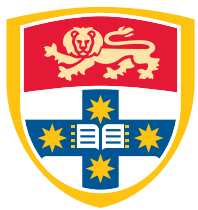
INFO1110 & COMP9001: Introduction to Programming

School of Information Technologies, University of Sydney



Copyright Warning

Week 1: First Steps

We will cover: Introduction to the unit, fundamental concepts, your first program

You should read: Sections 1.1 - 1.2, 2.1 of Sedgewick

An Introduction to Programming in Python. 7th Edition, Sedgewick, Wayne & Dondero. ISBN 9780134076430

Lecture 1: Introduction to Programming

Course overview and general guidelines

- This unit is *Introduction to Programming*.
- You will, I hope, by the end of this Semester, be able to write simple programs in Python and understand the basics of programming in general.
- This is a standard unit with extension material to keep everyone challenged.
- You will be expected to come to lectures and take notes. In your tutorial, ask questions and engage with the subject!
- We will do our best to teach you all — but *you* have to take responsibility for your own learning.

Where am I?

If you're wondering what you're doing in this unit, maybe something here will help:

- This unit is the lead-in to ALL the technical units in the School of IT
- This unit *does* require you to program!
- This unit has *no prerequisites*[1]

- (or possibly...) Actually yes, you've walked in to the wrong room.

[1] Except you know, having finished High School reasonably well and such. But note: we don't assume you can already program.

This unit

The unit is delivered in 13 weeks of lectures, with tutorials that you should attend. A register will be taken, and your attendance will be monitored.

There are *assessments* held during tutorials: missing them will not be a good idea as you'll miss out on the marks!

If you engage with the material, come to lectures, and *practice*, you will probably pass.☺

If you think you can learn it all by just reading the book and not programming, you will probably fail.☹

 ***Plagiarism — submitting someone else's work as your own — will not be tolerated. You MUST read and understand the University's policy documents on plagiarism. We use electronic means to identify potentially plagiarised work. You have been warned.***

What You Will Learn

Learning to program takes *work*!

There are some tricky concepts to get your head around:

- How does my code turn into a program?
- How does information get moved around in the computer?
- How do I turn this problem into a set of instructions to solve it?
- Why is the computer so stupid / why doesn't it know what I mean?

Most of you are in the same boat. You will learn a *lot* this semester — particularly if you are new to programming. You can do it!

Online resources: Canvas

This semester Canvas will be used for:

- Accessing your progressive grade
- Access to Lecture recordings
- Web links to other important places

canvas.sydney.edu.au

Online resources: edstem

We keep most of the course materials on edstem

Download lecture slides

Download tutorial exercises

Download assignment specification

Submit online assessments here!

edstem.org

Online resources: edstem

You will visit edstem.org quite often.

Announcements, information, updates and discussions

With the discussion forum, keep in mind:

When you post a question choose the most appropriate category.

Anyone posting code solutions to assessments will be banned, and may face further disciplinary action.

Any person doing anything inappropriate will face disciplinary action

edstem is an excellent platform for discussing and writing programs

Lectures are recorded

Available in canvas.sydney.edu.au

Audio is recorded and the screen. The screen images are only those shown on the display that is to the presenters left hand side.

Don't just listen or simply attend, take good notes. The better your notes the easier it is to revise.

Some, but not all, solutions to tutorial exercises will be provided.

Lecture slides are not full in detail

About The Coordinator/Lecturer

The unit coordinator and lecturer is
Dr. John Stavrakakis (John)

John manages all course related administration. Special consideration, special permission to enrol, student plagiarism, complaints/disputes, etc.

PhD in Computer Science. Specialise in 3D computer graphics. Coordinator/Lecturer for over 1500 students. Please be considerate of his time.

About the Teaching Assistant (TA)

The teaching assistant is **Mr. Tyson Thomas**

Prepares and delivers tutorial materials and conducts several course duties.

Bachelor of Computer Science and Technology (Adv), Strong interest in computational creativity and embedded systems

About our teaching team

We have a fantastic team of tutors. Each are talented in their own regard.

