School of Computer Science and Engineering

COMP9021

PRINCIPLES OF PROGRAMMING

Session 2, 2016

Contents

1 Course staff

Lecturer in charge: Eric Martin

> **Office:** building K17, room 409
>
> **Email:** emartin@cse.unsw.edu.au
>
> **Phone:** 9385 6936

Tutors: Manas Patra (m.patra@unsw.edu.au) and Peizhi Shao (z3385128@student.unsw.edu.au)

The lecturer in charge will be answering e-mails related to the course, provided that they do not require extensive or substantive answers. Questions that cannot be answered shortly are best raised in consultation. In order to provide maximal flexibility, there is no fixed consultation time. If you need a consultation please email the lecturer in charge for an appointment.

Another way to communicate with the lecturer in charge and fellow students is via WebCMS3, that allows all participants to the course to post questions and answers in relation to every resource available at the course website, accessible at

`http://www.cse.unsw.edu.au/~cs9021`

You will need to login to post your own messages or to reply to messages posted by other participants.

2 Course details

Units of credit: 6

No parallel teaching: only COMP9021 students attend the classes.

3 Course aims

This is a Level 0 course. It has no prerequisite. Like most Level 0 courses, it consists of bridging material in computing taught at an accelerated pace. It is a prerequisite to a number of courses including COMP9024 Data Structures and Algorithms, which itself is a prerequisite to most courses that can be taken as part of the Graduate Certificate in Computing (program 7543), the Graduate Diploma of Information Technology (program 5543), and the Masters of Information Technology (program 8543). Students who have already covered the material presented in this course can get exemption if they pass the corresponding exemption exam or have been exempted on the basis of academic background.

The aim of the course is to provide students with a solid foundation on fundamental programming concepts and principles, develop problem solving skills, and master the programming language python. Students will learn to design solutions to a broad range of problems and implement those solutions in the form of small to medium programs, using appropriate programming techniques and tools.

4 Student learning outcomes

- Know how to design, implement and test programs written in a language with procedural, object-oriented, and functional constructs.
- Be proficient in the python language, and know what happens behind the scene especially in terms of memory use.
- Be proficient in designing and implementing widgets.
- Have good knowledge of fundamental data structures and algorithms.
- Know how to design programs to solve small to medium scale problems.
- Be able to write clear, reliable, well-structured, well-tested, well-documented programs in python.
- Be proficient in the use of appropriate tools, debuggers in particular.
- Know how to represent data with linked lists, stacks, queues, heaps, and binary trees.
- Know how to use and be able to implement searching and sorting algorithms.

5 Overall approach to learning and teaching

You know that at university, the focus is on your self-directed search for knowledge. Lectures, labs, textbook and recommended reading, assignments and exams are all provided as a service to assist you in this endeavour. It is your choice as to how much work you do in this course, whether it is preparation for classes, completion of assignments, study for exams or seeking assistance or extra work to extend and clarify your understanding. You must choose the approach that best suits your learning style and goals in this course. Still note that the University expects you to do about 150 hours work for this course—including lectures, labs, and time spent on self-study and assignments.

Of course this will vary according to your aims. The course is designed in such a way that passing the course will only require a good understanding of the fundamental notions as well as good practical skills, thanks to regular work. If your aim is to obtain a high distinction then you will need to invest more time in this course.

6 Teaching strategies

Lectures introduce the material. Supervised labs in the first few weeks are meant to have individual contact, resolve any technical issue that might arise in relation to your working environment, and give you a chance to start practicing and get some immediate feedback on your work and answers to your questions. From week 5 onwards, one lab will still be supervised, and students who experience difficulties are very strongly encouraged to attend it and seek support. And of course, email and WebCMS3 are there for you to get help when needed, and you can also contact the LIC would you need a consultation. The lab programming exercises will guide you in building programming and problem solving skills, and to prepare you for the midterm and final exams. Also, from week 2 to week 11 included, programming quizzes will be released after the lecture and your answers should be submitted by midnight on Wednesday of the following week. This will help you master the fundamental notions and techniques that will have been presented during lectures, keep up to date with the current material, and give you confidence that you are well on track. The three assignments are where the bulk of the work will be done. Assignments will allow you to turn theory into practice, transform passive knowledge into active knowledge, design solutions to problems, and experience the many ways of making mistakes and correcting them when translating an algorithmic solution to an implementation.

7 Assessment

The assessment for this course will be broken down as follows.

