School of Computer Science and Engineering

COMP9021

PRINCIPLES OF PROGRAMMING

Session 2, 2018

Contents

1 Course staff

Lecturer in charge: Eric Martin

Office: building K17, room 409
Email: eric.martin@unsw.edu.au
Phone: 9385 6936

Help with consultation:

- Jiale Nie (z5148607@unsw.edu.au)
- Matthew Perry (z5075269@unsw.edu.au)
- Kongzhang Hao (z5028465@unsw.edu.au)
- Lianfeng (Richard) Luo (z5170521@unsw.edu.au)
- Yingcong (Julian) Zhu (z5141364@unsw.edu.au)
- Sarah Kazemi (z5141356@unsw.edu.au)

The lecturer in charge and course administrator 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 on Ed's forum. Students are encouraged to also answer any question and more generally, actively participate in any discussion which they can helpfully contribute to.

Consultation will be available from week 2 to week 12 included at the following locations and times:

- Monday from 6pm to 8pm in the bugle/horn labs (building J17, room 305, Sarah and Kongzhang)
- Tuesday from 4pm to 6pm in the viola/cello labs (building J17, room 302, Jiale)
- Wednesday from 6pm to 7pm in the bugle/horn labs (building J17, room 305, Richard and Julian)
- Wednesday from 1pm to 3pm in the 204 consultation room (building K17, Matthew)
- Thursday from 10am to 12pm in the 402 consultation room (building K17, Jiale)

- Friday from 12pm to 1pm in the 402 consultation room (building K17, Julian)
- Friday from 12pm to 1pm in the 403 consultation room (building K17, Richard)

Being held in a practical environment, these consultations are meant to provide personal support, resolve issues that cannot be addressed, or not easily so, through online discussion, and get feedback on own work (quizzes, assignments) if desired.

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 gain insights on what happens behind the scene especially in terms of memory use.
- 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.
- Be proficient in the use of appropriate tools, in particular for editing, testing and debugging.
- 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.
- Know how to use and be able to implement hash functions.
- Gain the opportunity to study the design and implementation of a variety of widgets.

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 Tuesdays, use problem solving to introduce the material; they are designed to help acquire good learning strategies and provide valuable insight. Extra 1 hour lectures, held on Thursdays, are meant to possibly present that part of the material that could not be covered during the Tuesday lecture, to further help students with the more mundane, syntactic aspects programming and Python and effective use of operating system and programming tools, and to

provide tips to quizzes and assignments if needed. Consultations are for individual contact, to help resolve more individual issues and get personal support for the homework. 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 noon on Tuesday 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 13, respectively.

7 Assessment

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

Assesment item	**Maximum mark**
10 weekly programming quizzes, worth 2.5 marks each, mark computed from 8 best	20
Assignment 1	10
Assignment 2	10
Midterm exam (3 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 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 current week. Your answers to the weekly quizzes should be submitted by noon on Tuesday of the following week. Every quiz will be worth up to 2.5 marks; the 8 best quizzes only will contribute towards the total mark for that assessment item.

Longer programming exercises, so-called lab exercises, 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. 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 **??**.

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 $\min(6, 8) = 6$.

For the midterm exam, which will take place in computer labs, you will have to write short programs.

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, always a Sunday), **Q** to the due date of a quiz (noon of that day, always a Tuesday), and **E** to an exam. Lecture contents is described very roughly, and subjected to adjustments.

Week	Date	Lecture contents	Assessment
1	**L**: 24 Jul; 26 Jul	Introduction to operators, lists, tuples, dictionaries, control structures, reading from files, printing, functions	
2	**L**: 31 Jul; 2 Aug	Functions from the random module, exceptions, strings, sets, modulo operations, Unicode, formatting More on lists, control structures, functions	
3	**L**: 7 Aug; 9 Aug **Q**: 7 Aug	Floating point operations and precision, slices, iterables, timing running time, plotting data More on lists in relation to time and space complexity, control structures, formatting	Quiz 1
4	**L**: 14 Aug; 16 Aug **Q**: 14 Aug	Operations on files and directories, system operations More on tuples, dictionaries, control structures, plotting data	Quiz 2

5	**L**: 21 Aug; 23 Aug **Q**: 21 Aug	Bases, bitwise operations, regular expressions More on sets, strings, formatting	Quiz 3
6	**L**: 28 Aug; 30 Aug **Q**: 28 Aug **A**: 2 Sep	Generator functions, use of modules such as collections, operator, itertools, argparse. Sorting, lambda expressions, 2-dimensional lists, numpy arrrays	Quiz 4 Assignment 1
7	**L**: 4 Sep; 6 Sep **Q**: 4 Sep **E**: 7 Sep	Recursion, memoisation, converting recursive implementations to iterative implementations	Quiz 5 Midterm exam
8	**L**: 11 Sep; 13 Sep **Q**: 11 Sep	Dynamic programming Classes, objects	Quiz 6
9	**L**: 18 Sep; 20 Sep **Q**: 18 Sep	Linked lists, stacks, queues	Quiz 7
		Mid-session recess	
10	**L**: 2 Oct; 4 Oct **Q**: 2 Oct	Trees, tries Inheritance	Quiz 8
11	**L**: 9 Oct; 11 Oct **Q**: 9 Oct	Hashing Decorators	Quiz 9
12	**L**: 16 Oct; 18 Oct **Q**: 16 Oct	Heaps Sorting	Quiz 10
13	**A**: 28 Oct		Assignment 2

10 Resources for students

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

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

This links to the Ed platform.

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. Jupyter notebook sheets will also be provided as complements to many of the sample programs and cover a wealth of material. They will also be duplicated as pdf documents. More will be said about it during the class.

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 good. The assessment of quizzes changes from 2 marks per quiz to 2.5 mark per quiz, with only the best 8 contributing to the total mark. Some of the material has been reorganised. The most important change is the much more extensive use of Jupyter notebook sheets, that play the role of lecture notes. There might also be "live programming exercises" to increase student participation during lecture.

12 Other matters

Lectures will take place each week of session, from week 1 to week 12, in Keith Burrows Theatre (K-J14-G5) on Tuesdays from 6pm to 9pm, and in Sir John Clancy Auditorium (K-C24-G17) on Thurdays from 1pm to 2pm. 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, jupyter notebook sheets, and pdf files at the latest by noon on the day when the lecture takes place.

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 Code of 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/