School of Computer Science and Engineering

COMP9021

PRINCIPLES OF PROGRAMMING

Session 1, 2017

Contents

1 Course staff

Lecturer in charge: Eric Martin

Office: building K17, room 409

Email: emartin@cse.unsw.edu.au

Phone: 9385 6936

Tutor for consultation: Manas Patra (m.patra@unsw.edu.au)

The lecturer in charge will be answering e-mails for personal matters that are not of relevance to other students, and provided that they do not require extensive or substantive answers. Questions that cannot be answered shortly should be raised in consultation. All questions that are of interest to the class should be asked using the Discussion utility of the Ed platform, used to manage the course. Students are encouraged to also answer any question and more generally, actively participate in any discussion which they can usefully contribute to.

Starting from week 2, the tutor will be available twice a week for 2 hours, on Tuesday from 4:00 to 6:00 and on Wednesday from 3:00 to 5:00, in the Horn lab, building J17, room 305. Being held in a practical environment, these consultations are meant to provide personal support and resolve issues that cannot be addressed, or not easily so, through online discussion.

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 many 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. Both are part of the general procedure for advanced standing, exemption, and substitution.

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, consultations, online discussions, textbook and recommended reading, quizzes, lab exercises 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 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, and are designed to provide insight and help acquire good learning strategies. Consultations are for individual contact, to help resolve more individual issues. Online discussions are for exchanges being part of a community, where everyone seeks support and provides support to others on any matter than is of interest to other students. From week 2 to week 11 included, programming quizzes will be released after the Tuesday lecture and your answers should be submitted by midnight on Monday of the following week. This will help you master the fundamental notions and techniques that will have been presented during lectures up to the previous week, keep up to date with the current material, and give you confidence that you are well on track. Students enrolled in the Friday class should start working on the quizzes as soon as they are released and so will get a chance to ask questions a few days later. 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. The same remark as for the quizzes applies to students enrolled in the Friday class.

7 Assessment

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

Assesment item	**Maximum mark**
10 weekly programming quizzes	20
Assignment 1	10
Assignment 2	10
Assignment 3	10
Midterm exam (2 hours)	20
Final exam (3 hours)	30

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 Tuesday lecture. Typically, you will have to complete incomplete programs, allowing you to check your understanding of the fundamental notions that will be presented during lectures up to the previous week (so as to advantage rather than disadvantage students enrolled in the Friday class). Your answers to the weekly quizzes should be submitted by midnight on Monday of the following week. Every quiz will be worth up to 2 marks.

Longer, unassessed programming exercises, so-called lab exercices, will be released from week 1 to week 12 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 exercises in this series will have one or more flagged questions, some of which will be exam questions modulo some small variations. For most exercises, up to two scaffolding stubs will be provided. Students should try and tackle the exercises from scratch, then from the less detailed stub if stuck, then from the more detailed stub if still stuck.

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.

Quizzes as well as assignments 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

UNSW has an ongoing commitment to fostering a culture of learning informed by academic integrity. All UNSW staff and students have a responsibility to adhere to this principle of academic integrity. Plagiarism undermines academic integrity and is not tolerated at UNSW. Plagiarism at UNSW is defined as using the words or ideas of others and passing them off as your own.

If you haven't done so yet, please take the time to read the full text of

UNSW's policy regarding academic honesty and plagiarism

The pages below describe the policies and procedures in more detail:

- Student Code Policy
- Plagiarism Policy Statement
- Plagiarism Procedure
- Student Misconduct Procedure

9 Course schedule

The following table outlines a provisional schedule for this course. In the **Date** column, **L** refers to the lectures, **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 Feb/3 Mar	Introduction to writing and executing python code Introduction to processing data, operators, lists, dictionaries	
2	**L**: 7 Mar/10 Mar	Control structures Random input Documentation and testing	
3	**Q**: 13 Mar **L**: 14 Mar/17 Mar	Lists, tuples Dictionnaries, sets Debugging	Quiz 1
4	**Q**: 20 Mar **L**: 21 Mar/24 Mar	Functions Identity versus equality, references	Quiz 2
5	**Q**: 27 Mar **L**: 28 Mar/31 Mar **A**: 2 Apr	Generators Files Regular expressions	Quiz 3 Assignment 1
6	**Q**: 3 Apr **L**: 4 Apr/7 Apr **E**: 7 Apr	Objects Inheritance Polymorphism	Quiz 4 Midterm exam
7	**Q**: 10 Apr **L**: 11 Apr	Recursion Comparisons between recursive and iterative procedures	Quiz 5
		Mid-session recess	
8	**Q**: 24 Apr **L**: 28 Apr	Linked lists Decorators	Quiz 6
9	**Q**: 1 May **L**: 2 May/5 May **A**: 7 May	Stacks Queues	Quiz 7 Assignment 2
10	**Q**: 8 May **L**: 9 May/12 May	Heaps Hashing	Quiz 8
11	**Q**: 15 May **L**: 16 May/19 May	Trees Binary search trees	Quiz 9
12	**Q**: 22 May **L**: 23 May/26 May	Sorting	Quiz 10
13	**A**: 4 Jun		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

The platform will switch from WebCMS to Ed when session starts.

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 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. Feedback from last session was very good, but this session will see important changes many of which are prompted by a large increase in student numbers, necessitating to adopt different teaching methods and ways to manage the class. One important change will be the use of the Ed platform, especially designed for computing courses, which should provide an improved experience in testing and submitting one's code, and running some of the code provided as part of the material. Another important change is that the lecture will be run twice a week. Students enrolled in the Friday class should be proactive and are strongly encouraged to watch the recording of the Tuesday class. Having gained some understanding through this recording, the Friday lecture will give them a chance to reinforce more than discover the material, ask questions and seek clarifications in a lecture theatre that is less "intimidating" than Matthews A, hence more suitable to asking questions of all kinds. Students enrolled in the Friday class should also make sure that they start working on the quizzes and assignments after they have been released so they can ask questions and seek support during the following lecture. Another change is providing scaffoldings for many lab exercises in the form of more or less detailed templates, which should guide those of the students who feel overwhelmed by some of the programming tasks.

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, from week 1 to week 12, in Mathews Theatre A (K-D23-201) on Tuesdays from 6pm to 9pm, and in Old Main Building 230 (K-K15-230) on Fridays from 11am to 2pm. Students are enrolled in either class and can attend the lecture for that class. Still lectures for both classes will be recorded any both recordings will be available to all students. The intention was to have the Friday class for students with no or limited programming experience, and the Tuesday class for the students with significant programming experience, reflected in the choice of program examples and the way they would be discussed. The lectures might not be so different, as it is not clear whether enrolment in either class is consistent with what was intended. Still, beginners will definitely benefit from attending a lecture that takes place in a much smaller lecture theatre, with fewer fellow students, and in particular feel more comfortable asking questions. Though lectures are recorded, attendance to lectures is highly recommended. 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 1.

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 page on Student Conduct.

You might also find the following web sites useful.

- CSE Help Desk:
  https://www.engineering.unsw.edu.au/computer-science-engineering/about-us/organisational-structure/computer-support-group/help-desk
- UNSW library: https://www.library.unsw.edu.au
- UNSW Learning center: http://www.lc.unsw.edu.au
- Ergonomics: https://safety.unsw.edu.au/manual-tasks
- Occupational Health and Safety policies:
  https://www.engineering.unsw.edu.au/computer-science-engineering/help-resources/health-safety/
- Equity and Diversity issues: https://student.unsw.edu.au/disability/