School of Computer Science and Engineering

COMP9021

PRINCIPLES OF PROGRAMMING

Session 2, 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: Matthew Perry (z5075269@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 utility of the platform used to manage the course (MessageBoard or Discussion). 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 6:30 to 8:30 in the Clavier lab (K14 LG20);
- on Friday from 1:00 to 3:00 in the Clavier lab (K14 LG20).

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. Other slots will be made available if necessary.

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

The 3 hour lectures, held on Thursdays, introduce the material, and are designed to provide insight and help acquire good learning strategies. Extra 1 hour lectures, held on Mondays, are meant to further help beginners with the more basic aspects programming and Python, but everyone is welcome to attend. 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 Thursday 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 up to the previous week, keep up to date with the current material, and give you confidence that you are well on track. 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. There will be two assignments, due at the end of week 6 and week 12, respectively.

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
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 Thursday 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 current week. Your answers to the weekly quizzes should be submitted by midnight on Wednesday of the following week. Every quiz will be worth up to 2 marks.

Longer 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. Lab exercises are not assessed. Solutions to lab exercises are released about one week after they have been made available.

The two 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. 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**: 24 Jul ; 27 Jul	Introduction to writing and executing Python code Introduction to operators, lists, dictionaries, strings, control structures	
2	**L**: 31 Jul ; 3 Aug	Control structures Functions Randomness and simulation Documentation and testing	
3	**L**: 7 Aug ; 10 Aug **Q**: 9 Aug	Lists, tuples Dictionaries, sets Debugging	Quiz 1
4	**L**: 14 Aug ; 17 Aug **Q**: 16 Aug	Strings, iterators, generators Selected modules Identity versus equality, references	Quiz 2
5	**L**: 21 Aug ; 24 Aug **Q**: 23 Aug	Operations on files Embedded structures Regular expressions	Quiz 3
6	**L**: 28 Aug ; 31 Aug **Q**: 30 Aug **A**: 3 Sep	Recursion, memoisation	Quiz 4 Assignment 1
7	**L**: 4 Sep ; 7 Sep **Q**: 6 Sep **E**: 8 Sep	Classes, objects Dynamic programming	Quiz 5 Midterm exam
8	**L**: 11 Sep ; 14 Sep **Q**: 13 Sep	Linked lists The numpy module	Quiz 6
9	**L**: 18 Sep ; 21 Sep **Q**: 20 Sep	Stacks Queues	Quiz 7
		Mid-session recess	
10	**L**: 5 Oct **Q**: 4 Oct	Trees, tries Inheritance	Quiz 8
11	**L**: 9 Oct ; 12 Oct **Q**: 11 Oct	Hashing Decorators	Quiz 9
12	**L**: 16 Oct ; 19 Oct **Q**: 18 Oct **A**: 22 Oct	Heaps Sorting	Quiz 10 Assignment 2

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 might 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, the main criticism being that the workload was excessive. To address this criticism, the number of assignments is being reduced from three to two. Last session, the lecture was delivered twice, the idea being that the class could be partitioned into beginners versus students with some programming background, so that the pace at which the material would be presented, the teaching style, and the focus of each lecture, would be different and appropriate to each cohort of students (of course, the basic expectations and learning objectives of the course are still the same for all). The experience was moderately successful as many students chose their lecture time based on personal convenience with respect to general timetabling, so both lectures had both beginners and students with some experience, so the very significant extra time needed to implement this strategy was more a cost to the lecturer than an investment for the students. This session, another technique will be tried, still to address the difficulties inherent to the class consisting of students with vastly different backgrounds and needs: an extra 1 hour lecture is being offered which in spirit, is between a lecture and a lab, meant to further help beginners with the more basic aspects programming and Python. Beginners are strongly advised to attend that extra lecture. Still beginners and non-beginners alike are welcome to attend.

Also, students are 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 Mondays from 3pm to 4pm, and in Keith Burrows Theatre (K-J14-G5) on Thursdays from 6pm to 9pm. Lectures for both classes will be recorded any both recordings will be available to all students. 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, with extra guidance provided during the Monday lectures.

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
- 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/