FIT1045: Algorithms and Programming Fundamentals in Python

Lecture 1

Introduction to Algorithms

Images at http://bloggertowordpresstestblog.blogspot.com.au/2009/11/michael-hansmeyer-solids-at-smallspace.html:

What's this unit about

Algorithms

- Sequence of instructions to solve problem
- Will cover variety of algorithms for different problems

The Python programming *language*

- Mean to communicate with computer and to implement algorithms
- Like any language: **practice** is key

Unit Objectives

- Learn to develop and reason about algorithms
- Be able to program them in Python
- Start to understand the limitations of algorithms

Transferrable skills from this unit

"Hard" skills

- Python programming
- Algorithm design and analysis

"Soft" skills

- Teamwork (workshops and tutorials)
- Planning (assignments, workshops)
- Communication (workshops and tutorials)

Overall

- **Problem solving…**

Computer Science enables people to do amazing things

Annie Easley;

Computer scientist, Mathematician, Rocket scientist with NASA for ~32 years
Helped develop the Centaur rocket system (got Cassini to Saturn!)
(left)

Admiral Grace Hopper;

Computer scientist, American Navy Admiral; involved in development of FORTRAN, COBOL
(right)

Ramanathan V. Guha;

Computer scientist, creator of RSS, worked on Google's custom search and adwords
(left)

Donald Knuth;

Computer scientist, mathematician, author. Developed the TeX typesetting system, and wrote the widely read "The Art of Computer Programming" (Turing award worthy)
(right)

Satoshi Nakamoto;

Computer scientist/cryptographer. Developer of the Bitcoin; ???
(left)

…by improving their thinking

“This knowledge [on algorithms] […] is a **general-purpose mental tool** that will be a definite aid to the understanding of other subjects, whether they be chemistry, linguistics, or music, etc. The reason for this […]: It has often been said that a person does not really understand something until after teaching it to someone else. Actually, **a person does not *really* understand something until after teaching it to a *computer***, i.e., expressing it as an algorithm...”

Donald Knuth

Requires investment of time and *attention*

Contact hours

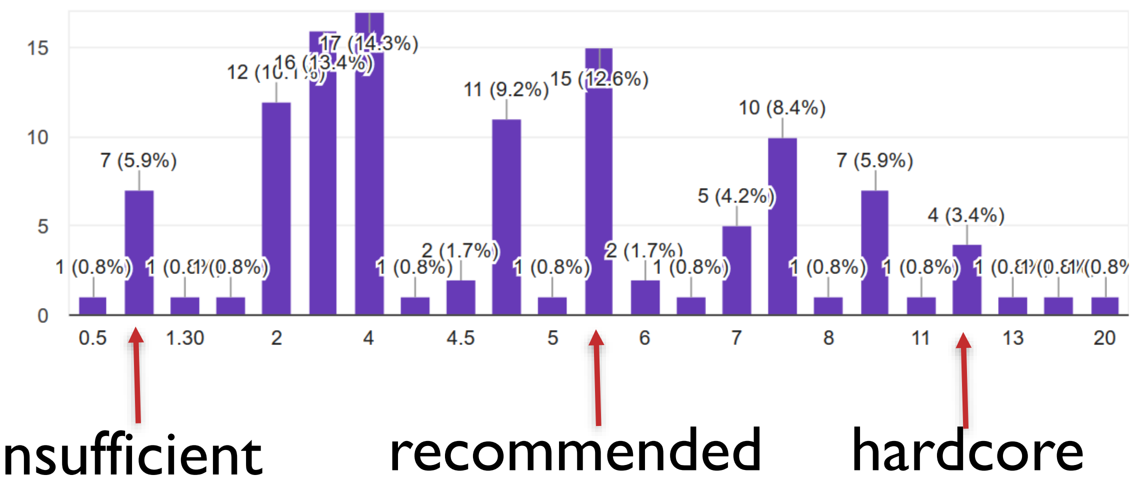
- Lectures (2h per week)
- Tutorials (2h)
- Labs/Workshops (2h)

Self study (5-7h)

- Practice (4-5h)
- Reading (1-2h)

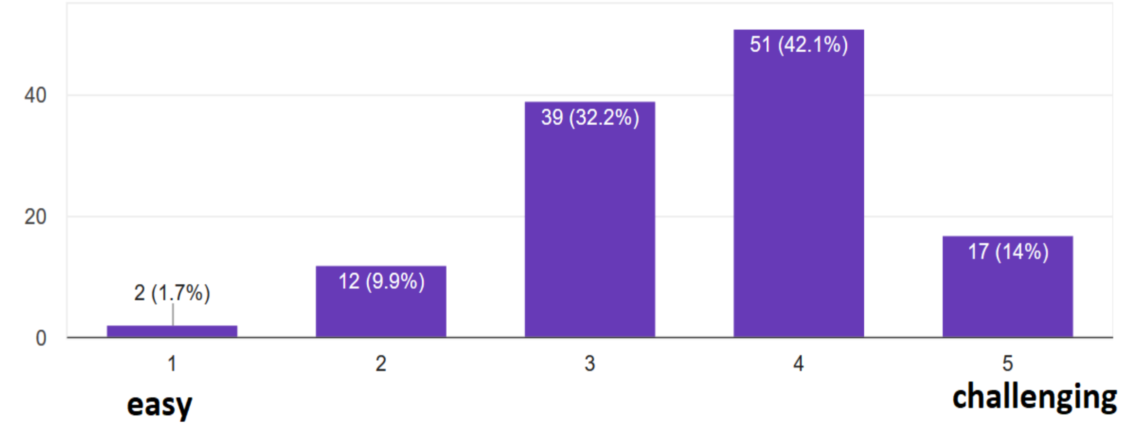
How much time do you spend on this unit every week (other than the 6 contact hours)

119 responses

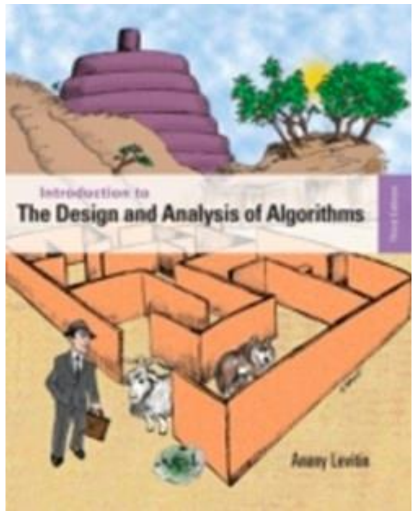


Now that you have completed all of your classes, How would you rate the difficulty of this unit?

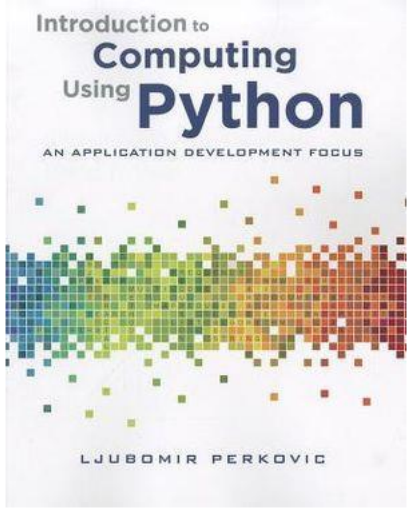
121 responses



Recommended literature



Levitin, A. (2012) *Introduction to the Design and Analysis of Algorithms* (3rd Edition) Pearson



Perkovic, L. (2012) *Introduction to Computing using Python: An Application Development Focus.* John Wiley & Sons, Inc.

Where am I?

