PROGRAMMING IN PYTHON I

Unit 10: (Introduction to) Classes in Python

Michael Widrich
Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

CLASSES IN PYTHON

Motivation

- Often, we want reusability and modularity of our code
  - Easier software design
  - Easier data modeling
- Object-oriented programming (OOP) tries to increase reusability and modularity
  - Uses objects, which can contain data and code
- Class-based programming or class-orientation is a style of OOP
  - Uses classes as templates for creating objects

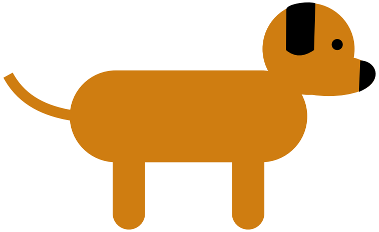
Object-oriented programming

- Object-oriented programming (OOP)
  - Programming paradigm based on the concept of objects
  - We will not go into details on the OOP paradigm in this course
- Objects (in OOP)
  - Combination of variables, functions, and data structures
  - Can contain state (data) and behaviors (procedures)
- Attributes (aka properties)
  - State (data) associated with an object
- Methods
  - Behaviors (procedures) provided by an object

Further reading:
https://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concep,
https://en.wikipedia.org/wiki/Object-oriented_programming

Objects: Example (1)

- Example: We want to create an object to describe a dog named "Bello"



Objects: Example (2)

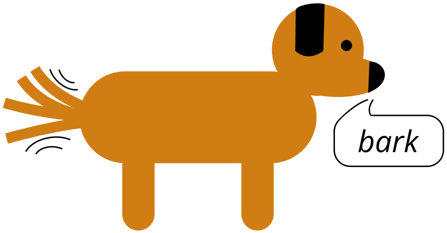
- Our dog-object can have attributes that hold values describing the name and fur color

Attributes
Name: "Bello"
Fur color: "brown"

Objects: Example (3)

- Our dog-object can have procedures that execute wiggling of its tail and barking

Methods



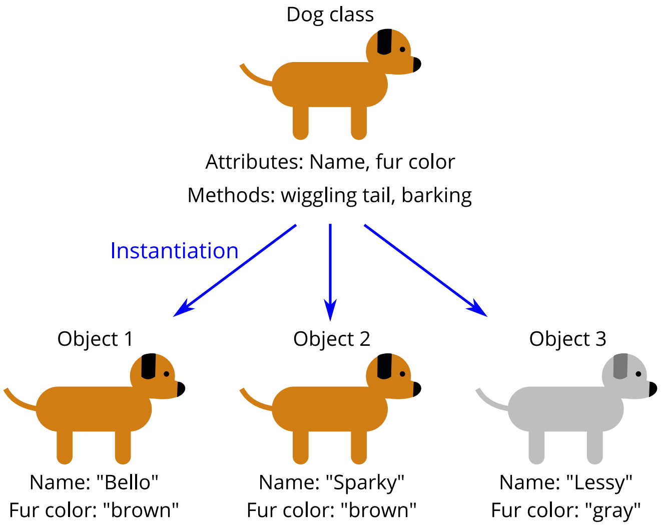
Wiggling tail
Barking

Class-based programming (1)

- Class-based programming
  - Uses classes to define objects (like a blue-print)
  - An object is an instance of a class that exists uniquely among all other objects
- Example: Assume we want to describe multiple dogs
  - We would first create a dog-class
  - The dog-class would contain attributes and methods that are used to describe a dog
  - If we want to describe an individual dog, we create an instance of our dog-class
  - Each instance is an individual object and contains a copy of the attributes and methods from our dog-class

  → We don’t have to re-write the code for a dog object for every dog!

Class-based programming (2)



Attributes: Name, fur color

Methods: wiggling tail, barking

Object 1: Name: "Bello", Fur color: "brown"

Object 2: Name: "Sparky", Fur color: "brown"

Object 3: Name: "Lessy", Fur color: "gray"

Class-based programming (3)

- We can also create (derive) new classes from existing classes
  - The new classes are referred to as child classes or sub classes
  - The classes the sub classes are derived from are referred to as parent classes, base classes, or super classes
- Sub classes can inherit attribute and method definitions from their base classes
  - Attributes/methods from parent classes are available in child classes but can be modified/extended

Class-based programming (4)

- Example: Assume we now to describe guard dogs that behave like our dog-class but also have a "guard" method
  - We can derive a guard-dog-class from our dog-class, which inherits the attribute and method definitions from the dog-class
  - We can add an additional "guard" method to our guard-dog-class
  - We can now create instances of our guard-dog-class

Classes in Python (1)

- Every object in Python is (indirectly) derived from the base class object
- We have already worked with classes in Python!
  - Example: Our integer objects are instances of the `int` class, which is derived from the object class
- Classes can be created using the class statement
  - Class names (by convention) should be CamelCase
  - Example: `class MyDog: ...` creates a class `MyDog`
- Similarly to functions, classes create a namespace
  - Variables that are created within the class or instance and are not attributes or returned via methods, only exist within the class or instance

Classes in Python (2)

- We can create an instance of a class by calling the class
  - Example: `my_dog = MyDog()` creates the instance `my_dog` from class `MyDog`
- Attributes and methods can be accessed using a dot `.`
  - Example: `MyDog.bark()` would call the `bark()` method of `MyDog`
- Python modules behave similar to classes
  - Example: `numpy.array()` would call the `array()` method of the module `numpy`

Classes in Python

- There is much more to classes is Python, which will not be covered in this course
  - Further reading:
    `https://docs.python.org/3/tutorial/classes.html`
- There is a whole world to object oriented programming
  - Further reading:
    `https://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concep`
    `https://en.wikipedia.org/wiki/Object-oriented_programming`

…we will not go into this further and instead jump to the Python code file for Unit 10