exam marks.**
  - ➡ **An overall unit mark of 50% or more.**

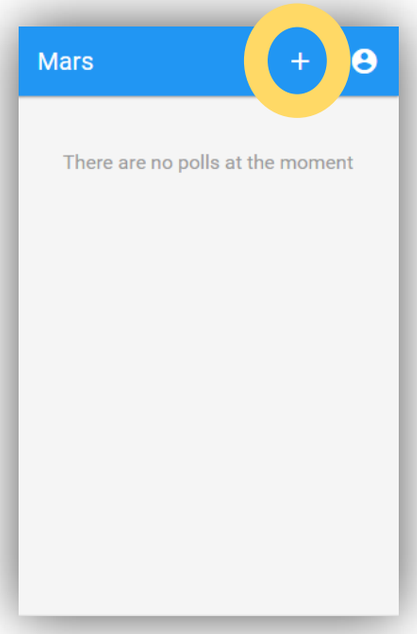
- If a student does not pass these hurdles then a mark of no greater than **49N** will be recorded for the unit.

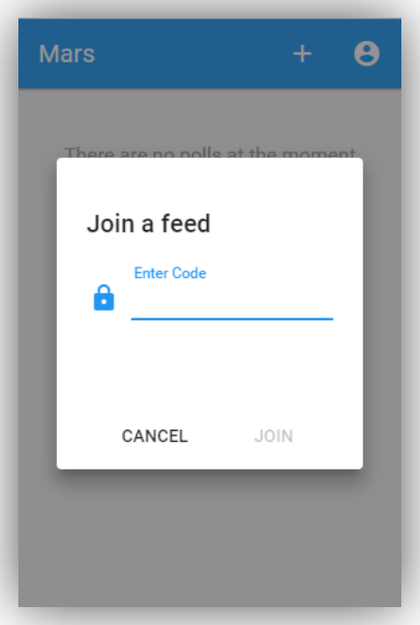


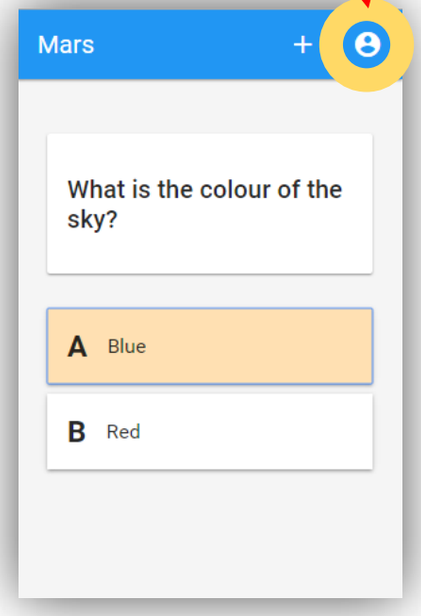
How we do lectures.

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: **QYZC01**
5. Answer questions when they pop up
6. Give yourself a Display Name and manage your feed subscriptions here









