CONQUER BLOCKS

PYTHON

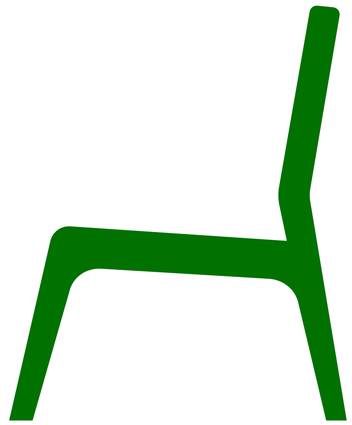
PROGRAMACION ORIENTEADA A OBJETOS (POO)

OBJETOS Y CLASES

CLASES → **Representaciones del mundo real**

Ejemplo:

SILLA

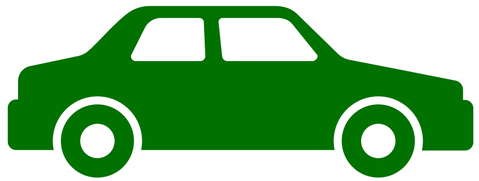


OBJETOS Y CLASES

CLASES → **Representaciones del mundo real**

Ejemplo:

COCHE

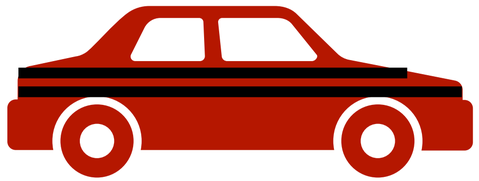


OBJETOS Y CLASES

OBJETOS → **Una instancia de esa clase**

Ejemplo:

MI COCHE



FERRARI

OBJETOS Y CLASES

OBJETO

Atributos

Datos

Motor

Puertas

Métodos

Funcionalidades

Acelerar

Girar

Tocar el claxon

OBJETOS Y CLASES

OBJETO

Atributos

Datos

- Motor
- Puertas

Métodos

Funcionalidades

- Ajustar la velocidad
- Girar
- Tocar el claxon

OBJETOS Y CLASES

EJEMPLO:

```
class Fruta:
    def __init__(self):
        self.nombre = "manzana"
        self.color = "rojo"
```

OBJETOS Y CLASES

EJEMPLO:

```
class Fruta:
    def __init__(self):
        self.nombre = "manzana"
        self.color = "rojo"
```

OBJETOS Y CLASES

EJEMPLO:

```
        nombre
class Fruta:
    def __init__(self):
        self.nombre = "manzana"
        self.color = "rojo"
```

OBJETOS Y CLASES

EJEMPLO:

```
class Fruta:
    def __init__(self):
        self.nombre = "manzana"
        self.color = "rojo"
```

función de inicialización

OBJETOS Y CLASES

EJEMPLO:

OBJETOS Y CLASES

EJEMPLO:

```
class Fruta:
    def __init__(self):
        self.nombre = "manzana"
        self.color = "rojo"

mi_fruta = Fruta()
```

PODEMOS ASIGNAR NUESTRA CLASE A UNA VARIABLE

OBJETOS Y CLASES

EJEMPLO:

```
class Fruta:
    def __init__(s
        self.nombr
        self.color

mi_fruta = Fruta()
```

```
print(mi_fruta.color)
print(mi_fruta.nombre)
```

✓ 0.0s

```
rojo
manzana
```

PODEMOS ACCEDER A LOS ATRIBUTOS

OBJETOS Y CLASES

EJEMPLO:

```
class Fruta:
    def __init__(s
        self.nombr
        self.color

mi_fruta = Fruta()
```

```
mi_fruta.color = "verde"
mi_fruta.nombre = "pera"
print(mi_fruta.color)
print(mi_fruta.nombre)
```

✓ 0.0s

```
verde
pera
```

REASIGNAR VALOR DE ATRIBUTOS

OBJETOS Y CLA

EJEMPLO:

```
class Fruta:
    def __init_
        self.no
        self.co

mi_fruta = Frut
```

