# Programming for Scientists

COMP1730 & COMP6730

Australian
National
University

# Introductions

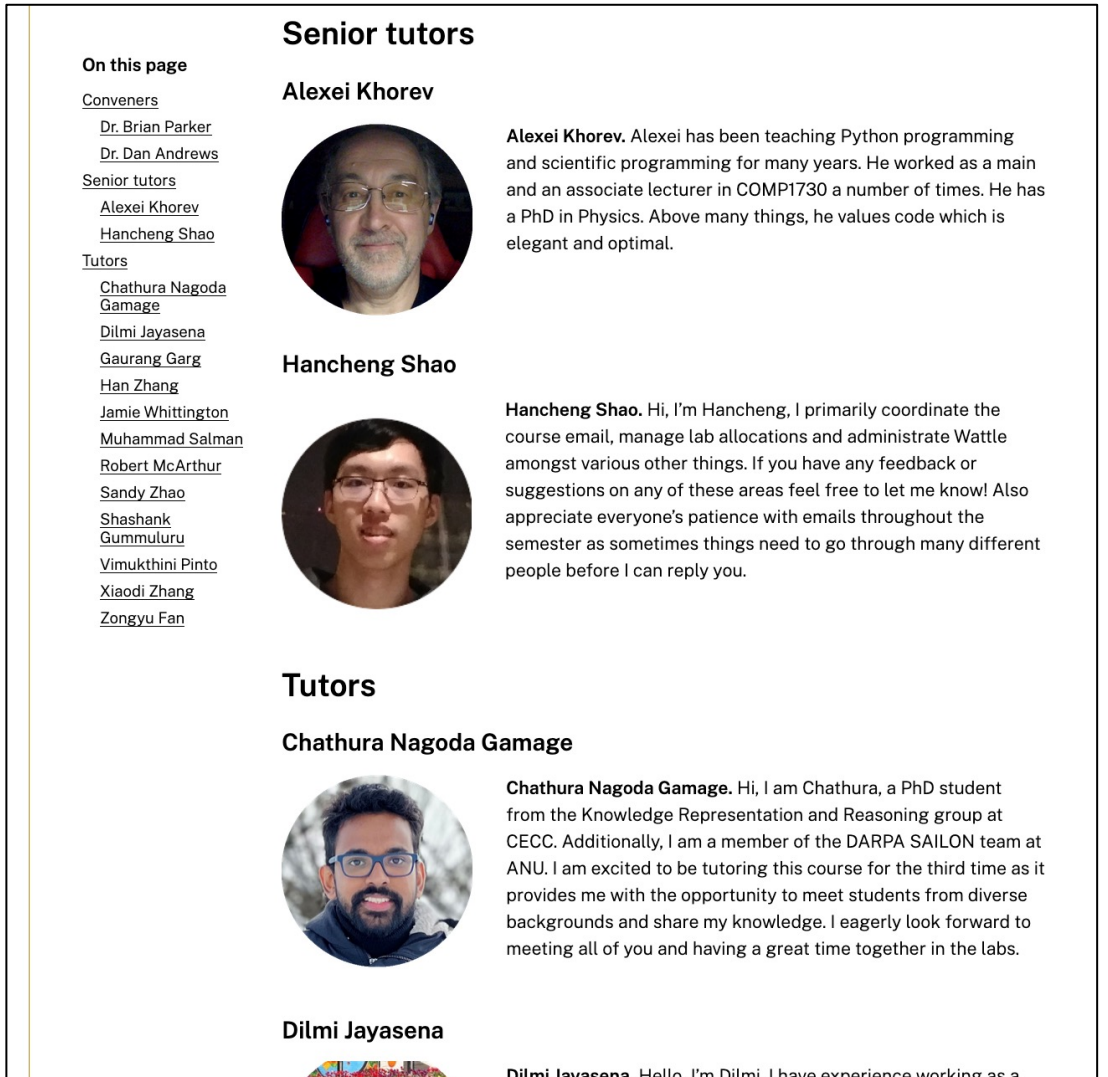
Australian National University

## Co-convenors Semester 1, 2024:

Dan Andrews

Brian Parker

https://comp.anu.edu.au/courses/comp1730/people/



On this page



### Senior tutors

#### Alexei Khorev

**Alexei Khorev.** Alexei has been teaching Python programming and scientific programming for many years. He worked as a main and an associate lecturer in COMP1730 a number of times. He has a PhD in Physics. Above many things, he values code which is elegant and optimal.

#### Hancheng Shao

**Hancheng Shao.** Hi, I'm Hancheng, I primarily coordinate the course email, manage lab allocations and administrate Wattle amongst various other things. If you have any feedback or suggestions on any of these areas feel free to let me know! Also appreciate everyone's patience with emails throughout the semester as sometimes things need to go through many different people before I can reply you.

### Tutors

#### Chathura Nagoda Gamage

**Chathura Nagoda Gamage.** Hi, I am Chathura, a PhD student from the Knowledge Representation and Reasoning group at CECC. Additionally, I am a member of the DARPA SAILON team at ANU. I am excited to be tutoring this course for the third time as it provides me with the opportunity to meet students from diverse backgrounds and share my knowledge. I eagerly look forward to meeting all of you and having a great time together in the labs.

#### Dilmi Jayasena

**Dilmi Jayasena.** Hello, I'm Dilmi. I have experience working as a

# Course structure

Australian National University

- Week 1 – Programming Basics, Variables and Expressions
- Week 2 – Functions and abstraction
- Week 3 – Code branching and Iteration, Strings
- Week 4 – Lists (Canberra Day holiday on the Monday)
- Week 5 – References, Dictionaries, Code best practices
- Week 6 – Modules, Classes, File IO

Dan

Teaching break

- Week 7 – Introduction to scientific libraries with NumPy, Debugging
- Week 8 – Data analysis with Pandas, Visualisation, Dictionaries, Sets
- Week 9 – Advanced functions, Errors and exceptions
- Week 10 – Computational complexity, Dynamic programming
- Week 11 – Computational methods in science and engineering
- Week 12 – Computational methods (cont), Exam revision