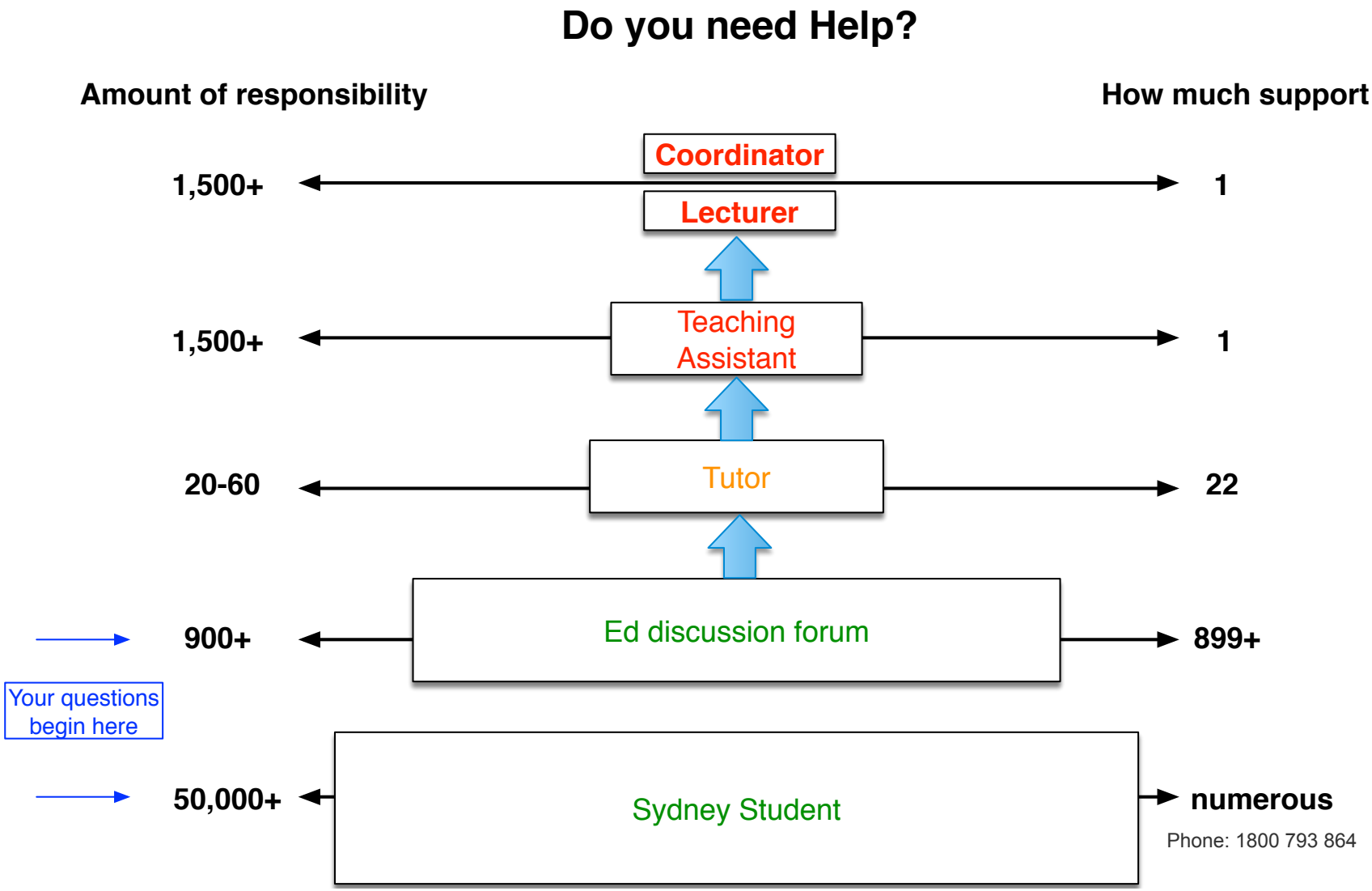
- Jonathan Du
- Frank Zhu
- Madeleine Wagner
- Brody Franks
- Alison Wong
- Vincey Au
- James Hardwick
- David Vo
- Eve Martin Jones
- Yuhao Wu
- Nahian-Al Hasan
- Rachel Dowavic
- Kelly Stewart
- Vincent Thong Nguyen
- Monica Lee
- Jonathan Chung
- Shenin Faizah
- Charles Christopher Hyland
- Jose Alejandro Vera Ospina
- Yining Guo

If you encounter any problems with our tutors, please contact the coordinator directly.

Where to get help?

1. Student admin → `https://sydneystudent.sydney.edu.au` or contact the Student Centre. Please check these first. E.g. timetable, passwords, payments, enrolments etc.
2. Ed discussion forum → `edstem.org`
3. Your tutor in your designated laboratory
4. The teaching assistant Tyson Thomas
5. Consultation time/place: 4 - 5pm Monday, Meeting room 302, Level 3, School of IT, Building J12

The hierarchy of help



Points That Sometimes Need Clarification

We are not out to get you: We want you to pass, but *you* have to do the work. If you feel that something has gone wrong, it might have, but *most things are fixable*.