Assesment item	**Maximum mark**
10 weekly programming quizzes	10
Assignment 1	10
Assignment 2	10
Assignment 3	10
Midterm exam (2 hours)	20
Final exam (3 hours)	40

The final mark will the arithmetic mean of all assessment items. To pass the course, you will need to get a total mark of 50 at least.

Programming quizzes will be released from week 2 to week 11 after the lecture. Typically, you will have to complete incomplete programs, allowing you to check your understanding of the fundamental notions that will be presented during lecture. In order to be marked, your answers to the weekly quizzes should be submitted by midnight on Wednesday of the following week, via WebCMS3, by just uploading some files as directed in the quiz specifications. Every quiz will be worth up to 1 mark.

Longer programming exercises will be released from week 1 to week 12 as part of the labs to help you practice in more depth the key material presented in the previous week and as a preparation for the midterm and final exams. More precisely, most labs will have one flagged programming question or more, some of which will be exam questions modulo some small variations.

The three assignments will be programming assignments. Each of the assignments will require you to develop problem-solving skills, the ability to design, implement and test solutions to problems, and to gradually acquire all the skills listed in Section 4.

Assignments will have to be submitted via WebCMS3, by just uploading some files as directed in the assignment specifications. Each assignment will be automatically assessed for correctness on a battery of tests.

The assignments give you the chance to practice what you have learnt and design solutions to common, small to medium scale problems. The learning benefits will be greater if you start working on the assignments early enough; do not leave your assignments until the last minute. See Section 9 to find out on when each of the three assignments is due (always by midnight on a Sunday). Assignments submitted late are subject to the following penalty. The maximum mark obtainable reduces by 1 mark per day late. Thus if students A and B hand in assignments worth 9 and 6, both two days late, then the maximum mark obtainable is 8, so A gets $\min(9, 8) = 8$ and B gets $\min(6, 8) = 6$.

Note that if you want help with your programs, you should make an appointment for consultation and present an up-to-date listing that is reasonably well laid out and documented. You must also bring evidence of having taken reasonable steps to solve the problem yourself. Do not send an email to the lecturer in charge of the form: *My program is attached. How does it come it does not work?*

For the midterm exam, which will take place in computer labs, you will have to write short programs, that will be variants of some of the flagged programs of the lab questions given before the midterm exam.

The format of the final exam will be the same as the format of the midterm exam.

It should be noted that no supplementary midterm exam will be offered. Students unable to attend the midterm exam due to illness should submit a request for special consideration within seven days after the exam. Students whose requests are granted will have their midterm component computed on the basis of their results in the final exam. Students whose requests are denied will receive zero mark for the midterm. A supplementary final exam will be offered only to students who submit a request for special consideration meeting the School's usual criteria (see the above) within seven days of the final exam, and perform at 50% or better in the midterm exam. (Students who were offered special consideration on the midterm exam and also request special consideration on the final will be handled at the discretion of the lecturer in charge, but will be expected to prove exceptional circumstances for both the midterm and final.) Please note that lodging an application for special consideration does not guarantee that you will be granted the opportunity to sit the supplementary exam. A supplementary final exam will only be offered to students who have been prevented from taking an end of session examination, and whose circumstances have improved considerably in the period since the exam was held. Students who apply for special consideration must be available during this period to sit the exam. No other opportunities to sit the final exam will be offered.

8 Academic honesty and plagiarism

What is Plagiarism?

Plagiarism is the presentation of the thoughts or work of another as one's own.[1] Examples include:

- direct duplication of the thoughts or work of another, including by copying material, ideas or concepts from a book, article, report or other written document (whether published or unpublished), composition, artwork, design, drawing, circuitry, computer program or software, web site, Internet, other electronic resource, or another person's assignment without appropriate acknowledgement;
- paraphrasing another person's work with very minor changes keeping the meaning, form and/or progression of ideas of the original;
- piecing together sections of the work of others into a new whole;
- presenting an assessment item as independent work when it has been produced in whole or part in collusion with other people, for example, another student or a tutor; and
- claiming credit for a proportion a work contributed to a group assessment item that is greater than that actually contributed.[2]

For the purposes of this policy, submitting an assessment item that has already been submitted for academic credit elsewhere may be considered plagiarism.

[1] Based on that proposed to the University of Newcastle by the St James Ethics Centre. Used with kind permission from the University of Newcastle

[2] Adapted with kind permission from the University of Melbourne.

Knowingly permitting your work to be copied by another student may also be considered to be plagiarism.

Note that an assessment item produced in oral, not written, form, or involving live presentation, may similarly contain plagiarised material.

The inclusion of the thoughts or work of another with attribution appropriate to the academic discipline does not amount to plagiarism.