1. Introduction to unit
2. Unit structure and assessment
3. Getting help
4. What are algorithms

Staff

Mario Boley (Chief Examiner, Clayton and Malaysia)

mario.boley@monash.edu

Michael Kamp (Lecturer, Clayton)

michael.kamp@monash.edu

Ganesh Krishnasamy (Lecturer, Malaysia)

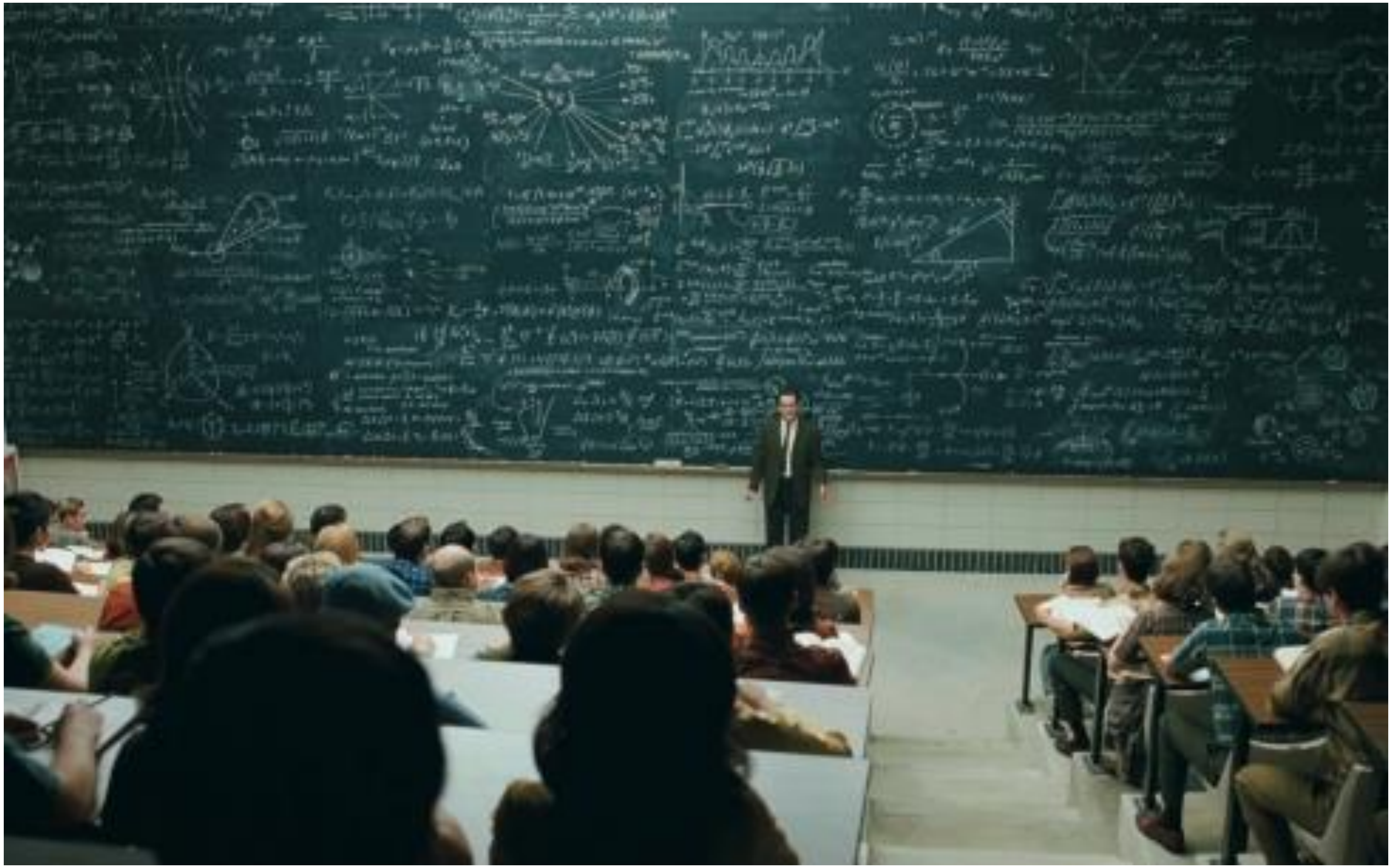
ganesh.Krishnasamy@monash.edu

Simon Teshuva (Head Tutor)

