Clase 3 Parte 3: Programación

Programación Orientada a Objetos (Continuación).

Temario

- Programación Orientada a Objetos en Python
  - Constructores
- Encapsulamiento y ocultamiento de información
- Ejemplos

Miembros

- En POO se denomina:
- Datos miembros, a las variables de instancia
- Funciones miembros, a los métodos

- Hay dos tipos de miembros, tanto para datos como para métodos:
  - De instancia
  - De clase

Por lo cual se pueden definir:

* variables de instancia y métodos de instancia

* variables de clase y métodos de clase

Miembros de instancia

Supongamos tener creada la clase Auto

```
class Auto:
def __init__(self,mar,mod):   #constructor
        self.__marca=mar
        self.__modelo=mod

def verMarca(self):           #método de instancia
        return self.__marca
.............
```

Variables de instancia

Miembros de instancia

- Los miembros de instancia, ya sean variables o métodos, son utilizados cuando se trabaja con instancias de una clase

```
a1= Auto("Ford",1987)
a2= Auto("Peugeot",1995)
a3= Auto("Fiat",2008)
```

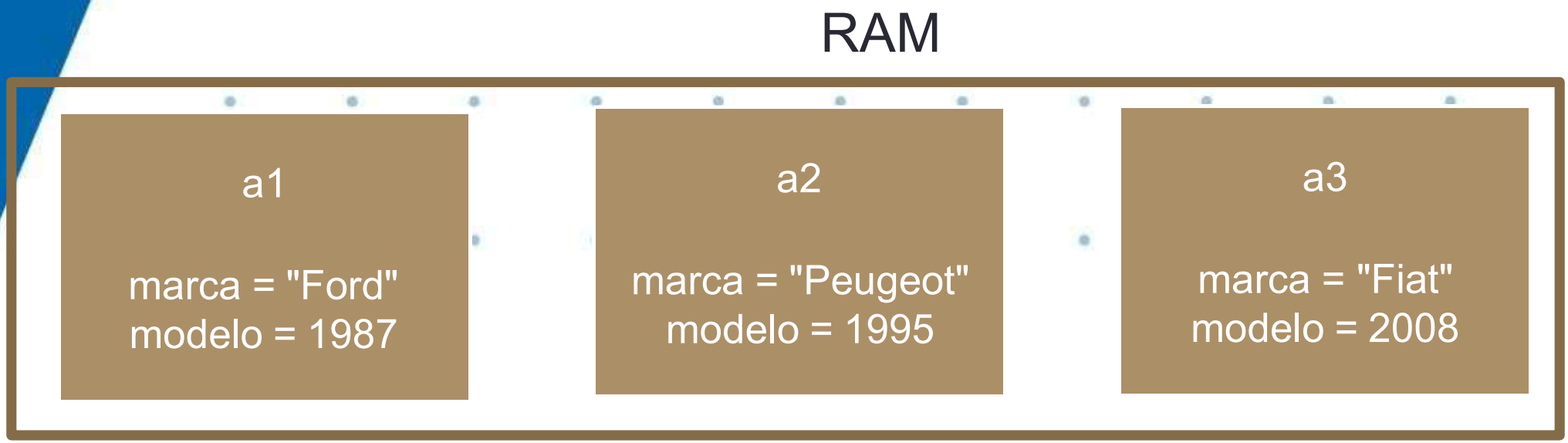
a1 , a2 y a3 son instancias de la clase Auto.

```
print ('Marca:', a1.verMar())
print ('Marca:', a2.verMar())
print ('Marca:', a3.verMar())
```

verMar() es un método de instancia, se aplica sobre objetos o instancias de la clase Auto.
Retorna el contenido de la variable de instancia marca

