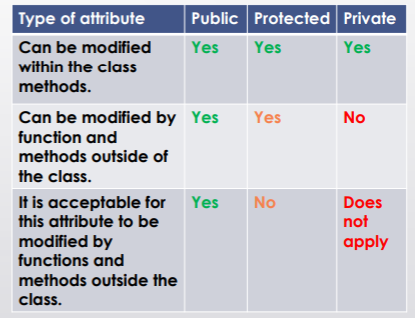
**Object-oriented Programming**

* Objects:
  + Contain both data (state) and functionality (method)
  + State: things that the object knows about itself (attributes)
  + Methods: functions that can be executed by the object
  + object instantiation: robot = Thymio()
* Constructor: functions that create a new object instance (every class automatically uses name of class as name of constructor function)
  + def \_\_init\_\_(self, name, hero\_class, max\_lifepoints = 100):   
    define inputs to assign default values in our methods  
    these attributes have to be given when creating instance
* Types of Attributes: “what an object has”
  + Private: \_\_name double underscore
  + Protected: \_name single underscore
  + 
* Method:
  + A function that is called to act on a specific instance, “what an object can do”
  + Accessed using dot notation + ()
  + self refers to the object that the function is invoked upon
  + All methods defined in a class that operate on objects of that class will have self as their first parameter (to serve as reference to the object itself)
  + Definition parameters = Invocation parameters (self is implicit) + 1 (self)
  + Helper function: function used by a class to assist in a task, but does not belong to the class
  + Mutator: mutates or changes the internal state of the object
* Special Methods
  + \_\_dict\_\_ :d displays attributes and values of the object as dictionary
  + \_\_eq\_\_ : this special method is called when A == B is used, where A and B are 2 objects of our custom class
  + \_\_call\_\_ : defines what happens when you call your custom object as a function and pass it some arguments
  + \_\_str\_\_ : defines what happens when you attempt to convert your custom object into a string type object
* Properties: similar to attributes, protected by methods like setter and getter.
  + current\_lifepoints = property(get\_current\_lifepoints, set\_current\_lifepoints)
  + Automatically executes the getter method for my\_hero.current\_lifepoints
  + Similarly for setter method
* Setters & Getters
  + Getters: fetches the current value stored in an attribute
  + Setters: rewrites the value set in an attribute
* Return instance example:
  + def halfway(self,target):  
     mx = (self.x + target.x)/2  
     my = (self.y + target.y)/2  
     return Point(mx,my)
* Equality
  + Shallow: == operator, compares the references (True for aliases)
  + Deep: compares the values
  + If 2 variables refer to same object, they have both shallow & deep equality
* Docstrings: strings proceeding class definition

**Inheritance**

* Hero object <<has-a>> weapon object
* Inheritance: create a class which reuses methods and attributes from another class
  + Subclass is-a parent class
  + class Sword is-a class Weapon
  + class Staff is-a class Weapon