simon.teshuva@monash.edu

FIT1045.clayton-x@monash.edu

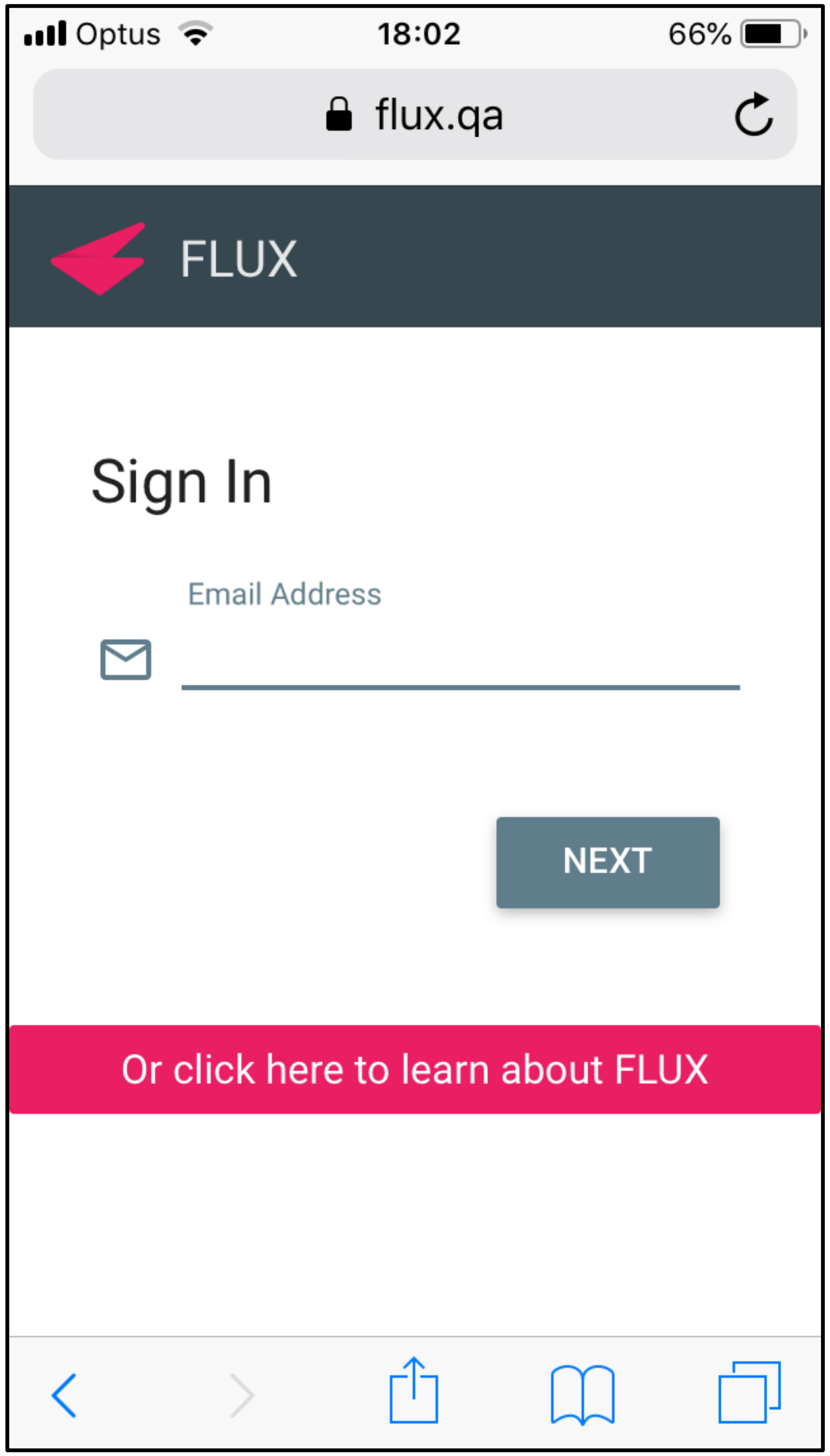
Conventional Lectures



http://www.pedagogygeek.com/category/uncategorized/

Quiz time

1. Visit https://flux.qa
2. Log in (your Authcate details)

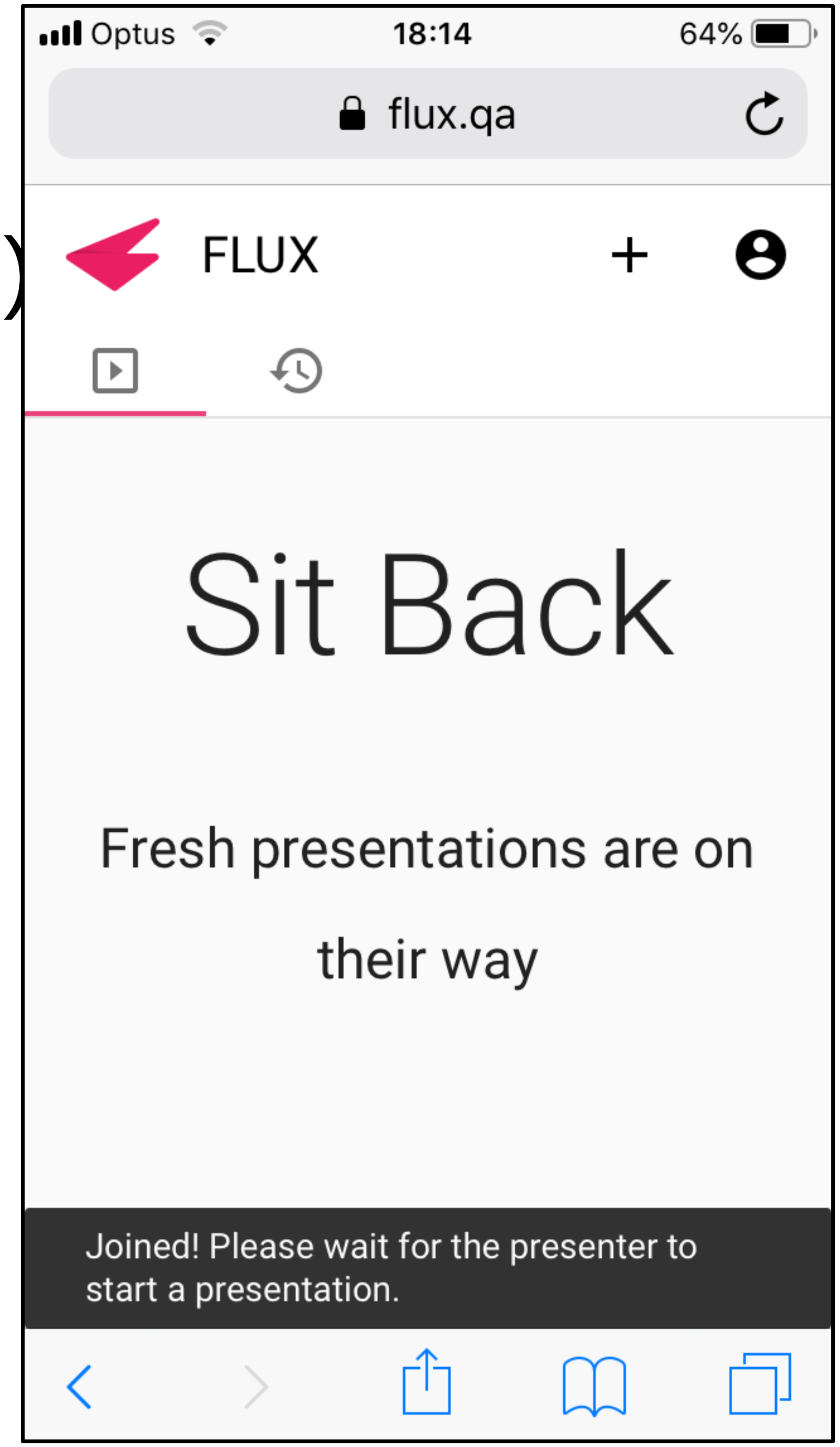


Sign In

Email Address

Quiz time

1. Visit https://flux.qa
2. Log in (your Authcate details)
3. Touch the + symbol
4. Enter your audience code
   - Clayton: AXXULH
   - Malaysia: LWERDE
5. Answer questions



Sit Back

Fresh presentations are on their way

Joined! Please wait for the presenter to start a presentation.

Assessment

In-semester Marks (max of 60%)

- Assignment Part 1 (10%)
- Assignment Part 2 (12%)
- Workshops (19%)
- Tutorial Preparation (8%)
- In-semester test 1 (3%)
- In-semester test 2 (8%)

Exam (40%) (2 hours)

Hurdles (see Unit Guide)

To pass this unit a student must obtain:

- 40% or more in the unit's examination, and
- 40% or more in the unit's total non-examination assessment, and
- an overall unit mark of 50% or more.

If a student does not pass these hurdles then a mark of no greater than **45-NH** will be recorded for the unit.

Assessment: Assignment Parts 1 and 2 (10%, 12%)

- Complex programming tasks covering different types of important problems
- Individual work in your own time
  - Part 1 due week 6
  - Part 2 due week 11
- **IMPORTANT:** you **need** to explain your code to demonstrators in interviews (weeks 7 and 12). Not doing the interview yields 0 marks for respective assignment
- Last semester many students simply did not come to their interview, leading to many students getting 0
- Past students consistently recommend: **"start early, plan ahead!!!"**

Assessment: Workshops (max 19%)

- All about programming practice with Python
  - Active learning
  - Preparation for assignments
- Marked workshops released in weeks 1,2,3,4,5,7,8,9,10
  - Assisted programming in workshop (instructor is around to help)
  - Finish the rest in your own time
  - **Workshop *n*** must be uploaded on Moodle by midnight on Tuesday of **week *n+1***
  - In weeks 7 and 12 we do assignment interviews. There is still a worksheet for the earlier week (6 and 11), but it is not assessed.
- Each week is worth 2.5 marks to maximum of 19 marks
  - Mark is given based on number of tasks solved…*and ability to explain solution to instructor*
  - Students are marked individually (even if work done as a pair)

Assessment: Workshops cont. (max 19%)

- Workshops from Workshop 2 will contain an advanced problem
  - Optional bonus activity.
  - The advanced problem will **not** be marked.

Assessment:
Tutorial preparation (max 8%)

- Every tutorial from week 2 onwards contains a preparation question (weeks 2-12)
  - **Tutorial prep *n*** must be posted in the Moodle forum of your tutorial by midnight on Tuesday of **week *n***
- Each week is worth 1 mark to maximum of 8 marks
  - Mark is given based on whether there was a genuine attempt to answer the question

Assessment:
In-semester tests (3%, 8%)

2x in-semester tests

- In week 4, and week 9, on Moodle
- Timed test, ~30 minutes duration
- Covers content taught in first 3 and 8 weeks respectively
- Designed to give you a feeling for exam-style questions

Assessment:
Final Exam (40%) (2 hours)

During Exam period

Contains questions on:

- Definitions
- Programming/Python
- Analysis of algorithms
- Algorithm design
- Running algorithms by hand
- And more…

