ENGG1811: Computing For Engineers

Introduction

Week 1: Friday 6th June, 2025

Friday 16:00 - 18:00 | OboeME304

Today

General Course Information & Expectations

Arithmetic Operations in Spyder & Python

Lab Tips

General Course Information & Expectations

Introductions

Brendan Mabbutt

b.mabbutt@student.unsw.edu.au

Degree: Advanced Mathematics
Hobbies: Running, F1
Facts: Switched from engineering

Adrian

Degree: Renewable Energy Engineering
Hobbies: Basketball, Physics
Facts: Ab-Soul Appreciator

Now your turn: Introductions

Some icebreakers I'll ask, try to answer some of them for someone next to you:

- How many hours of sleep did you get last night?
- What is your degree?
- Any interesting facts about yourself?

Be sure to get to know the people around you during the lab — it's an important part of learning in this course!

Emails

- Has everyone received the welcome email?
- Ensure that your UNSW student email (Outlook) is activated.
  - If you haven't set it up yet, visit: `https://www.unsw.edu.au/myit/services/account-access-email/email`
- Please Please Please send me an email if you have any questions once or ever related to the labs. If you have more general issues with this course such as for the self-directed labs or setup, contact the course account at en1811@cse.unsw.edu.au.
- If you feel there is a situation out of your control affecting your performance in this course, consider applying for special consideration through UNSW: `https://www.student.unsw.edu.au/special-consideration`. Feel free to loop us in about this. The same goes for ELPs.

What is ENGG1811?

- This course serves as an introduction to programming with engineers from the non-electrical disciplines in mind
  - We will often motivate problems and ideas by leaning on physical examples and 'real'-world systems
  - You will NOT come-away from this course knowing how to build your own website, or exactly what is going on inside your computer when you program
  - You will come-away from this course knowing, instead, how to automate calculations and to create meaningful data that is crucial for an engineer

What will be doing in ENGG1811?

- The plan is to use Python as our programming language of choice in order to learn common programming and computation techniques
  - Our IDE will be Spyder
  - To use spyder at home, use the Anaconda package manager — see the setup guide on the course website.
- We will also provide a brief introduction to Excel & Matlab

Assessments

Item	Weighting	Due Date
Labs	20%	Each week in class
Assignment 1	20%	Week 7
Assignment 2	20%	Week 10
Final Exam	40%	Exam Period

The final exam will also have a **40 % hurdle**, meaning you will need to get at least 40% on the final exam and at least 50% overall to pass the course.

Important Resources

- ENGG1811 course website
  - Contains all the information/resources you need throughout the term: lab exercises, lecture slides, assignments, sample exams, timetables etc
  - Help sessions/PASS sessions and their timetabling will also be available here
- Discourse forum
  - Moderated forum to ask course specific questions
  - You will most likely get your question answered by a tutor, but students often help — and we encourage you all to do so
- Live-coding session
  - One of the tutors will go through optional problems once a week throughout the term. This will be Friday 1-2PM. Official details will be released on the course website.

Labs

- Our labs will be held each Friday from 16:00 - 18:00
- Labs are marked from weeks 2-5 and weeks 7-10
  - 2 marks are allocated to finishing **AND** explaining your lab solutions
  - 1 mark is allocated to answering a MCQ which is available *only* during your lab-time
  - It is strongly encouraged that each week's worth of lab questions be attempted in the week they were released: lab exercises are released Monday of the previous week.
  - You are free to leave early once you get your work marked off!
  - You may use either your own computer or a lab computer for the labs. You can also access the lab computer environment at home — see the setup guide on the course website.
- There are two virtual labs on Excel & Matlab that are released sometime in weeks 3/4 and weeks 7/8 — you usually have about 2-3 weeks to complete them

Labs (Cont)

- Only under legitimate excuses will we let your lab-work be marked in a later week
  - You must either have made a valid attempt in the lab but ran out of time or, if you couldn't make a lab, sent an email giving a valid reason! In both cases, you must have your work ready by next week. Limit of 1 time in each case otherwise you will be marked only if time permits at the end.
  - You can get work marked in another lab if you can't make it
- You will be tested and vetted on the work you give us
  - Be under the expectation that you will need to explain your solutions
  - If we find something wrong, you are either expected to address it immediately, or for you to go back on your own and fix it
  - We may decide to alter the detail of a question, or ask you to reproduce a technique, to ensure that you know exactly what you have done
  - It is painfully obvious to us tutors when you do not understand your work or have plagiarised

