Resources / Course Outline

Course Outline

Contents

Course Details

Course Code	COMP9021
Course Title	Principles of Programming
Convenor	Dr Rachid Hamadi (/users/z2286838)
Admin	Dr Rachid Hamadi (/users/z2286838)
Tutors	Matthew Perry (https://webcms3.cse.unsw.edu.au/users/z5075269) Terry Zhuo (https://webcms3.cse.unsw.edu.au/users/z5191493) Ye Lin (https://webcms3.cse.unsw.edu.au/users/z3326191)
Classes	**Lectures** : Monday 18:00 - 20:00 Online Wednesday 18:00 - 20:00 Online Lectures will be conducted **online** through **Blackboard Collaborate Ultra** , which is accessible through Moodle Consultations will be conducted either **face-to-face** or **online** through **Blackboard Collaborate Ultra** , which is accessible through Moodle Both **lectures** and **online consultations** will be **recorded** and accessible through Moodle. Timetable for all classes (/COMP9021/22T3/timetable)
Consultations	See Timetable for all classes (https://webcms3.cse.unsw.edu.au/COMP9021/22T3/timetable)
Units of Credit	6
Course Website	http://cse.unsw.edu.au/~cs9021 (https://webcms3.cse.unsw.edu.au/COMP9021/22T3/)

Handbook Entry	http://www.handbook.unsw.edu.au/postgraduate/courses/current/COMP9021.html (http://www.handbook.unsw.edu.au/postgraduate/courses/current/COMP9021.html)

Course Summary

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.

Assumed Knowledge

Though there is no assumed knowledge, **mathematical maturity** and **familiarity with some form of programming** might reduce the amount of time and efforts that will have to be dedicated to the course, as the **learning curve is steep** and **programming can be very time consuming** , especially for **beginners** .

Student Learning Outcomes

After successfully completing this course, students will be able to:

1. Know how to design, implement and test programs written in a language with procedural, object-oriented, and functional constructs.
2. Be proficient in the Python language, including advanced syntax and programming techniques.
3. Gain insights on what happens behind the scene when operating on Python data types, with an understanding of efficiency and memory use.
4. Have a first acquaintance with fundamental data structures and algorithms.
5. Know how to design programs to solve small to medium scale problems.
6. Be able to write clear, reliable, well-structured, well-tested, and well-documented programs.
7. Be proficient in the use of appropriate tools, in particular for editing, testing and debugging.
8. Know how to plot data in various ways, record animation movies.
9. Be exposed to a variety of problems related to more specialised fields and taught in other courses (such as Turing machines, k-clustering, Prolog, Nash equilibrium, cryptography, and fractals).
10. Gain the opportunity to study the design and implementation of a variety of widgets.

This course contributes to the development of the following graduate capabilities:

Graduate Capability	Acquired in
Scholars capable of independent and collaborative enquiry, rigorous in their analysis, critique and reflection, and able to innovate by applying their knowledge and skills to the solution of novel as well as routine problems	Practice exercises, Quizzes, Assignments
Entrepreneurial leaders capable of initiating and embracing innovation and change, as well as engaging and enabling others to contribute to change	Practice exercises, Quizzes, Assignments
Professionals capable of ethical, self- directed practice and independent lifelong learning	Practice exercises, Quizzes, Assignments

Global citizens who are culturally adept and capable of respecting diversity and acting in a socially just and responsible way	Practice exercises, Quizzes, Assignments

