

School of Computer Science and Engineering

COMP9021

PRINCIPLES OF PROGRAMMING

Term 2, 2021

Contents

1 Course staff and access to resources 3

2 Course details 4

3 Course aims 4

4 Student learning outcomes 4

5 Overall approach to learning and teaching 5

6 Teaching strategies 5

7 Assessment 6

8 Special consideration 7

9 Academic honesty and plagiarism 8

10 Course schedule 9

11 Additional resources 10

12 Course evaluation and development 13

13 Other matters 13

1 Course staff and access to resources

Lecturer in charge: Eric Martin

Office: building K17, room 409
Email: eric.martin@unsw.edu.au
Phone: (02) 9065 5587

Tutors:

- Matthew Perry (matthew.perry@unsw.edu.au)
- Terry Zhuo (terry.zhuo@unsw.edu.au)

Online lectures will take place in weeks 1–2, 4–5, 7–10 on Mondays and in weeks 1–5, 7–10 on Wednesdays, between 6pm and 8pm.

Online consultation will be available from week 1 to week 10:

- Tuesday, from 4pm to 6pm (Matthew) and from 6pm to 8pm (Terry)
- Friday, from 4pm to 6pm (Matthew) and from 6pm to 8pm (Terry)

Both lectures and consultations will be recorded.

Whether you want to join live lectures or consultations, or watch recordings of previous lectures or consultations, you should log into Moodle. Clicking on COMP9021 will take you to the minimalistic Moodle page for the course, whose only purpose is to provide you with the links to lectures and consultations. Click on *Lectures and Consultations* (that is purposely pretty much all you can do...), then select the relevant category (Monday Lectures, Tuesday Consultations, Wednesday Lectures, Friday Consultations) and then the session of your choice, to take part in it live or watch its recording.

The rest of the action all happens on the Ed platform: access to course material, MessageBoard to ask and answer questions, applications to write, run and test code...

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. As they can be more personalised, 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. 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.

Everything is done online so it makes no difference whether you are in Sydney or overseas. The final exam will also be online.

2 Course details

Units of credit: 6

No parallel teaching: only COMP9021 students attend the classes.

3 Course aims

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, videos, notes, samples programs, recommended reading, practice exercises, quizzes, assignments and final exam 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, study of the more advanced material to deepen your knowledge, completion of assignments, or seeking assistance to clarify your understanding and get personal feedback. You must choose the approach that best suits your learning style and goals in this course. How much time you will devote to this course will vary greatly depending on your learning style and objectives. 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 will discuss part of the notes, some of which come with automatically generated videos. To fully benefit from the lectures, you should beforehand study the available notes and videos. The Wednesday lectures will also discuss quizzes and assignments as they are released. Lectures are designed to help you acquire good learning strategies, provide valuable insight, and improve your problem solving skills. 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 Wednesday lecture and your answers should be submitted by noon on Wednesday of the following week (except for quiz 5, released in week 5 and due in week 7); details on submission will be provided during lectures. 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 by Monday 10am of week 7 and

week 11, respectively. There will be a final exam, run online at a date and of a duration that are still to be determined.

7 Assessment

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

Assesment item	**Maximum mark**
8 weekly programming quizzes, worth 3 marks each	24
2 assignments, worth 13 marks each	26
Final exam (a to be determined number of hours)	50

The final mark will be 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 Wednesday 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 Wednesday of the following week (except for quiz 5, released in week 5 and due in week 7).

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. They are due by Monday 10am of week 7 and week 11, respectively.

Other programming exercises, so-called practice exercises, will be regularly released to help you become very competent programmers. Practice exercises are not assessed. Solutions to practice exercises are released about one week after they have been made available.

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 full late day. Thus if students A and B hand in assignments worth 12 and 9, both two days late, then the maximum mark obtainable is 11, so A gets $\min(12, 11) = 11$ and B gets $\min(9, 11) = 9$.

Expectations for the final exam 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 (not so) limited time. The date of the final exam will be made public by Central admin towards the end of session, and the duration of the exam will be decided then.

8 Special consideration

If your work in this course is affected by unforeseen adverse circumstances, you should apply for Special Consideration. If your request is reasonable and your work has clearly been impacted, then

- for an assignment, you may be granted an extension;
- for the Final Exam, you may be offered a Supplementary Exam

Note the use of the word "may". None of the above is guaranteed. It depends on you making a convincing case that the circumstances have clearly impacted your ability to work.

No Special Consideration for the quizzes is possible, as otherwise it would not be possible to give fast and effective feedback to the class, both in terms of marks and solutions.

UNSW handles special consideration requests centrally (in the Student Lifecycle division), so all special consideration requests must be submitted via the UNSW Special Consideration website.