Submission of Assignments

Submission details will be specified on each assignment part

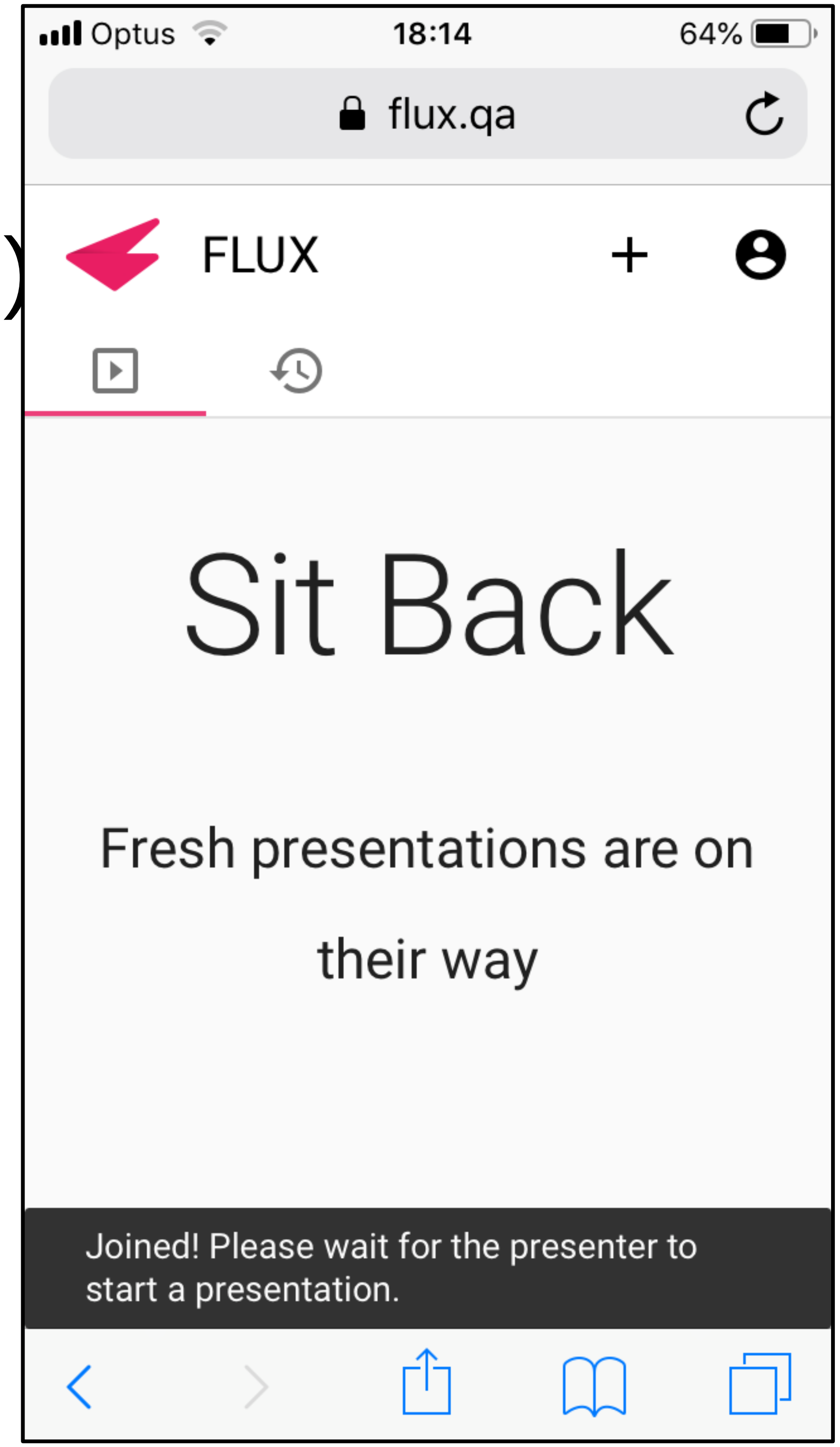
- You will submit your assignment parts through Moodle
- Whenever you submit you must complete a submission form
- Late submission will have 10% off the *maximally achievable* assignment marks per day (including weekends)
- Assignments submitted 7 days after the due date will not be accepted.

Extensions

- Compelling and *exceptional* reasons only (must be *unforeseeable* circumstances) via special consideration request
- Send to FIT1045.clayton-x@monash.edu with supporting documentations BEFORE deadline
- Don't stop working!

Quiz time

1. Visit https://flux.qa
2. Log in (your Authcate details)
3. Touch the + symbol
4. Enter your audience code
    - Clayton: AXXULH
    - Malaysia: LWERDE
5. Answer questions



Sit Back

Fresh presentations are on their way

Joined! Please wait for the presenter to start a presentation.

Week	Lecture a	Lecture b	Assessment
1	Introduction to Algorithms	Introduction to Python	
2	Functions and Selection	While-loops and Sequences	Workshops due from week 2-11 (excluding 7)
3	For-loops and Sequences	Tables and Matrices	
4	Decomposition	Understanding Python	Test 1 in tutorial (3%)
5	Sorting Problem	Invariants	
6	Introduction to Complexity	Search Problem	Assignment Part 1 (10%)
7	Divide and Conquer	Divide and Conquer cont.	Interview in workshop
8	Recursion	Graph Traversal, Stacks, and Queues	
9	Graphs, Stacks and Queues	Transform and Conquer	Test 2 in tutorial (8%)
10	Solving Linear Systems	Combinatorial Optimisation Problems	
11	Brute Force	Backtracking	Assignment Part 2 (12%)
12	Complexity Classes	Revision	Interview in workshop

Where am i?

1. Introduction to unit
2. Unit structure and assessments
3. Getting help
4. What are algorithms?

Help is Available

We want to help you succeed!

- Lecturer
- Tutors
- Consultations
- Moodle forums
- PASS

PARTICIPATE & ACHIEVE BETTER RESULTS

PEER-ASSISTED STUDY SESSIONS (PASS)

- Available for FIT1045
- Receive support and mentoring from a senior student successful in this unit
- Weekly study sessions to keep you up-to-date
- Learn through samples, teamwork, and group study games
- Fine-tune study skills
- Make new friends

THE PASS EQUATION

1 hour of PASS =

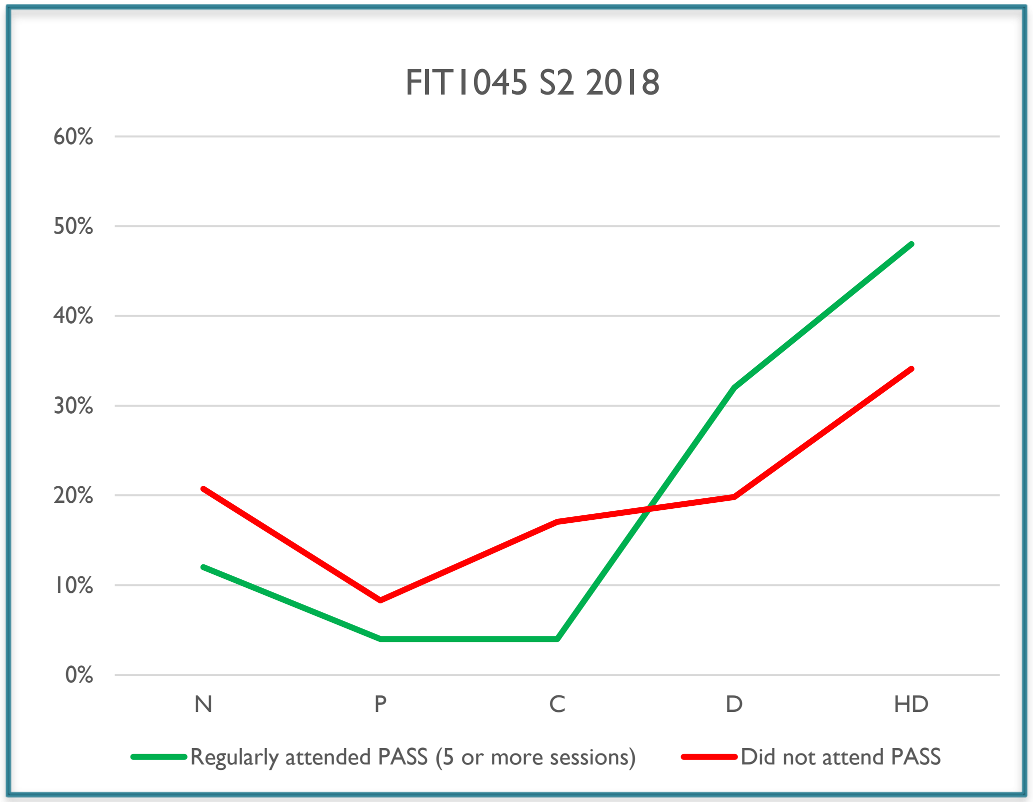
3 hours struggling on your own

PARTICIPATE & ACHIEVE BETTER RESULTS

PASS GETS RESULTS

Students who regularly attend PASS

- are more likely to score a D or HD
- are less likely to fail the unit



The comparative PASS results for FIT1045 last year

PARTICIPATE & ACHIEVE BETTER RESULTS

SIGN UP TO PASS

- Sign up for PASS via Allocate+ in Week 1
- Select a FIT1045 PASS session that fits your schedule
- No spots left in Allocate+? Drop into the session anyway. Most classes do not have full attendance
- For any other requests, please email pass.registration@monash.edu
- PASS study sessions begin in Week 2. See you there!

Seek assistance as a preventative measure

Take the following relevant preventative measures as soon as possible, if you are falling behind in your studies:

- Study difficulties: Discuss any difficulties you are experiencing with your tutor or lecturer.

  These staff members can assist you in identifying your problem areas and explore the options available to you in your course.

