Cheat sheet: classes

Concept	How to define it	How to use it	Notes

Constructor	`class MyClass:`
	`  def __init__(self):`
	`  ...`
		`my_obj = MyClass()`	Called when creating an instance of a class.

Instance attribute	`class MyClass:`
	`  def __init__(self):`
	`  self.my_attribute = ...`
		`my_obj.my_attribute`	A variable that is bound to an object. Its value can be different across instances.

Class attribute	`class MyClass:` `  my_attribute = ...`	`MyClass.my_attribute` `my_obj.my_attribute`	A variable that is bound to a class. Its value is the same across all instances.

Instance method	`class MyClass:`
	`  def my_method(self):`
	`  ...`
		`my_obj.my_method()`	A function that is bound to an object. Its implementation has access to the instance (`self`).

Class method	`class MyClass:`
	`  @classmethod`
	`  def my_method(cls):`
	`  ...`
		`MyClass.my_method()`
		`my_obj.my_method()`
			A function that is bound to a class.

Static method	`class MyClass:`
	`  @staticmethod`
	`  def my_method():`
	`  ...`
		`MyClass.my_method()`
		`my_obj.my_method()`
			A function that is bound to a class. Its implementation does not have access to the class or the instance.

Special method	`class MyClass:`
	`  def __getitem__(self, idx):`
	`  ...`
		`my_obj[42]`	Reserved method name for implementing syntactic sugar.
			List of all special method names.

“Private” attribute/method	`class MyClass:`
	`  def _my_method(self):`
	`  self._my_attribute = ...`
		`self._my_method()`
		`self._my_attribute`
			Indicates that the attribute/method should only be used from inside the class and is not part of the public interface.

Cheat sheet: inheritance

Concept	**Example**	**Notes**

Superclass / parent class	`class Parent:`
	`  def my_method(self):`
	`  ...`	

Subclass / child class	**`class Child(Parent):`** `  pass`	Inherits methods from parent class.

`super()`	`class Child(Parent):`
	`  def my_method(self):`
	`  **super().my_method()**`
	`  ...`
		Delegates method calls to the parent class.

Abstract class
Abstract method
	`from abc import ABC, abstractmethod`
	
	`class AbstractParent(`**`ABC`**`):`
	`  `**`@abstractmethod`**
	`  def my_abstract_method(self):`
	`  pass`
		Cannot be instantiated (calling `AbstractParent()` raises an error).
		Requires child classes to implement all abstract methods.