Special consideration requests must be accompanied by documentation, which will be verified by Student Lifecycle. Do not email the course convenor directly about special consideration.

If you cannot attend the Final Exam because of illness or misadventure, then you must submit a Special Consideration request, with documentation, through MyUNSW within 24 hours of the exam. If your request is reasonable, then you will be awarded a Supplementary Exam

Note that UNSW expects you to be available to sit Supplementary Exams if required. If you are awarded a Supplementary Exam and do not attend, then your exam mark will be zero.

For further details on special consideration, see the UNSW Student website.

If you are registered with Disability Services, please forward your documentation to Course Convenor within the first two weeks of term.

9 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.

10 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 (10am of that day, always a Monday), and **Q** to the due date of a quiz (noon of that day, always a Wednesday). Lecture contents is described very roughly, only very indicative, and subjected to adjustments.

Week	Date	Lecture contents	Assessment
1	**L**: 31 May, 2 Jun	Introduction to operators, strings, lists, tuples, dictionaries, control structures, reading from files, printing, functions.	
2	**L**: 7 Jun, 9 Jun **Q**: 9 Jun	Functions from the random module. Exceptions. Sets. More operations on strings and lists. More on control structures. Arithmetic operators. Positional and keyword arguments, default values, use of star for function parameters and arguments.	Quiz 1
3	**L**: 16 Jun **Q**: 16 Jun	String formatting. Various forms of comprehension. Representations in a given base, conversions between bases. The Unicode character set. Sorting. Lambda expressions. Floating point computations. Iterators. Timing program execution.	Quiz 2
4	**L**: 21 Jun, 23 Jun **Q**: 23 Jun	Walrus operator. Zipping and plotting sequences. Finer use of timeit, insights on the complexity of operations on lists versus sets. Global versus local variables. More operations on sets. Simple versus complex types as default arguments. Bitwise operators.	Quiz 3
5	**L**: 28 Jun, 30 Jun **Q**: 30 Jun	More on generator expressions. Slices. Insights on space allocation for lists. Working with paths in a platform-independent manner. Reading from and writing to csv files. More use of the standard library: defaultdict and Counter from collections, product from itertools... Generator functions.	Quiz 4

6			
7	**L**: 12 Jul, 14 Jul **A**: 12 Jul **Q**: 14 Jul	Named tuples. Operations with infinities. Regular expressions. Lists of lists. Numpy arrays, types, reshaping, broadcasting, arithmetic and Boolean operators. More on plotting, creating animations with matplotlib. Command line arguments.	Assignment 1 Quiz 5
8	**L**: 19 Jul, 21 Jul **Q**: 21 Jul	Inner functions. Recursion. Memoisation, using default arguments or the lru_cache decorator. The ___defaults___ attribute of functions. Itertools module: permutations. Turning recursive designs into iterative designs. Specialised operations on strings.	Quiz 6
9	**L**: 26 Jul, 28 Jul **Q**: 28 Jul	Classes and objects. Special methods, in particular for the implementation of operators. Customised exceptions. More on function parameters. Insights on object creation and initialisation and on OO syntax. Abstract classes. The fraction module.	Quiz 7
10	**L**: 2 Aug, 4 Aug **Q**: 4 Aug	Dynamic programming. Inheritance. Decorators. Properties, getters and setters. Presentation of more optional material and modules for specialised applications: beautifulsoup for web crawling, PIL for image processing, pygame...	Quiz 8
11	**A**: 9 Aug		Assignment 2

11 Additional resources

Announcements, jupyter notebook sheets, pdf files, sample programs, links to automatically produced 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, and the provided material is self-contained. Still, you are encouraged to also spend time reading books or tutorials or watching videos, most of which teach programming in more or less the same way, quite different from the approach I am advocating.

Jupyter notebook sheets, together with pdf files produced from those, will be provided as notes. Some of the notes are complemented with automatically produced videos. 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 very 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:

Graphical User Interfaces with Tk

does the job perfectly.

12 Course evaluation and development

Student feedback on the 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 partly based on this feedback. For last term, some students have reported poor audio quality; this will be addressed using a better microphone. (It could have been addressed during session had I been made aware of this then; this kind of feedback does not have to wait for the course to be surveyed at the end of session to be provided!)

13 Other matters

Practical work can be conducted either on Ed 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.

- UNSW library: https://www.library.unsw.edu.au
- UNSW Learning center: http://www.lc.unsw.edu.au
- Occupational Health and Safety policies: https://www.gs.unsw.edu.au/policy/documents/ohspolicy.pdf
- Equity and Diversity issues: https://student.unsw.edu.au/disability/