COMP2041 22T2 — Course Introduction

https://www.cse.unsw.edu.au/~cs2041/22T2/

COMP(2041|9044) Staff

Convenor, Lecturer Andrew Taylor

    Admin Dylan Brotherston

    Tutors Aisha Nauman, Angeni Bai, Bridget McCarthy, Elliot Rotenstein, Jack Peng, Jesse Merhi, Max Owen, Nasser Malibari, Rexana Jiang, Sage Barreda-Pitcairn, Sunny Chen, Zander Zhuang, Zelun Li, Zhenqi Wang

Course Goals

- First programming courses deal with . . .
    - one language (C or Python at CSE)
    - one program
    - small(ish) tightly-specified examples
    - narrow aspects of programming (e.g. basics, correctness)
- COMP(2041|9044) deals with . . .
    - other languages (Shell)
    - combining multiple programs to solve problems
    - larger (less-small) less-specified examples
    - tools for working with software (e.g. git)
    - configuring systems (e.g package managers, mounting)
- get you to the point where:
    - you could build a package
    - put it on github
    - and have people download & use it
- TLDR - COMP2041/COMP9044 will expand your coding skills

Assumed Knowledge

At the start of this course you should be able to:

- write, debug, test programs in C or Python
  - OK for COMP2041/COMP9044 if you don't know C
  - basic Python knowledge will be assumed
  - COMP9021, COMP1531, learning Python elsewhere, sufficient
- appreciate the use of abstraction in computing

Changes from recent years

- no web frontend/backend programming
    - moved to COMP6080
    - script to scrape/download web data covered
- no Perl
    - Python will be used to teach same material
    - Perl much less important that when COMP2041 started
    - basic Python assumed, more covered

Lectures

- Monday, 12:00—14:00; Thursday 14:00—16:00;
  delivered via YouTube
  - you will have email about how to access the event
  - feel free to ask questions via chat
  - lectures recorded and linked from course home page
- present a brief overview of theory
- focus on practical demonstrations of coding
- demonstrate problem-solving (testing, debugging)
- Lecture slides available on the web before lecture.

Tutorials

- Tutorials start in week 1.
    - please wear a mask in face-to-face classes
    - online classes are via Blackboard Collaborate
- tutes clarify lecture material
- work through problems related to lecture topics
- give practice with design *(think before coding)*
- answers available on the class webpage Friday afternoon

To get the best out of tutorials

- attempt the problems yourself beforehand
- ask if you don't understand a question or how to solve it
- Do *not* keep quiet in tutorials . . . talk, discuss, . . .
- Your tutor may ask for your attempt to start a discussion.

Lab Classes

Each tutorial is followed by a two-hour lab class.

- Several exercises, mostly small implementation/analysis tasks

- Aim to build skills needed for assignments, exam

- Aim to give experience applying tools/techniques

- Done individually

- Submitted via `give`, before Tuesday 09:00 following week

- Automarked (with partial marks) — 15% of final mark

- Labs may include challenge exercises:

    - may be silly, confusing, or impossibly difficult

    - 95% possible for labs without completing any challenge exercises

Weekly Tests

From week 3, weekly tests:

- programming tests
- immediate reality-check on your progress.
- done in your own time under self-enforced exam conditions.
- Time limit of 1 hour
- Automarked (with partial marks) — 10% of final mark
- best 6 of 8 tests used to calculate the 10%
- any violation of test conditions $\Rightarrow$ zero for whole component

Assignments

- Assignments give you experience applying tools/techniques to larger programming problems than lab exercises

- Assignments will be carried out individually.

- They *always* take longer than you expect.

- Don't leave them to the last minute.

- There are late penalties applied to maximum marks, typically 2%/hour — organising your time $\Rightarrow$ no penalty

Code of Conduct

CSE offers an inclusive learning environment for all students.

In anything connected to UNSW, including social media,
these things are student misconduct and will not be tolerated:

- racist/sexist/offensive language or images

- sexually inappropriate behaviour

- bullying, harassing or aggressive behaviour

- invasion of privacy

Show respect to your fellow students and the course staff

Plagiarism

Cheating of any kind constitutes academic misconduct and carries a range of penalties. Please read course intro for details.

Examples of inappropriate conduct:

- groupwork on individual assignments (discussion OK)
- reading someone else's solution before stating an assignment
- allowing another student to copy your work
- getting your hacker cousin to code for you
- purchasing a solution to the assignment

Plagiarism

- Labs, tests, assignments must be entirely your own work.

- You can not work on assignment as a pair (or group).

- Plagiarism will be checked for and *penalized*.

- Plagiarism may result in suspension from UNSW.

- Scholarship students may lose scholarship.

- International students may lose visa.

- Supplying your work to any another person may result in loss of all your marks for the lab/assignment.

Final Exam

- online practical exam, during exam period; you complete from home
- closed-book — limited on-line language documentation available
- some multiple-choice/short-answer questions, similar to tut questions.
- some questions will ask you to read shell, Python, regex, . . .
- six (probably) implementation questions, similar to lab exercises
- most marks for questions which ask you to write shell or Python
- also may ask you to answer written questions
- you *must* score 18+/45 on the final exam to pass course

Assessment

- 15% Labs
- 10% Weekly Programming Tests
- 15% Assignment 1 — due week 7
- 15% Assignment 2 — due week 10
- 45% Final Exam

Above marks may be scaled to ensure an appropriate distribution
To pass you must:

- score 50/100 overall
- score 18/45 on final exam

For example:

55/100 overall and 17/45 on final exam $\Rightarrow$ **55 UF** not 55 PS

How to Pass this Course

- coding is a *skill* that improves with practice
- the more you practise, the easier you will find assignments/exams
- do the lab exercises
- take weekly tests seriously
- start the assignments early
- practise programming outside classes
- treat extra tutorial questions like exam practice