As you already know, **object-oriented programming (OOP)** is a programming paradigm that is based on the concept of objects. Objects interact with each other and that is how the program functions. Objects have states and behaviors. Many objects can have similar characteristics and if you need to work with them in your program it may be easier to create a **class**for similar objects. Classes represent the common structure of similar objects: their data and their behavior.

**Declaring classes**

Classes are declared with the keyword class and the name of the class:

# class syntax

class MyClass:

var = ... # some variable

def do\_smt(self):

# some method

Generally, a class name starts with a capital letter and it is usually a noun or a noun phrase. The names of the classes follow the **CapWords**convention: meaning that if it's a phrase, all words in the phrase are capitalized and written without underscores between them.

# good class name

class MyClass:

...

# not so good class name:

class My\_class:

...

Classes allow you to define the data and the behaviors of similar objects. The behavior is described by **methods.**A method is similar to a function in that it is a block of code that has a name and performs a certain action. Methods, however, are not independent since they are defined within a class. Data within classes are stored in the **attributes** (variables) and there are two kinds of them: [class attributes](https://hyperskill.org/learn/step/6661) and [instance attributes](https://hyperskill.org/learn/step/6661). The difference between those two will be explained below.

**Class attribute**

A class attribute is an attribute shared by all instances of the class. Let's consider the class **Book**as an example:

# Book class

class Book:

material = "paper"

cover = "paperback"

all\_books = []

This class has a string variable material with the value "paper", a string variable cover with the value "paperback" and an empty list as an attribute all\_books. All those variables are class attributes and they can be accessed using the [dot notation](https://hyperskill.org/learn/step/6661)with the name of the class:

Book.material # "paper"

Book.cover # "paperback"

Book.all\_books # []

**Class attributes** are defined within the class but outside of any methods. Their value is the same for all instances of that class so you could consider them as the sort of "default" values for all objects.

As for the **instance variables,**they store the data unique to each object of the class. They are defined within the class methods, notably, within the \_\_init\_\_ method. In this topic, we'll deal with the class attributes, but don't worry – you'll have plenty of time to learn more about instance attributes.

**Class instance**

Now, let's create an instance of a **Book**class. For that we would need to execute this code:

# Book instance

my\_book = Book()

Here we not only created an instance of a **Book**class but also assigned it to the variable **my\_book**. The syntax of instantiating a class object resembles a [function call](https://hyperskill.org/learn/step/6661): after the class name, we write parentheses.

Our my\_book object has access to the contents of the class **Book**: its attributes and methods.

print(my\_book.material) # "paper"

print(my\_book.cover) # "paperback"

print(my\_book.all\_books) # []

**Conclusion**

Well, those were the very basics of classes in Python. Classes represent the common structure of similar objects, their attributes, and their methods. There are class attributes and instance attributes. Class attributes are common to all instances of the class.

All in all, classes are an extremely useful tool that can help you optimize your code and organize the program in a logical and readable way. We hope you'll make use of them!

class Angel:

color = "white"

feature = "wings"

home = "Heaven"

class Demon:

color = "red"

feature = "horns"

home = "Hell"

Angel=Angel()

Demon=Demon()

print(Angel.color)

print(Angel.feature)

print(Angel.home)

print(Demon.color)

print(Demon.feature)

print(Demon.home)

Are methods in Python dependent or independent compared to functions?

Dependent

Imagine you have created the Wild\_Animal class. It works fine but it can be improved. Look at the first line. How can we change the name in the first line according to PEP8? Print the name (only the name itself) in the box below.

class Wild\_Animal:

get\_food\_itself = "yes"

live\_in\_nature = "yes"

live\_with\_man = "no"

According to **PEP8**, class names should follow the **CapWords** (CamelCase) convention, without underscores.

### ✅ Corrected class name:

WildAnimal

The code below defines the class **House** and an object of that class. However, the code is somewhat incorrect. Fix the mistakes but do NOT change variable or attribute names (they are necessary for checking).

class House:

construction = "building"

elevator = True

def \_\_init\_\_(self, contruction):

construction = contruction

# object of the class House

new\_house = House("building")

class House():

def \_\_init\_\_(self, construction="building", elevator=True):

self.construction = construction

self.elevator = elevator

new\_house = House("building")

## What is a class?

A class represents common structures of similar objects (behavior and data).