Teaching Strategies

- **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. Lectures will be online, live and recorded with Blackboard Collaborate Ultra through **Moodle** on the days and times lectures have been timetabled. For instructions on how to see the lectures live and access the recordings later, see Blackboard Collaborate Ultra (https://student.unsw.edu.au/blackboard-collaborate-ultra) . You will also be able to access (from COMP9021 Moodle shell) and watch the recorded lectures any time after they become available.
- **Consultations** : Help resolve more individual issues and get personal support for the homework, clarify concepts, get feedback, practice better. There will be **six** 2-hour consultations (see Timetable for all classes (https://webcms3.cse.unsw.edu.au/COMP9021/21T3/timetable)). They will be run by enthusiastic, very helpful, and very competent tutors either **face-to-face** or **online** using Blackboard Collaborate Ultra through **Moodle** . Online consultation sessions will also be recorded. You will also be able to access (from COMP9021 Moodle shell) and watch the recorded online consultation sessions any time after they become available.
- **Online forums discussions** : Are for exchanges, for being part of a community, where everyone seeks support and provides support to others on any matter that is of interest to other students. We will be using **Ed** platform for forums discussions.
- **Coding quizzes** : Help you master the fundamental concepts 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. You will be submitting your coding quizzes in **Ed** platform.
- **Assignments** : 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. You will also be submitting your assignments in **Ed** platform.

Teaching Rationale

At university, as you know, the focus is on your self-directed search for knowledge. Lectures, consultations, online discussions, videos, notes, sample programs, recommended readings, practice exercises, coding 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 coding quizzes and assignments, study for final exam, 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 concepts as well as decent practical skills, thanks to consistent regular work. If your aim is to obtain High Distinction then you will need to invest more time in this course.

Student Conduct

The **Student Code of Conduct** (Information (https://student.unsw.edu.au/conduct) , Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)) sets out what the University expects from students as members of the UNSW community. As well as the learning, teaching and research environment, the University aims to provide an environment that enables students to achieve their full potential and to provide an experience consistent with the University's values and guiding principles. A condition of enrolment is that students *inform themselves* of the University's rules and policies affecting them, and conduct themselves accordingly.

In particular, students have the responsibility to observe standards of equity and respect in dealing with every member of the University community. This applies to all activities on UNSW premises and all external activities related to study and research. This includes behaviour in person as well as behaviour on social media, for example Facebook groups set up for the purpose of discussing UNSW courses or course work. Behaviour that is considered in breach of the Student Code Policy as discriminatory, sexually inappropriate, bullying, harassing, invading another's privacy or causing any person to fear for their personal safety is serious misconduct and can lead to severe penalties, including suspension or exclusion from UNSW.

If you have any concerns, you may raise them with your lecturer, or approach the School Ethics Officer (mailto:ethics-officer@cse.unsw.edu.au) , Grievance Officer (mailto:grievance-officer@cse.unsw.edu.au) , or one of the student representatives.

Plagiarism is defined as (https://student.unsw.edu.au/plagiarism) using the words or ideas of others and presenting them as your own. UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Plagiarism and Academic Integrity (https://student.unsw.edu.au/plagiarism)
- UNSW Plagiarism Procedure (https://www.unsw.edu.au/content/dam/pdfs/governance/policy/2022-01-policies/plagiarismprocedure.pdf)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism. In particular, you are also responsible that your assignment files are not accessible by anyone but you by setting the correct permissions in your CSE directory and code repository, if using. Note also that plagiarism includes paying or asking another person to do a piece of work for you and then submitting it as your own work.

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 (https://student.unsw.edu.au/plagiarism)

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

- Student Code Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)
- Student Misconduct Procedure (https://www.gs.unsw.edu.au/policy/documents/studentmisconductprocedures.pdf)
- Plagiarism Policy Statement (https://www.gs.unsw.edu.au/policy/documents/plagiarismpolicy.pdf)
- Plagiarism Procedure (https://www.unsw.edu.au/content/dam/pdfs/governance/policy/2022-01-policies/plagiarismprocedure.pdf)

You should also read the following page which describes your rights and responsibilities in the CSE context:

- Essential Advice for CSE Students (https://www.engineering.unsw.edu.au/computer-science-engineering/about-us/organisational-structure/student-services/policies/essential-advice-for-cse-students)

Assessment

Item	Topics	Due	Marks
Six Coding Quizzes	Various and worth 3.5 marks each	Friday Weeks 3, 4, 5, 7, 8, and 9 @ 9pm	21%
Assignment 1	Procedural programming	Monday Week 7 @ 10am	14%
Assignment 2	Object oriented programming	Monday Week 11 @ 10am	15%
Final exam	Everything covered in the course	Exam period	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, at least, 50.