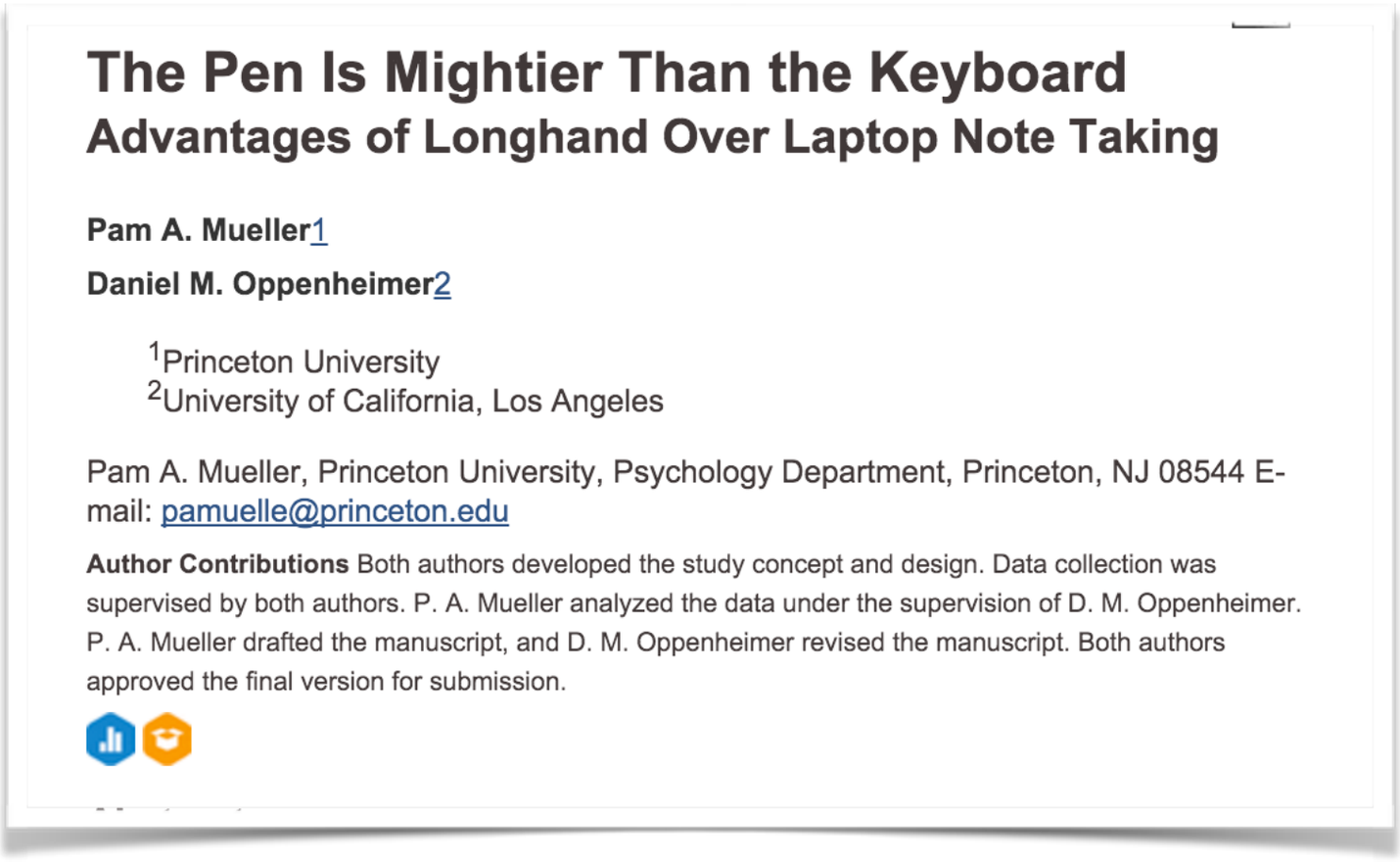
What is this unit about

Rules of the house…

Some advice

Teaching team

If possible take notes…



The Pen Is Mightier Than the Keyboard
Advantages of Longhand Over Laptop Note Taking

Pam A. Mueller[1]
Daniel M. Oppenheimer[2]

[1]Princeton University
[2]University of California, Los Angeles

Pam A. Mueller, Princeton University, Psychology Department, Princeton, NJ 08544 E-mail: pamuelle@princeton.edu

Author Contributions Both authors developed the study concept and design. Data collection was supervised by both authors. P. A. Mueller analyzed the data under the supervision of D. M. Oppenheimer. P. A. Mueller drafted the manuscript, and D. M. Oppenheimer revised the manuscript. Both authors approved the final version for submission.

(slides are usually not self-standing **on purpose**)

Recommended Reading

- **MIPS Assembly Language Programming.** Britton.

- **Problem Solving with algorithms and data structures using Python.** Miller & Ranum.
  [available online under CC license:
  https://interactivepython.org/runestone/static/pythonds/index.html]

- **Data Structures & Algorithms in Python.** Goodrich, Tamassia & Goldwasser.

Help is Available

- Lecturer
- Tutors
- Help Room Consultations (every week TBA)
- Moodle
- Co-ordinators
- Administration Officers

Consultations

Julian García

Tuesday: 10AM to 11AM

Office 230, 25 Exhibition Walk, Clayton

Do you have any form of condition (medical, disability other) that impacts on your ability to study?

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

- Notetakers and Auslan interpreters
- Readings in alternative formats
- Adaptive equipment and software
- Alternative arrangements for exams

For further information and details about how to register:

Email: disabilitysupportservices@monash.edu
Phone: 03 9905 5704
Web: monash.edu/disability

ALSO:

http://www.monash.edu/health/counselling

Study Hacks by Cal Newport

On Preparation:

Drizzle Test Preparation Over Many Days
How early should you start studying? This post lays out the basic philosophy preached in Straight-A. Put simply: start early; work in little batches.

Use Focused-Question Clusters to Study for Knowledge Based Tests
How should you study for classes that require you to know a large number of facts and concepts? I overlooked these classes in Straight-A (as many of you subsequently brought to my attention.) In this post I rectify this oversight. It was originally written for multiple choice tests, but the advice is relevant for any exam requiring a large amount of memorized information.

Pseudo-Work Doesn't Equal Work and Studying is a Technical Skill
How are some high-scoring students able to escape the stress of the grind lifestyle? These two early posts, from a series titled "The Straight-A Gospels," lay out the core philosophical ideas behind the mysterious, yeti-like low-stress 'A.' I recommend a quick review before diving too deep into exam period chaos.

http://calnewport.com/blog/2008/04/28/monday-master-class-the-study-hacks-guide-to-exams/

Studying is a skill

What is this unit about

Rules of the house…

Some advice

Teaching team

Head Tutor

Brendon Taylor

- **Email**: brendon.taylor@monash.edu

- **Consultation** Times:
  - TBA

Lecturer

Dr Muhammad Fermi Pasha

- Room No. 2-4-23
  School of Information Technology
  Sunway, Malaysia

- **Email**: Muhammad.FermiPasha@monash.edu

- **Consultation** Times:
  - TBA

Lecturer

Dr Julian Garcia

- **Room 230**, 25 Exhibition Walk, Clayton
- Tel: 9905 3654
- **Email**: Julian.Garcia@monash.edu
- **Consultation** Times:
  - Tuesdays, 10AM

Looking forward!