Encapsulating data

A "class" is a specified prototype describing a set of properties that characterize an object. Each class has a data structure that can contain both functions and variables to characterize the object.

The properties of a class are referred to as its data "members". Class function members are known as its "methods", and class variable members (declared within a class structure but outside any method definitions) are known as its "attributes".

Class members can be referenced throughout a program using dot notation, suffixing the member name after the class name, with syntax of *class-name.method-name*() or *class-name.attribute-name*.

A class declaration begins with the **class** keyword, followed by a programmer-specified name (adhering to the usual Python naming conventions but beginning in uppercase) then a : colon. Next come indented statements optionally specifying a class document string, class variable attribute declarations, and class method definitions – so the class block syntax looks like this:

class ClassName :

    ''' class-documentation-string '''

    class-variable-declarations

    class-method-definitions

The class declaration, which specifies its attributes and methods, is a blueprint from which working copies ("instances") can be made.

All variables declared within method definitions are known as "instance" variables and are only available locally within the method in which they are declared – they cannot be directly referenced outside the class structure.

Typically, instance variables contain data passed by the caller when an instance copy of the class is created. As this data is only available locally for internal use it is effectively hidden from the rest of the program. This technique of data "encapsulation" ensures that data is securely stored within the class structure and is the first principle of Object Oriented Programming (OOP).

Hot tip

It is conventional to begin class names with an uppercase character and object names with lowercase.