Brian

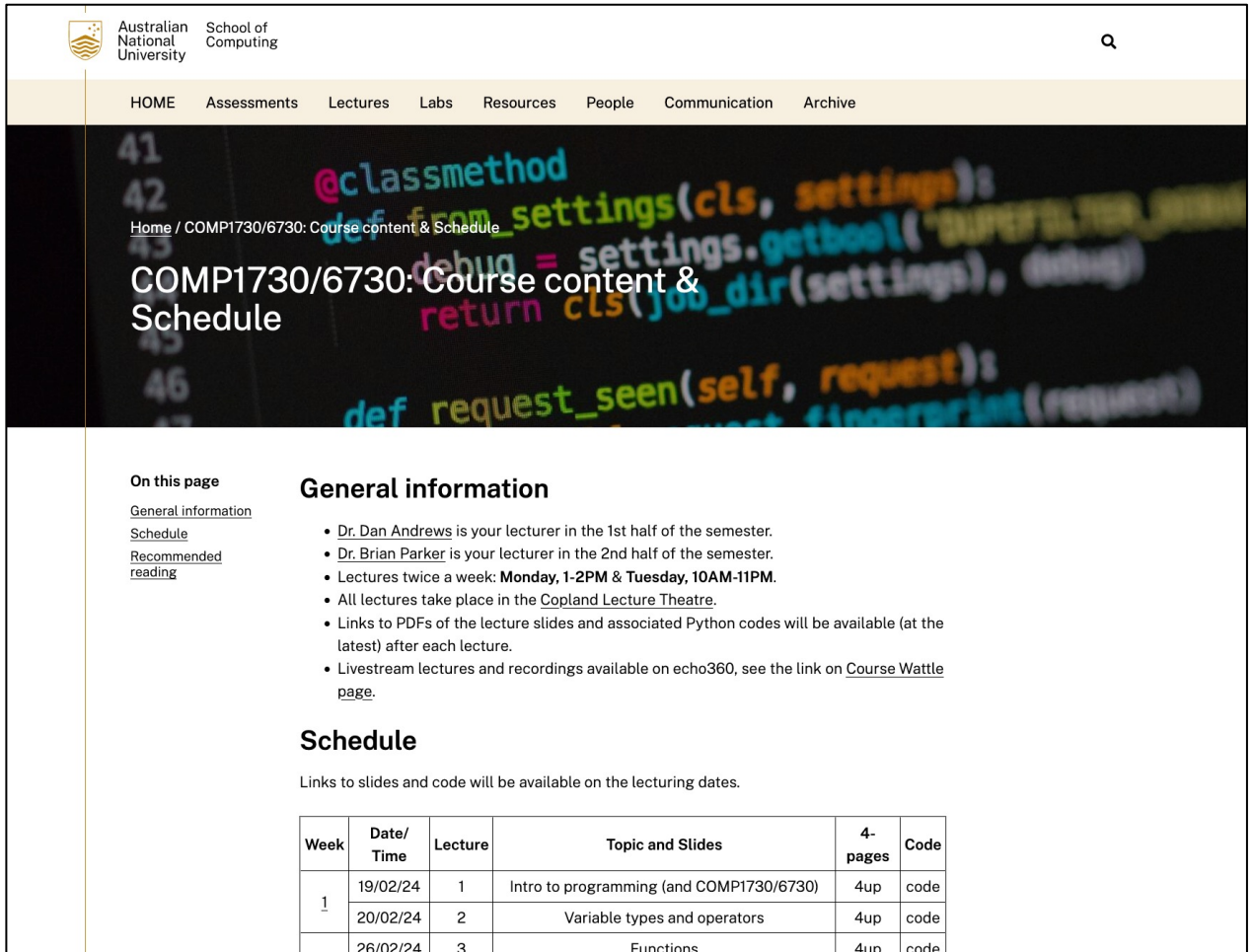
See: https://comp.anu.edu.au/courses/comp1730/lectures/

# Lecture slides and code examples:

Australian National University

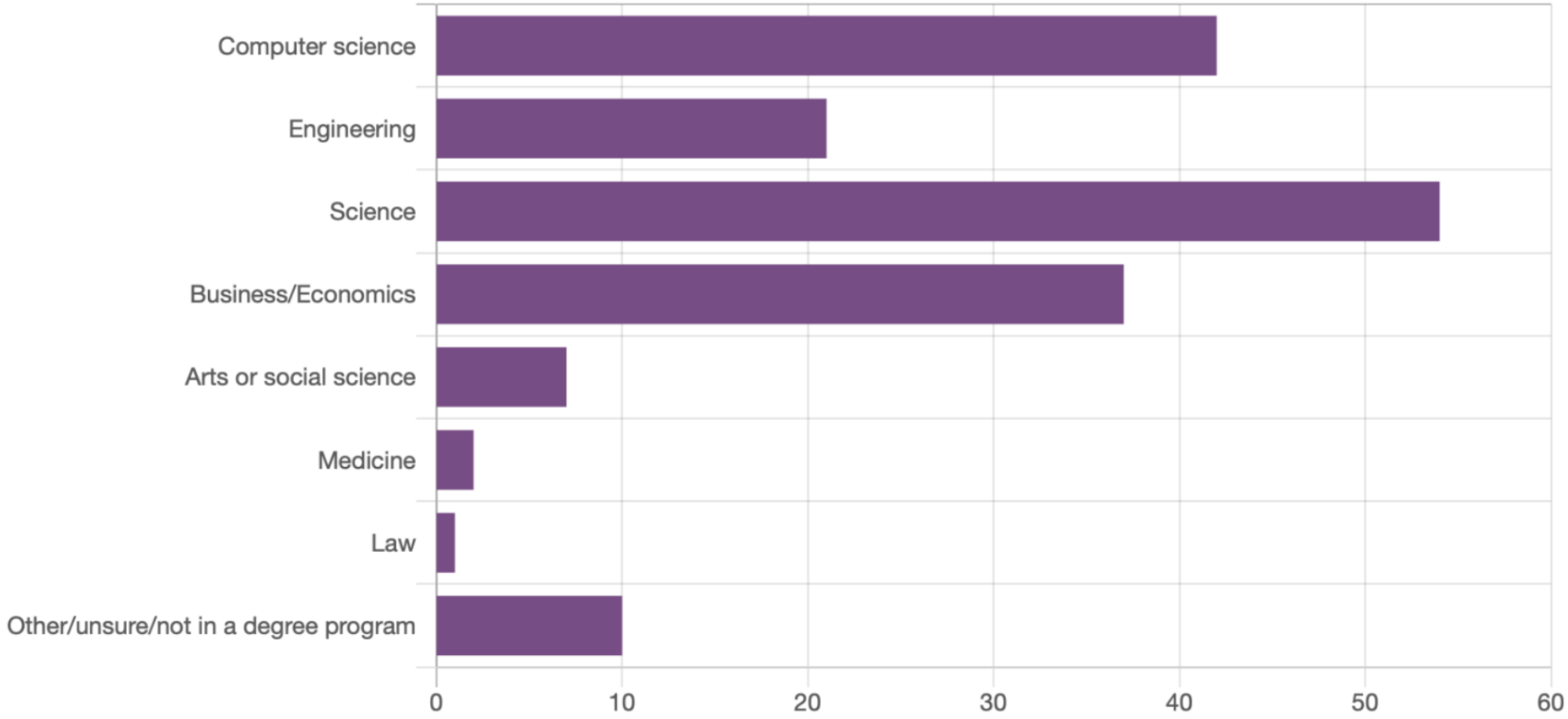
- Course website:

https://comp.anu.edu.au/courses/comp1730/lectures/



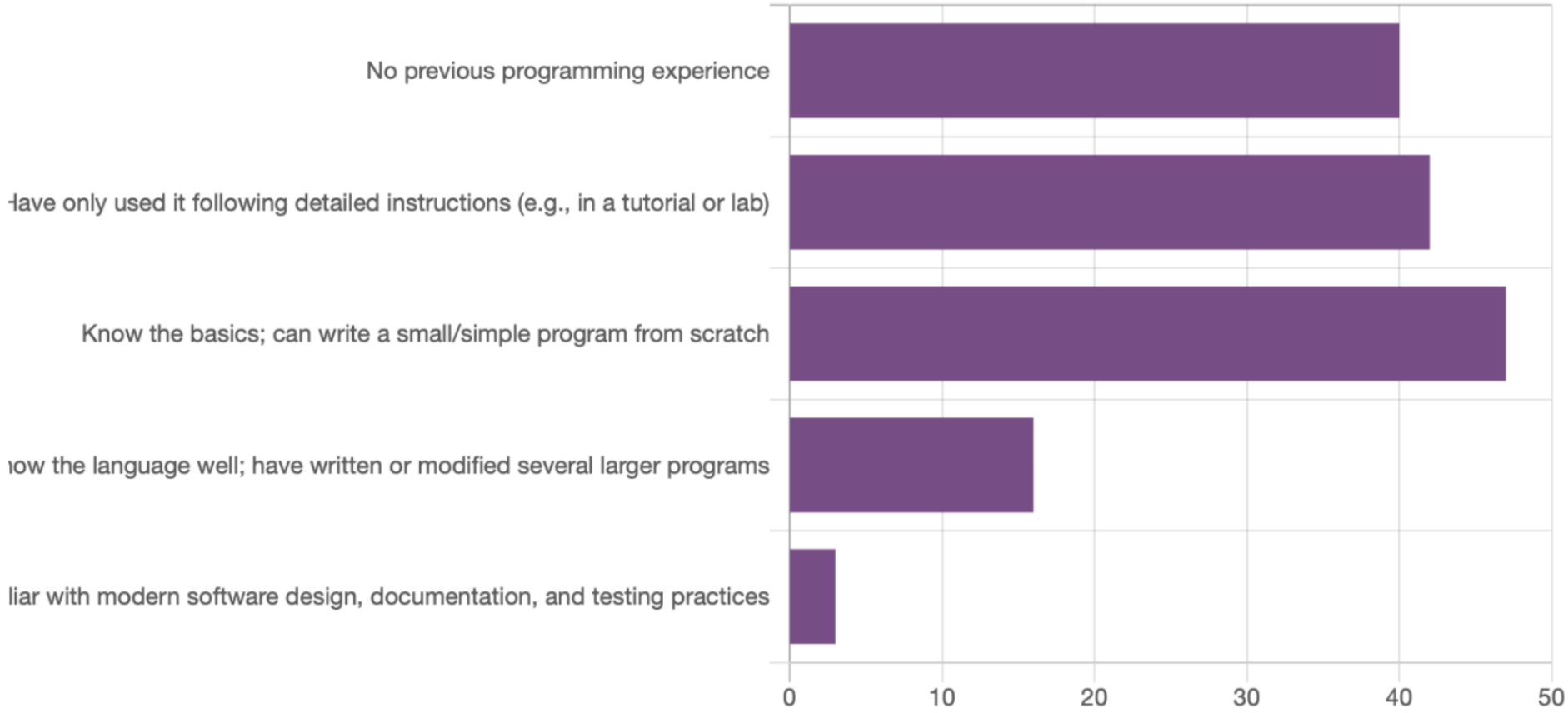
# About you – results of demographic survey

Australian National University



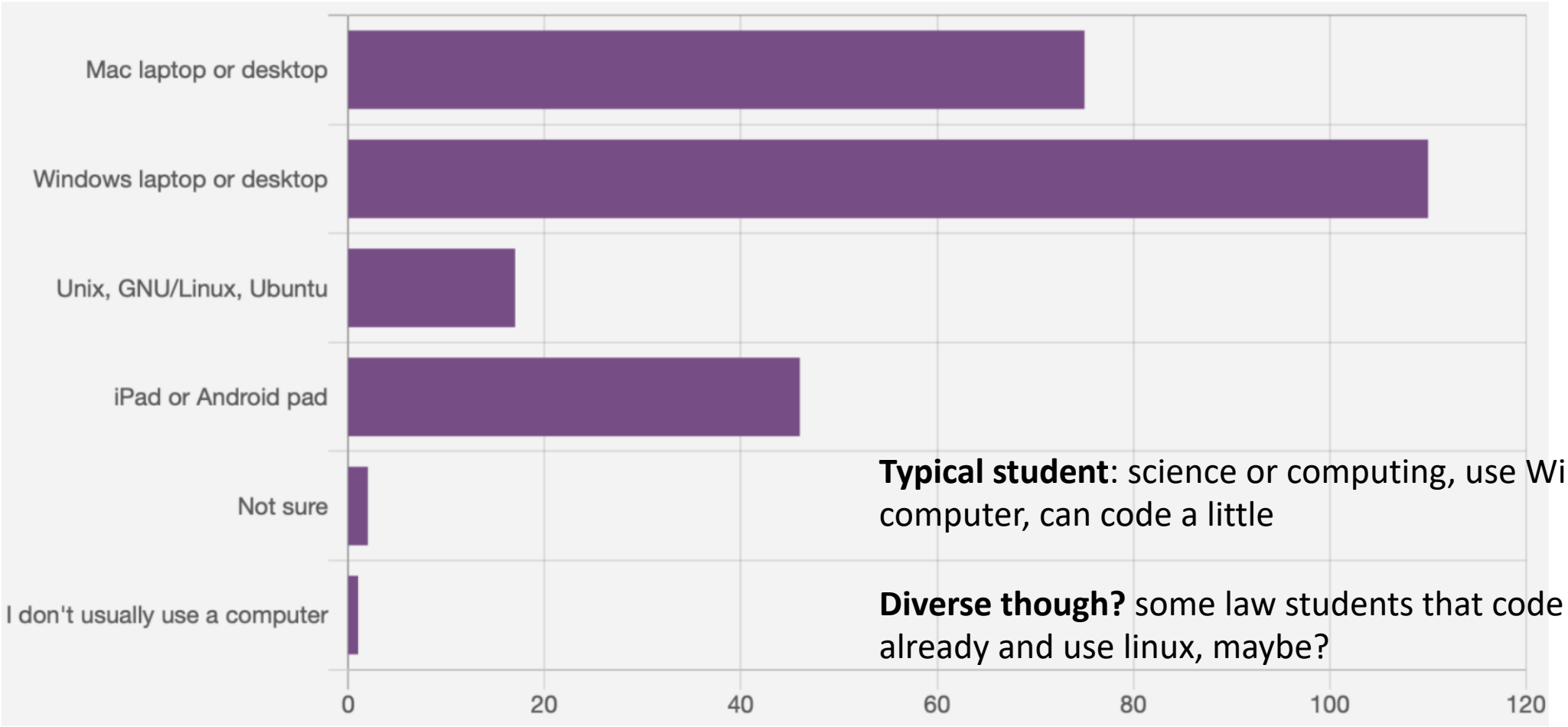
# About you – results of demographic survey

