

Introduction to Programming

Dr. John Stavrakakis

School of Computer Science, University of Sydney



COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of the University of Sydney pursuant to Part VB of the Copyright Act 1968 (**the Act**).

The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.

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 0: 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 attend lectures, tutorials 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.

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 Computer Science
- This unit *does* require you to program!
- This unit has *no prerequisites*^[1]
- 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. 😞

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

Tutorial timetable may change during week 1 and week 2!

Please check your unikey email for any updates to your timetable

The time may change,

OR

The venue may change,

OR

both!

Learning to program takes *time and effort!*

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!

This semester Canvas will be used for:

- Accessing your progressive grade
- Access to Live Lecture recordings
- Access to online Lecture recordings (required)
- Access to *extra* online lecture recordings for Advanced (INFO1910 required)
- Web links to other important places

canvas.sydney.edu.au



≡ 2021_S2C_INFO1110_COMP9001_ND



Account



Help



Dashboard



Courses



Calendar



Inbox



History



Studio



OLE

2021 Semester 2

Home

Ed Discussion

Announcements

Assignments

Modules

Recorded Lectures

Marks

Reading List

Zoom

FE Student Portal

Unit outlines

Recent Announcements



Zoom tutorial links
INFO1110 Labs Code Day Start Ti...

Posted on:

Aug 4, 2021 at 23:41

INFO1110 COMP9001 Introduction to Programming




[Unit of Study Information INFO1110](#) ↗

[Unit of Study Information COMP9001](#) ↗


[ICT Help](#) ↗ | [Academic Honesty](#)

This unit is an essential starting point for software developers, IT consultants, and


Online lectures via Canvas (cont.)




THE UNIVERSITY OF SYDNEY




Account




Help




Dashboard




Courses




Calendar




Inbox



History



Studio



OLE

≡ 2021_S2C_INFO1110_COMP9001_ND > Modules

2021 Semester 2

Home

Ed Discussion

Announcements

Assignments

Modules

Recorded Lectures

Marks

Reading List


Zoom

FE Student Portal

Unit outlines

▼ Unit Information

 Staff and Contact Details

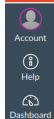
 Lecture Slides and Course Resources

 Course Discussions - Ed

▼ Tutorial Information

▼ Online Lectures

 Week 1



≡ 2021_S2C_INFO1110_COMP9001_ND > Pages > Week 1

2021 Semester 2

Home

Ed Discussion

Announcements

Assignments

Modules

Recorded Lectures

Marks

Reading List

Zoom

FE Student Portal

Unit outlines

Week 1

Live lecture - Week 1 - Course Introduction

To be published when available

Online lecture - Week 1 L02 - Programming basics

Python syntax

In order for the compiler to turn your code into a working program, the code has to obey a lot of syntax rules. You will pick up many of these as you go along, but let's just list a few now too:

- Some words are *reserved*—e.g., `import`, `int`, `float`, `if`, `else`. You can't use these for variable names.
- Variable names can't begin with numbers.
- Expressions are delineated with parentheses ()
- Array items are accessed with brackets []
- Control flow statements should end with colons :
- Strings are delimited by double quotes " " or single quotes ' '
- The code is only executed if it is indented correctly (ouch!)

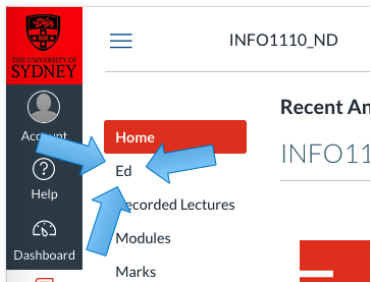
It's very easy to get things wrong. Don't worry. Most things are fixable. [DIVE IN!](#)

We keep most of the course materials on Ed

Download lecture slides, lab exercises, assessment specification

Submit online assessments here!