- Language and learning online can help you with study methods, language skills and work presentation (organised by the library) http://www.monash.edu.au/lls/llonline/

- Student life and support services can be found at http://monash.edu/students/support/ and include: Health services, support and services, clubs and sports etc

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

Special consideration

- If something beyond your control is affecting your performance on assessment we might be able to do something for you!
- Please send requests with documentation to the role account: **FIT1045.clayton-x@monash.edu**
- See special consideration policy (https://www.monash.edu/exams/changes/special-consideration)

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.

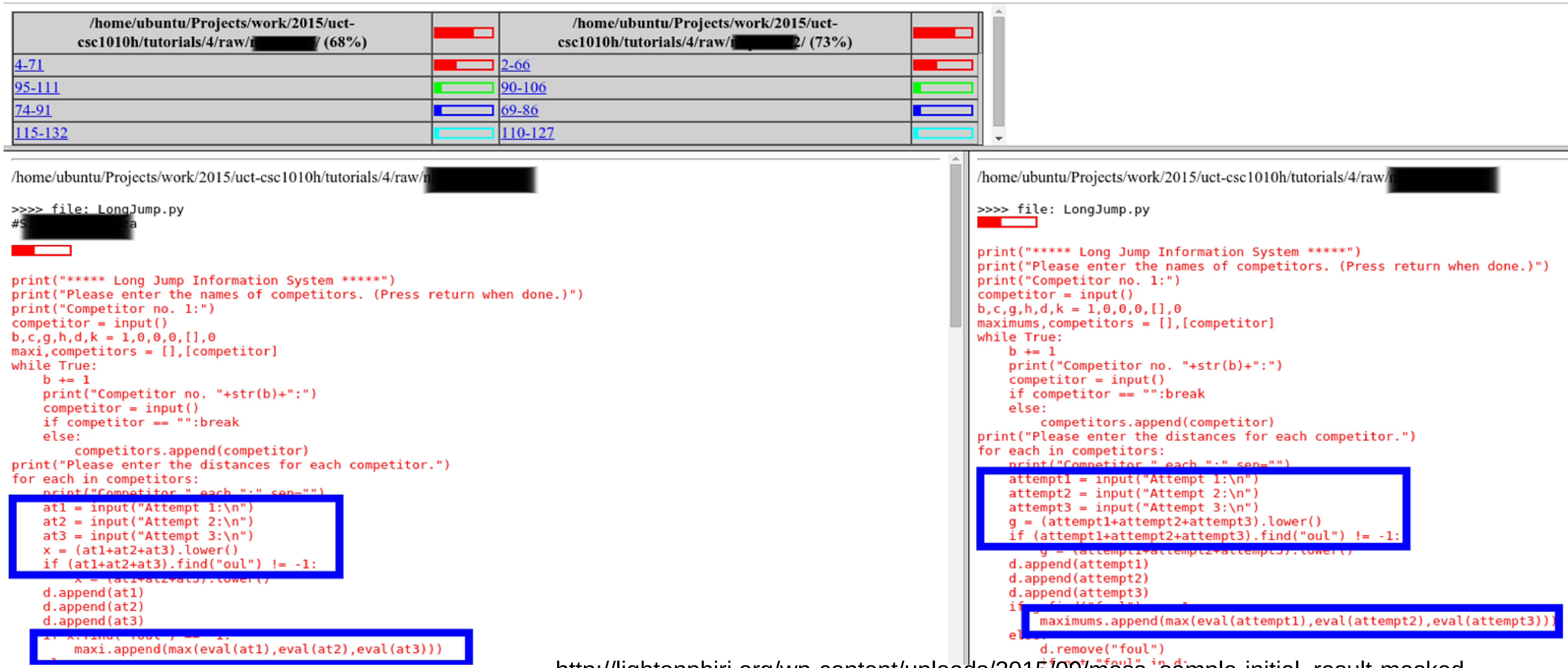
- **Collusion:** Unauthorised collaboration on assessable work with another person or persons.

- **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

MOSS



/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/[redacted]/ (68%)	🟥	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/[redacted]2/ (73%)	🟥
4-71	🟥	2-66	🟥
95-111	🟩	90-106	🟩
74-91	🟦	69-86	🟦
115-132	▭	110-127	▭

/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/[redacted]

```
>>>> file: LongJump.py
#[redacted]

print("***** Long Jump Information System *****")
print("Please enter the names of competitors. (Press return when done.)")
print("Competitor no. 1:")
competitor = input()
b,c,g,h,d,k = 1,0,0,0,[],0
maxi,competitors = [],[competitor]
while True:
    b += 1
    print("Competitor no. "+str(b)+":")
    competitor = input()
    if competitor == "":break
    else:
        competitors.append(competitor)
print("Please enter the distances for each competitor.")
for each in competitors:
    print("Competitor " each ":" sep="")
    at1 = input("Attempt 1:\n")
    at2 = input("Attempt 2:\n")
    at3 = input("Attempt 3:\n")
    x = (at1+at2+at3).lower()
    if (at1+at2+at3).find("oul") != -1:
        x = (at1+at2+at3).lower()
    d.append(at1)
    d.append(at2)
    d.append(at3)
    if x.find("foul") == -1:
        maxi.append(max(eval(at1),eval(at2),eval(at3)))
```

/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/[redacted]

```
>>>> file: LongJump.py

print("***** Long Jump Information System *****")
print("Please enter the names of competitors. (Press return when done.)")
print("Competitor no. 1:")
competitor = input()
b,c,g,h,d,k = 1,0,0,0,[],0
maximums,competitors = [],[competitor]
while True:
    b += 1
    print("Competitor no. "+str(b)+":")
    competitor = input()
    if competitor == "":break
    else:
        competitors.append(competitor)
print("Please enter the distances for each competitor.")
for each in competitors:
    print("Competitor " each ":" sep="")
    attempt1 = input("Attempt 1:\n")
    attempt2 = input("Attempt 2:\n")
    attempt3 = input("Attempt 3:\n")
    g = (attempt1+attempt2+attempt3).lower()
    if (attempt1+attempt2+attempt3).find("oul") != -1:
        g = (attempt1+attempt2+attempt3).lower()
    d.append(attempt1)
    d.append(attempt2)
    d.append(attempt3)
    if
        maximums.append(max(eval(attempt1),eval(attempt2),eval(attempt3)))
    else:
        d.remove("foul")
        if not "foul" in d:
```

http://lightonphiri.org/wp-content/uploads/2015/09/moss_sample-initial_result-masked-021.png

Where am I?

1. Introduction to unit
2. Unit structure and assessment
3. Getting help
4. What are algorithms

Learning outcomes:

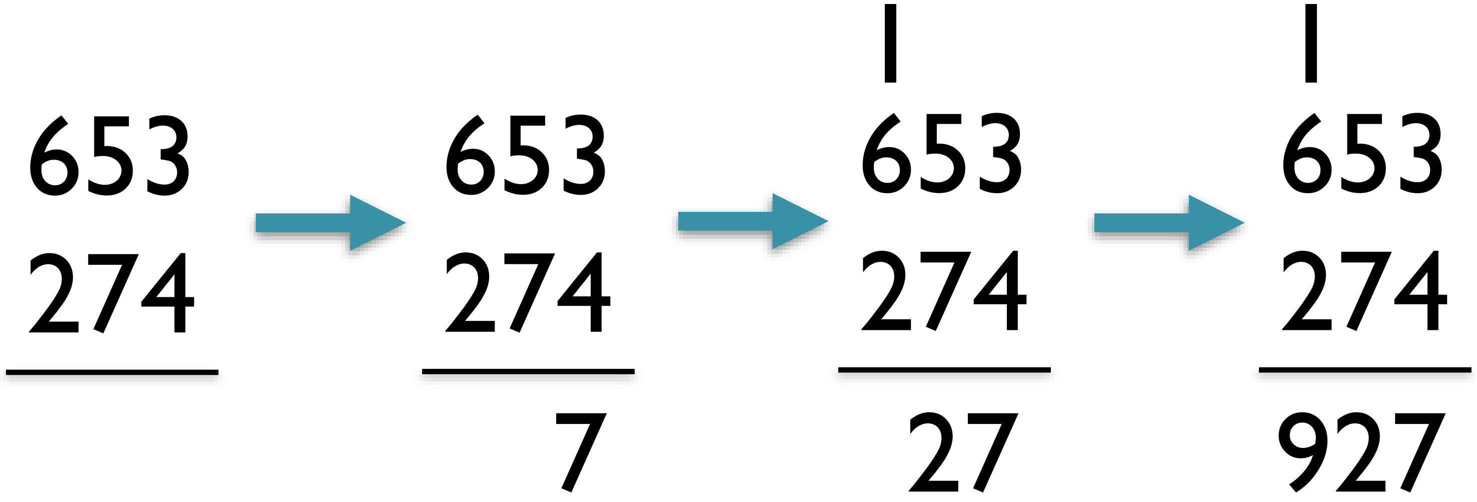
- 1, translate between problem descriptions and program design with appropriate input/output representations

Image at: vindelz.wordpress.com/2009/11/algorismi1.jpg

al-Khwarizmi
(c. 780 – 850)

What is an Algorithm?