The Learning Centre website is main repository for resources for staff and students on plagiarism and academic honesty. These resources can be located via:

http://www.lc.unsw.edu.au/plagiarism

The Learning Centre also provides substantial educational written materials, workshops, and tutorials to aid students, for example, in:

- correct referencing practices;
- paraphrasing, summarising, essay writing, and time management;
- appropriate use of, and attribution for, a range of materials including text, images, formulae and concepts.

Individual assistance is available on request from The Learning Centre.

Students are also reminded that careful time management is an important part of study and one of the identified causes of plagiarism is poor time management. Students should allow sufficient time for research, drafting, and the proper referencing of sources in preparing all assessment items.

Assignments will be checked. The penalties for copying range from receiving no marks for the assignment, through receiving a mark of 00 FL for the course, to expulsion from UNSW (for repeat offenders). Allowing someone to copy your work counts as plagiarism, even if you can prove that it is your work.

We are aware that a lot of learning takes place in student conversations, and don't wish to discourage those. However, it is important, for both those helping others and those being helped, not to provide or accept any programming language code in writing; this leads to plagiarism penalties for both the supplier and the copier of the codes. Use a piece of paper to write down your ideas, but take it away when the discussion is over.

If you are new to studying in Australia, be aware that attitudes to plagiarism at UNSW may be different from those in your home country. In brief, and for the purposes of COMP9021, plagiarism includes copying or obtaining all, or a substantial part of your assignments (C code). Note that if you copy code from another student or non-student with acknowledgement, you will not be penalised for plagiarism, but you are unlikely to get any marks for the copied material. If you use code found in a publication (on the internet or otherwise) then the marks you get for this will be at the marker's discretion, and will reflect the marker's perception of the amount of work you put into finding or adapting the code, and the degree to which you understand the code.

Note also that there is a big difference between being able to understand someone else's code, and writing that code yourself from scratch. A computer programmer has to be able to write code from scratch. The assignments provide opportunities for you to develop the skills necessary to write your own code. Use these opportunities!

9 Course schedule

The following table outlines a provisional schedule for this course. In the **Date** column, **L** refers to the lecture, **A** to the due date of an assignment (midnight of that day), **Q** to the due date of a quiz (midnight of that day), and **E** to an exam. Lecture contents is described very roughly, and subjected to adjustments.

Week	**Date**	**Lecture contents**	**Assessment**
1	**L**: 28 Jul	Introduction Writing and executing code Operators	
2	**L**: 4 Aug	Assignments Control flow Strings Documentation and testing	
3	**Q**: 10 Aug **L**: 11 Aug	Lists, tuples Dictionnaries, sets Debugging	Quiz 1
4	**Q**: 17 Aug **L**: 18 Aug	Functions Identity versus equality, references	Quiz 2
5	**Q**: 24 Aug **L**: 25 Aug **A**: 28 Aug	Generators Files Regular expressions	Quiz 3 Assignment 1
6	**Q**: 31 Aug **L**: 1 Sep	Objects Inheritance Polymorphism	Quiz 4

7	**Q**: 7 Sep **L**: 8 Sep **E**: 9 Sep	Recursion Simulations	Quiz 5 Midterm exam
8	**Q**: 14 Sep **L**: 15 Sep	Linked lists Decorators	Quiz 6
9	**Q**: 21 Sep **L**: 22 Sep	Stacks Queues	Quiz 7
	A: 2 Oct	Mid-session recess	Assignment 2
10	**Q**: 5 Oct **L**: 6 Oct	Heaps Hashing	Quiz 8
11	**Q**: 12 Oct **L**: 13 Oct	Trees Binary search trees	Quiz 9
12	**Q**: 19 Oct **L**: 20 Oct	Sorting	Quiz 10
13	**A**: 30 Oct		Assignment 3

10 Resources for students

Announcements, lecture notes, example programs, jupyter notebooks, lab exercises and solutions, quizzes and assignment specifications are made available at the course's homepage:

http://www.cse.unsw.edu.au/~cs9021

There is no required textbook, though you can consider the following as the "official" textbook for this course:

Bill Lubanovic: *Introducing Python: Modern Computing in Simple Packages.* O'Reilly Media

For the syntactic aspects of the language, the official documentation will be complemented with Jupyter notebook sheets. Lecture notes will cover some conceptual and algorithmic topics, meant to provide insight and not repeat what is being abundantly described is easily available books and online resources. Often, a Google search will be the most effective way to get answers to your questions.

Here are some recommendations, but you will very certainly come across other resources, and you are encouraged to share your great findings with everyone. . .

