School of Computer Science and Engineering

COMP9021

PRINCIPLES OF PROGRAMMING

Trimester 1, 2019

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:

- Gaganjot Singh (gaganjotsingh@student.unsw.edu.au)
- Matthew Perry (matthew.perry@unsw.edu.au)
- Xin Li (xin.li8@student.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 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 1 to week 11 included at the following locations and times:

- Monday from weeks 2 to 9 included and in week 11, from 6:00pm to 7:30pm, in the kora lab (building J17, rooms 307, Gaganjot and Xin)
- Wednesday from weeks 2 to 11 included, from 2pm to 4pm in the organ lab (building K14, room 19, Gaganjot, Matthew and Xin)
- Friday from weeks 1 to 8 included and in week 10, from 5pm to 6:30pm in the kora lab (building J17, room 307, Matthew and Xin)

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 mastered the material covered in this course in previous academic studies can get exemption; see the rules for credit transfer and exemption for details.

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, including advanced syntax and programming techniques.
- Gain insights on what happens behind the scene when operating on Python data types, with an understanding of efficiency and memory use.
- Have a first acquaintance with 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 plot data in various ways, record animation movies.
- Be exposed to a variety of problems related to more specialised fields and taught in other courses (Turing machines, k-clustering, Prolog, Nash equilibrium, cryptography, fractals...)
- 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, practice 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 sufficient understanding of the fundamental notions as well as decent 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 two 2 hour lectures, held on Wednesdays and Thursdays, use problem solving to present the material; they are designed to help acquire good learning strategies and provide valuable insight. Consultations are for individual contact, to help resolve more individual issues and get personal support for the homework, clarify concepts, get feedback, practice better. Online discussions are for exchanges, for 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 1 to week 9 included, with week 6 excluded, programming quizzes will be released after the Thursday lecture and your answers should be submitted by noon on Thursday of the following week (except for quiz 4, released in week 5 and due in week 7). 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 10, respectively. There will be a final exam that will be preceded by a practice exam. You can think as the practice exam and the final exam as two instances of the exam, as two chances to do well, since the one where you perform best will be the one that counts in the computation of your mark for the course. The exam has a lot in common with job interviews, where you have limited time to demonstrate that you can write a few lines of code that correctly solve a reasonable, rather standard exercise, the main challenge being to manage the stress.

7 Assessment

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

Assesment item	**Maximum mark**
8 weekly programming quizzes, worth 2.5 marks each	20
2 assignments, worth 10 marks each	20
Practice final exam (2 hours) Final exam (2 hours)	60 60 (will take max of both)

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 1 to week 9, with the exclusion of week 6, 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 noon on Thursday of the following week (except for quiz 4, released in week 5 and due in week 7). Every quiz will be worth up to 2.5 marks.

Other programming exercises, so-called practice exercises, will be released from week 1 to week 10 to help you master the key material presented in the previous weeks and help you become a competent programmer. Practice exercises are not assessed. Solutions to practice 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 $\min(6, 8) = 6$.

The final exam will be a 2 hour practical exam, taking place in the labs. Expectations are not high: you will have to demonstrate that you can write short programs that correctly implement the

specifications and pass a number of tests, for a number of rather simple problems. The main difficulty is to control the possible stress of having to produce correct code in limited time in a lab environment. There will be a practice exam, held in week 11, on Thursday, between 6pm and 8pm, that will be run exactly as the final exam (whose date is not known yet and will be made public by Central admin towards the end of session). You should normally perform at least as well in the final exam as in the practice exam, still it could be the other way around. To make sure the circumstances are as advantageous as possible to you, your mark for the exam component of the course will be computed as the maximum of the mark for the practice exam and the mark for the final exam.

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

Plagiarism detection software will be run for all quizzes and assignments, and penalties will be applied to those of the students who will be caught.

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 Thursday), and **E** to the practice exam. Lecture contents is described very roughly, and subjected to adjustments.

Week	Date	Lecture contents	Assessment
1	**L**: 20 Feb; 21 Feb	Introduction to operators, lists, tuples, dictionaries, control structures, reading from files, printing, functions.	
2	**L**: 27 Feb; 28 Feb **Q**: 28 Feb	Functions from the random module, exceptions. Base systems, modulo operations. Strings, Unicode, formatting. Lists and sets, with a view on time complexity, plotting, timing.	Quiz 1
3	**L**: 6 Mar; 7 Mar **Q**: 7 Mar	Slices, lists with a view on space complexity. Operations on files and directories, system operations. Control structures, functions	Quiz 2
4	**L**: 13 Mar; 14 Mar **Q**: 14 Mar	The collections and matplotlib modules. Floating point operations and precision, bitwise operations. Regular expressions.	Quiz 3
5	**L**: 20 Mar; 21 Mar **Q**: 21 Mar	The itertools and numpy module. Iterables, generators, lambda expressions. 2-dimensional lists, numpy arrays and operations.	Quiz 4
6	**A**: 31 Mar		Assignment 1
7	**L**: 3 Apr; 4 Apr **Q**: 4 Apr	The functools module. Recursion, memoisation. From recursive implementations to iterative implementations	Quiz 5
8	**L**: 10 Apr; 11 Apr **Q**: 11 Apr	Classes, objects. Object-oriented programming. Special methods.	Quiz 6
9	**L**: 17 Apr; 18 Apr **Q**: 18 Apr	Dynamic programming. Inheritance. Decorators.	Quiz 7

10	**L**: 24 Apr **Q**: 25 Apr **A**: 28 Apr	Properties. Search techniques.	 Quiz 8 Assignment 2
11	**L**: 30 Apr **E**: 2 May	Mixins. Sorting.	 Practice exam

10 Resources for students

Announcements, jupyter notebook sheets, pdf files, sample programs, introductory videos, practice exercises and solutions, quizzes and assignment specifications are made available at the course's homepage. The link that follows lets you log into the Ed platform:

https://edstem.org/login

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.

Other Jupyter notebook sheets, together with pdf files produced from those, will be provided as "lecture notes". These Jupyter notebook sheets offer many advantages over the more traditional lecture notes: you can edit the cells that make up a Jupyter notebook sheet, you can add or delete cells, you can run the contents of cells that contain code, allowing you to guess what the output will be and check that your guess is correct, letting you play a more active role when you learn from existing code. These Jupyter notebook sheets have been carefully designed to cover an extensive part of the Python language and include, besides all the basics, advanced syntax and programming techniques, more than you will find in most textbooks, all presented in the context of interesting problems, most of which should be relevant to the practical problems you will have to tackle in the workplace or in other courses.

Here are some recommendations for further reading, 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. As we change from a 12 week semester (with mid-semester break) to a 10 week trimester (without semester break, still made possible with 4 hours of lecture per week for the remaining weeks and "created" in week 6), there are significant changes to most aspects of the course in order to fit the trimester model.

12 Other matters

Lectures will take place on Wednesdays and Thursdays from week 1 to week 5 and from week 7 to week 9, on Wednesday of week 10 and on Tuesday of week 11 in Central Lecture Block 7 (K-E19-104). 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 beginning 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/.

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/