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color

mi_fruta = Fruta("manzana", "rojo")
print(mi_fruta.color)
print(mi_fruta.nombre)
```

✓ 0.0s

```
rojo
manzana
```

OBJETOS Y CLA

EJEMPLO:

```
class Fruta:
    def __init_
        self.no
        self.co

mi_fruta = Frut
```

Parámetros

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color

mi_fruta = Fruta("manzana", "rojo")
print(mi_fruta.color)
print(mi_fruta.nombre)
```

✓ 0.0s

```
rojo
manzana
```

OBJETOS Y CLA

EJEMPLO:

```
class Fruta:
    def __init_
        self.no
        self.co

mi_fruta = Frut
```

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
mi_fruta = Fruta("manzana", "rojo")
print(mi_fruta.color)
print(mi_fruta.nombre)
```

Parámetros

Asignamos los parámetros a los atributos

✓ 0.0s

```
rojo
manzana
```

OBJETOS Y CLA

EJEMPLO:

```
class Fruta:
    def __init_
        self.no
        self.co

mi_fruta = Frut
```

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color

mi_fruta = Fruta("manzana", "rojo")
print(mi_fruta.color)
print(mi_fruta.nombre)
```

Argumentos

✓ 0.0s

```
rojo
manzana
```

OBJETOS Y CLASES

EJEMPLO:

```
manzana = Fruta("manzana", "rojo")
platano = Fruta("platano", "amarillo")
kiwi = Fruta("kiwi", "verde")
```

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
```

OBJETOS Y CLASES

EJEMPLO:

```
manzana = Fruta("manzana", "rojo")
platano = Fruta("platano", "amarillo")
                          verde")
```

```
print(platano.nombre, platano.color)
print(kiwi.nombre, kiwi.color)
print(manzana.nombre, manzana.color)
```

✓ 0.0s