Miembros de instancia

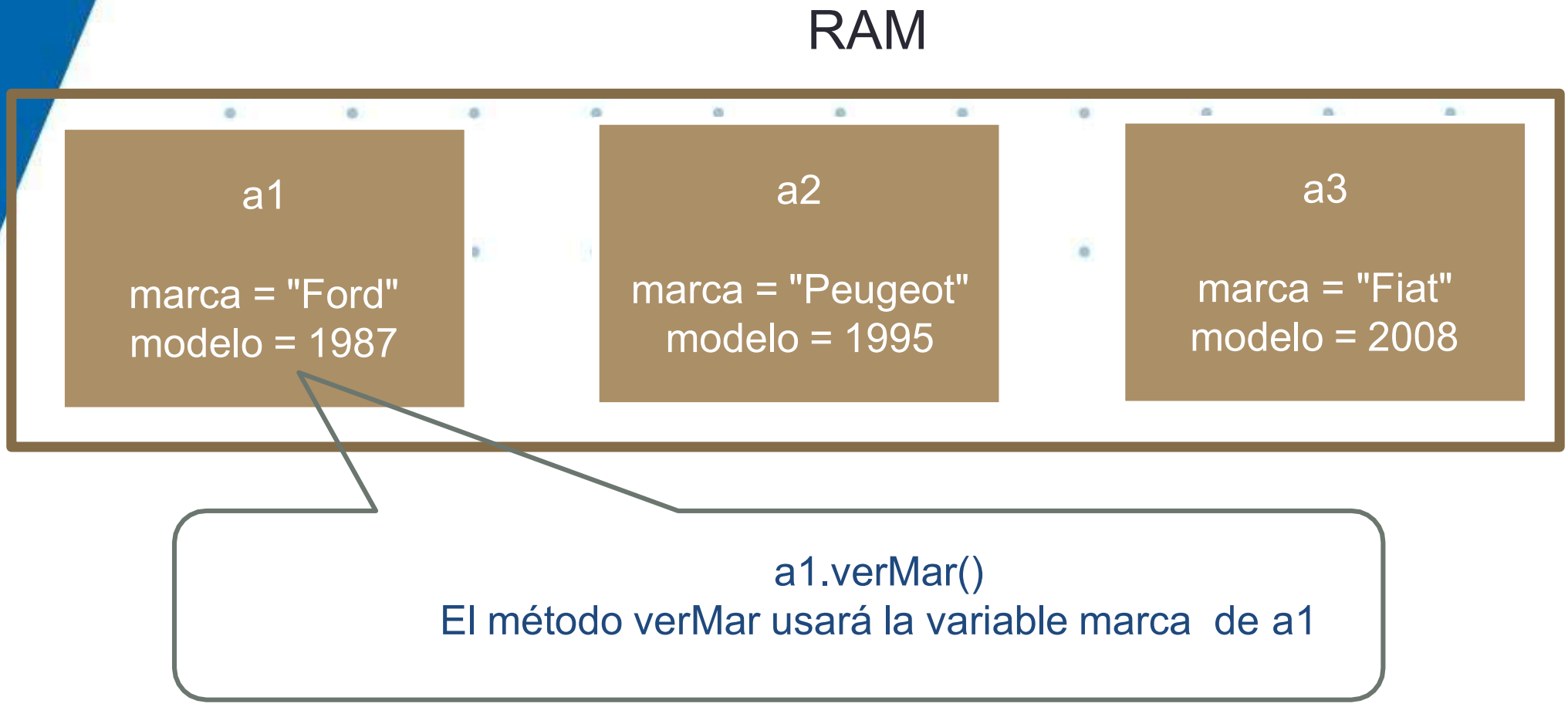
- Cuando se instancia una clase, el compilador genera en memoria una "copia" de la clase para cada instancia creada.