Question: what is 653+274?



$$\begin{array}{r} 653 \\ 274 \\ \hline \end{array} \rightarrow \begin{array}{r} 653 \\ 274 \\ \hline 7 \end{array} \rightarrow \begin{array}{r} 1 \\ 653 \\ 274 \\ \hline 27 \end{array} \rightarrow \begin{array}{r} 1 \\ 653 \\ 274 \\ \hline 927 \end{array}$$

Instructions:

```
write given numbers on top of each other
for each column:
        find result digit for column
        “carry over” potential overflow
```

Definition

[Levitin, p3]

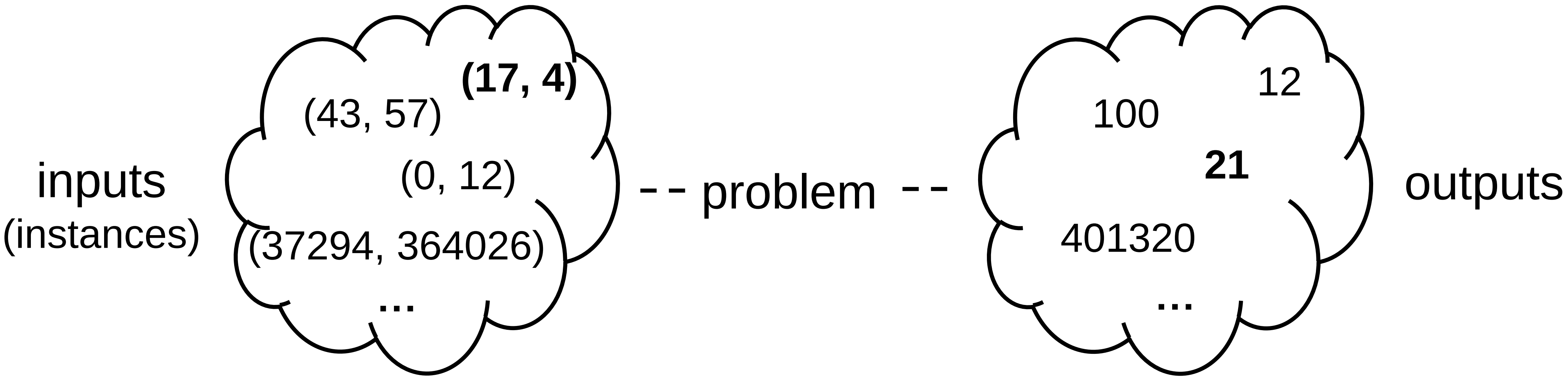
“An ***algorithm*** is a sequence of instructions for solving a *problem.*”

But what is a problem?

A *computational* problem is a (typically) infinite collection of questions (called *inputs* or *instances*), each of which has at least one correct associated answer (called *output*).

Example: *Addition*

- 17+4 is an input (question) of the comp. problem *Addition*
- The output (answer) to this instance is 21



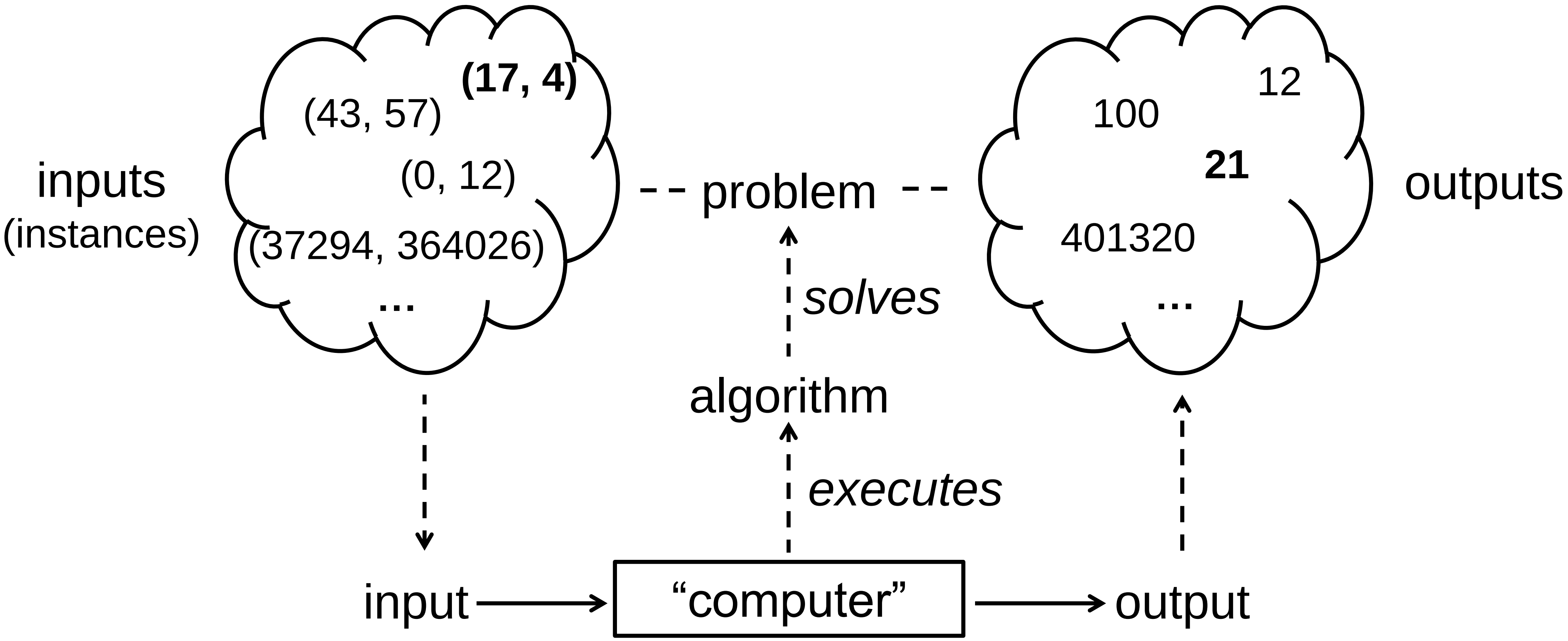
But what is a problem?

A *computational* problem is a typically infinite collection of questions (called *inputs* or *instances*), each of which has at least one correct associated answer (called *output*).