```
platano amarillo
kiwi verde
manzana rojo
```

OBJETOS Y CLASES

EJEMPLO:

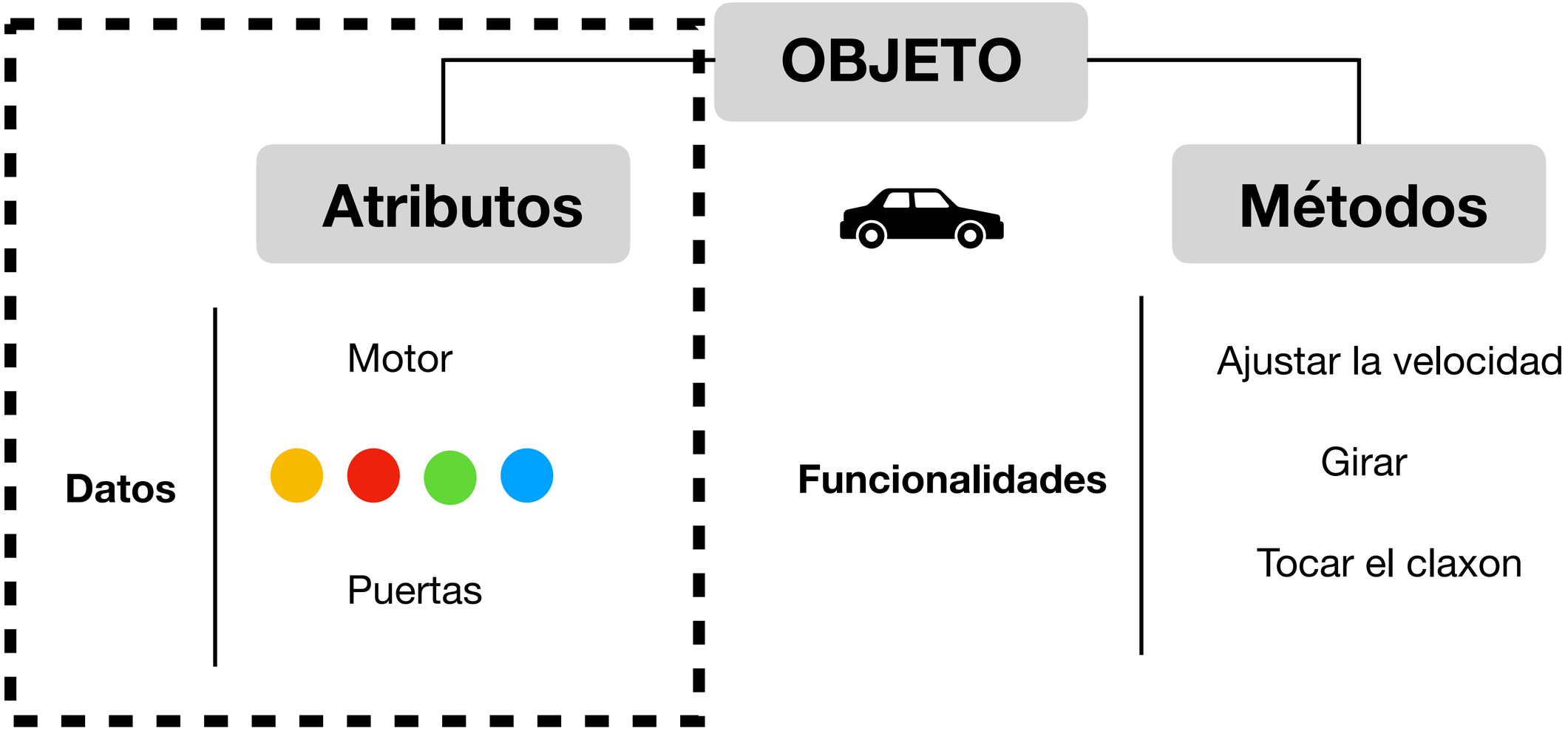
```
manzana = Fruta("manzana", "rojo")
platano = Fruta("platano", "amarillo")
kiwi = Fruta("kiwi", "verde")
```

INSTANCIAS / OBJETOS

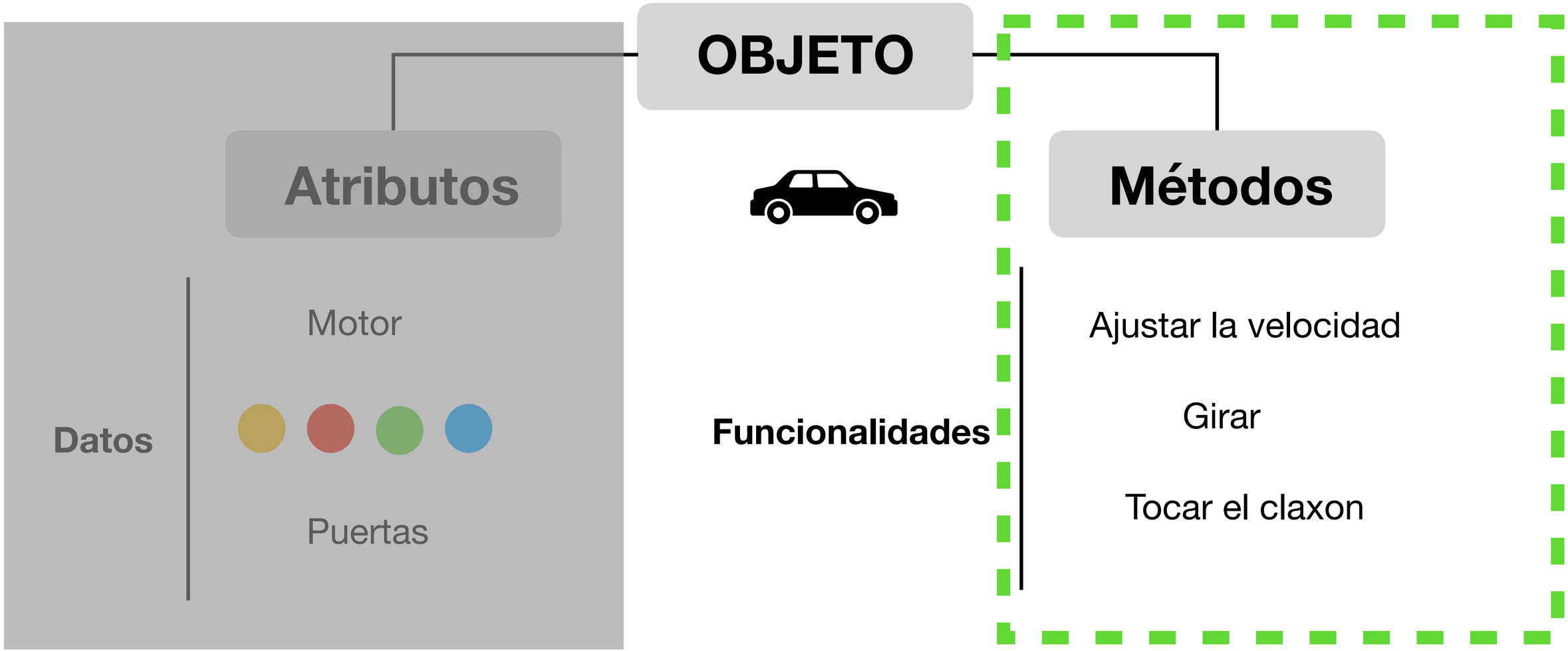
```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
```

CLASE

OBJETOS Y CLASES



OBJETOS Y CLASES



OBJETOS Y CLASES

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
    def details(self):
        print("mi " + self.nombre +
              " es " + self.color )

manzana = Fruta("manzana", "roja")
manzana.details()
```

