Introduction to object-oriented programming

Ben Thomas

b.a.thomas@ucl.ac.uk

Overview

- Introduction to Jupyter
- Introduction to object oriented programming (OOP)
- Key concepts in OOP
    - Encapsulation
    - Data abstraction
    - Inheritance
    - Polymorphism
- Conclusion

Introduction to Jupyter

- Jupyter notebooks[1] are human-readable documents that contain both executable code (e.g. Python) and descriptions, results etc.
- "Executable documents"
- Client-server application that allows you to edit and run notebooks in a web browser.

[1] http://jupyter.org/

Introduction to object oriented programming (OOP)

- Conventionally a program is created through a collection of functions and procedures.
- An object oriented program is:
  - Based on *classes*.
  - Operates using a collection of interacting objects.
- In OOP, objects process data and can send and receive messages, possibly from other objects.

- A class definition can be compared to the recipe for baking a cake[2].
- A recipe (class) and a cake (an instance/object of the class).
- To bake a cake you need ingredients and instructions.
- A class needs variables and methods.
- Classes may have variables and methods that are common and others that differ.

---

[2] http://www.python-course.eu/objectoriented*programming*.php

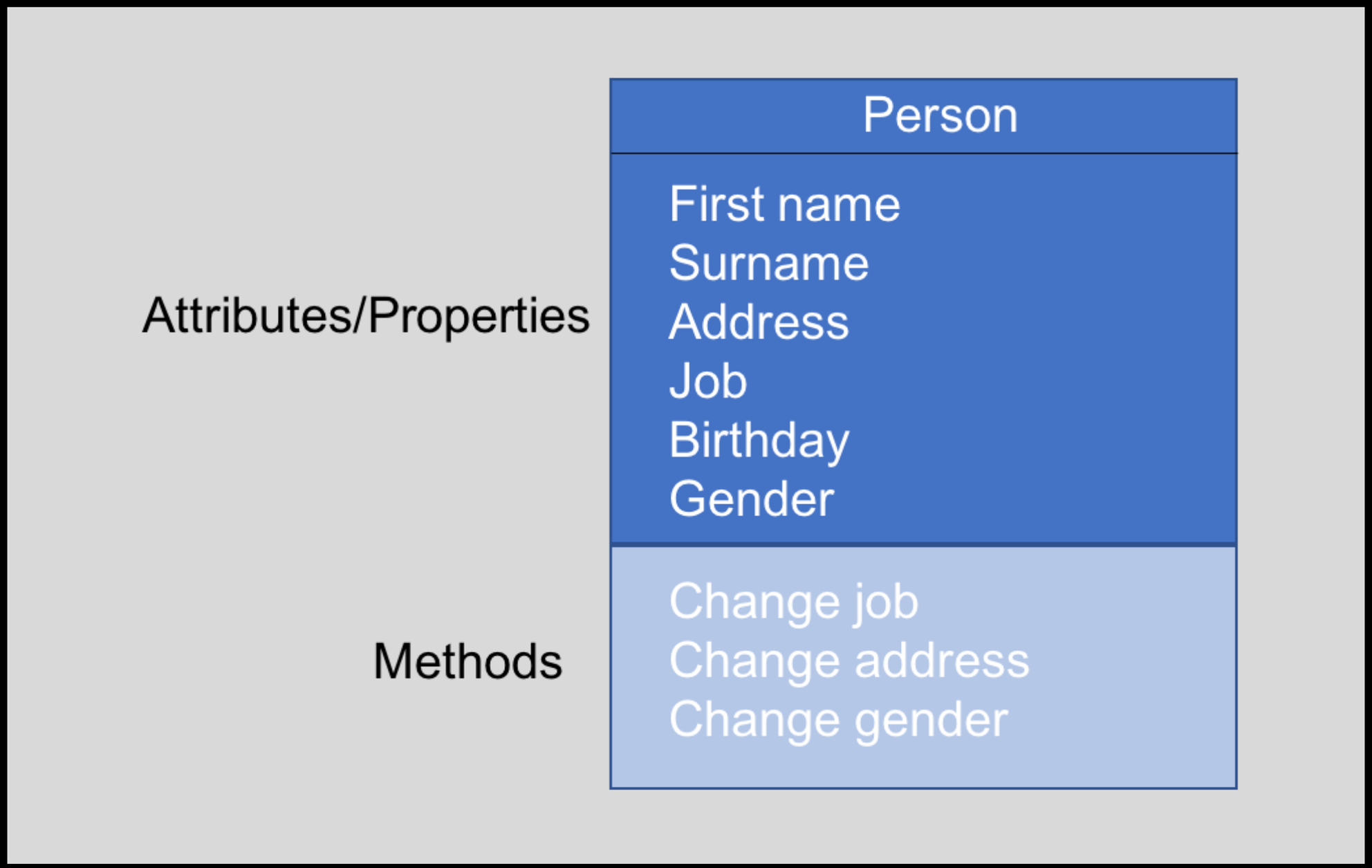
Classes

- A class defines a data type.
- This type contains variables, properties and methods that describe *something*.
- The set of values of an object's attributes is called its *state*.
- An object comprises both state and the behaviour defined by a object's class.