Australian National University



# About you – results of demographic survey

Australian National University




**Typical student**: science or computing, use Windows computer, can code a little

**Diverse though?** some law students that code well already and use linux, maybe?

# Introductory Lecture - format

Australian National University

- Orientation to python – let's look at some code first
- Learning to program
- Reference books and other reading
- Variables and Expressions (part I)

AND (at the end):

- Admin:
  - Lab class enrolments
  - Assessment
  - Other announcements

# Hello, World!

Australian National University

- Brian Kernighan, of C fame, is attributed to be responsible for the first ‘Hello, world!’ program.
- It is much simpler to implement this in Python than C for the early 1970’s

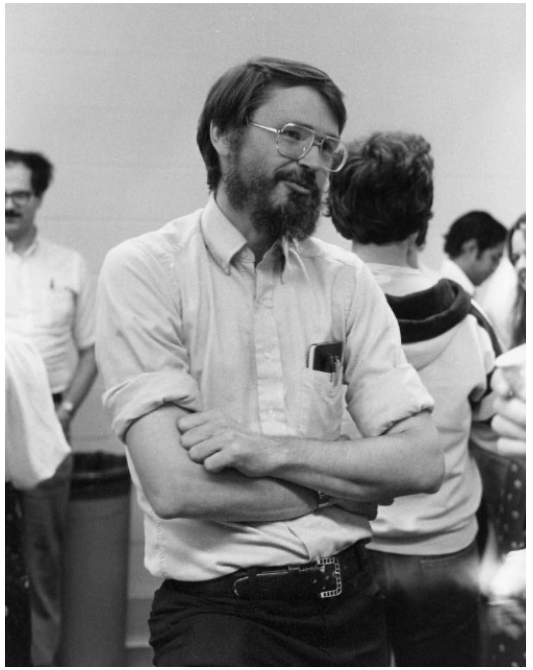
In C:

```
main( ) {
    extern a, b, c;
    putchar(a); putchar(b); putchar(c); putchar('!*n');
}

a 'hell';
b 'o, w';
c 'orld';
```

Source: Wikipedia

In Python (in the interpreter interactive mode):

```
In [1]: print("Hello, world!")
Hello, world!
```



Brian Kernighan, ca. 1972

# Example: Running python programs?

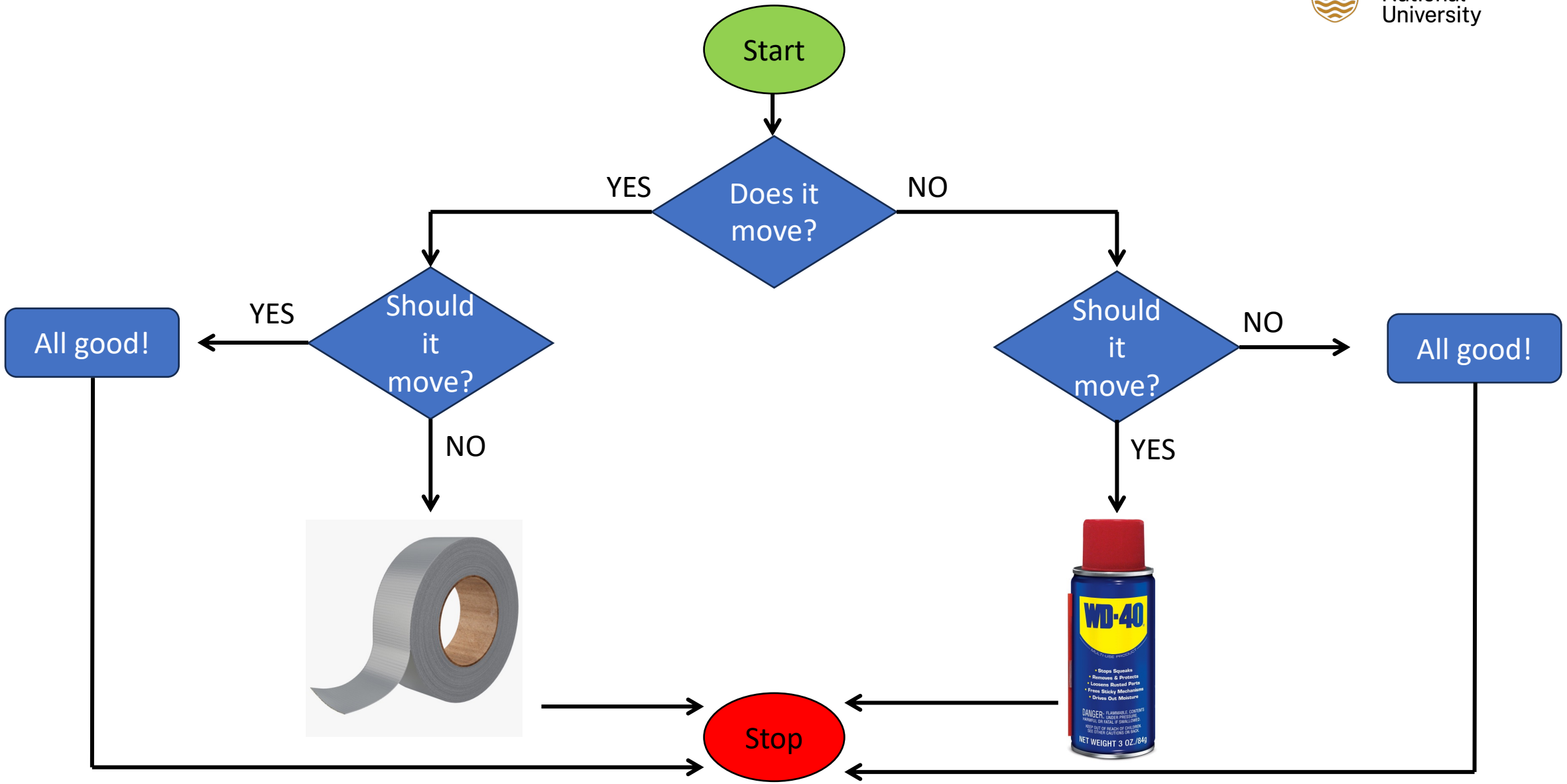
Australian National University

- Code_L1_1.py

- Interactive mode python
- Script mode – files that end with *.py
- Running in a terminal
- Running in an Integrated Development Environment

