Introduction to Python

Session 07: Block 2. Part I

Introduction to Object Oriented Programming (OOP)

Classes
Objects

Functional/Structured programming

- **Variables**
- **Functions**
    - **Group of statements**: easier to read and debug.
    - Smaller programs by eliminating **repetitive code.**
    - Divide a long program into functions. Decomposition of a problem into smaller sub-problems and assemble them in a workflow script.
    - **Reuse** of the same functions in several programs.
- **Modules/Libraries** that are collections of variables and functions.

upf.

Object Oriented Programming

Attributes: Variables that define the **state** of the object

Methods: Define the **behavior** of the object.
A method is a function associated to an object.

Concept in the real world.

Class vs **Object Instance**

upf.

Object Oriented Programming

Class: Defines the structure: attributes and methods

Student
Name
Surname
Identification Number
Birth date

Instances: Specific realization of any class. Exist in a given program execution

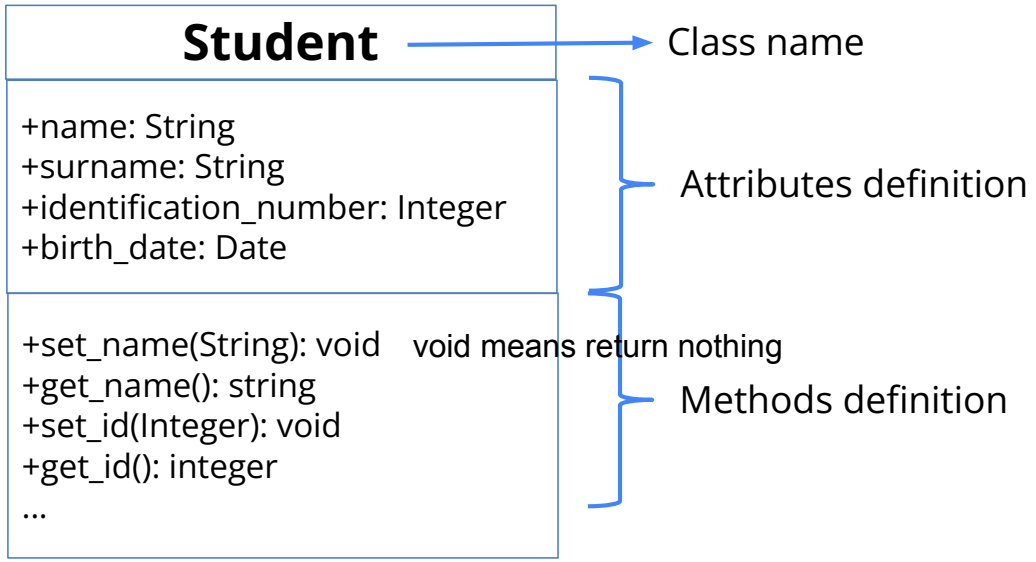
Name: Antonio
Surname: Gómez
Identification Number: 1234
Birth date: 1/1/1990