"It's not fair": *This is as fair as you will ever get, ever again.* Everyone gets the same information and has access to the same resources. Everyone has the same requirements.

Reminders: At University you won't be reminded of when things are due: you have to organise your time and plan accordingly.

Assignments take time: The amount of time is not proportional to how many marks you can get from it, because assignments and quizzes and exams serve very different purposes.

"Introductory" does NOT mean "easy": No, it really doesn't. (Would you think "introductory quantum mechanics" was going to be simple?)

Points That Sometimes Need Clarification (cont.)

Attendance is not enough. We have had students, after they get their exam results, saying they deserved to pass, because they

1. tried really hard
2. went to every lecture
3. really need to pass

None of these are sufficient.

You are responsible. Now you're at University, you need to understand that, while we do care, and we do want you to do well, it's not just up to us. Our role is to help you to learn, to present you with all the resources you need, or the opportunities to get them: there's no spoon-feeding.

Points That Sometimes Need Clarification (cont.)

Language. If you're having trouble understanding written or spoken instructions, get help as soon as you can. Student Services has many resources to help you improve your understanding of English, organise your time, study, write reports, etc. Use them!

Time. You're probably enrolled full time, doing four 6cp units, or equivalent. Expect to spend about 10 hours per week for each 6cp unit.

Seriously, commit yourself to do 10 hours of *actual study*.

About you

- You're enrolled in INFO1110 & COMP9001 (I hope)
- or you're just interested in programming
- You have probably not done (much) programming before
- You are highly motivated!

When you want help

1. Look at your notes and course text referred to as Sedgewick.
2. Look on ed *before* asking the same question someone else has asked! ☺
3. Ask your friends (don't copy their code! you won't LEARN!)
4. Ask your tutor directly
5. The consultation time is a good chance to get a direct response on administration matters. Remember, this is not a tutorial, you cannot receive help for your assignments or the course materials explained in great detail.

Contacting staff

You should *definitely*

- post questions on ed if you're having problems,
- contact your tutor if you'd like more challenging things to do
- be really specific in your questions
- for serious/complex course issues, first email the coordinator and visit the consultation time.

You should *not*

- expect an immediate response
- expect all questions on ed to be answered by teaching staff
- post this question → "`my thing doesn't work`"
- expect to pass just by showing up.
- beg for more marks at any time.
- be late on the day of an assessment. **You won't be able to take it later**
- expect help on *beginning* of the course at the end
- writing this email → "`hi im in ur class can i get sum help`"

Assessments

What you'll have to do to do well

Assessments

Assessments include Assignments, Tasks, Quizzes and Computer examination.

All assessments are *individual* work.

You must get ≥40% of each major component to be permitted a pass: the major components are the *progressive mark* (Tasks, Quizzes, and Assignments combined), and the *paper exam mark*.

You must also get a combined mark of at least 50% in total, of course[2].

Here are some examples of how this works, in case the above isn't clear:

ProgMark 44%, Final Exam Mark 50%, total 48%: FAIL
ProgMark 75%, Final Exam Mark 35%, total 55%: FAIL
ProgMark 22%, Final Exam Mark 80%, total 51%: BUT FAIL

[2] If you are having trouble working this out, you are already in trouble.

Important Dates

- Week 1-13 one or more tasks due
- Week 5 Computer examination (45 minutes)
- Week 6 Assignment I due (10pm Monday 16 April)
- Week 7 Quiz 1 (50 minutes)
- Week 11 Computer examination (90 minutes)
- Week 12 Assignment II due (11pm Sunday 3 June)
- Week 12 Quiz 2 (50 minutes)
- ~~Week 15-18 Computer examination (90 minutes)~~

All the assessment details and schedule are available on CUSP[3]:

CUSP website for INFO1110

CUSP website for COMP9001

[3] Assignment I & II are correct in these slides

Quizzes 15%

- Week 7 worth **5%**
- Week 12 worth **10%**

These will assess your knowledge and skills for the course.

Closed book. 40-50 minutes time.

Covers all material from lectures and tutorials

You may write actual programming code, be neat.

These are on scheduled to be completed during your tutorial.
Absence or Lateness on the day will result in zero marks.[4]

[4] exception is Special Consideration

Practical assessment 10%

The practical assessment in this course is similar to an assignment. You will be given a problem to solve, allowed time to work on the solution. Complete this task with a lab computer.

These are on scheduled on Saturdays of specific weeks in the semester and are visible on your timetable.