# What does this look like in Python?

Australian National University



# In Python:

Australian National University

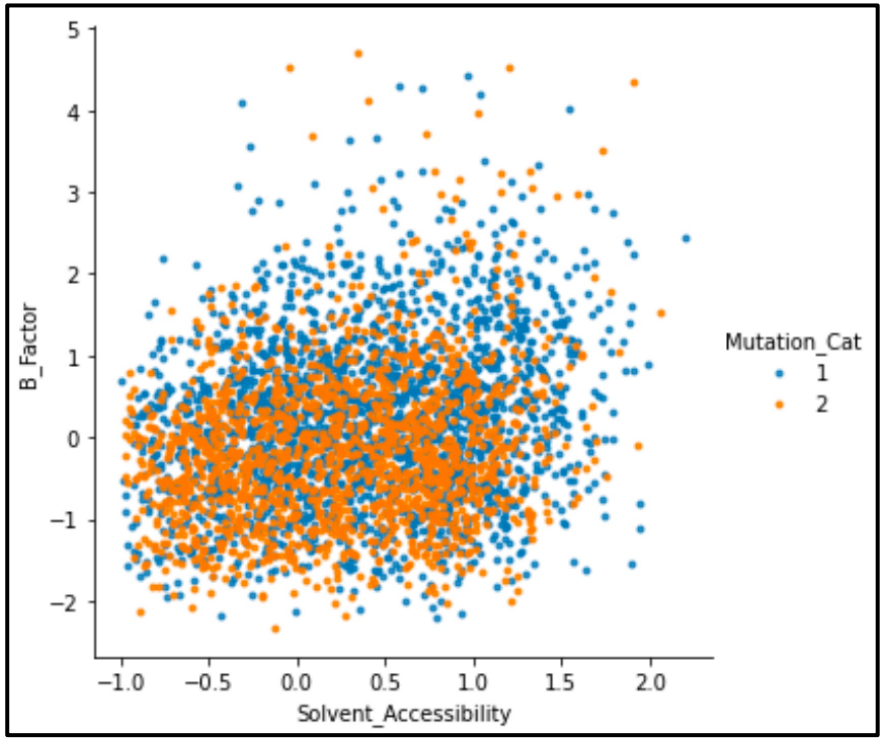
Code_L1_2.py

```
"""
Implementation of a time-tested algorithm to fix
all known mechanical problems.
"""

def advice_bot(moves_str, shouldit_str):
    """ Ponders problem and provides solution text
    moves_str - (str) response to 'does it move' question
    shouldit_str - (str) response to 'should it move' question
    """

    if moves_str == shouldit_str:
        return 'All good'
    elif moves_str == 'Y' and shouldit_str == 'N':
        return 'Apply duct tape'
    elif moves_str == 'N' and shouldit_str == 'Y':
        return 'Use WD-40'


print("START")

print("Does it move? (Y/N):")
response1 = input()

print("Should it move? (Y/N):")
response2 = input()

advice_response_str = advice_bot(response1, response2)

print("Solution: " + advice_response_str + "!")

print("STOP")
```

# More complicated: Python as a toolbox

Australian National University

```
import pandas as pd
import seaborn as sns
from sklearn.tree import DecisionTreeRegressor

mutations = pd.read_csv('/Users/dan/ownCloud/work/lectures/'
                        +'comp1730_prog_for_scientists/2024_S1/'
                        +'code_examples_in_lectures/lecture1/'
                        +'data_file.csv')

sns.lmplot(x="Solvent_Accessibility",
           y="B_Factor",
           data=mutations,
           fit_reg=False,
           hue='Mutation_Cat',
           legend=True,
           markers='.',
           x_jitter=True,
           y_jitter=True)

train_data = mutations.iloc[0:, [26, 30]].values
labels = mutations.iloc[0:,[6]].values

tree_clf = DecisionTreeRegressor(max_depth=2)
tree_clf.fit(train_data, labels)

# Solvent_Accessibility: 4.9%
# B-factor: -0.9 (low, -ve)
prediction = tree_clf.predict([[ 0.049, -0.9]])
print("Residual function prediction: " + str(prediction[0]))
```



# What is programming?

COMP1730 & COMP6730

Reading:

Chapter 1: Downey, *Think Python,*

Chapter 1: Sundnes, *ItSPwP*

*OR*

Sections 1& 2: https://docs.python.org/3/tutorial/index.html

Australian
National
University

# Why Python?

Australian National University

- This is not a course on programming in python.  It is a course on programming that uses python
- Why python?
  - Python is a very popular programming language
    - Especially in science and engineering
  - Open source, available on most platforms
  - Huge external code libraries for doing just about everything, in:
    - Data Science
    - Machine Learning
    - Bioinformatics…
- We will use python 3 (beware older books that are python 2)

python™

Python creator, Guido van Rossum in 2019

# This course

Australian National University

- Is a first course in programming, in python
  - Focused in transferable, practical skills
  - Coding languages come and they go – but the good coding practice is relevant to all languages
  - Useful to those in science and engineering.
  - Not foremost teaching commercial software engineering
- A beginners course – no prior experience required.  But this doesn't mean we are going to go slowly, or that it will be easy!
- We will use python 3

# Coding as a craft

Australian National University

- Some recommendations:
  - Read widely and write code frequently.  Practice, practice, practice.
  - This won't end well if it is already week 9 and this is the first time you are looking at the course.
  - *Textbooks and reading*: if you only attend one part of this course, make sure it is the tutorials.  Though, these will be very hard if you don't at least attend lectures or do the course reading
  - In the beginning, as you start to write your first programs, it might feel bad as you make all the beginners mistakes.  Don't worry and keep trying.  Everyone starts here.
  - Error messages are your friend…

# Reading: Course Textbooks

Australian National University

Alex Downey (2016) *Think Python*, 2nd Edition

Sundnes (2020) Introduction to Scientific Programming with Python (ItSPwP)

(electronic copies available at: https://comp.anu.edu.au/courses/comp1730/resources/)