Name: Alba
Surname: González
Identification Number: 3456
Birth date: 1/1/1992

Name: Agapito
Surname: Garcia
Identification Number: 2827
Birth date: 21/10/1992

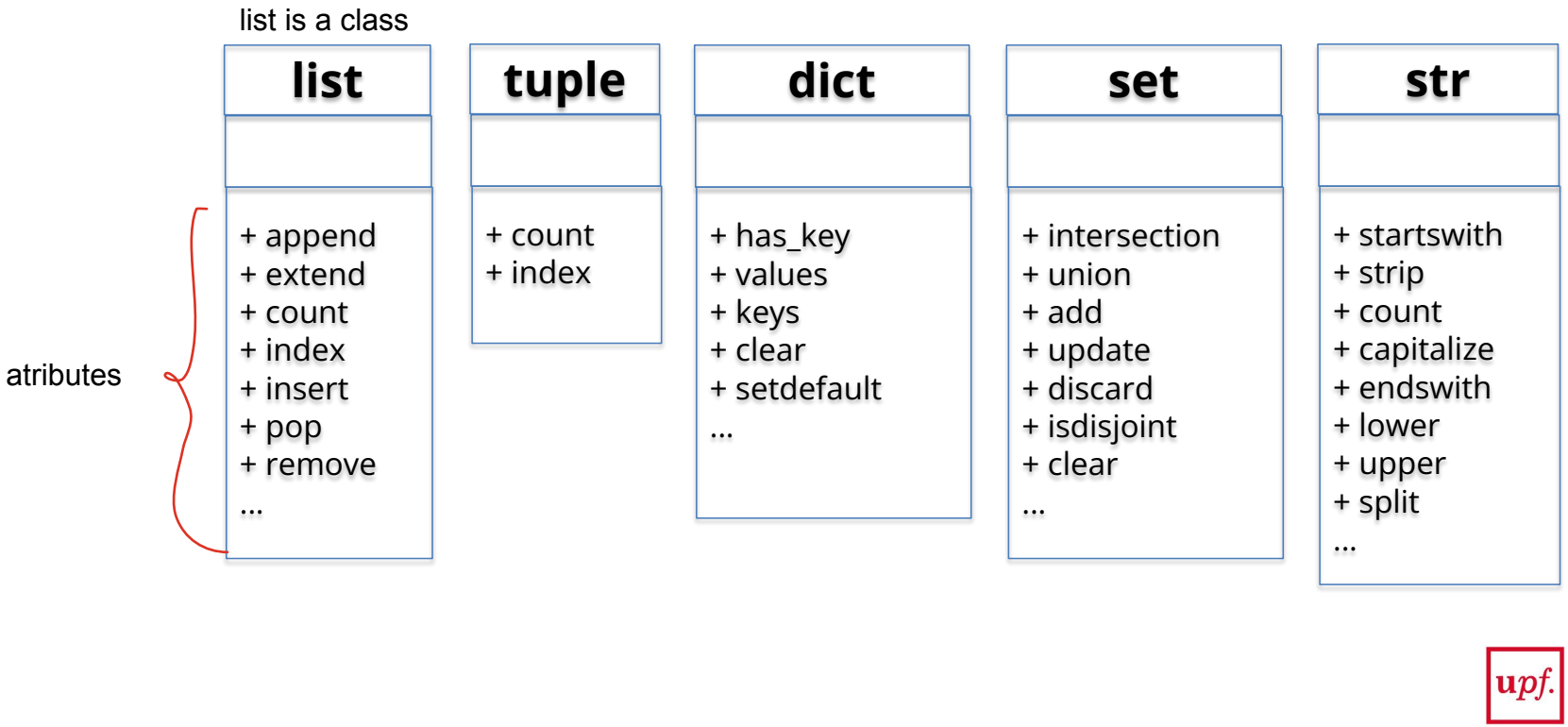
upf.

Class representation: UML diagram



To call the method of an object: `object_name.method()`

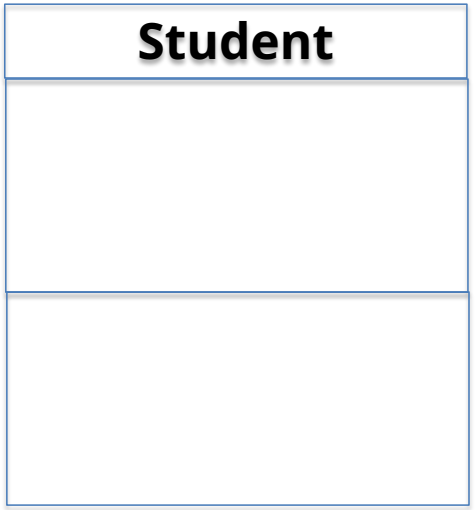
In Python, **everything** is an object !