Demo: The course website, Moodle and Discourse

- We will go through how to access the course website via Moodle and explore the important pages on the course website, such as:
  - Labs
  - Lectures
  - Grades
  - Timetable
  - Getting Started
  - Help Sessions
- We will look at how to post on the Discourse forum. We will go through:
  - Ensuring that if you post your `lab` code or `assessment` code to get help from a tutor on the forum, you make it a `private`. Preferably, try to ask questions in a general way so that sending code isn't required. Remember, there will also be help sessions where tutors can look at your code.
  - How to make the post anonymous if you would like
  - Choosing the correct category

Tips for First Time Programmers

- Programming is a skill that builds up incrementally ! It is very similar in this regard to learning mathematics, a language, or even an instrument
  - All the content builds upon itself, so please pay close attention to each week's material, as it will serve as a foundation for future weeks.
- Just like any skill , you are guaranteed to have a period of growing pain and so you need to be patient with yourself during those moments
- Do not be afraid to experiment
  - I promise nothing will break if you decide to try something out or if you wonder what happens if we do x thing or y thing instead
  - Experimentation is the best way to figure out why something works the way it does

Tips for First Time Programmers (Cont.)

- Give yourself much leeway when debugging
  - Most of your time in fact will be spent figuring out why something didn't work rather than what you should do
  - We will go over common debugging strategies and tips next week
- Ask for help when needed, but enjoy the process of struggling and figuring things out yourself

Working Together & AI

- You are allowed to work with others on labs; however, you are not permitted to directly copy code. Remember, you must be able to explain your code for the lab.
- You are not allowed to work with others on assignments, as they are designed to be completed individually.
- Using AI for assignments or labs is strictly prohibited. Using AI for debugging or other programming tasks is also strongly discouraged.
  - You risk receiving an automatic zero.
  - Assignments are checked by AI detection tools.
  - You will not have access to AI during the final exam (remember there is **HURDLE**), so relying on AI may negatively affect your long-term progress.

Arithmetic Operations in Spyder & Python

Demo: Spyder IDE

- Here, we will go through how to access the Spyder IDE on the lab computers. We will cover:
  - Creating a directory for ENGG1811 and a subdirectory for each lab using the file manager
  - The Spyder code editor and the run button, which we won't be using today
  - The Spyder console, where you can perform single-line calculations and view the output of any programs
  - The help panel, where we can access current variable values. We will go through some tricks here later such as using the up/down keys to get previous commands and using `clear` to remove the clutter of the consol without deleting the data.
- You can use Ctrl + to zoom in and Ctrl - to zoom out in Spyder. This will be useful in demonstrating your code — I have bad eyes :(

Accustoming Ourselves with Spyder

- At this stage, we will be thinking of Spyder as a glorified calculator
- We can go to the console at the bottom right of our IDE and ask it to compute expressions such as `2 + 3`, `15*3 - 2/4`, `2*(3+5)`, `(1 / (2 / (3 / (4))))`, etc
- For the entirety of the lab today, you can do all the exercises in the console

(Probably) New Operators

Three 'new' operators you all might not have seen before:

- `**` — the double asterisk operator is what we use for exponentiation
  - `2 ** 2` gives `4` because $2^2 = 4$
  - `3 ** 4` gives `81` because $3^4 = 81$
- `%` — the percentage sign is what we use to calculate the remainder of a division problem
  - `4 % 3` gives `1` because the remainder of $4 \div 3$ is 1
  - `400 % 365` gives `35` because the remainder of $400 \div 365$ is 35
- `//` — the double slash operator returns the integer portion of a division problem
  - `10 // 3` gives `3` because $10 \div 3 = 3\frac{1}{3}$ and the whole/integer part is 3
  - `400 // 365` gives `1` because $400 \div 365 = 1\frac{35}{365}$ and the whole/integer part is 1

Optional Exercises

- We are usually familiar with `^` to mean exponentiation — what is `^` reserved for in Python instead?
- Play around with `**`, `%`, `//` using negative and decimal numbers to see if they behave in ways that meet your expectations

Precedence

Operator
`()`
`**`
Unary `+, -`
`*, /, %, //`
Binary `+, -`

- Order of operators is bottom-down. For example, we will do `()` before we do `*`.
- **Question:** What is unary `-` as opposed to binary `-`?
- We must use `()` (parentheses) for brackets in Python. The `{}` and `[]` brackets have different meanings in Python syntax.

Lab Tips

Lab Tips

- Exercise 1
  - These must be done with as few parentheses as possible
- Exercise 2
  - You need to work out how many years and *left over* days there are in $1,000,000$ minutes
    - For example, if there are 400 days total the answer will be 1 year and 35 days
    - You can print out the number of whole years on one line, and the number of left-over days on another line
    - If you finish early, please also try finding the number of leftover minutes and seconds
  - The answer is given at the bottom, so please check this before you get "marked off"

Feedback

Feel free to provide anonymous feedback about the lab!

Feedback Form