# CM2015: Programming with Data

# Summary

Arjun Muralidharan

12th October 2020

# Contents



# List of Figures

# List of Listings

# 1 Setting up your programming environment

## 1.1 Jupyter Setup

Install Jupyter Lab using the following command.

```
pip3 install jupyterlab
```

Run Jupyter Lab using the following command.

```
jupyter lab
```

The interface allows you to create individual `.py` files, or entire *notebooks* that contain *cells*. Each cell is a small set of commands to execute and view the output of inline. A cell can contain code or markdown.

## 1.2 Python Basics

Python relies on whitespace indentation, and comments are given using the `#` sign. Code blocks are initiated with a colon, after which the contents of the block are indented one level.

```
# This is a simple Python script
for x in array:
        if x < pivot:
                less.append(x)
        else:
                greater.append(x)
```

Everything in Python is an *object*, making methods available on e.g. variables, strings, data structured, modules or even functions. In a notebook, pressing "Tab" will expose the available methods.

Python assigns variable by reference, i.e. it uses binding. This means that when a variable $a$ is assigned to another variable $b$, changes to $a$ will also reflect in $b$.

Python is *strongly typed* but will make implicit type conversions when obvious. To find the type of a variable, use the following function.

```
isinstance(a, int)
isinstance(a, (int, float))
```

This returns `true` if the provided variable is of the given type. In the second example, a tuple is used, where the function checks if the variable is of any type provided in the tuple.

Check if an object is iterable:

```
# Returns true if the argument is iterable
isiterable(a)
```

```
# Example: Convert an iterable structure to a list
if not isinstance(x, list) and isiterable(x):
        x = list(x)
```

Python can import other Python files as modules, and even use variables from those files with new names.

```
import some_module as sm
from sm import PI as pi, g as gf

r1 = sm.f(pi)
r2 = gf(6, pi)
```

Python supports all the standard arithemtic and logical operators. One operator worth highlighting is the *is* operator.

```
# Returns true if both variables are referencing the same object
a is b
a is not b

# NOTE: These are not the same as
a == b
a != b
```

The standard scalar types are `int`, `float`, `str`, `bytes`, `bool`, and `None`.

While strings can be written with single or double quotes, in Python we can use triple quotes for strings spanning multiple lines.

```
c = '''
This is a multi-line
string that has a lot
of text in it
'''
```