✓ 0.0s

```
mi manzana es roja
```

Los métodos son funciones relacionadas con objetos

OBJETOS Y CLASES

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
    def details(self):
        print("mi " + self.nombre +
              " es " + self.color )
```

MÉTODO

Los métodos son funciones relacionadas con objetos

OBJETOS Y CLASES

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
    def details(self):
        print("mi " + self.nombre +
              " es " + self.color )
```

MÉTODO

Los métodos son funciones relacionadas con objetos

OBJETOS Y CLASES

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
    def details(self):
        print("mi " + self.nombre +
              " es " + self.color )
```

MÉTODOS

Los métodos son funciones relacionadas con objetos

OBJETOS Y CLASES

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
    def details(self):
        print("mi " + self.nombre +
              " es " + self.color )
```

AL PASAR *SELF* A TODOS LOS METODOS PODEMOS USAR LOS MISMOS ATRIBUTOS EN TODOS LOS MÉTODOS DE NUESTRA CLASE

OBJETOS Y CLASES

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
    def details(self):
        print("mi " + self.nombre +
              " es " + self.color )
```

A

B

AUNQUE AMBOS METODOS ESTAN HACIENDO COSAS COMPLETAMENTE DIFERENTES, PUEDEN ACCEDER A LOS MISMO ATRIBUTOS GRACIAS A *SELF*

OBJETOS Y CLASES

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
    def details(self):
        print("mi " + self.nombre +
              " es " + self.color )

manzana = Fruta("manzana", "roja")
manzana.details()
```

Llamada al método

✓ 0.0s

```
mi manzana es roja
```

OBJETOS Y CLASES

ERRORES COMUNES — No usar self para un atributo —

```
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
        caducidad = "01.03.2023"
    def details(self):
        print("mi " + self.nombre +
              " es " + self.color )
        print("caduca el ", caducidad)

manzana = Fruta("manzana", "roja")
manzana.details()
```

