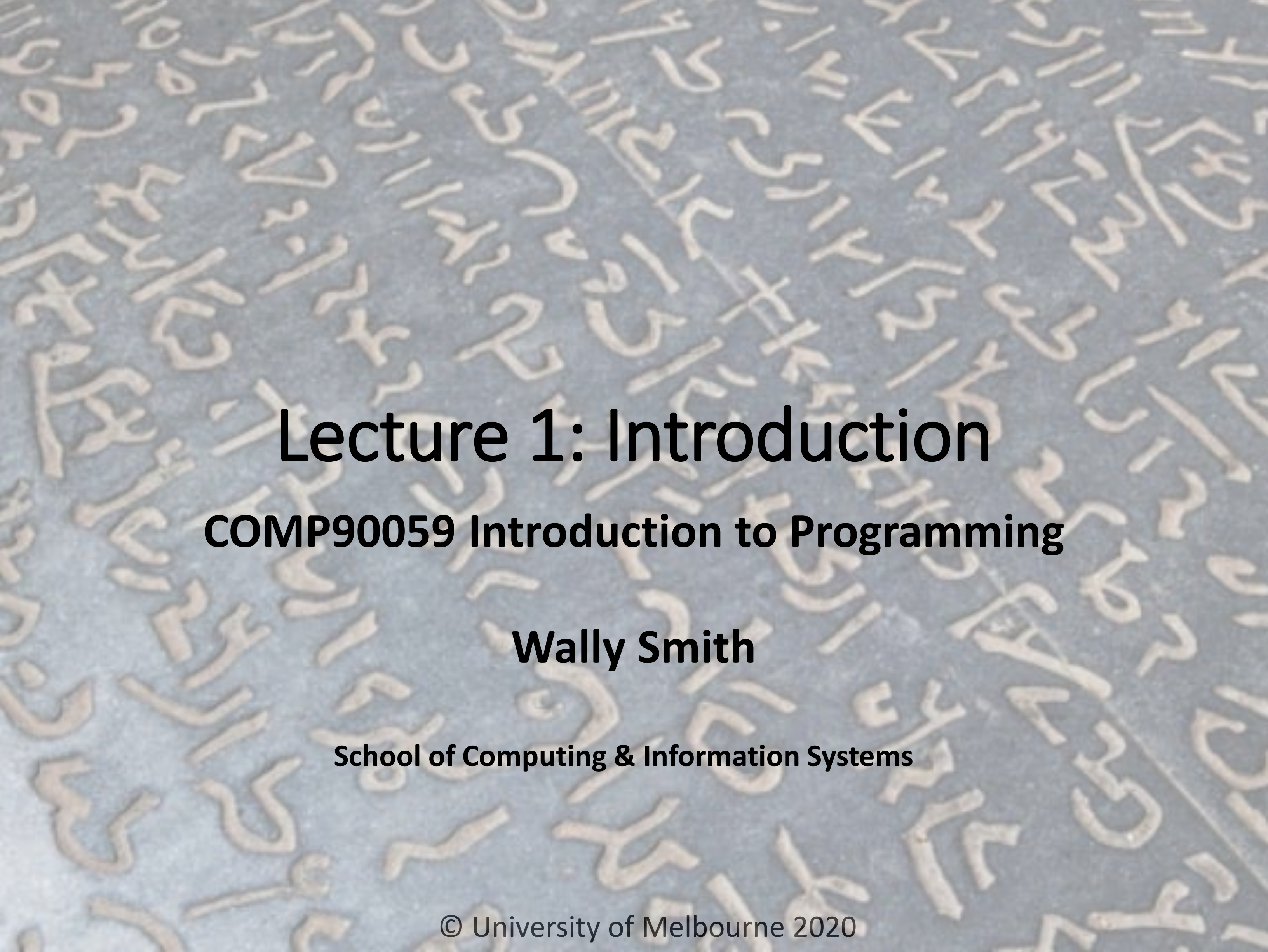


Lecture 1: Introduction

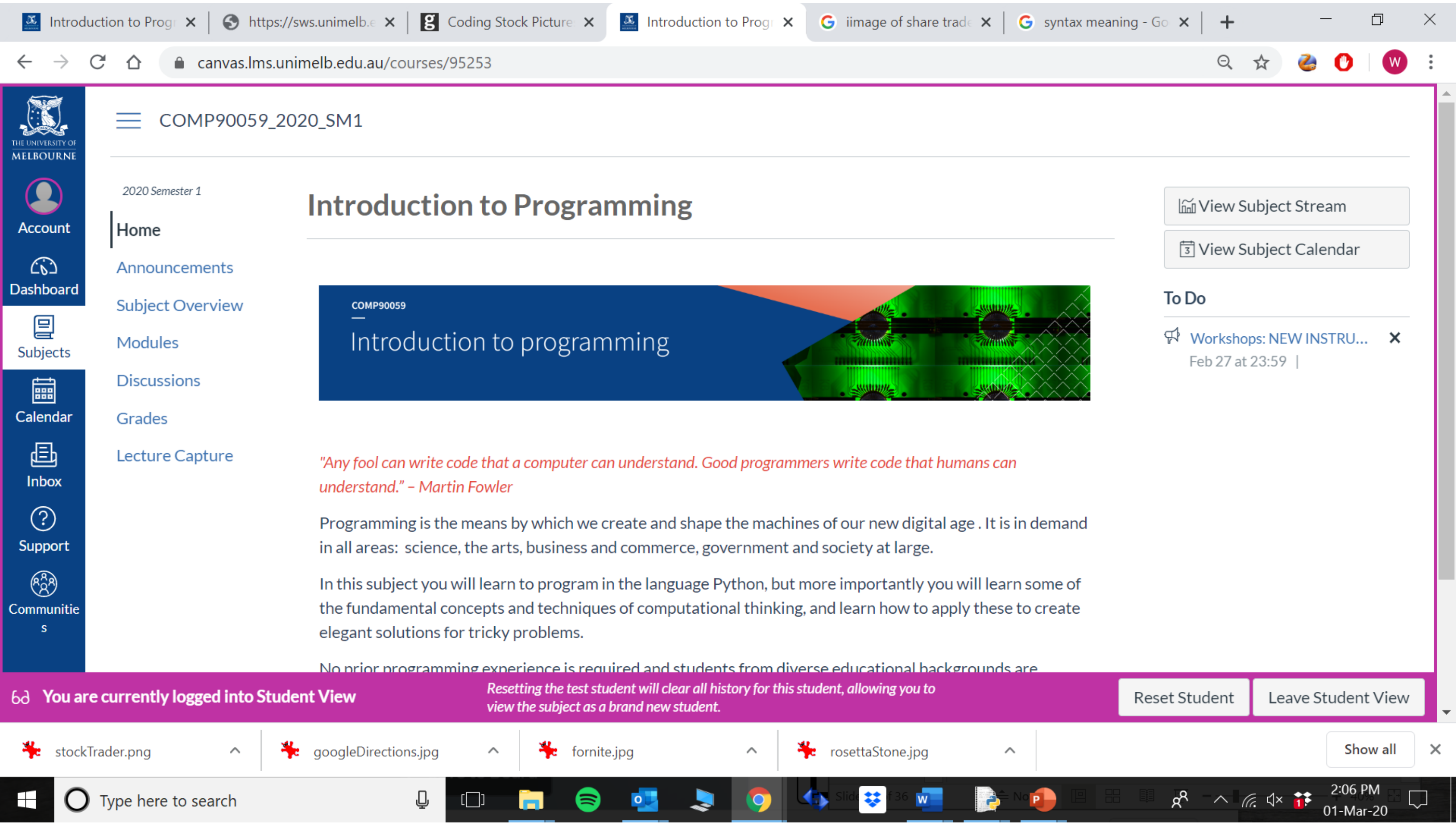
COMP90059 Introduction to Programming

Wally Smith

School of Computing & Information Systems

Lecture Overview

1. What to expect: lectures, workshops, assessment
2. What is this subject about?
   - coding in Python
   - computational thinking
   - theory and concepts of programming