Defining new classes

create an object

`class`: Defines a new class of objects

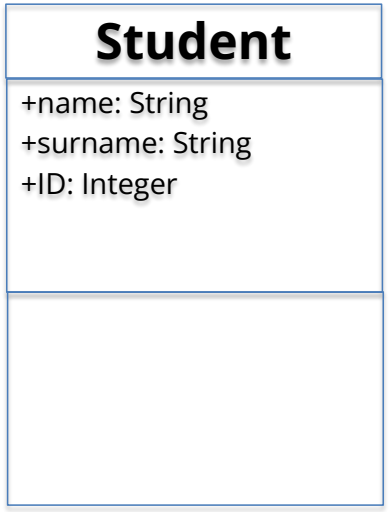



```
class Student:
    pass
```

```
student_instance1 = Student()
print(student_instance1)
<__main__.Student object at 0x106ae3850>
```

Defining new classes

`__init__`: method automatically called after the object is created to **initialize** the instance:



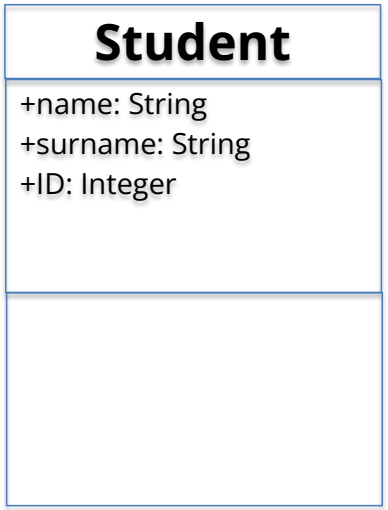

```
class Student():

    def __init__(self, name, surname, NIE):

        self.name = name
        self.surname = surname
        self.NIE = NIE
```

the underscore options are not for user purpose, unless we kknow what we are doing

Defining new classes

self: represents the current instance of the class inside the class definition.

- First parameter in methods (automatically passed when called)
- Access object attributes

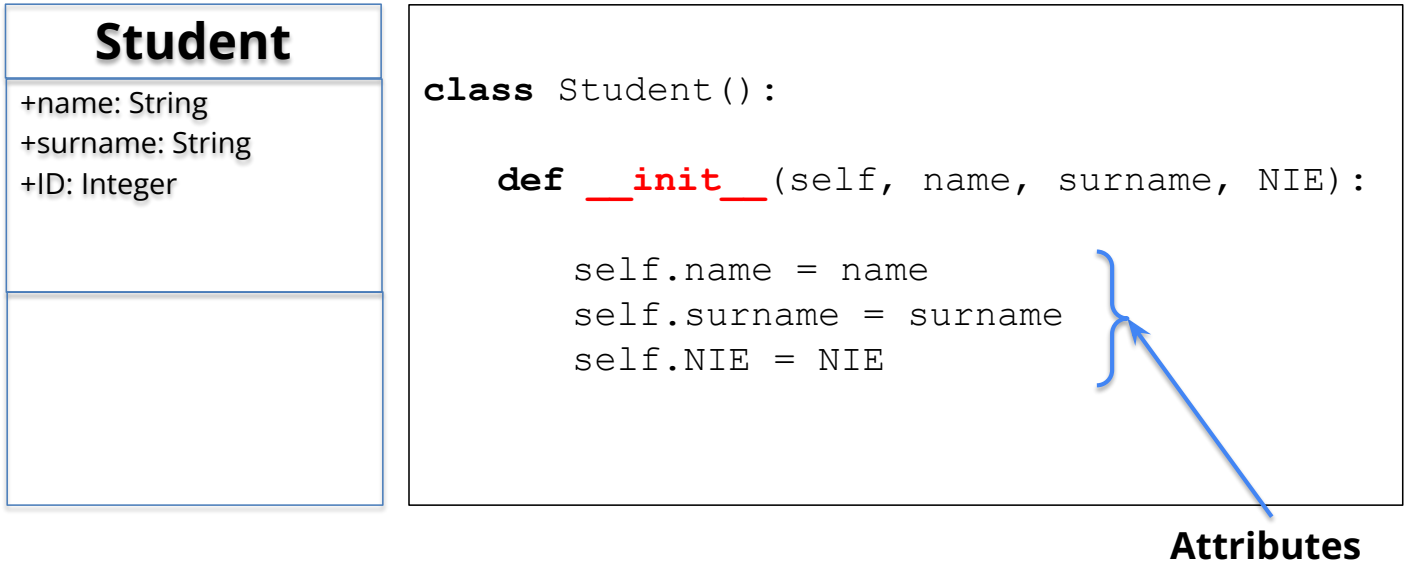



```
class Student():

    def __init__(self):
        print(id(self))

s = Student()
print(id(s))
```