Addition Problem

Input: two numbers n and m

Output: the sum $n + m$



Definition

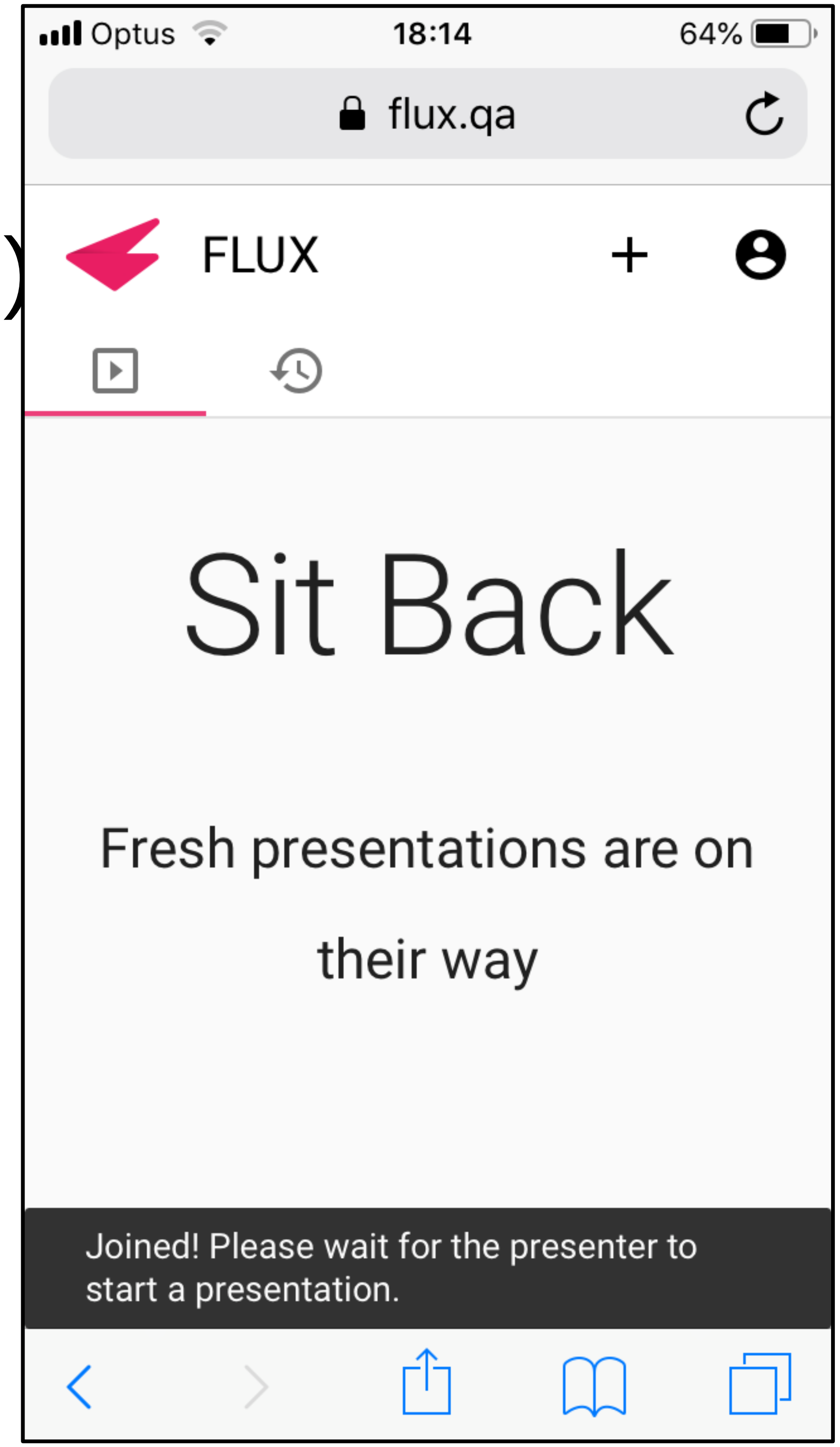
[Levitin, p3]

“An ***algorithm*** is a sequence of instructions for solving a ***problem***, i.e., for obtaining a required output for any legitimate input in a finite amount of time.”

- **Input**
- **Output**
- **Finiteness**

Quiz time

1. Visit https://flux.qa
2. Log in (your Authcate details)
3. Touch the + symbol
4. Enter your audience code
   - Clayton: AXXULH
   - Malaysia: LWERDE
5. Answer questions



Sit Back

Fresh presentations are on their way

Joined! Please wait for the presenter to start a presentation.

Image at: vindelz.wordpress.com/2009/11/algorismi1.jpg

What is an Algorithm?

al-Khwarizmi
(c. 780 – 850)

Problem: 1203 + 98 = ?

$$\begin{array}{l} 1203 \\ \underline{98} \end{array}$$

Instructions:

```
write given numbers on top of each other
for each column:
        find result digit for column
        “carry over” potential overflow
```

Image at: vindelz.wordpress.com/2009/11/algorismi1.jpg

What is an Algorithm?

al-Khwarizmi
(c. 780 – 850)

Problem: 1203 + 98 = ?

$$\begin{array}{r} 1203 \\ 98 \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{r} 1203 \\ 98 \\ \hline 1 \end{array}$$

Instructions:

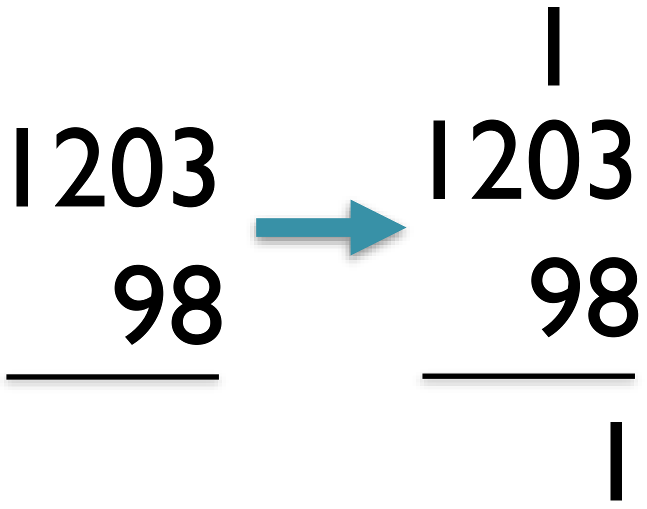
```
write given numbers on top of each other (right-aligned)
for each column:
        find result digit for column
        "carry over" potential overflow
```

What is an Algorithm?

Image at: vindelz.wordpress.com/2009/11/algorismi1.jpg

al-Khwarizmi
(c. 780 – 850)

Problem: 1203 + 98 = ?



1203
98
→
1
1203
98
1

Instructions need to be **definite**

Instructions:

write given numbers on top of each other **(right-aligned)**
for each column **(from right to left)**:
    find result digit for column
    “carry over” potential overflow

Image at: vindelz.wordpress.com/2009/11/algorismi1.jpg

What is an Algorithm?

al-Khwarizmi
– 850)

Problem: 1203 + 98 = ?

```
          1
1203     1203
  98  →    98
____     ____
            1
```

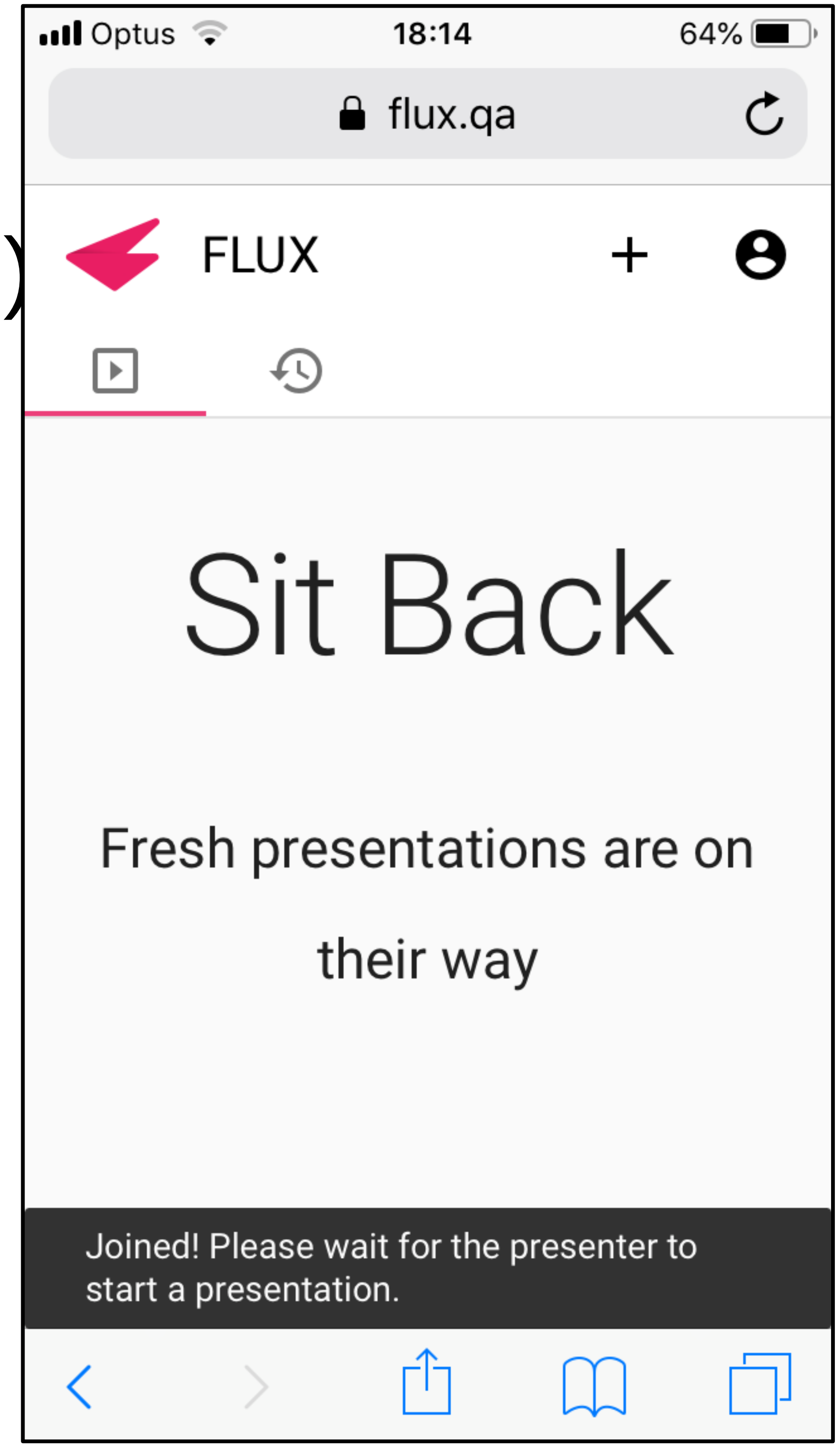
“computer” needs to be able to **effectively** carry out instruction