3. Start coding & thinking computationally!
   - print, input, variables, assignments, commenting
   - Exercises - from easy to difficult to very difficult
4. Tools we will use
   - IDLE
   - GROK learning platform



Introduction to Programming

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand." – Martin Fowler

Programming is the means by which we create and shape the machines of our new digital age . It is in demand in all areas: science, the arts, business and commerce, government and society at large.

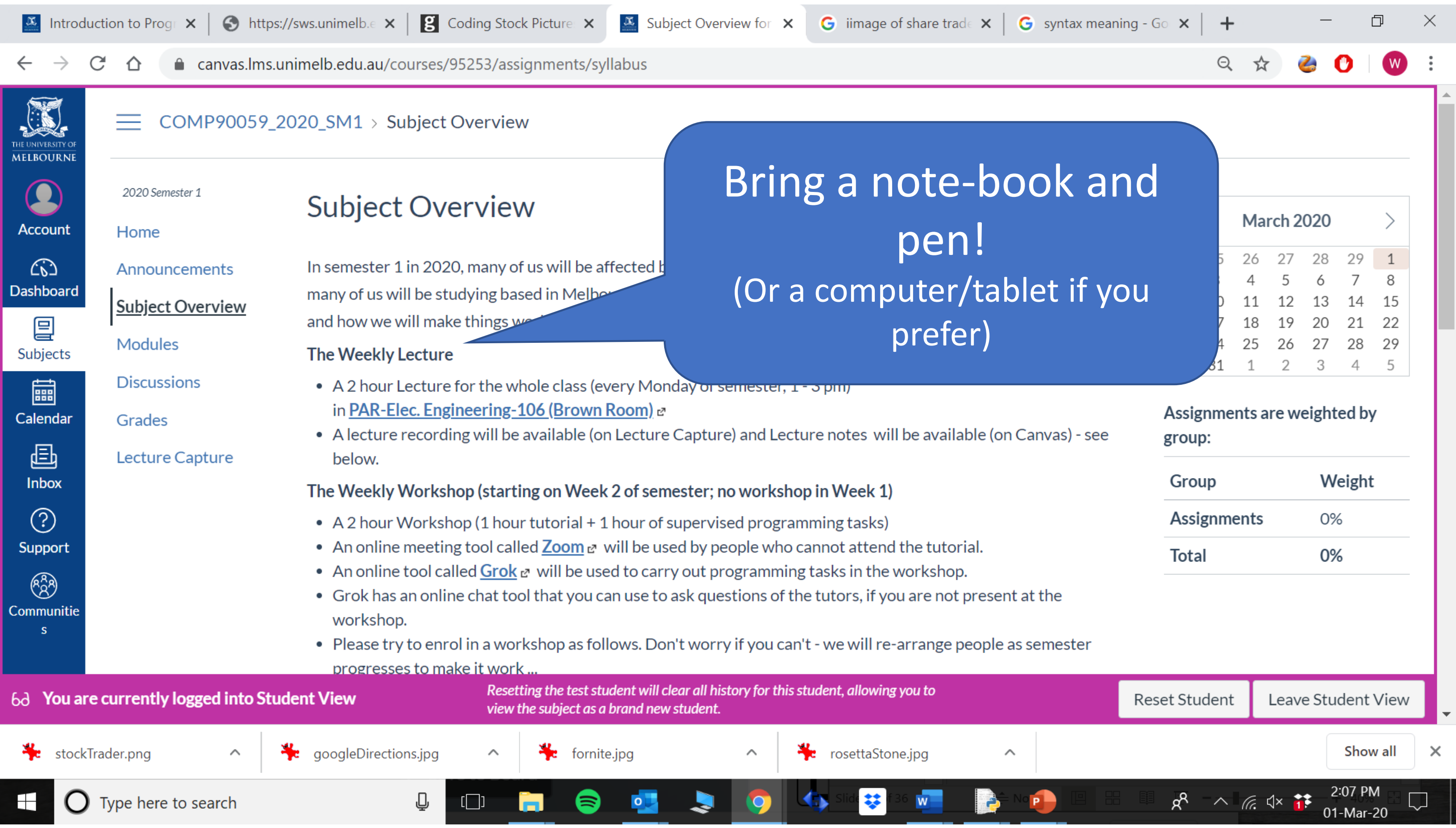
In this subject you will learn to program in the language Python, but more importantly you will learn some of the fundamental concepts and techniques of computational thinking, and learn how to apply these to create elegant solutions for tricky problems.

