**Python OOP**

**Overview**

|  |
| --- |
| As you have seen from the earliest code examples in this course, it is not compulsory to organise your code into classes when you program in Python. You can use functions by themselves, in what is called a *procedural* programming approach. However, while a procedural style can suffice for writing short, simple programs, an object-oriented programming (OOP) approach becomes more valuable the more your program grows in size and complexity. |

**Difference between Procedure Oriented and Object Oriented Programming**.

* Procedural programming creates a step by step program that guides the application through a sequence of instructions. Each instruction is executed in order.
* Procedural programming also focuses on the idea that all algorithms are executed with functions and data that the programmer has access to and is able to change.
* OOP is much more similar to the way the real world works; it is analogous to the human brain. Each program is made up of many entities called objects.
* In addition, OOP uses data fields where Procedural Programming uses procedures.

The most important principle of object orientation is encapsulation: the idea that data inside the object should only be accessed through a public interface - that is, the objects methods.

**What is an Object?**

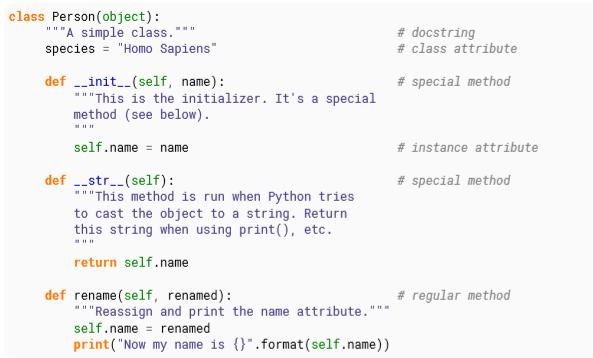
* Objects are the basic run-time entities in an object-oriented system.
* They may represent a person, a place, a bank account, a table of data or any item that the program must handle.
* When a program is executed the objects interact by sending messages to one another.
* Objects have two components, Data(i.e attributes) and behaviours (i.e methods/functions).

**What is a Class?**

Classes are a way of grouping together related data and functions which act upon that data. It is a kind of data type, just like a string, integer or list. When we create an object of that type, we call it an instance of a class.

**Defining and using a class (Section 38.1 on the module notes)**

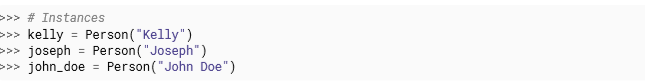
Here is an example of a simple class which can be used to rename a person’s name.



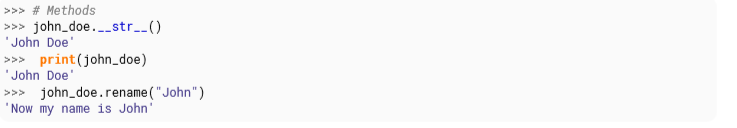
There are a few things to note when looking at the above example.

1. The class is made of attributes (data) and methods (functions).
2. Attributes and methods are simply defined as normal variables and functions
3. The *\_\_init\_\_()* method is equivalent to a constructor in other languages. It is the method that is first run when you create a new object.
4. Attributes that apply to the whole class are defined, and are called class attributes.
5. Attributes that apply to specific objects are called instance attributes, defined inside *\_\_init\_\_()* method.
6. Every python method accepts *self* (object in question). This is the recommended convention.
7. Python has no real concept of private elements.
8. Some of the class methods have the following form *\_\_functionname\_\_(self, other)\_\_*. All such methods are called magic methods. See relevant documentation.

Now let’s make a few instances of our Person class!

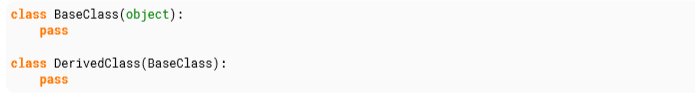


We can execute the methods of the class using the same dot operator .:



**Basic inheritance**

Inheritance is based on similar ideas used in other object oriented languages like Java, C++. A new class can be derived from an existing class as follows.



The BaseClass is the already existing (parent) class, and the DerivedClass is the new (child) class that inherits (or subclasses) attributes from BaseClass.

Python allows multiple and multilevel inheritance.

**Listing all Class Members**

The dir() function can be used to get a list of the members of a class.

For example:



**Task 1**

Briefly describe a possible collection of classes which can be used to represent a music collection (for example, inside a music player), focusing on how they would be related by composition. You should include classes for songs, artists, albums and playlists. Hint: write down the four class names, draw a line between each pair of classes which you think should have a relationship, and decide what kind of relationship would be the most appropriate.

For simplicity you can assume that any song or album has a single “artist” value (which could represent more than one person), but you should include compilation albums (which contain songs by a selection of different artists). The “artist” of a compilation album can be a special value like “Various Artists”. You can also assume that each song is associated with a single album, but that multiple copies of the same song (which are included in different albums) can exist

Write a simple implementation of this model which clearly shows how the different classes are composed. Write some example code to show how you would use your classes to create an album and add all its songs to a playlist. **Hint**: if two objects are related to each other bidirectionally, you will have to decide how this link should be formed – one of the objects will have to be created before the other, so you can’t link them to each other in both directions simultaneously!