COMP1730/COMP6730
Programming for Scientists

Recap and where to now?

Announcements

* Practice Examination Available in Wattle.
* Oral Examination Sign-Up in Wattle.
  - Will remain open until 11:55pm Wednesday 10 June.
  - If you just want to do the Wattle exam, you don't have to do anything.

Lecture outline

* Recap of the course
* What else is in Python
* Learning other languages
* And then what?

The Basic Building Blocks

* We started with an introduction to the fundamental programming constructs:
  - Variables, statements, expressions and data types (L1 & L3).
* Then moved on to the different ways of changing the flow of execution.
  - Functions and functional abstraction (L2, L4, L14 & L22).
  - Control flow - `if/else`, `for` and `while` (L5 & L6).

Data Types

* We looked in detail at a few different data types:
  - `str` (L9)
  - `float` (L11)
  - `list` and `tuple` (L8 & L13)
  - `dict` and `set` (L16)
* And the differences between immutable types and mutable types (L13 & L16).

Writing Good Code

* We have focused a lot of energy on trying to write good code:
  - Code quality - comments, docstrings, naming, organisation, efficiency (L7)
  - Testing and debugging - unit tests using `pytest` (L10)
  - Error handling and exceptions (L18)
  - Design and organisation and standards (L19)

Other topics

* And finally some other topics we thought were important:
  - Data science - useful in general but also for the major assignment (L12).
  - Files and IO - how to read and write (large amounts of) data (L15).
  - Complexity - how our algorithms scale as data gets larger (L17).
  - Modules - how Python works with modules (L21).

* So was this a thorough treatment of everything in Python?

* Not even close!
* Used roughly 10 out of approximately 200 standard library modules. See this list for a complete breakdown.
* Used roughly 15 out of the 1400+ in the Anaconda Python distribution.
* And that's not even counting the wealth of other Python libraries that have been developed.

Object Oriented Programming

* Conceptually, the biggest missing piece is likely *object oriented programming*.

```
class NewClass(object):
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2

    def m1(self):
        return 2 * self.p1 + self.p2
    ...
```

And Python has a package to do just about everything

* Data Science and Machine Learning - `numpy`, `SciPy` and `Scikit-learn`.
* Image Processing - `scikit-image`, `Pillow` and `OpenCV`.
* GIS - `shapely` and `geopandas`.
* Audio - https://wiki.python.org/moin/Audio/.
* Videos, games, databases, webpages, you name it, someone has written a Python module to do it.

* And are you limited to Python?

* What does the following C# function do?

```
public int Fun_A(List<int> sequence)
{
    int total = 0;
    foreach (int i in sequence)
    {
        if (i > 0)
        {
            total += i;
        }
    }
    return total;
}
```

* We said at the start that this was going to be a programming course that uses Python, not a Python course.
* Learning a second language will be much easier than learning the first one was.
* Concepts like loops, if/else, functions, types, etc. are all used in most modern programming languages.
* You just need to learn the different syntax, but the approach can stay the same.

Other Programming Courses at ANU

* COMP1100 - Programming as Problem Solving
* COMP1110 - Structured Programming
* COMP1600 - Foundations of Computing
* These courses focus less on the *syntax* and programming language, and more on how language features and models are used to solve problems.

Other ANU Computing Topics

* Software Engineering
* Data Science
* HCI
* AI and Machine Learning
* Computer Systems
* Security
* and many more.