No prior programming experience is required and students from diverse educational backgrounds are

You are currently logged into Student View

Resetting the test student will clear all history for this student, allowing you to view the subject as a brand new student.

Reset Student | Leave Student View



Bring a note-book and pen!

(Or a computer/tablet if you prefer)

Subject Overview

In semester 1 in 2020, many of us will be affected b
many of us will be studying based in Melbo
and how we will make things w

The Weekly Lecture

- A 2 hour Lecture for the whole class (every Monday of semester, 1 - 3 pm) in PAR-Elec. Engineering-106 (Brown Room)
- A lecture recording will be available (on Lecture Capture) and Lecture notes will be available (on Canvas) - see below.

The Weekly Workshop (starting on Week 2 of semester; no workshop in Week 1)

- A 2 hour Workshop (1 hour tutorial + 1 hour of supervised programming tasks)
- An online meeting tool called Zoom will be used by people who cannot attend the tutorial.
- An online tool called Grok will be used to carry out programming tasks in the workshop.
- Grok has an online chat tool that you can use to ask questions of the tutors, if you are not present at the workshop.
- Please try to enrol in a workshop as follows. Don't worry if you can't - we will re-arrange people as semester progresses to make it work ...

5	26	27	28	29	1
3	4	5	6	7	8
0	11	12	13	14	15
7	18	19	20	21	22
4	25	26	27	28	29
31	1	2	3	4	5