Instructions:

write given numbers on top of each other (right-aligned)
for each column (from right to left):
    find result digit for column
    “carry over” potential overflow

Quiz time

1. Visit https://flux.qa
2. Log in (your Authcate details)
3. Touch the + symbol
4. Enter your audience code
   - Clayton: AXXULH
   - Malaysia: LWERDE
5. Answer questions



Sit Back

Fresh presentations are on their way

Joined! Please wait for the presenter to start a presentation.

Definition

[Levitin, p3]

“An ***algorithm*** is a sequence of ***unambiguous*** instructions for solving a ***problem***, i.e., for obtaining a required output for any legitimate input in a finite amount of time.”

- **Input**
- **Output**
- **Finiteness**
- **Definiteness**
- **Effectiveness**

Example: Greatest Common Divisor

$$\frac{18}{24} = \frac{3}{4}$$

$$\text{gcd}(18, 24) = 6$$

<u>Greatest Common Divisor Problem</u>

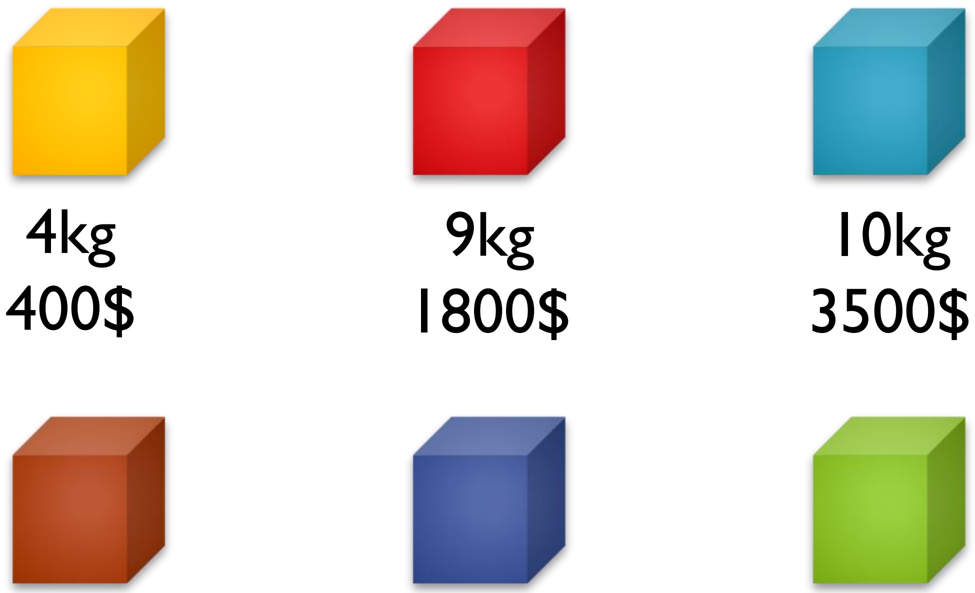
Input: two non-negative integers `m` and `n`

Output: greatest common divisor of `m` and `n`

Do you already know an algorithm to solve this problem?

Example: Robbing a Museum

http://www.unusualbag.com/



4kg
400$

9kg
1800$

10kg
3500$

20kg
4000$

2kg
1000$

1kg
200$

Museum Robbing Problem

Input: collection of items (with a weight and dollar value) and a knapsack (with a set capacity)

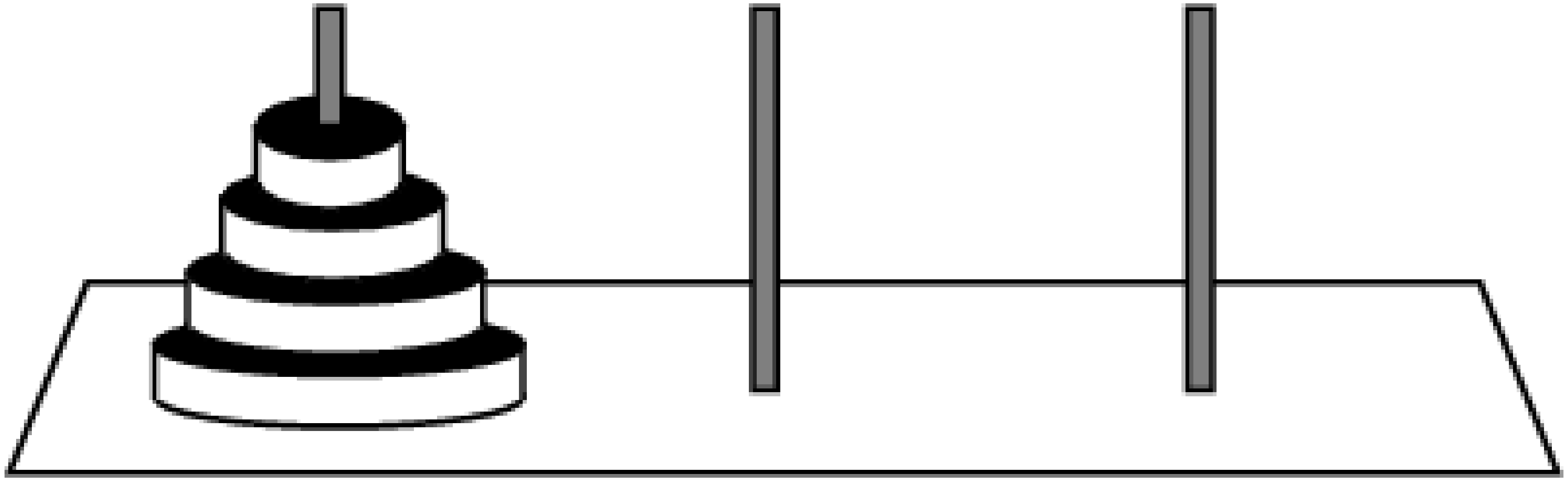
Output: collection of items to put in the knapsack to maximise value?

Example: Moving disks

Towers of Hanoi Problem

Input: stack of disks on a peg, where every disk is smaller than the one directly below

Output: sequence of moves that transfers all disks to a different peg without ever having to stack a larger disk on a smaller



Before Next Lecture

Log onto the FIT1045 Moodle site

Make sure you have flux.qa set up

Attempt the online quiz module on Moodle

Think about how to write an algorithm to solve the Greatest Common Devisor problem