There is no Internet to be used during this assessment.

Practical assessment 10%

- Week 5 worth **0%**
- Week 11 worth **10%**
- ~~Week 15-18 (Examination period) worth **10%**~~

This shows us that you know how to use a computer, but more importantly complete a programming problem given a specification.

Each assessment builds on the previous knowledge. It is important that you stay on top of the course!

Absence or Lateness on the day will result in zero marks.[5]

[5] exception is Special Consideration

Tasks 10%

Small programming exercises to be completed online with edstem

There will be both easy and hard questions. They can take anywhere from 5 - 60 minutes to complete.

Due dates are throughout the semester

Late submissions will be penalised at 50% per day.

Why so many assessments?

It is a reflection of how involved you are in the course.

Your study time, used correctly, will make these relatively easy. This is reasonable assumption of any University Student.

Most of your preparation time is practice tasks and tutorial exercises with *actual programming*

Final exam.

- The exam will be 2 hours long with 10 minutes reading time
- It will contribute 50% to your final grade.
- You will be permitted 1 A4-size sheet of paper with notes on it (yes, both sides)[6].

[6] unless you have Möbius paper but that's your own fault if you do

Automatic Marking

We use several ways to test your program. Generally, input vs output.

We give your program input data and compare the output with what we expect.

The *kind* of testing that is done by automatic marking is made known to you. Not everything is known to you.

Pay attention to the assignment specification to derive your own test data before you submit.

Automatic Marking (cont.)

For example, suppose you have a program called `MyNameIs`, which should print out a friendly message when run with the input "Bruce", and a question otherwise:

```
> python MyNameIs Bruce
Gday mate.
> python MyNameIs Bill
Is your name not Bruce?
```

If, when we run your program with the same input, we get this, then you would fail a test:

```
> python MyNameIs Bruce
Like, dude!
```

The above output would *fail* the test.
Similarly this would also fail:

```
> python MyNameIs Bruce
Is your name not Bruce?
```

Automatic Marking (cont.)

We have many tests for each program you write. To get full marks you must pass each test.

We also keep tests *hidden*: you won't know until after the deadline how you've gone on those.

That means you'll have to really understand your code and think carefully about all kinds of possible inputs to ensure your program will handle them properly.

More details on submission

The program *must be written in Python* and must compile and run on the lab computers or other platform that is specified in the assignment

The proportion of tests your code passes will be used for your automatic mark for each assessment.

You may be required to *explain* your code to your tutor, or to the Unit Coordinator (i.e., me). If you can't explain what it does, you won't get a mark.[7]

Don't get other people to do your assignments for you. Really.

[7] If you can't understand it, why would you submit it?

Plagiarism

 ***We run tests on your submissions to see how similar they are to those of other students.***

The software we use is very good at helping us detect plagiarism, so don't do it.

 ***We have failed students the course on the basis of plagiarism.***

Sharing is caring...is learning

Outside assessments we want you to pursue the discussion of programming problems, ideas, approaches, suggestions with your peers

If you can discuss the problem and receive feedback, then debug and correct your own code, you would have the equivalent of mining skill +1

If you can resolve someone else's programming problem without rewriting their code, you would have repair skill +1

There are many different kinds of programming skills!

Recommendation: survey as much as you can of this landscape and shift your character from explorer to navigator!

Times and places

INFO1110 & COMP9001 lectures weeks 1-13.

- 1 - 2pm Monday Chemistry Lecture Theatre 1

INFO1110 & COMP9001 Tutorials weeks 1-13.

- Shown on your timetable. One 2 hour session.
  Go to the tutorial you are scheduled for
  Do not miss class, except for illness, emergencies, etc
  Get help from staff if you feel you are falling behind

INFO1110 & COMP9001 Computer Examination

- Week 5, Saturday 14 April
- Week 11, Saturday 26 May
- Week 15 - 18, To Be Announced

Outside the course: Python help desk

This is not affiliated with this course.

A program to provide additional help with Python programming. Run by volunteer students!

Starting from **week 3**, and runs every week of semester

Details about time and place to be posed on edstem

Technology Society (SUITS)

This is not affiliated with this course.

They run various events throughout semester, both fun and professional.

Meet people

They have a shiny website: `https://suits.org.au`

Outside the course: Competitive Online Gaming Society

This is not affiliated with this course.

They run various events throughout semester, mostly fun and distracting.

Gaming not the same as programming but it is close.

Meet people

They spend more time on gaming than their own website: Competitive Online Gaming Society

Welcome to the beginning!