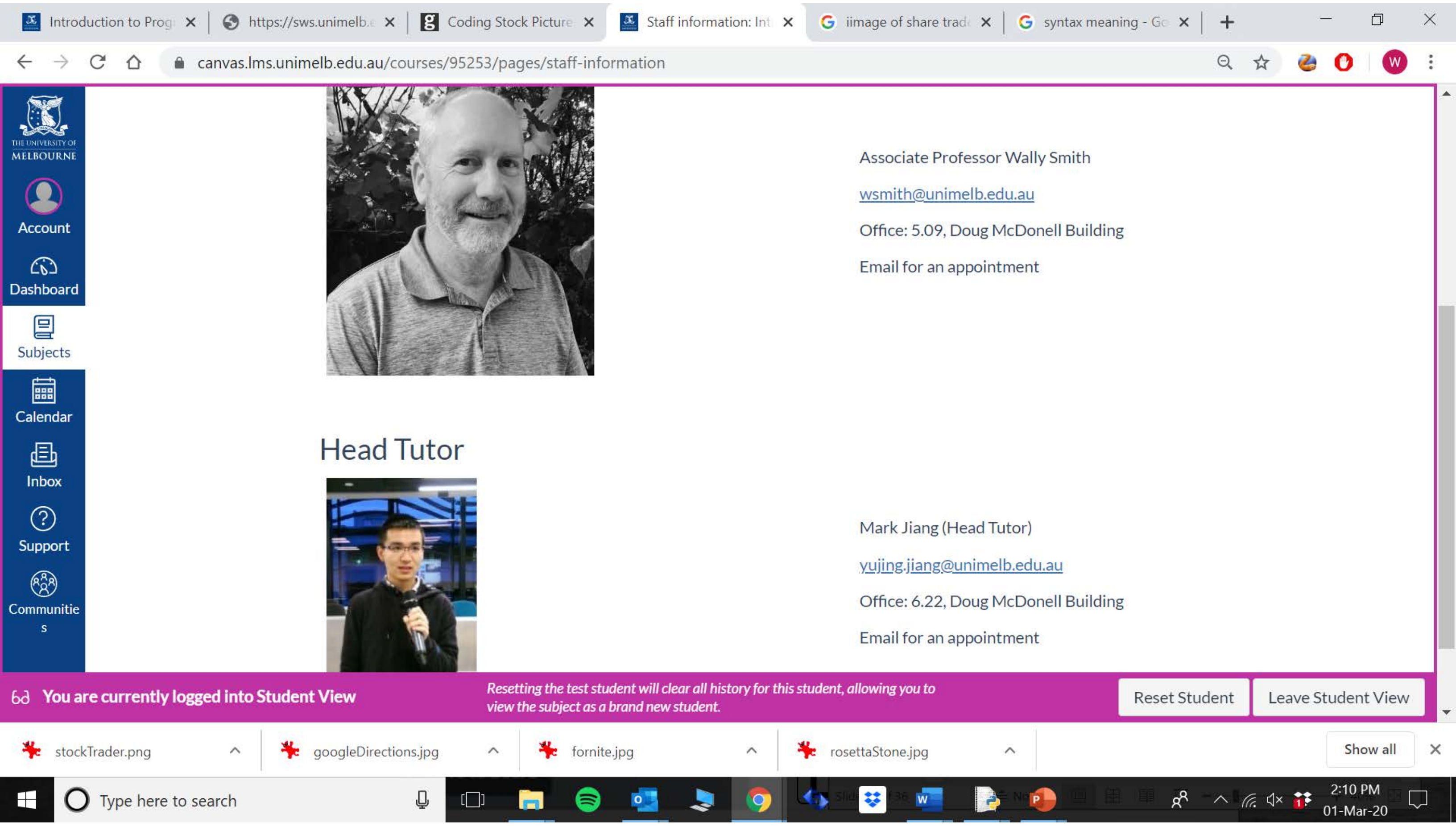
Assignments are weighted by group:

Group	Weight
Assignments	0%
Total	0%

You are currently logged into Student View

Resetting the test student will clear all history for this student, allowing you to view the subject as a brand new student.

Reset Student | Leave Student View



Associate Professor Wally Smith

wsmith@unimelb.edu.au

Office: 5.09, Doug McDonell Building

Email for an appointment

Head Tutor

Mark Jiang (Head Tutor)

yujing.jiang@unimelb.edu.au

Office: 6.22, Doug McDonell Building

Email for an appointment

You are currently logged into Student View

Resetting the test student will clear all history for this student, allowing you to view the subject as a brand new student.

The Weekly Workshop (starting on Week 2 of semester; no workshop in Week 1)