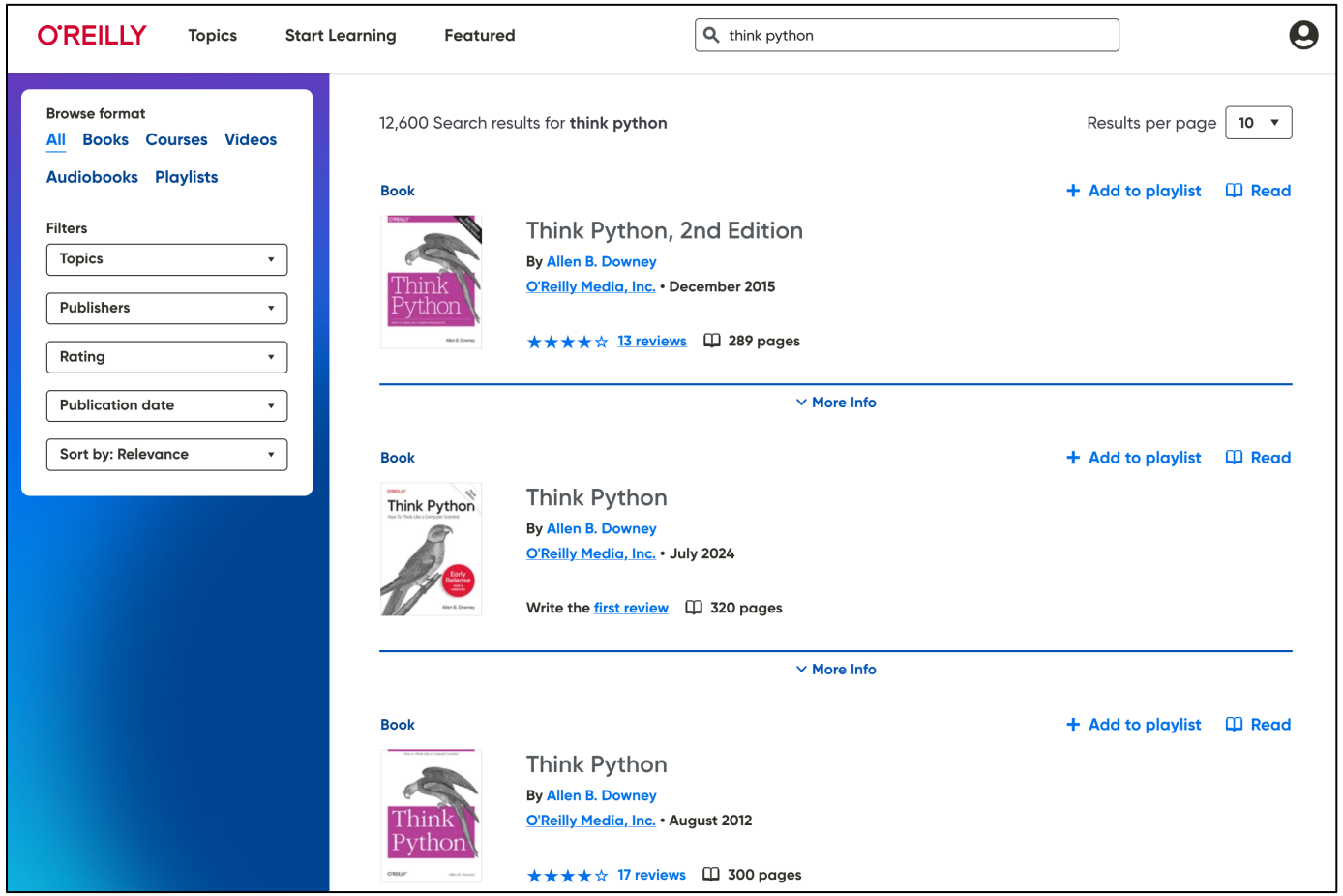
- Other good resources:
  https://docs.python.org/3/tutorial/index.html
- Other books:
  Al Sweigart (2015) Automate the Boring Stuff with Python
  Bill Lubanovic (2019) Introducing Python, 2nd Edition

- When reading other python books, make sure they are python 3!
- Be careful with web resources – some are great (eg. docs.python.org). Many aren't.

# Safari Books (ANU library subscription)

Australian National University

https://www.oreilly.com/library-access/




Or, as a PDF file: https://greenteapress.com/wp/think-python-2e/

# Variables (part I)

COMP1730 & COMP6730

<u>Reading:</u>

Chapter 2 : Downey, *Think Python*

Chapter 2, Sundnes, *ItSPwP*

*OR*

*https://docs.python.org/3/tutorial/index.html*

Australian
National
University

# Variables – what are they?

Australian National University

- Contain a program's data whilst it is executing
- Assignment statements:

```
>>> message = 'And now for something completely different'
>>> n = 17
>>> pi = 3.141592653589793
```

Downey (2015) Think Python, 2nd Ed.

- In memory – the 'state' of the program:

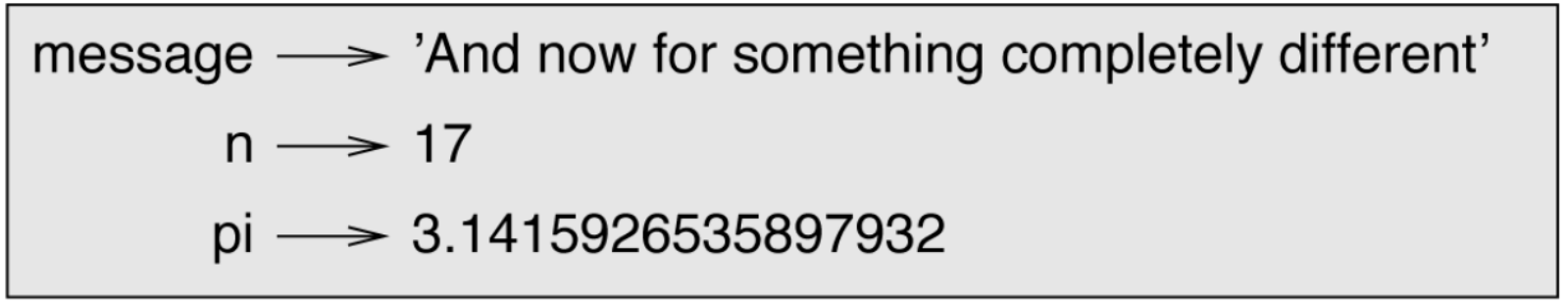



Figure 2-1. State diagram.

Code_L1_4.py

# Types of variables (in python)

Australian National University

- All variables have a type – and you will get an error is you store an incompatible value in the wrong type (eg. a string value in an integer variable type)
- Or try to do something inappropriate with a data type (eg, print an integer as a string)
- Basic data types:
  - int – integer
  - float – decimal values
  - str – strings of one or more characters
  - bool – Boolean values, `True` or `False`
- And variables that contain multiple values of basic data types:
  - List and Tuple – sequences an index
  - Dict – a hash, key-value pairs

**Table 1-2:** Common Data Types

| Data type | Examples |
|---|---|
| Integers | `-2, -1, 0, 1, 2, 3, 4, 5` |
| Floating-point numbers | `-1.25, -1.0, -0.5, 0.0, 0.5, 1.0, 1.25` |
| Strings | `'a', 'aa', 'aaa', 'Hello!', '11 cats'` |

Sweigart (2019) Automate the Boring Stuff with Python, 2nd Ed.