upf.

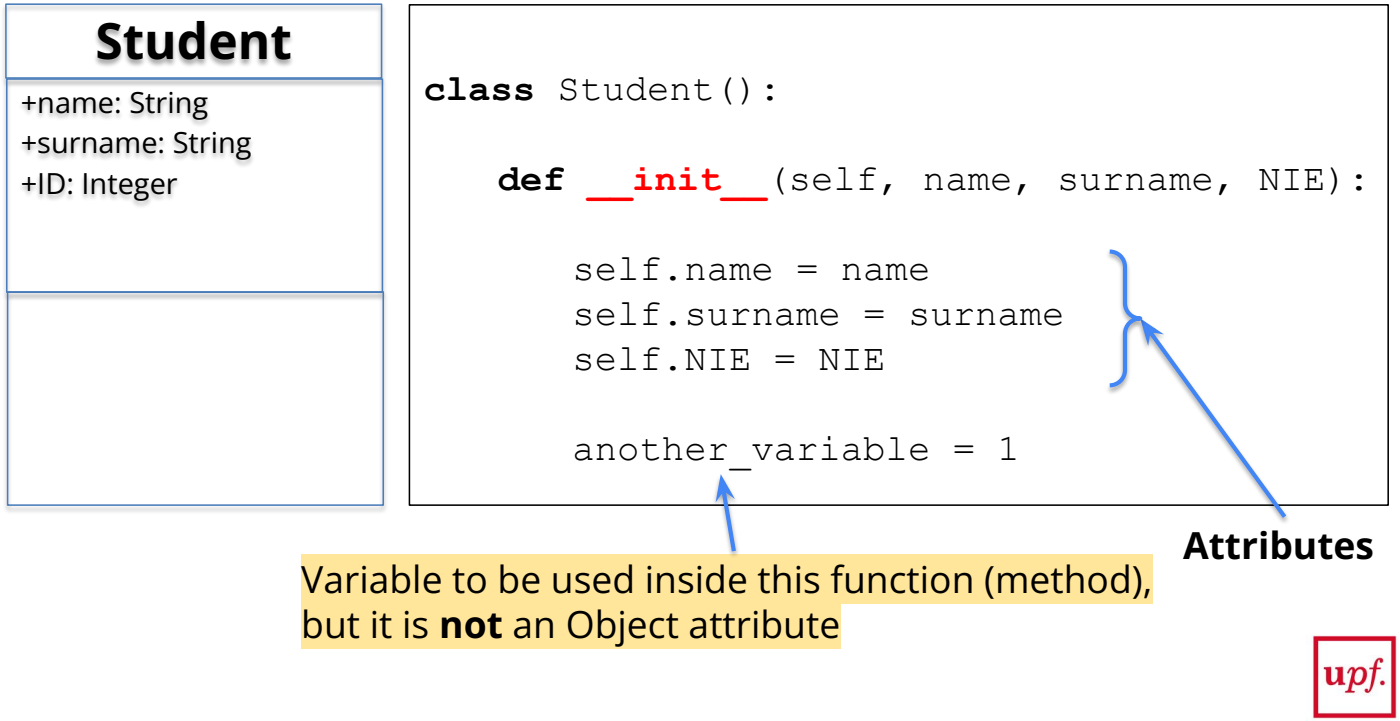
Defining new classes

Attributes: Variables that define the state of the object: **`self.attribute_name`**