- A 2 hour Workshop (1 hour tutorial + 1 hour of supervised programming tasks)
- An online meeting tool called Zoom will be used by people who cannot attend the tutorial.
- An online tool called Grok will be used to carry out programming tasks in the workshop.
- Grok has an online chat tool that you can use to ask questions of the tutors, if you are not present at the workshop.
- Please try to enrol in a workshop as follows. Don't worry if you can't - we will re-arrange people as semester progresses to make it work ...
  For everyone who is affected by the travel restrictions, i.e., if you will be studying from China, you will be given access to one of FOUR ONLINE Zoom workshops (W9 Mon 4:15-6:15; W2 Tues 10-12; W4 Tues 12 - 2, W8 Fri 9 - 11). Note that these will change to run at more convenient times in China. If you are enrolled in a different workshop, please unenrol now and wait for the ONLINE workshops to open.
  For everyone who is NOT affected, i.e. if you are NOT studying from China during semester, please enrol in one of the following workshops (W1 Mon 4:15 - 6:15; W7 Tues 12:00 - 2:00; W3 Weds 3:15-5:15; W5 Thur 11:00 – 1:00; W6 Thur 3:15 - 5:15). Space will become available during Week 1 of semester.
- See COMP90059 in the Handbook - for latest timetable information

Private Study

- Private study outside of class is essential for you to develop a good understanding of this subject. You should spend an additional 5 hours per week: to work on Assessment tasks (see below), to review and carry out exercises and examples from class, and to carry out private study with Additional Resources.

Ground covered in COMP90059 overall

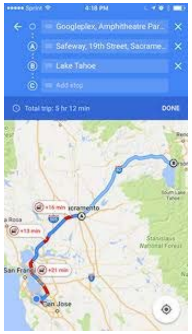
- Learn to use primitive data types and data structures
- Understand basic programming concepts
- Write simple applications that relate to a specific domain
- Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.
- Test applications

Assessment

- Assignment 1 (10%) a programming task due in Week 6
- Assignment 2 (20%) a programming task due in Week 12
- 10% on Mid-Semester Test held in the Lecture of Week 8.
  (If you are unable to attend this Lecture in person then your final examination mark will be weighted as 70% of your overall grade to make up.)
- Final Examination (60%) Assessing all class content (except where clearly stated as non-assessable)
  Duration : 3 hours + 15 minutes reading time

- **Hurdle requirement:** Final exam must be passed to pass the subject ie. 30 or more out of 60 marks.

What is a computer program?



Field	Value
Name	...siness Machines Corporation
Symbol	IBM
Exchange	NYSE
Exchange Name	New York Stock Exchange
Instrument	Equity
Market Open	9:30
Market Close	16:00
Market Capitalization	223,700,000,000
EBITDA	26,799,000,000
Shares Outstanding	1,142,848,384
Shares In Float	1,140,911,000
Average Daily Volume	3,749,070
52 Week Range	158.76 - 210.69
Short Ratio	3.900
EPS	13.765
EPS Current Year	0
EPS Next Year	0
EPS Next Quarter	0
P/E	14.050
Dividend/Share	3.200
Dividend Yield	1.650

IBM : DAILY International Business Machines C...
F : DAILY Ford Motor Co.
GOOG : DAILY Google Inc.
GM : DAILY General Motors Company
Place
For Help, press F1

A set of instructions given to a computer for it to execute a task.

Programming involves ...

- Understanding the task to be carried out.
- Designing a structured and precise process to carry out the task.
- Writing instructions to the computer in its special language for how to execute this process.

What does this code do?

```
principal = eval(input('Enter the initial principal: '))
apr = eval(input('Enter the annual interest rate: '))
numberYears = int(input('Enter the number of years: '))
apr = apr/100
for i in range(numberYears):
        principal = principal * (1 + apr)
print('The final value is: ', round(principal))
```

Note how it is made up of STATEMENTS which are executed by the computer in order.

What makes a piece of code good?

Commenting the code (= easier to understand)

```
# a program to calculate the future value of a sum
# based on: the initial principal, the annual percentage interest rate & number of years

#get the data from the user
principal = eval(input('Enter the initial principal: '))
apr = eval(input('Enter the annual interest rate: '))
numberYears = int(input('Enter the number of years: '))

#convert given apr to a fraction
apr = apr/100

#increase the principal value by the apr for each year
for i in range(numberYears):
        principal = principal * (1 + apr)

#report back the final value
print('The final value is: ', round(principal))
```

Natural Language vs Programming Language

Natural languages

- complex but flexible grammar rules
- meaning relies on context, and ambiguity is resolved by the listener
- missing or wrong elements might not matter too much
- extensive vocabulary, with subtle differences in meaning
- is understood at the speed of conversation

Programming languages

- have simple but strict grammar rules
- meaning must be clear and precise
- missing or wrong elements leads to a complete breakdown
- a small vocabulary of powerful expressions
- understood and acted on VERY QUICKLY in milliseconds or less

SYNTAX

Syntax is the name for the rules of a language.

