mybook

Jane Doe

26/05/2022

Table of contents

Preface

This unit provides an overview of programming languages and why you might want to programme. It is preparation for the R (**?@sec-rlang**) **?@sec-rlang** and Python (Chapter 2) units, which introduce the two programming languages used at CIM[^introduction-1].

1 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

2 Python

2.1 Variables

Variables are named containers for data and they allow us to call them when we need them. We select a name and assign something to it. We can then just call the name to do something with the variable.

```python
text = "Hello Warwick"
text
```

```
'Hello Warwick'
```

Variables can be changed at any time.

```python
text = "Hello London"
text
```

```
'Hello London'
```

! Naming variables

It is strongly advised that you give the variables meaningful names that describe what they are doing. This will help you when you return to the programme.
The naming conventions section of the Python style guide gives examples of what Python will allow.

2.2 Data Types

Python is a dynamically typed language. Unlike languages like Java or C++, Python does not require the type declaration or for a variable to be declared in advance. We can declare a variable with some data and Python will infer the type when it uses it.

In statically typed language, a variable needs to be declared before it is used. In Java, we would type:

```
int anInt;
anInt = 5;
```

`anInt` will then be an integer when it is compiled and run.

In Python, we could declare a variable and then assign a different type of data later to it.

```
an_int = 5
print(an_int)
an_int = "Oh dear"
an_int
```

```
5
```

```
'Oh dear'
```

As you may come to find, some Python operations expect data in a certain format. It is good to understand the typing and to be careful when declaring variables that you are not changing an existing variable.

Python does have some core data types:

2.2.1 String

A string of alphanumeric characters (AKA, a text).

```
"Hello Warwick"
```

2.2.2 Integer

A whole number, like 1,2,3

```
1
```

2.2.3 Float

A number with a decimal place, like `1.0`, `2.0`, `3.0`

```
1.0
```

2.2.4 Collections

These basic types can be put together into collections.

A list is a collection of items. These can be of any type, including lists and dictionaries.

```
a_list = [1,2,"a"]
```

If we know where some data is, we can access it directly, which we call *slicing*.

Like variables, you can reassign data in a list.

```
a_list[2]
print(a_list[2])
a_list[2] = "b"
a_list
```

```
a
```

```
[1, 2, 'b']
```

However you need to be careful with this and to know the length of the list.

```
a_list[3]
```

```
IndexError: list index out of range
```

You should get an error saying that list "list index out of range". This is because Python starts counts from 0, not 1.

We can find the size of the list, if we do not already know it, using:

```
len(a_list)
```

```
3
```

How might you find the last element in a list using this?

2.2.4.1 Tuples and sets

Tuples are like lists and operate in much the same way. However, they are immutable. Once the elements are created, they cannot be changed.

Sets operate in the same way as lists but can only contain one of each value that you want to use. A list can be `[1,1,2,3]` where a set would be `[1,2,3]`.

2.2.5 Dictionaries

Dictionary, like lists, are collections. Unlike lists, they collect pairs of key : values. The key is a name of something and the associated value. Like lists, dictionaries are reassignable.

```
a_dict = {"name": "Tom", "surname":"Thumb", "place":"London"}
print(a_dict)
a_dict["surname"] = "Thumbelina"
a_dict
```

```
{'name': 'Tom', 'surname': 'Thumb', 'place': 'London'}
```

```
{'name': 'Tom', 'surname': 'Thumbelina', 'place': 'London'}
```

Understanding dictionaries will help if you process any data formatted in JSON.

Python has a collections library that can be used for different containers.

2.3 Summary

In this section, we introduce the Python language and the core data types. The concept of types and how Python sees them is discussed.

2.4 Further Reading

Lutz, M. 2013. Learning Python (5 ed). O'Reilly Media, Sebastopol

Driscoll, M. 2014. Python 101

Sweigart, A. 2019. Automate the Boring Stuff (2 ed), No Starch Press, San Francisco

Python Style Guide

Static vs Dynamic Typing

Part I

Introduction to R

This unit will provide a basic understanding of the R language and will cover data handling and visualisation.

Why R?

R is a free and widely used statistical programming language. Employees of companies such as Google and Microsoft use R to process, analyse and visualise data. Work in R is supported by code written by leading academics in mathematics, statistics and many other areas. If you work in data analysis in your future career then knowing how to use R will be a very useful skill.

Where is R used?

Learning outcomes

By the end of this unit, you'll be able to:

1. Do this
2. Do that

Requirements

Preparing for the unit

Before we start the face to face session please you should:

1. Install R and RStudio (instructions can be found here: (Chapter **installingr?**))
2. Download exercise files (Data and code) from here

Instructions for install R and RStudio on Mac can be found here.

3 Summary

In summary, this book has no content whatsoever.

References

Knuth, Donald E. 1984. "Literate Programming." *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.