Técnicamente hay tres variables marca y tres variables modelo. Son las variables de instancia de cada objeto

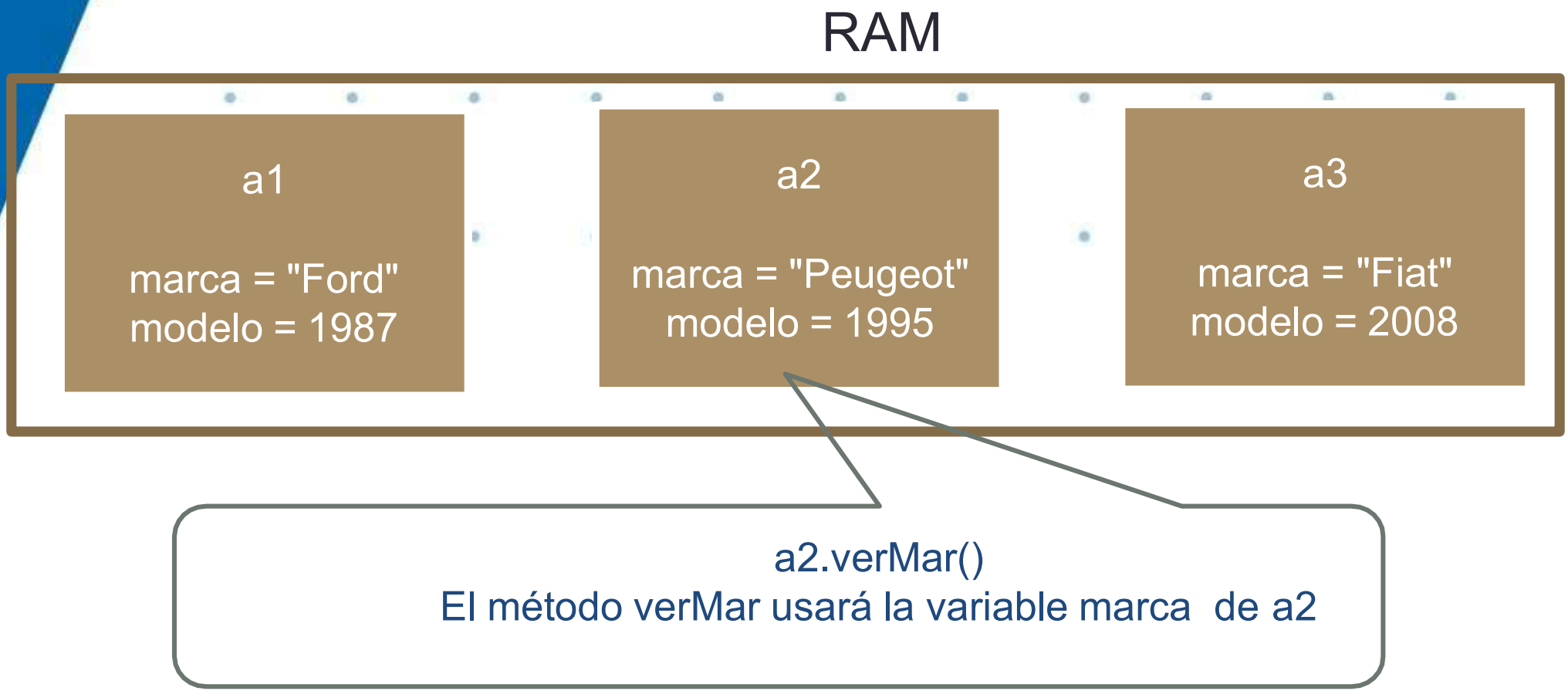
Miembros de instancia

- Los métodos de instancia son los que acceden a cada variable dependiendo de quién sea el objeto receptor del mensaje.