It defines what counts as a correct statement in that language.

There are many different computer languages:

e.g. Python, JAVA, C, C++, COBOL, FORTRAN, ...

Each one is suited to writing instructions for different kinds of tasks, each one has its own syntactic rules.

ALGORITHM

An algorithm is a sequence of steps for the task, or a part of the task, that will work for every situation that we might encounter:

- Consists of a finite number of instructions
- Each individual instruction is well defined
- Describes a process that eventually halts after arriving at a solution to a problem
- Solves a general class of problems

EXERCISE: defining a process to arrange a meeting

- Six people need to sign an agreement. They all need to be physically present to sign at the same time. It will take just 30 mins to sign.
- They all have busy working lives in the CBD of Melbourne, and they need to arrange a date and time during working hours when they can do this.

- Identify the steps of the process you would follow to arrange a convenient meeting time for the group. Assume that you can contact each person separately by email, but you cannot share their email contacts with others in the group. Do not include the use of meeting tools in your process.

A human robot

Imagine Wally is a robot who only understands a programming language with the following commands. Could you ‘program’ him to walk from one side of the room to the other?

- raise left foot
- raise right foot
- lower left foot
- lower right foot
- move left foot forwards
- move right foot forwards
- move left foot backwards
- move right foot backwards
- turn left
- turn right

This works - but it's a bit laborious!

- raise left foot
- move left foot forward
- lower left foot
- raise right foot
- move right foot forward
- lower right foot
- raise left foot
- move left foot forward
- lower left foot

Making new commands … new functions

What if we could make new commands by connecting existing commands?

define step right foot:

- raise right foot
- move right foot forward
- lower right foot

define step left foot:

- raise left foot
- move left foot forward
- lower left foot

Now we can command the robot through a simpler program …

```
step right foot
step left foot
step right foot
step left foot
…
```

Repeating instructions… iteration

What if we get the robot to repeat a command X times?

```
repeat
    step right foot
    step left foot
    until obstacle
```

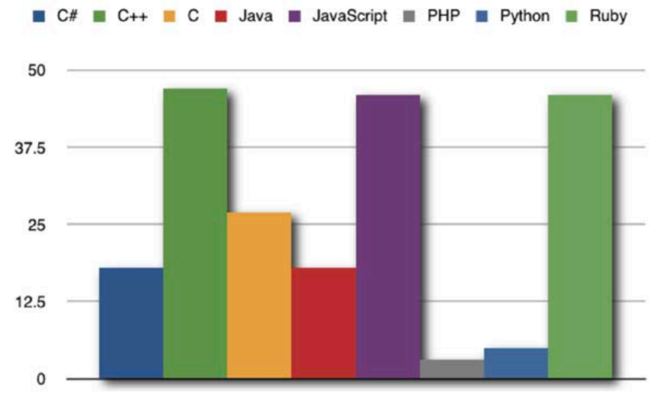
We define **obstacle** as detecting something in the way that stops movement going forward.

