The Object Oriented Paradigm

In this worksheet you are to:

1. Enhance the definitions of each item used in OOP with a simpler, more user-friendly definition
2. Provide a code example in Python
3. That code examples MUST be commented properly
4. All references must be in Harvard format[1]. Use a tool like citethisforme.com[2] as your referencing tool.

Classes

A class is a blueprint for an object that will use that class. We can think of class using an example of a sketch of a parrot with labels. It contains all the details about the name, colours. size etc. Classes can contain both attributes (data) and methods (functions or processing) tha(are related to the class.

Encapsulation

Encapsulation may refer to a mechanism of restricting the direct access to some components of an object, such that users cannot access state values for all variables of a particular object. Encapsulation can be used to hide both data members and data functions or methods associated with an instantiated class or object.

Abstraction

Abstraction occurs when a programmer hides any irrelevant data about an object or an instantiated class to reduce complexity and help users interact with a program more efficiently.

Inheritance

Inheritance allows us to inherit attributes and methods from the base or parent class. This is useful as we can create sub-classes and access all functionality from our parent class. Then we can overwrite and add new functionalities without affecting the parent class.

[1] **How to Cite Different Sources with Harvard Referencing**
https://www.student.unsw.edu.au/citing-different-sources
[2] **Create Harvard, APA & MLA citations**
https://www.citethisforme.com/

Generalisation

This concept revolves around designing classes and methods so that they can be reused. With some design pre-planning the classes and methods can be created so that future redundant code is reduced and the re-usability of code is increased. The concept links closely with inheritance to include generalisations of objects in the parent class and specialisations in child classes.

Polymorphism

This is the ability of objects and methods to perform differently in different contexts. For example, the plus sign will perform differently when working with strings or numeric data types in Python.