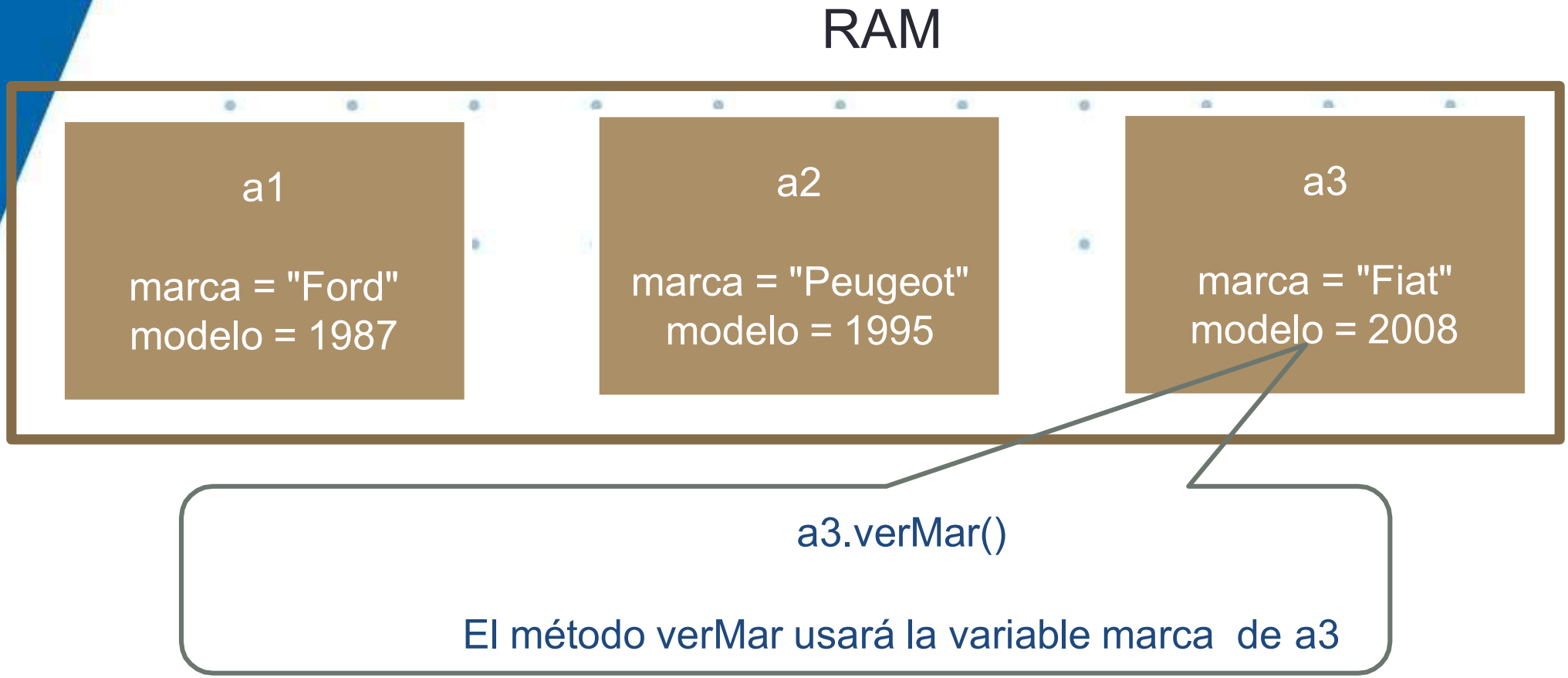
Miembros de instancia

- Los métodos de instancia son los que acceden a cada variable dependiendo de quien sea el objeto receptor del mensaje.



Miembros de instancia

- Los métodos de instancia son los que acceden a cada variable dependiendo de quién sea el objeto receptor del mensaje.



