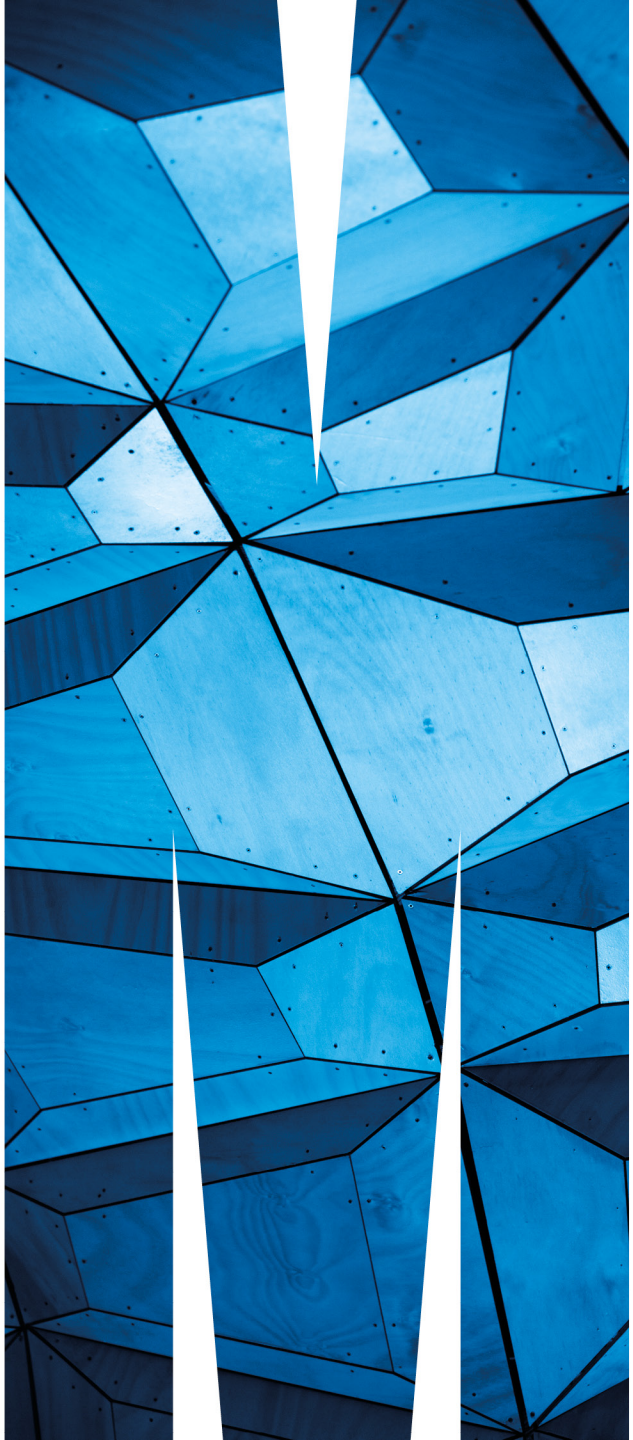
MONASH University

MONASH
INFORMATION
TECHNOLOGY

FIT9136: Algorithms and Programming Foundations in Python

Week 1:
Introduction to Programming and Algorithms
Python Basic Data Types



GROUP OF EIGHT AUSTRALIA

FIT9136 Unit Schedule

- Week 1: Introduction to Programming and Algorithms Python Basic Data Types
- Week 2: Python Basic Elements
- Week 3: Control Structures
- Week 4: Built-in Data Structures
- Week 5: Classes and Variable Scope
- Week 6: Abstract Data Types
- Week 7: Binary Trees and Binary Search Trees
- Week 8: Testing, Exception Handling, and External Libraries
- Week 9: Complexity, Searching and Sorting Algorithms
- Week 10: Recursion and Divide-and-Conquer
- Week 11: Greed, Brute-force, Backtracking
- Week 12: Review of the Unit

Find overview on Moodle > Unit Previews

Quick question

- What is programming?
  - A science?
  - Or an art?

Small caveat

- Programming is an art. (A scientifically based art)
- You only get good at performing an art by practise.
- Very few artists start off as brilliant.
- YOU MUST MAKE MISTAKES TO LEARN PROGRAMMING
- This includes struggling to know what to do.

