Introduction to Programming with Python

Part 5

Objects and Classes
We already heard that there are different data types in Python - e.g. strings, lists, dictionaries, tuples, files, etc.

Each of those data types is defined by a class: the class of strings, the class of files, etc.
  - The class determines what you can and cannot do with data of this type.

Any particular variable you create is an instance of a class.
What is a class?

- We already heard that there are different data types in Python - e.g. strings, lists, dictionaries, tuples, files, etc.

- Each of those data types is defined by a class: the class of strings, the class of files, etc.
  - The class determines what you can and cannot do with data of this type.

- Any particular variable you create is an instance of a class.

Analogy:
- class of strings — a particular string (e.g. `a = 'Hello'`)
- humans — a particular person (e.g. Jon)
- chairs — your chair in your office
Recall the mysterious phrase ’Everything in Python is an object’.

- An *object* is an instance of a class.
- It has a name, a value and a place in memory.
  - E.g. the string `a = 'Hello'`, once you ’declare’ it (create it), is an object, with name `a` and value ’Hello’, and is an instance of the class of strings.
    - (note: objects in Python cannot exist without a name - as opposed to humans and chairs.)
One object might be referred to by two different names. For example, try:

```python
a = 'hello'
b = 'hello'
a is b
```

On the other hand, like Jon might have an identical twin brother, two different objects might have the same value in Python.

```python
a = 'hello'
b = 'he'
c = 'llo'
d = b+c
a == d
a is d
```
Object-oriented programming
Defining your own classes

Just like we can define our own functions next to Python’s built-in functions, we can also define our own classes, next to Python’s existing classes.

- When you define a class, you specify the type of data, possible properties (attributes) and which kind of things you can do with it (methods).

- For example, if you want to simulate evolution, you may want to define a new class ‘DNA sequence’: a string that can only have elements ‘C’, ‘G’, ‘A’ and ‘T’, which can make a copy of itself, with some elements changed, etc...
Object-oriented programming
Defining your own classes: sub- and superclasses

- Classes can be defined as *subclasses* of other classes.
  - For example, when making a database of people working at IST:
    Superclass: 'employee', subclasses: 'scientist' and 'support'
    Subclasses of 'scientist': 'student', 'postdoc' and 'professor'

- Subclasses *inherit* the methods and properties from their superclass.
  - All employees have a salary (attribute), and all employees can get a salary raise (method).
  - Scientists are member of a group (attribute)
  - Unique attribute of class 'students': before qual / after qual
  - Unique attribute of class 'professors': tenure / no tenure
You can define your own classes as subclasses of built-in Python classes.

- For example, you can make your own subclass of strings, 'DNAsequence'.
- A derived class can override any methods of its base class or classes

You can redefine built-in operators (e.g. +) to have a new functionality for your own classes.