# Problem Solving with Python

Ryan Greenup
*17805315*

James Guerra
*19045229*

WESTERN SYDNEY UNIVERSITY

## Contents

```
code /home/ryan/Dropbox/Studies/QuantProject/Current/Python-Quant/ & disown
```

Here's what I gatthered from the week 3 slides

## 1 Introduction

Python has become one of the most used programming languages to date. It can be applied in Web and Software Development, Desktop GUI's and Science to name a few https://www.python.org/about/apps/. In the first part of this report, we will use python to solve mathematical problems which are difficult to solve using just a pen and paper. We wanted to show that there are numerous ways to approach a problem in python. To do this, we used the help of Prof. Roozbeh Hazrat's textbook, "Mathematica®: A Problem-Centered Approach". This led to understanding the different modules and functions available in python, which allowed us to explore different methods to solve the problems. This will effectively provide insight into the importance and advantages of computational thinking. The second part of this report focuses on dynamical systems, in particular chaos theory, Mandelbrot's set and fractals. Dynamical systems became an intriguing theory to research and with the help of python we can visually display the mathematics behind it.

## 2 Basics

Learning the following fundamental features of python is going to be essential for the upcoming problems. It begins with understanding:

- Lists
- Functions
- Loops

### 2.1 Lists

Lists are one of the most useful data types and are widely used within our research. Lists can hold multiple pieces of data and hence can represent matrices, co-ordinate pairs, vectors, etc. An example of a list is:

```
lst = [1, [2, 3], "4"]
```

*The preceding code is a list containing a range of data types.*

## 2.2 Functions

Simply put, functions are blocks of code that perform tasks. They are either built-in or programmer defined. The key benefits of a function is code re-usability, readability and to break down bigger tasks. An example of a programmer defined function is:

```
def sum(m, n):
    return m + n
```

*The preceding code requires two arguments which will return their sum.* https://www.tutorialspoint.com/

## 2.3 Loops

Needing to perform a task multiples times is majorly common when programming. It is certainly a tedious task hard coding repetition, especially over a large range like $10^{10}$ for example. Hence, loops are created as a means to mitigate this monotonous task by sequentially passing through iterable objects such as lists, **ranges**, **strings** etc. An example is:

```
for item in range(1, 10):
    print(item**2)
```

*The preceding code will display squared numbers ranging from 1 to 9.*

# 3 Computational Thinking

**Problem 1**
Let m be a natural number and $A = \frac{(m+3)^3+1}{3m}$ . Find all the integers m less than 500 such that A is an integer.

**Solution**
Let $S = \{m_1, m_2, ..., m_k\}$ be the solution set to $A \epsilon Z$. Where $m_k$ is an integer that satisfies $A$. Now, consider the following statement

$$\begin{aligned}(m+3)^3 + 1 &\equiv 0 \ (mod\ 3m)\\ m^3 + 9m^2 + 27m + 28 &\equiv 0 \ (mod\ 3m)\\ m^3 + 28 &\equiv 0 \ (mod\ 3m)\end{aligned}$$

By inspection:
$m = 1$ gives:

$$\begin{aligned}(1)^3 + 28 &\equiv 0 \ (mod \ 3 \cdot 1) \\ 29 &\not\equiv 0 \ (mod \ 3)\end{aligned}$$

$m = 2$ gives:

$$\begin{aligned}(2)^3 + 28 &\equiv 0 \ (mod \ 3 \cdot 2) \\ 36 &\equiv 0 \ (mod \ 6) \\ 0 &\equiv 0 \ (mod \ 6)\end{aligned}$$

$$\vdots$$

$m = 449$ gives:

$$\begin{aligned}(499)^3 + 28 &\equiv 0 \ (mod \ 3 \cdot 499) \\ 124251527 &\equiv 0 \ (mod \ 1497) \\ 527 &\not\equiv 0 \ (mod \ 6)\end{aligned}$$

It turns out by inspection $S = \{2, 14\}$ is the solution set.
Now let's see how we approach this in python.

```
p = lambda n: ((n+3)**3 + 1)
print([i for i in range(1, 500) if (p(i) % (3*i)) == 0])

[2, 14]
```

Understanding the mathematics first can sometimes allow you to transfer the same principles into python. First call python's lambda function to create a mathematical function $p(n) = (n + 3)^3 + 1$, use the modulo operator % to find if the remainder does in fact equal 0 (mod $3m$), then place this inside a list comprehension to return a list of all $n$ values.

# 4 Dynamical Systems

# 5 Appendices

# A Appendix A: Data Types

## A.1 Strings

Strings are a list of characters put together. A character in the string can be accessed by typing

```
string_name[index(s)]}
```

. Strings have numerous methods which can be used to amend them.

## A.1 Integers, Floats and Complex Numbers

Integers, floats and complex numbers are all numeric type variables. They allow for mathematical operations and incrementing.

## A.2 Booleans

Boolean values take two forms, either 'True' or 'False'. Sometimes conditions or statements need to be checked such as, "$3 < 10$", python will interpret this and will display 'True' since the statement is true.
https://www.w3schools.com/python/python$_booleans.asp$