# Every variable has a type

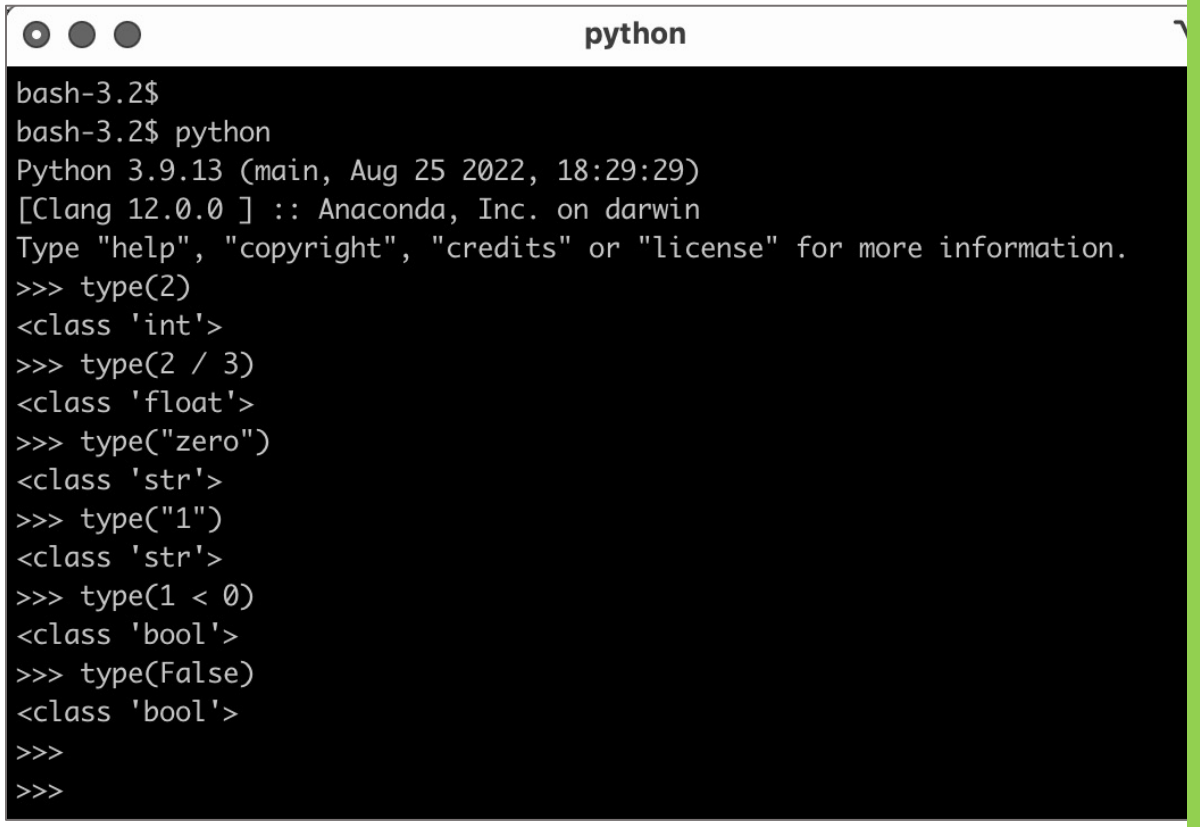
Australian National University

Variable types in python:

- Integers (type `int`)
- Floating-point numbers (type `float`)
- Text strings (type `str`)
- Truth or Boolean values (type `bool`)

Variable types determine what we can do with values (and sometimes what the result is)

- The `type()` function tells us the type of a variable:




Code_L1_5.py

# Numeric types: `int`

Australian National University

- `int` types represent the mathematical integers (positive and negative whole numbers) (0, 1, 2, -1, -17, 4096,…)
- Values of type `int` have no inherent size limit in python

```
>>>
>>> 2 ** (2 ** 2)
16
>>> 2 ** (2 ** (2 ** 2))
65536
>>> 2 ** (2 ** (2 ** (2 ** 2)))
200352993040684646497907235156025575044782547556975141926501697371089405955631145308950613088093334810103823434290726318182294938211881266886950636476154702916504187191635158796634721944293092798208430910485599057015931895963952486337236720300291696959215610876494888925409080591145703767520850020667156370236612635974714480711177481588091413574272096719015183628256061809145885269982614142503012339110827360384376787644904320596037912449090570756031403507616256247603186379312648470374378295497561377098160461441330869211810248595915238019533103029216280016056867010565164675056803874152946384224484529253736144253361437372908830379460127472495841486491593064725201515569392262818069165079638106413227530726714399815850881129262890113423778270556742108007006528396332215507783121428855167555407334510721311242739956298271976915005488390522380435704584819795639315785351001899200002414196370681355984046403947219401606951769015611972698233789001764151719005113346630689814021938348143542638730653955296969138802415816185956110064036211979610185953480278716720012260464249238511139340046435162386756707874525946467090388654774348321789701276445552940909202195958575162297333357615955239488529757995402847194352991354376370598692891375715374000198639433246489005254310662966916524341917469138963247656028941519977547770313806478134230959619096065459130089018888758808473362595606544488850144733570605881709016210849971452956834406197969056546981363116205357936979140323632849623304642106613620022017578785185740916205048971178182040018728293994344618622432800983732376493181478984811945271300744022076568091037623999203492023906626264491909167985461515778839060397720759279378852241294301017458086862263369284725851403039615558564330385450688652213114813638408384778263790
```

- Note: can’t use commas to format integers for readability
  - Write `128736` not `1,282,736`

# Numeric types: `float`

Australian National University

- Floating-point numbers (type `float`) approximate the mathematical real numbers
- Values of type float have limited range and limited precision
  - Min/max ± 1.79 x $10^{308}$
  - With a few exceptions to this limit
  - Though this is the typical limit – the actual limits depend on the python implementation
- Type `float` also has special values ± `inf` (infinity) and `nan` (not a number)

# String variables

Australian National University

- Strings (type `str`) represent text
- A string literal is enclosed in single or double quote marks

```
>>> "Hello world"
'Hello world'
>>> '4" long'
'4" long'
```

- A string (in python) can contain other types of quote mark, but not the one used to **delimit** it
- More about strings (so much more) in a coming lecture

# Suggested Exercises

Australian National University

- Complete Exercises 2-1 and 2-2 on Page 18 & 19 of *Think Python*.