MONASH
University

Introduction to Python

The Evolution of Python

- Invented in Christmas of 1989 by a Dutch programmer
  - Guido van Rossum
  - "Python" is named after Guido's favorite British comedy series "Monty Python's Flying Circus"

- Python 2 vs. Python 3:
  - Python 3 was released to address various design decisions and inconsistency in Python 2 (and its subsequent releases 2.x)
  - Python 3 is backward incompatible with Python 2.x
  - Note: We will use Python 3.8 (or Python 3.7/3.6) for this unit

Why Python?

- Python is a general-purpose programming language that can be used to literally develop any kind of programs.
- Python is a *simple* programming language that is easy to learn and use.
- Python is supported with a rich collection of libraries or packages (i.e. *ready-to-use* code) to build sophisticated programs.

"**Life is short**
(You need Python)"

--*Bruce Eckel*
ANSI C++ Committee member

What Can Python Do?

- Scripting
  - Crawling
  - Calculation
- Website development
- Visualization
- Data Science
  - Computer Vision
  - Natural Language Processing
  - Speech Recognition
- Desktop application with GUI

With enough time and effort, Python can do whatever you want it to do!

Who Uses Python? [Chapter 1, Learning Python by Mark Lutz (2013)]

MONASH
University

Running Python Programs

How to Execute Python Programs?

- Python programs can be executed by using a Python interpreter in two modes:
  - Interactive mode
  - Script mode

- Interactive mode:
  - Start up the Python interpreter by running the command “python” at the prompt of a command-line terminal
  - Type the Python statements at the interactive mode prompt represented by >>>

```
Python 3.5.2 |Anaconda 2.5.0 (x86_64)| (default, Jul  2 2016, 17:52:12)
[GCC 4.2.1 Compatible Apple LLVM 4.2 (clang-425.0.28)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

How to Execute Python Programs? (continue)

- Script mode:
  - Create the Python source file (a.k.a. the *module* file) with the file extension of `.py` using any text editor
  - Pass the source file (e.g. `program.py`) as an argument to the "python" command:

```
python program.py
```

Python IDE: Anaconda (Jupyter Notebook)

- Integrated Development Environment (IDE):
  - More manageable and convenient to develop Python programs
  - Equipped with a set of utilities and libraries (packages)
  - Assist programmers to edit, build, and debug their programs
  - E.g. IDLE (the default Python IDE), **PyCharm**, Jupyter Notebook (an interactive web-based IDE)

- Anaconda:
  - Python distributions especially for data science.
  - Bundled with more useful packages for scientific computation and data analysis (e.g. NumPy, SciPy, Pandas, etc.)

Running Python with PyCharm

[PyCharm Launcher Window]

Programming Tools

- Stack Overflow
  - Largest Q&A site about programming
  - 19M questions, 29M answers, 12M users

- GitHub
  - Largest host of source code
  - >100M repositories including 28M public ones, 40M users

MONASH
University

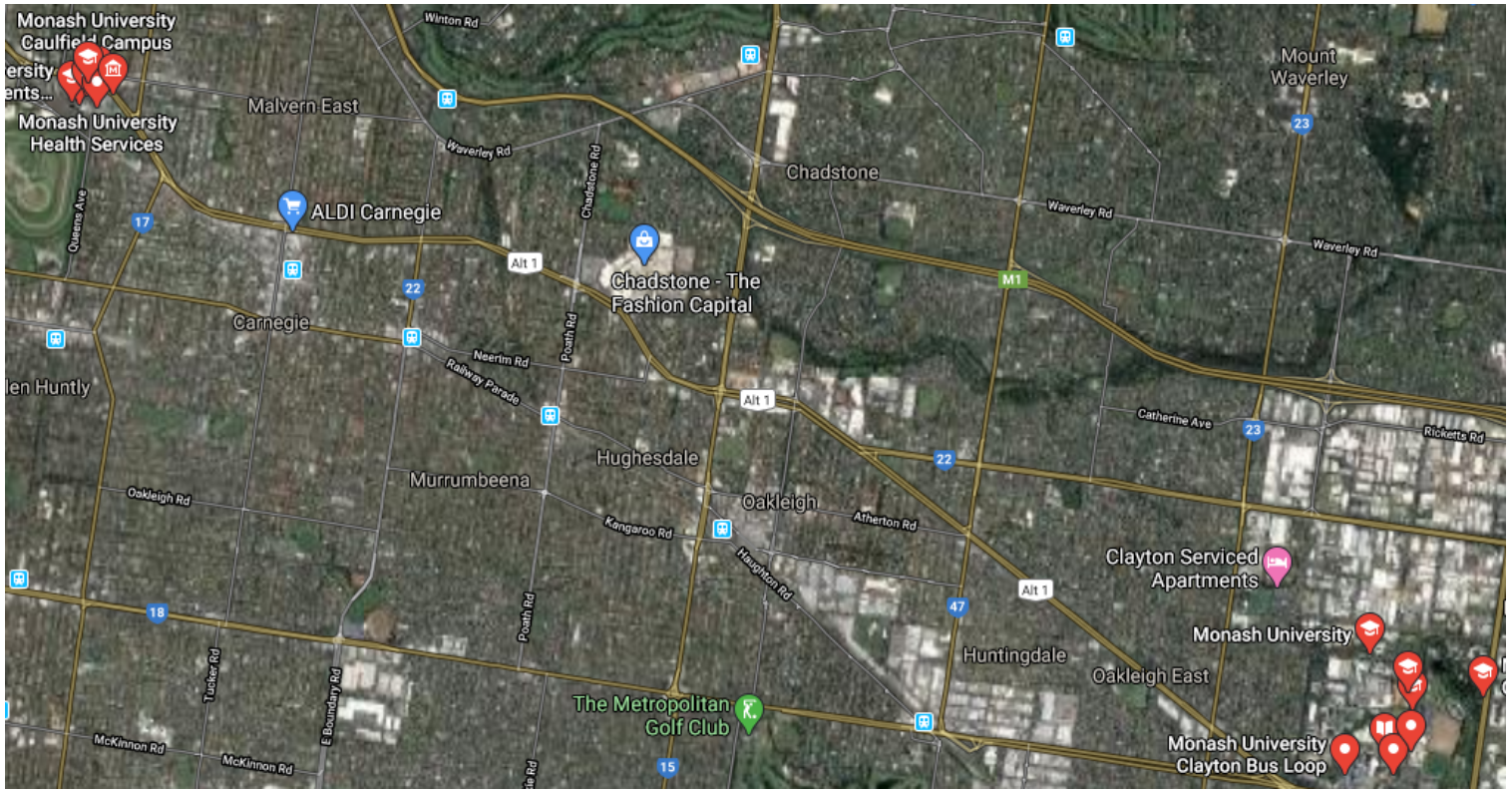
Fundamental Concepts of Computational Thinking

Abstraction & Programs & Algorithms

- What is abstraction?
  - Intends to ignore irrelevant details but focusing only on the essential ones
- What is algorithm?
  - The general solution designed for a specific problem
  - Specifies a sequence of step-by-step instructions with the flow of control indicating how each of the instructions should be executed
- What is program?
  - A solution implemented with a specific programming language to solve a computational problem

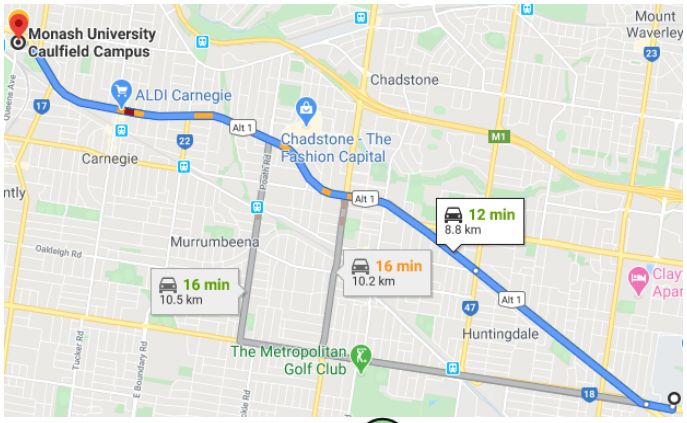
Let's do some computational thinking…

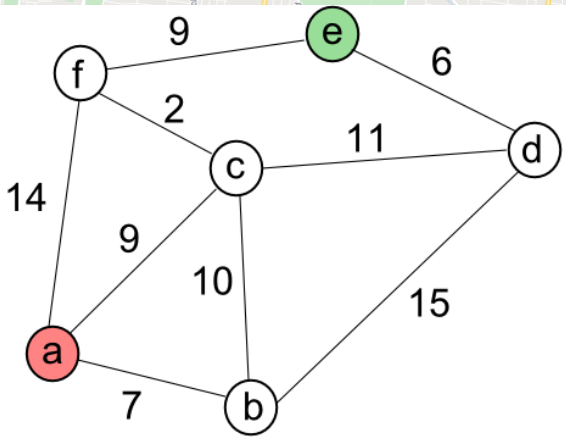
- How would you drive to Monash University Clayton Campus from Caulfield Campus?



Solution

- abstraction
  - Shortest path between 2 points
  - All roads and streets
- algorithm
  - Finding all potential paths
  - Calculate the distance
- program
  - Detailed source code in Python