Course Schedule

The following table outlines a **provisional** schedule for this course. The contents of the lectures are described **roughly** and are subject to **adjustments.**

Week	Lectures	Assignments	Quizzes
1	Introduction to operators, strings, lists, tuples, dictionaries, control structures, reading from files, printing, functions.		
2	Functions from the random module. Exceptions. Base systems, modulo operations. Unicode character set. Sorting, lambda expressions.		
3	Approximation in computations. String formatting. Lists and sets, with a view on time complexity, plotting, timing. Slices, lists with a view on space complexity.	Assignment 1 released	Quiz 1 due Friday 9pm
4	Operations on files and directories, system operations. Default arguments. Bitwise operations. The collections and matplotlib modules.		Quiz 2 due Friday 9pm
5	Special modules. Generator functions. 2-dimensional lists, numpy arrays and operations. Regular expressions.		Quiz 3 due Friday 9pm
6			

7	More special modules. Recursion. Memoisation. From recursive implementations to iterative implementations	Assignment 1 due Monday 10am Assignment 2 released	Quiz 4 due Friday 9pm
8	Classes, objects. Object-oriented programming. Special methods.		Quiz 5 due Friday 9pm
9	Dynamic programming. Inheritance. Decorators.		Quiz 6 due Friday 9pm
10	Searching. Sorting.		
11		Assignment 2 due Monday 10am	

Resources for Students

Announcements, Jupyter notebook sheets, pdf/html files, sample programs, links to automatically produced videos, practice exercises and solutions, coding quizzes and assignment specifications are made available in Ed platform.

The following link lets you log in to the Ed platform: https://edstem.org/login (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, tutorials, or watching videos, most of which teach programming in more or less the same way, quite different from the approach we are advocating in this course.

Jupyter notebook sheets, together with pdf/html 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 certainly come across other resources, and you are encouraged to share your great findings with everyone.

For easy introductions to Python, the following texts are recommended:

- John Zelle: Python Programming: An Introduction to Computer Science (https://fbeedle.com/our-books/23-python-programming-an-introduction-to-computer-science-3rd-ed-9781590282755.html)

- Brad Miller and David Ranum: Problem Solving with Algorithms and Data Structures Using Python (https://runestone.academy/runestone/static/pythonds/index.html)
- Allen B. Downey: How to think like a computer scientist: Learning with Python (https://runestone.academy/runestone/static/thinkcspy/index.html)

For students with a good knowledge of Python already, the following texts are recommended:

- Luciano Ramalho: Fluent Python (http://shop.oreilly.com/product/0636920032519.do)
- David Beazley and Brian K. Jones: Python Cookbook (http://shop.oreilly.com/product/0636920027072.do)

Official references are richer and often invaluable:

- The Python Tutorial (https://docs.python.org/3.9/tutorial/)

They also offer the most complete coverage of the language:

- The Python Standard Library (https://docs.python.org/3.9/library/index.html)

Every week, there will be a widget, but to understand all aspects of their code, some resources are necessary. The following official reference does the job perfectly:

- Graphical User Interfaces with Tk (https://docs.python.org/3/library/tk.html)

Course Evaluation and Development

This course is evaluated each term using the myExperience survey system at the end of the term.

In the previous offerings of this course, some students suggested to give more weight to assignments. This advice will be followed. As a consequence, there will be a smaller weight on the final exam.

Your feedback is important and will be considered to improve future offerings of this course. Students are also encouraged to provide informal feedback during the term, and let the lecturer and tutors know of any problems as soon as they arise. Suggestions will be listened to very openly, positively, constructively and thankfully, and every reasonable effort will be made to address them as soon as possible.

Resource created 3 months ago (Sunday 04 September 2022, 06:51:38 PM), last modified 2 months ago (Friday 30 September 2022, 06:04:57 PM).