Miembros de clase

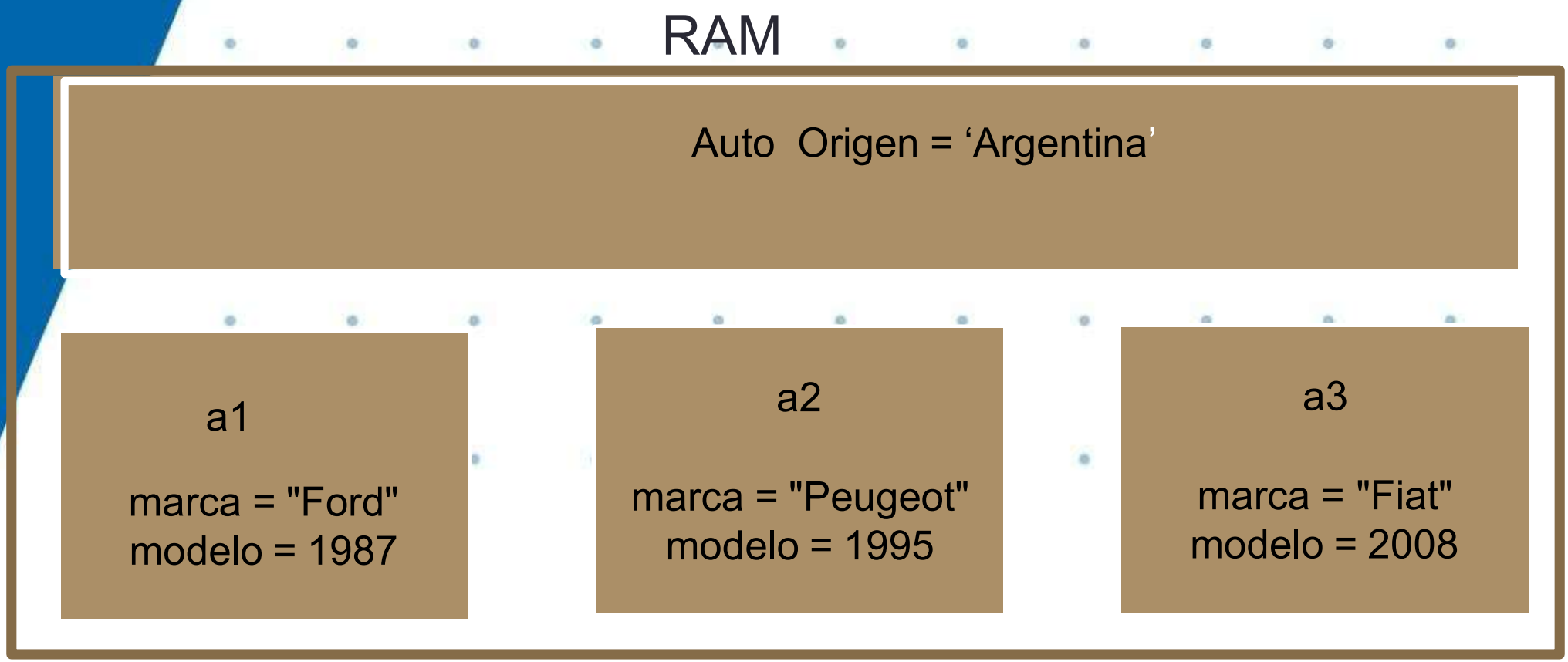
Las variables de clase son elementos que guardan información común y compartida por todas las instancias de una clase.

Se la define luego del encabezado de la clase, como si fuera una variable global. Su nombre comienza con mayúscula. Hay una sola copia de dicha variable.

```
class Auto:
    Origen='Argentina'

    def__init__(self, mar, mod):

        self.__marca = mar
        self.__modelo = mod
```

Declaración de una variable de clase (pública)

Miembros de clase



La variable Origen es una variable de clase y es común a todas las instancias. Hay una sola copia. Por eso muestra el valor Argentina para cualquier instancia.

Miembros de clase

Las formas de acceder a una variable de clase son:

- `NombreDeClase.nombreVariable`
- `self.nombreVariable`

Esta variable de clase, nos permite utilizarla sin necesidad de instanciar, es decir, de crear un objeto perteneciente a esta clase. Se le puede dar valor aún cuando no haya ningún objeto de la clase creado.
Si está definida como pública, se puede acceder a ella solo con colocar el nombre de la clase y el nombre de la variable:

print (’Fabricado en:’, Auto.Origen)

Encapsulamiento y ocultamiento

En Python:

- la clase permite encapsular la representación interna y la implementación de los métodos.