```
from heapq import heappush, heappop
# based on recipe 119466
def dijkstra_shortest_path(graph, source):
    distances = {}
    predecessors = {}
    seen = {source: 0}
    priority_queue = [(0, source)]

    while priority_queue:
        v_dist, v = heappop(priority_queue)
        distances[v] = v_dist

        for w in graph[v]:
            vw_dist = distances[v] + 1
            if w not in seen or vw_dist < seen[w]:
                seen[w] = vw_dist
                heappush(priority_queue,(vw_dist,w))
                predecessors[w] = v

    return distances, predecessors
```

A bit more thinking…

- Following a recipe when cooking
- Travelling to University
- Putting on clothes
- Beating Super Mario Bros.

Basic Elements of Python: Objects and Variables

Syntax and Semantics

- Each programming language is defined by its own syntax and semantics
- Syntax:
  - A set of rules that defines how program instructions are constructed from various symbols and structures of a specific language
  - Programs constructed with invalid syntax will cause *syntax errors*
- Semantics:
  - Meaning associated with a program or its individual instructions that defines what the program is intended to achieve
  - Semantic errors (*logic errors*) happen when a program is syntactically well formed but did not produce the expected result

Objects in Python

- Objects:
  - Core elements that Python programs manipulate on
  - Pieces of memory locations in the computer that containing (holding) a specific type of data value or literal

- Each object is associated with a specific data type (or object type)
  - How a program can manipulate it?
  - What kind of operations that a program can perform on it?

Examples of Python Objects

- How many Python objects in this program?

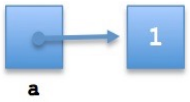
```
a = 1
b = 2
result = a + b
```

  - Both literal values ‘1’ and ‘2’ are objects of type *integer* (`int`)
  - The sequence of characters “The addition of a and b is” is object of type *int* (`int`)

- To find out the type of an object with the built-in Python function `type()`:
  - E.g. `type(1)` or `type(result)`

Variables in Python

- Variables:
  - A means of associating a label to a value stored in the computer memory
  - Python: a label is a *reference* to a object
  - E.g.: `a = 1`



- A same variable can be associated with a number of different objects that could have a different value or a different data type

```
a = 1
a = 100
a = "Hi Python"
```

Python variables do not have a data type but the associated objects do; and the type of the object is determined by the literal that it contains.

Naming Rules in Python

- Variable names in Python:
  - Can only contain: lowercase letters (a-z), uppercase letter (A-Z), digits (0-9), underscore (_)
  - Case sensitive
  - Cannot begin with a digit
  - Cannot be keywords (reserved words) in Python

and	as	assert	break	class	continue
def	del	elif	else	except	finally
for	from	global	if	import	in
is	lambda	nonlocal	not	or	pass
raise	return	try	while	with	yield
False	None	True			

Naming Conventions in Python

- When using multiple words as variable names:
  - Use a single underscore (_) as the *delimiter* between words (lowercase)
    - E.g.: `student_number, number_list`
  - Use the camelCase style
    - E.g.: `studentNumber, absentStudentNumber`

Use either one and be consistent throughout your programs

Variable names should be ***meaningful*** and usually *self-explained* with the kind of data that they represent (e.g. `a` vs. `studentNumber`?)

Naming Conventions in Python

- Python Enhancement Proposals (PEP)
  - "*Variable name should be lowercase, with words separated by underscores as necessary to improve readability.*"
  - https://www.python.org/dev/peps/pep-0008
- Google Python Style Guide
  - "*Function names, variable names, and filenames should be descriptive; eschew abbreviation. In particular, do not use abbreviations that are ambiguous or unfamiliar to readers outside your project, and do not abbreviate by deleting letters within a word.*"
  - https://google.github.io/styleguide/pyguide.html
- A Neural Model for Method Name Generation from Functional Description
  - 26th IEEE International Conference on Software Analysis, Evolution, and Reengineering, 2019

Week 1 Summary

- So far, we have discussed:
  - Python as a high-level programming language
  - Execution of Python programs
  - Computational thinking: abstraction, algorithm, and program.
  - Python object and variables

- Next week:
  - Core data types in Python
  - Operators and expressions
  - Statements and assignments
  - Standard input and output in Python

Reminder: Labs begin next week.