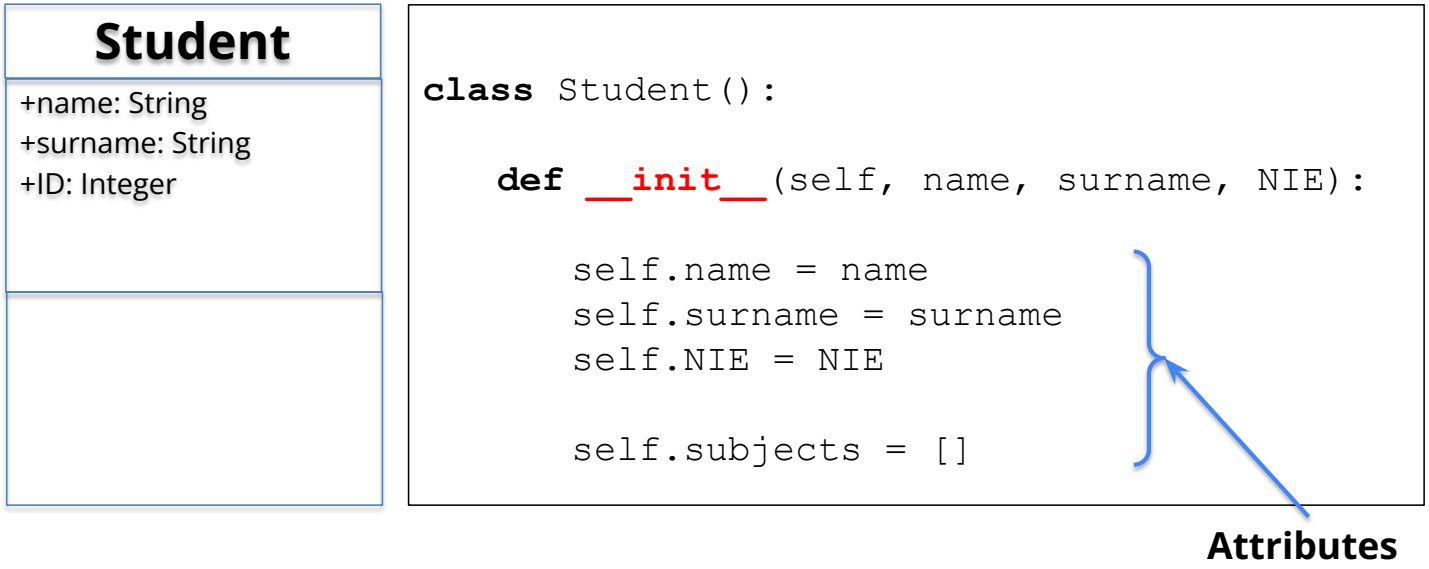
Defining new classes

Attributes: Variables that define the state of the object: **`self.attribute_name`**



Defining new classes

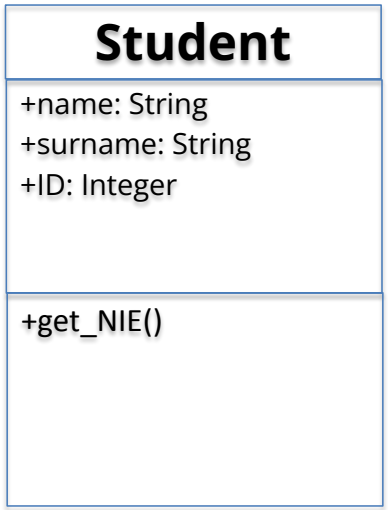
Attributes: Variables that define the state of the object: **`self.attribute_name`**