Ed



You will visit Ed 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

Ed is an excellent platform for discussing and writing programs

2 lectures per week:

- Lecture 1 → Live recordings (Zoom Lecture, 1 hour)
- Lecture 2 → Online recordings (you must watch these, 40m - 2 hour)
- Lecture 3 (Advanced only) → Online recordings

Available in canvas.sydney.edu.au

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

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

Lecture slides are not full in detail

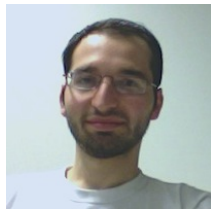
- Review existing material
- Further practice
- More open for questions

Starting week 1

Attendance optional

The entire session is recorded and viewable online

If you cannot attend that is OK



The unit coordinator and lecturer is
Dr. John Stavrakakis

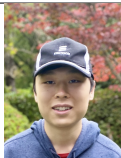
PhD in Computer Science
Specialises in 3D computer graphics

Overall course administration and design

Coordinator/Lecturer for over **800** students.

Please be considerate of his time.

About the Teaching Assistants (TA)



Andrew Xu

Lost money in crypto :(



Xinwei Luo

COH2 and cats cats cats

The teaching assistants help with the preparation and delivery of the course contents. Seminars, labs, tutorials, quizzes, assignments, challenges, computer examinations. They also conduct several other duties.

About our teaching team

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

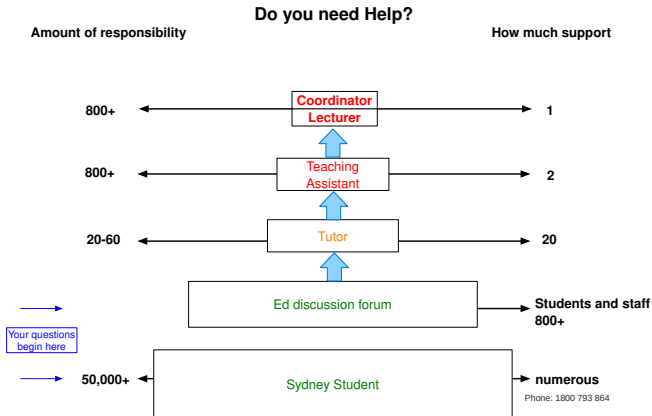
- Has Brooke
- Christopher Irving
- Steven Condell
- Hanin Zenah
- Jung (Monica) Lee
- Xiaowen Hu
- Hamish Croser
- Alexander Tan
- Jason Lai
- Mengqi (Grace) He
- Ziyun (Erik) Chi
- Pranav Alavandi
- Imran (Michael) George Ashshiddiq Podbury
- Tiancheng Mai
- Judd Zhan
- Amir Harambasic
- Andre Georgis
- Vincent Qu

If you have issues with tutors, please contact one of the TA 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 → [Ed](#)
- 3 Your tutor in your designated laboratory
- 4 Contact a TA for administration → [Ed](#) (private thread)
- 5 The teaching assistants email: [Andrew Xu](#), [Xinwei Luo](#)
- 6 TA INFO1910 [Advanced](#) [Alan Robertson](#)
- 7 Consultation with [Dr. John Stavrakakis](#):
16:15 - 17:15 Monday: <https://uni-sydney.zoom.us/j/81472012634>

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

- ❶ tried really hard
- ❷ went to every lecture
- ❸ 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*.

- You're enrolled in INFO1110 or INFO1910 or 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 lab, you cannot receive help for your assignments or the course materials explained in great detail.

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 teaching assistant and visit the consultation time with the coordinator.

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.

The *progressive mark* is the sum of all Quizzes and Assignments.

You must get $\geq 40\%$ of the progressive mark and you must get $\geq 40\%$ of the *final exam mark* to be *eligible* to pass.

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.

All the assessment details and schedule are available on Unit of Study website:

UoS information for INFO1110

UoS information for INFO1910

UoS information for COMP9001

INFO1910 Advanced has different assessments with different weights. Please check the UoS page, announcements from the lecturer, and from your tutor.