OBJETOS Y CLASES

ERRORES COMUNES — No usar self para un atributo —

```
class Fruta:
    def __init__(self, nomb
        self.nombre = nombr
        self.color = color
        caducidad = "01.03.
    def details(self):
        print("mi " + self.
              " es " + self
        print("caduca el ",

manzana = Fruta("manzana", "roja")
manzana.details()
```

```
mi manzana es roja

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 12
      9         print("caduca el ", caducidad)
     11 manzana = Fruta("manzana", "roja")
---> 12 manzana.details()

Cell In[12], line 9, in Fruta.details(self)
      6 def details(self):
      7     print("mi " + self.nombre +
      8           " es " + self.color )
----> 9     print("caduca el ", caducidad)

NameError: name 'caducidad' is not defined
```

OBJETOS Y CLASES

ERRORES COMUNES
— No usar self para un atributo —

```
class Fruta:
    def __init__(self, nomb
        print("mi " + self.
            " es " + self
        print("caduca el ",

manzana = Fruta("manzana", "roja")
manzana.details()
```

```
mi manzana es roja

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 12
      8             " es " + self.color )
----> 9     print("caduca el ", caducidad)

NameError: name 'caducidad' is not defined
```

Caducidad es una variable local de __init__. Al no ser globar el metodo details no sabe de su existencia

OBJETOS Y CLASES

```python
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
        self.caducidad = "01.03.2023"
    def details(self):
        print("mi " + self.nombre +
              " es " + self.color )
        print("caduca el ", self.caducidad)

manzana = Fruta("manzana", "roja")
manzana.details()
```

✓ 0.0s

```
mi manzana es roja
caduca el  01.03.2023
```

Convertir *caducidad* en un atributo

— SOLUCIÓN —

```python
class Fruta:
    def __init__(self, nombre, color):
        self.nombre = nombre
        self.color = color
    def details(self):
        caducidad = "01.03.2023"
        print("mi " + self.nombre +
              " es " + self.color )
        print("caduca el ", caducidad)

manzana = Fruta("manzana", "roja")
manzana.details()
```

✓ 0.0s

```
mi manzana es roja
caduca el  01.03.2023
```

Definir *caducidad* dentro del metodo adecuado

MÉTODO __INIT__

```
class mi_clase:
    def __init__(self):
        self.attrib = []
```

- Es ejecutado automáticamente con cada nueva instancia de una clase (No necesitamos llamarlo como al resto de métodos)
- Es donde inicializamos los atributos
- Tiene el nombre reservado

MÉTODO __INIT__

```
class mi_clase:
    def __init__(self):
        self.attrib = []
```

```
class Fruta:
    def __init__(self, clr):
        self.color = clr

manzana = Fruta("roja")
```

self = el objeto *manzana*

self = the *apple* object its*elf*

MÉTODO __INIT__

```python
class mi_clase:
    def __init__(self):
        self.attrib = []
```

```python
class Fruta:
    def __init__(self, clr):
        self.color = clr

platano = Fruta("amarillo")
```

self = el objeto *plátano*

self = the *banana* object it*self*

EN OTROS LENGUAJES DE PROGRAMACIÓN...

Javascript

```javascript
function Fruta(nombre, clr) {
    this.nombre = nombre;
    this.color = clr;
}
```

- *this*
- *función constructora*

Python

```python
class Fruta:
    def __init__(self, nombre, clr):
        self.nombre = nombre
        self.color = clr
```

- *self*
- *class*

ESTO ES LO QUE CONFORMA LA PROGRAMACION ORIENTADA A OBJETOS

- SET DE PRINCIPIOS DE PROGRAMACIÓN

- TRABAJAMOS CON OBJETOS EN VEZ DE CON DATOS Y PROCEDIMIENTOS SEPARADOS

- UNIMOS DATA Y FUNCIONALIDAD EN UNA SOLA ESTRUCTURA CREANDO OBJETOS QUE INTERACTUAN ENTRE ELLOS

ESTO ES LO QUE CONFORMA LA PROGRAMACION ORIENTADA A OBJETOS

FUNCIONES	→	**MÉTODOS**
VARIABLES (CON PREFIJO *SELF*)	→	**ATRIBUTOS**
INSTANCIA DE UNA CLASE	→	**OBJETO**

CONQUER BLOCKS