# Course Organisation

COMP1730 & COMP6730

Australian
National
University

# Course Admin, Information and Contacts

Australian National University

- https://comp.anu.edu.au/courses/comp1730/
- Wattle for announcements, forums, quizzes, surveys and assignment submission
- Recordings of lectures are available on Wattle
- Read the Wattle news and announcements!
- To ask a question:
  - Use the discussion forum on Wattle
  - Ask your tutor in labs
  - For private matters, use the course email: comp1730@anu.edu.au
  - Always use your ANU email address, to avoid the spam filters
  - **Please don't email the course convenors directly** – these emails will be ignored

# Schedule overview

Australian National University

- Two lectures per week
- All lectures will be presented live and will be recorded
- Follow content and schedule: https://comp.anu.edu.au/courses/comp1730/lectures/
- One 2-hour lab per week, starting from Week 2
  - **Before Fri 23rd Feb - Sign-up for a lab class** with MyTimetable (linked via Wattle): https://mytimetable.anu.edu.au/odd/student
- Assessments will be due at **11:55pm on Sunday** of weeks when due (unless otherwise specified): https://comp.anu.edu.au/courses/comp1730/assessments/
- You are expected to spend another 6 hours per week studying the course:
  - doing the recommended reading
  - solving all lab exercises, and
  - time spent to practice coding

# Drop-In Sessions

Australian National University

- As of this semester, we are continuing weekly drop-in sessions for 1-to-1 tutor contact
- Times will be announced in later in 1st Term
- **Python installation help sessions**:
  - Tues – 3-5pm – Birch Building, Lab 1.08
  - ~~Weds – 4-6pm – Hanna Neumann Bldg, Computer Lab 1.24~~
  - Thurs – 11am-1pm – room N114, CSIT Building (#108)

# Assessment (preliminary)

Australian National University

| Component/Link | Weight | Release date | Due date/Exam date |
|---|---|---|---|
| Homework 1 | 3% | 26/02/2024 (Wk 2) | **03/03/2024, 23:55PM** |
| Homework 2 | 3% | 04/03/2024 (Wk 3) | **10/03/2024, 23:55PM** |
| Homework 3 | 3% | 18/03/2024 (Wk 5) | **23/03/2024, 23:55PM** |
| Homework 4 | 3% | 25/03/2024 (Wk 6) | **14/04/2024, 23:55PM** |
| Homework 5 | 3% | 15/04/2024 (Wk 7) | **21/04/2024, 23:55PM** |
| Project Assignment | 35% | 22/04/2024 (Wk 8) | **10/05/2024, 23:55PM (Fri Wk 10)** |
| In-lab project assessment | Mandatory discussions with a tutor in weeks 11 & 12 following the due date. **If absent, your project mark will be zero.** | | |
| Final Exam | 50% | N/A | TBA |

See: https://comp.anu.edu.au/courses/comp1730/assessments/

- Final exam:
  - In-person, in computing labs
  - COMP1730 & COMP6730 at different times
  - Not a hurdle assessment

- Assignment:
  - Individual assignment is a take-home programming assignment
  - There will be a *viva* component of the assignment assessment
    - Held during weeks 11 and 12 at same times as your usual lab session
    - Students are expected to have a thorough knowledge of their own work and be able to speak in detail about their answers and solutions

- The assessment scheme will be final at end of Week 2. Any changes will be announced.

# Academic honesty

Australian National University

- Submitted code will be checked computationally for evidence of plagiarism.
- If evidence of plagiarism is found in individual homework problems, the mark for that individual homework will not be posted, until all homeworks have been assessed. In the context of all homeworks if it is decided there is evidence of repeated plagiarism, students will be interviewed for possible action of academic misconduct.
- The take-home assignment and exam will also be checked for evidence of academic misconduct.
- **What is okay**: for the homework, discussing the programming problems and approaches to solve them with other students is allowed, provided that no code is exchanged and that the final solution and code is written individually. In this case, the other students involved in the discussion must be listed in a comment at the top of the homework.
- For the final exam and take-home assignment must be individual work. You may not discuss the questions or your answers with anyone (this includes any on-line forum).
- Note that in all cases every line of code submitted must be fully written by you from scratch (and not just a modified copy of a version from the internet), and must be fully understood and explainable by you. Sufficient inline comments should be provided to make clear that you understand the code.
- Note on large language models and other code generators: generative AI models such as github copilot, chatGPT, Bing chatbot etc can be used by students for the homeworks and take-home assignment to explore solutions and understand their own code. They will not be allowed for the final exam. But in all cases the final code submitted by the student must be fully written and understood by the student, as described above.
- If you are unsure, please ask your tutor or the convenors.

# Assessment

Australian National University

- All assignment deadlines are hard – no late submissions will be accepted.  Unless previous permission has been granted.
- Extension requests and late submissions **require documentary evidence**, such as a medical certificate
- Regarding deferred assessments and special consideration, please read: https://www.anu.edu.au/students/program-administration/assessments-exams
- Please note that "*any submitted work may be subject to an additional oral examination*", which can change the assessment mark in any way.

# Useful Links:

Australian National University

- Install python - if you want to start, follow the instructions to install python (via Anaconda) on your laptop: https://comp.anu.edu.au/courses/comp1730/labs/install/

- Lab materials - this is where to find the labs: https://comp.anu.edu.au/courses/comp1730/labs/

- And the assessment description:

  https://comp.anu.edu.au/courses/comp1730/assessments/

# Wattle Discussion forum

Australian National University

**In general, this is where you should go to ask questions**

**3 simple rules:**

1. Read before you post.
   - Before posting a question, check if your question has already been answered

2. Give you post a good, descriptive topic
   - Don't write '*A question*'. Write something like '*Variable assignment: why does the value not change?*'

3. You **may not post** solutions to assignment problems

These rules are good etiquette and apply to any online forum.

# Important tasks:

Australian National University

1. Complete the **demographic information questionnaire** on course Wattle page
2. Sign up to a **lab class!**
   - Do this via myTimetable: https://mytimetable.anu.edu.au/odd/student
   - Link also accessible from Wattle page
   - Do this by end of Week 1 (Fri 23rd Feb)
   - Labs start in Week 2
   - Homework 1 is also due in Week 2
   - In-lab assessment starts in Week 2
3. Login to STREAMS: https://cs.anu.edu.au/streams/
   - This will create an account for you on the lab computers
4. **Prepare for the labs!** Attend lectures, read lab instructions – and attempt some of the exercises before attending your lab
5. Make sure you have a working python programming environment:
   - Install Anaconda on your own computer:
   - Go to: https://www.anaconda.com/download
   - Current installation will give you python 3.9 or later
   - Includes that Spyder IDE as part of installation
   - For more tips and detailed instructions: https://comp.anu.edu.au/courses/comp1730/labs/install/
   - Or, install another python3 implementation
   - Or, verify that you can reliably use the lab computers