Instances and objects

- Terms are used interchangeably.
- An instance is an object of a class created while a program is running (run-time).

Example class



Encapsulation

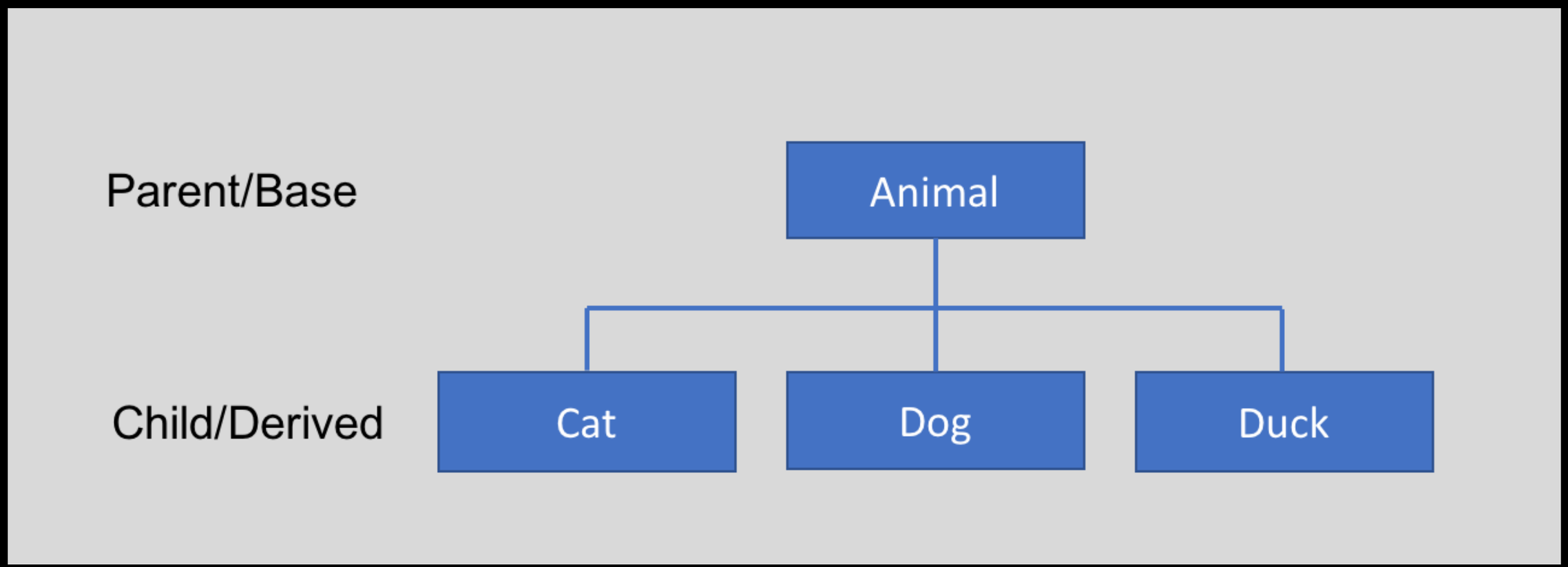
- In general terms, encapsulation is the restricting of access to the components of an object.
- The stuff inside the object can't be seen from the outside.
- Classically achieved with *getters* and *setters* to control access to internal data.
- Can (should) prevent an object being put into an incorrect state.

Data abstraction

- *Abstraction* is the process of hiding all but the relevant information about a given object.
  - Aims to reduce complexity.
- *Encapsulation* focuses on the how to hide the implemenation details.
  - This can be achieved through the wrapping/ hiding of data/methods (*Information hiding*).

Inheritance

- Classes can be extended from one or more other classes.
- The *child* class inherits attributes and methods from the *parent* class.
- The child is then customised for its particular purpose.



Polymorphism

- Polymorphism is an important feature that facilitates the use of different types from a single interface.
- Types of polymorphism:
  - *Sub-typing*: Instances of different classes related by a common parent class. This is what the term "polymorphism" normally refers to.
  - *Ad hoc*: Polymorphic functions behave differently depending on argument types. *Function overloading*
  - *Parametric*: Write code without stating specific types. The same code can be used for different types. A.K.A *generics* or *generic programming*.

Abstract classes

- An abstract class defines methods that a child must override.
- Parent names the method, but the child provides the implementation.

Suggestions

- Typically you should not need more than three levels of inheritance.
- The choice to use OOP depends on application.
- Try working out a design of hierarchies, methods etc. before implementing it.

Summary

- Key concepts in OOP
  - Encapsulation
  - Data abstraction
  - Inheritance
  - Polymorphism