Defining new classes

Methods: Function associated to an object that defines the **behavior** of the object:

```
def method_name(self, ...)
```



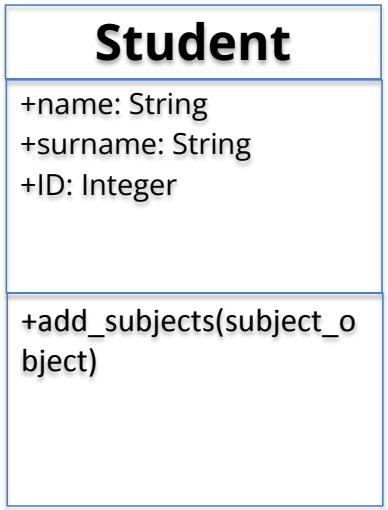

```
class Student():

    def __init__(self, name, surname, NIE):

        self.name = name
        self.surname = surname
        self.NIE = NIE
        self.subjects = []


    def get_NIE(self):
        return self.NIE
```

upf.

Defining new classes

Methods: Function associated to an object that defines the **behavior** of the object:

`def method_name(self, ...)` every method inside a class must use a self



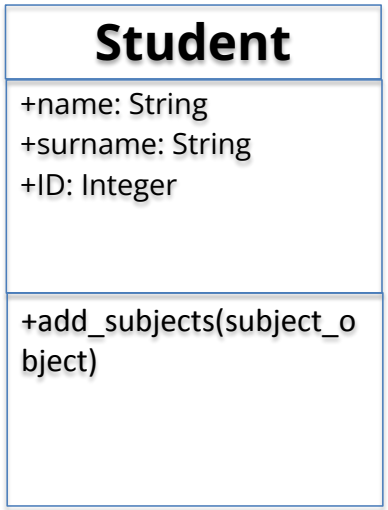

```
class Student():

    def __init__(self, name, surname, NIE):

        self.name = name
        self.surname = surname
        self.NIE = NIE
        self.subjects = []


    def add_subject(self, subject_object):
        self.subjects.append(subject_object)
```

upf.

Defining new classes

Methods: Function associated to an object that defines the **behavior** of the object:

```
def method_name(self, …)
```



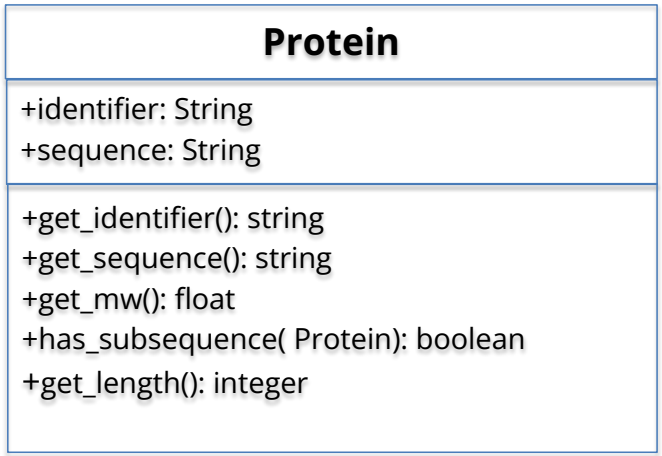

```
s1 = Student(name="Xavi",
surname="Jalencas", NIE=1234)
s1.get_NIE()
```

upf.

Exercises. Session 07

Create a python script called **<uID>_S07.py** with the following:

1) Define a new class named Protein, with the following definition. If necessary, you can define the additional methods you need



2) Modify the FASTA_iterator generator function to yield Protein objects instead of tuples. In each iteration, the function must yield a Protein Object:

```
FASTA_iterator(fasta_filename)
```

upf.