For easy introductions to python, I recommend:

John Zelle: Python Programming: An Introduction to Computer Science

They can be complemented with:

Brad Miller and David Ranum: Problem Solving with Algorithms and Data Structures Using Python

and with:

Allen B. Downey: How to think like a computer scientist: Learning with Python

For students with a good knowledge of Python already, I recommend:

Luciano Ramalho: Fluent Python

and

David Beazley and Brian K. Jones: Python Cookbook

Official references are richer and often invaluable:

The Python Tutorial

They also offer the most complete coverage of the language:

The Python Standard Library

Every week, there will be a widget, but to understand all aspects of their code, some resources are necessary. The official reference:

Tkinter 8.5 reference: a GUI for Python

does the job perfectly.

11 Course evaluation and development

Student feedback on this course, and on the lecturing in this course, will be obtained via electronic survey at the end of session. Student feedback is taken seriously, and continual improvements are made to the course based in part on this feedback. Here is the answer to the feedback that was received at the end of last session.

> It is always difficult to cater for the needs of a variety of students, some being expert programmers already, others being total beginners. This session will try and bring some simplifications, but it is a fine balancing act. The distinction between what are the basic expectations and what is more advanced, optional material will be made clearer. More

time might be spent on more basic concepts and techniques. There will be a broader range of application programs, with a number of more practical programs, though here too we have to be careful as problems are meant to support some fundamental techniques and constructs, and practical problems cannot always play this role. Last session assignments 2 and 3 were not independent, and assignment 3 was probably too hard; this will be kept in mind for the assignments this session.

Students are very strongly encouraged to let the lecturer in charge know of any problems, as soon as they emerge. Suggestions (or even complaints!) will be listened to very openly, positively, constructively and thankfully, and every action will be taken to fix any issue or improve the students' learning experience.

12 Other matters

Lectures will take place each week of session, in Webster Theatre B (F Hall B) (K-G15-290), on Thursdays from 6pm to 9pm, from week 1 to week 12. Attendance to lectures is highly recommended, but lectures are recorded and the link to the recordings will be provided after the first lecture has taken place. The material that will be covered during a given lecture will be posted on the web as sample programs and occasionally some lecture notes (pdf format), at the latest by noon on the day when the lecture takes place. Support for the syntactic aspects of the language is provided in the form of Jupyter notebook sheets, all made available in week 0.

You should have enrolled in one of six labs:

- Monday 16:00 - 17:30, Strings Lab J17 302 (K-J17-302)
- Monday 17:30 - 19:00, Strings Lab J17 302 (K-J17-302)
- Monday 19:00 - 20:30, Strings Lab J17 302 (K-J17-302)
- Tuesday 12:30 - 14:00, Strings Lab J17 302 (K-J17-302)
- Tuesday 16:00 - 17:30, Strings Lab J17 302 (K-J17-302)
- Tuesday 17:30 - 19:00, Strings Lab J17 302 (K-J17-302)

All labs will run from week 2 to week 5 included. The Monday 19:00 - 20:30 lab will run till week 12 included. Labs are meant to give you a good start, resolve potential technical issues, and help you acquire the practical skills. All labs run for 3 weeks, except for the Monday 19:00 - 20:30 lab which runs for the whole session; irrespectively of which lab you are enrolled in to start with, you are welcome to attend the Monday 19:00 - 20:30 lab from week 5 onwards if you need support or feedback.

Attendance to labs is highly recommended, but not compulsory. Lab questions for a given week will be posted on the web at the latest by the end of the previous week, and solutions will be posted at the end of the week.

Practical work can be conducted either on the School's lab computers or on your own computer. If your computer is a Windows machine then you might consider installing Linux. Information on doing so is available at

`http://taggi.cse.unsw.edu.au/FAQ/Running_your_computer/`

The labs will help you get used to Unix and the setup of the computing laboratory. A good starting point to learn more about the computing environment and available resources is

`http://taggi.cse.unsw.edu.au/FAQ/`

You should have read carefully the yellow form.

You might also find the following web sites useful.

- CSE Help Desk: `http://www.cse.unsw.edu.au/~helpdesk`
- UNSW library: `http://info.library.unsw.edu.au`
- UNSW Learning center: `http://www.lc.unsw.edu.au`
- Ergonomics: `https://www.ohs.unsw.edu.au/hs_hazards/ergonomics.html`
- Occupational Health and Safety policies: `http://www.cse.unsw.edu.au/ohs/`
- Equity and Diversity issues: `http://www.studentequity.unsw.edu.au/`