Python

The Python programming language

- Guido van Rossum created Python in the early 1990s
- Python is popular:
  - Python code is clear and close(r) to natural language
  - Extensive libraries
  - Python is free
  - It has a supportive community
- Useful resources at *www.python.org*

The relative proportion of profanities per line in code written in different languages



Source(s):

http://ilovecharts.tumblr.com/post/3463898034/amount-of-profanity-in-git-commit-messages-per

Install Python

- We will use Python v3.6 or later
- Programs are developed in Python IDLE (Interactive DeveLopment Environment)
- You just enter your code as text and the Python interpreter turns it into machine code for you
- Get a copy of python for your own machine at home - there are free versions for Windows, MacOS and Linux
  http://www.python.org/download/
- Portable version (USB) http://portablepython.com/
- Advanced Python Distribution (for scientific experimentation)
  http://www.enthought.com/products/edudownload.php

Grok Learning environment

- GROK Learning is the web-based programming environment we will be using for the duration of this subject in your labs:

  https://groklearning.com/course/unimelb-comp90059-2020-s1/
- All you need to access the system is a browser, an internet connection and your GROK account
- Different modes of working in GROK: code, run, mark, terminal
- To access GROK, you will need to login using your university email

Some first steps with Python

The print statement

A **print** statement evaluates an expression, or many expressions, and displays them, separated by one space

Syntax:

```
print (<expression>, ..., <expression>)
```

Examples:

```
>>>print('hello')
hello
>>>print (3)
3
```

variables, assignment & types (a first look)

A **variable** is something we create in a program to store a value under the name we give it.

Variables are of different **types**, and they can be given values through **assignment** statement.

Two important variable types

A string variable is a sequence of characters

```
name = 'Katie Mills'
print(name)
>>>
Katie Mills
```

An integer variable is a whole number

```
price = 17
print(price)
>>>
17
```

The **input** statement

An **input** statement puts a prompt message on the screen, and then return the value of the string that the user enters into the keyboard, terminated by the return key.

Syntax:

```
<string variable> =input(prompt)
```

Example: name = input('Enter your name: ')

or to get an integer from the user ...

```
<integer variable>=int(input(prompt))
```

Example: age = int(input('Enter your age in year: '))

or to get Python to decide what the user has given us ...

```
<variable> = eval(input(prompt))
```

Example programs and their output

```
#Say "Hello World"
print('Hello World')
>>>
Hello World
```

```
#Ask a lot of questions
print('How are you?')
print('Who are you?')
print()
print('What is your name?', 'Where do you live?', 'What is your favourite colour?')
>>>
How are you?
Who are you?

What is your name? Where do you live? What is your favourite colour?
```

```
#Say Your Name A lot
name = input('Enter your name: ')
print(name)
print(name, name)
print(name * 5)
name = name+" "
print(name * 5)
```

```
>>>
Enter your name: Sharon
Sharon
Sharon Sharon
SharonSharonSharonSharonSharon
Sharon Sharon Sharon Sharon Sharon
```

```
#Do some simple sums
string1 = input('Enter a number: ')
string2 = input('Enter another number: ')

number1=int(string1)
number2=int(string2)

sum = number1 + number2
product = number1 * number2
ratio = number1 / number2
power = number1 ** number2

print('Sum = ', sum)
print('Product = ', product)
print('Ratio = ', ratio)
print('Power = ', power)
```

```
>>>
Enter a number: 3
Enter another number: 4
Sum =  7
Product =  12
Ratio =  0.75
Power =  81
```

EXERCISE: Write a program to ...

- Print out the names of five countries in the world.
- Tells us what the sum of 57 + 82 is.
- Take the user's favourite animal name and make it appear on the screen one hundred times.
- Take the user's name and produces the text of the Happy Birthday song for that person.
- Take a sum of money in one currency and tell the user how much it is worth in another currency. (DIFFICULT)
- Take a person's birth date and then tell them approximately how many days they have been alive. (VERY DIFFICULT!)

Things to do before next class

- Install Python on your computer.
- Log on to GROK and try the first exercise with the turtle.
- Check out the resource about Python listed in CANVAS.
- Have a go at the program exercises from class - but don't worry if you find them too hard for now.
- Workshops start in Week 2 - look out for instructions about re-enrolling.