- El ocultamiento de un atributo o método viene determinado por su nombre:

- ***si el nombre comienza con dos guiones bajos (y no termina también con dos guiones bajos) se trata de un atributo o método privado***,

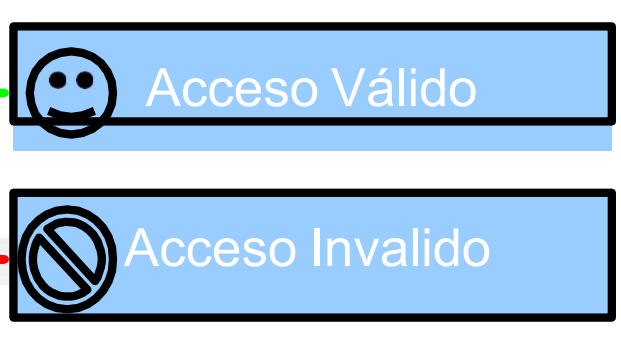
- en caso contrario es público.

Ocultamiento de información en Python

```
class Ascensor:
    def __init__(self, __pisoActual):
        self.__pisoActual= __pisoActual
        self.disponible = True

    def subir(self):
        self.__pisoActual=self.__pisoActual+1

    def bajar(self):
        self.__pisoActual=self.__pisoActual-1

    def getPisoActual(self):
        return self.__pisoActual

    def __setPisoActual(self, __pisoActual):
        self.__pisoActual=__pisoActual

ascensor1 = Ascensor(0)

ascensor1.disponible = True

ascensor1.getPisoActual()

ascensor1.__pisoActual=__pisoActual

ascensor1.__setPisoActual(2)
```

Miembros Privados

Miembros Públicos



Ventajas de la POO

- Reusabilidad del código mediante herencia y encapsulamiento
- Fácil entendimiento de la lógica del programa
- Facilidad en el mantenimiento y expansión
- Fácil documentación y diseño del programa

Ejemplo 1

Defina una clase llamada "Mascota" con 4 atributos: nombre de la mascota, especie (perro, gato, tortuga, etc.), edad en años y nombre del dueño.

Defina un constructor que reciba los cuatro campos.

```
class Mascota:

    def __init__(self,nom,esp,due,ed):
        self.__nombre=nom
        self.__especie=esp
        self.__duenio=due
        self. __edad=ed

    def verNom(self):
        return self.__nombre

    def verEdad(self):
        return self.__edad

    def verEsp(self):
        return self.__especie

    def verDuenio(self):
        return self.__duenio
```

Constructor

```
def modNom(self,otro):
    self.__nombre=otro

def modEd(self,otra):
    self.__edad=otra

def modEsp(self,otra):
    self.__especie=otra

def modDuenio(self,otro):
    self.__duenio=otro
```

Ejemplo 2

Dada la definición de la clase Mascota del ejercicio anterior ¿cuáles de las siguientes sentencias de instanciación son correctas?

```
Mascota m
m = new Mascota ("Fufu", "conejo")
m = Mascota ("Fido", "perro")
m = Mascota ("Silver", "caballo", 10, "Llanero")
m = new Mascota ()
m = Mascota('beto','perro','Jorge',5)
```

Ejemplo 3

Agregue a la clase "Mascota" un método de instancia llamado "hablarConElDueño" que dependiendo de la especie haga su sonido característico (ladrar, maullar, relinchar, etc,)

```
def hablarConElDue(self):
    if self.__especie=="perro":
        hace="Estoy ladrando“
    elif self.__especie=="gato":
        hace="Estoy maullando"
    elif  self.__especie=="caballo":
        hace= "Estoy relinchando"
    elif self.__especie== "canario":
         hace="Estoy piando"
    elif self.__especie=="conejo":
          hace= "Estoy chillando"
    else:
          hace= "Soy un animal que no habla"
    return hace
```

Ejemplo 4

Haga un programa que instancie diferentes mascotas, las guarde en una lista y luego recorra la colección haciendo "hablar" a las mascotas.

```
#Aplicación

listamascotas = []
m= Mascota ("Manuelita", "tortuga", "Elena", 4)
listamascotas.append(m)
m = Mascota ("merlin", "gato", "Pedro", 5)
listamascotas.append(m)
m= Mascota ("Silver", "caballo", "Llanero", 10)
listamascotas.append(m)

for a in listamascotas:
    print (a.hablarConElDue())
```