Assessment	Due Date	Weighting
Assignment 1	Week 4 23:59 05/09/21	10%
Quiz 1	Week 7 16:00 20/09/21	10%
Quiz 2	Week 10 16:00 18/10/21	10%
Assignment 2	Week 12 23:59 07/11/21	20%
Final Exam	Formal Exam Period	50%

What are the assessments for?

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 lab exercises with *actual programming*

Assessments are essential practice for the final examination.

Assess your knowledge and skills for the course

Closed book. 30 minutes time

Covers *any* material from lectures and labs

You may need to write actual programming code

For illness or misadventure please file for special consideration.

Assignments are asking you to solve much larger problems that may take hours to compose, test, debug and implement.

They are intentionally challenging and are available to complete for 14 days from release to deadline. Start early!

They are partly marked by your tutor and there will also be a component of your mark from an automatic tester.

You may be required to explain your code to the teaching team. If you cannot, you will get 0.

Final exam.

- The exam will be 2 hours long with 10 minutes reading time
- It will contribute 50% to your final grade.
- The exam will be *closed book* but there will be reference pages provided. The reference pages will be accessible before the examination.

More information on Ed [About the final examination and pass requirements](#)

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 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.

Automatic Marking (cont.)

We may grade the assessment using different, but similar, data.

For example write a program that prints the addition of two numbers.

Test	First number	Second number	Expected	Computation
1	3	4	7	$3 + 4 = 7$
2	5	1	6	$5 + 1 = 6$

```
1 if first_number == 3:
2     print(7)
3
4 if first_number == 5:
5     print(6)
```

Is this program correct?

The program *must be written in Python* and must compile and run on the lab computers or other platform that is specified in the assessment. There are public tests, pre-deadline hidden test cases and post-deadline private test cases.

- public tests show where you pass/fail and reveal some information about the nature of the test
- hidden tests show does not disclose what was tested
- pre-deadline - shows you pass/fail
- post-deadline - only be performed after the deadline

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.^[3]

Further information [Assessment procedures 2011](#)

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

^[3]If you can't understand it, why would you submit it?



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 different kinds of academic dishonesty, so don't do it.

The software we use is very good at detecting and alerting staff for dishonest practices. Staff are able to confirm such activity through investigation. So please don't do it.



We have failed students in the course on the basis of academic dishonesty



Plagiarism, Collusion, Contract cheating — submitting someone else's work as your own — will not be tolerated. You MUST read and understand the University's policy documents on **Academic dishonesty and plagiarism**. We use both electronic and human means to identify dishonesty. You have been warned.

What can you cannot expect, or ask help for

All assessments in this course are an individual's effort

You cannot discuss with your friends about the assessment, or share any code. You cannot describe, dictate, explain, draw, or communicate in any way how you solved the assessment or even ideas on how to solve it.

You cannot ask staff about the assessment or expect any help on those activities. The assessments are a measure of your understanding and acquired skill of the course.

You need to complete the assessment using the course materials and other specified reference materials (such as the reference to API documents).

If the work you copy is not your own, then you will be reported for Academic Integrity.

Private study - every week!

- Textbook readings, revise previous week, review upcoming materials, write lots of programs! save them in your Ed workspaces!

Lectures weeks 1-13

- 1 hour session. Monday afternoon.
Please see your timetable. [For link, see here](#)
- 1 hour (average) session. [Online in Canvas see here](#)

Seminar weeks 1-13

- 1 hour session. [9am Wednesday See here](#)

Labs weeks 1-13

- Shown on your timetable. One 2 hour session.

USU clubs and societies

- USYD ROCKETRY
- MADSOC - Movement and Dance Society
- SUMS - Sydney University Maths Society
- WASABI - Japanese cultural exchange
- TAEKWONDO
- SCA - Society for Creative Anachronism
- SYNCSS - Sydney Computing Society
- SYDNEY MOTOR SPORT
- SUGEX - Sydney University Global Exchange Society
- SUCC - Chess club

These are not affiliated with this course.

Meet people

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

Reading for this week

Sections 1.1 - 1.2, 2.1 of Sedgewick

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

Online version found in our local library:

<https://library.sydney.edu.au/>

Welcome to the beginning!

