

# Classes

*Python*

# Repaso

# Funciones

## Ejemplo: Factorial

$$1! = 1$$

$$2! = 2(1) = 2$$

$$3! = 3(2)(1) = 6$$

$$4! = 4(3)(2)(1) = 24$$

$$5! = 5(4)(3)(2)(1) = 120$$

# Ejemplo: Fibonacci

## The Fibonacci Sequence

**1,1,2,3,5,8,13,21,34,55,89,144,233,377...**

$$1+1=2$$

$$1+2=3$$

$$2+3=5$$

$$3+5=8$$

$$5+8=13$$

$$8+13=21$$

$$13+21=34$$

$$21+34=55$$

$$34+55=89$$

$$55+89=144$$

$$89+144=233$$

$$144+233=377$$

# Classes

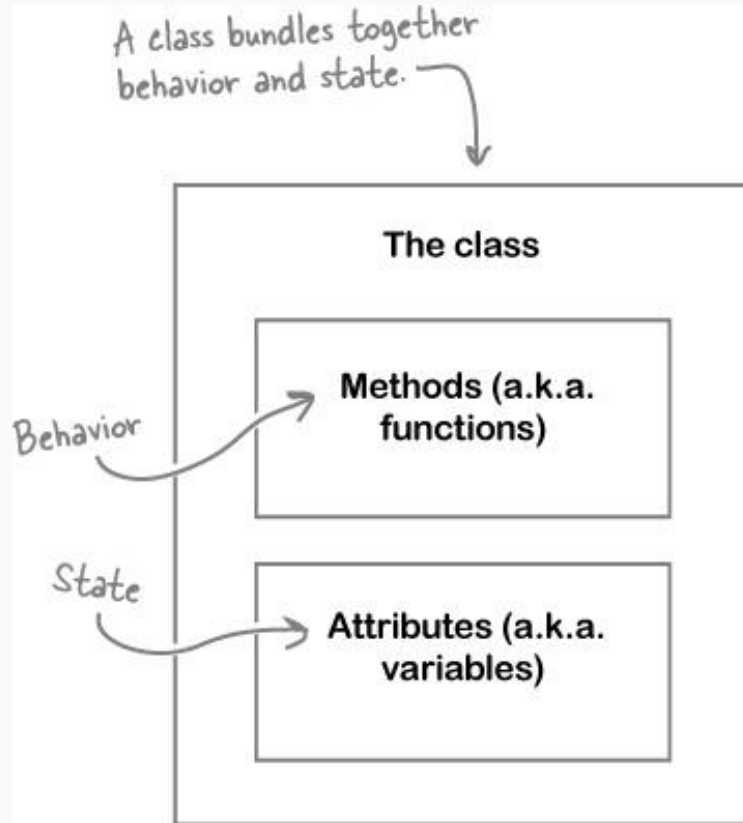
# Definición

- Una **clase** representa el comportamiento y estado de un objeto.

comportamiento -> función

estado -> variables

# Definición



# Definición

- Los **objetos** se crean a partir de **clases** a través de la **instanciación**.
- En Python:

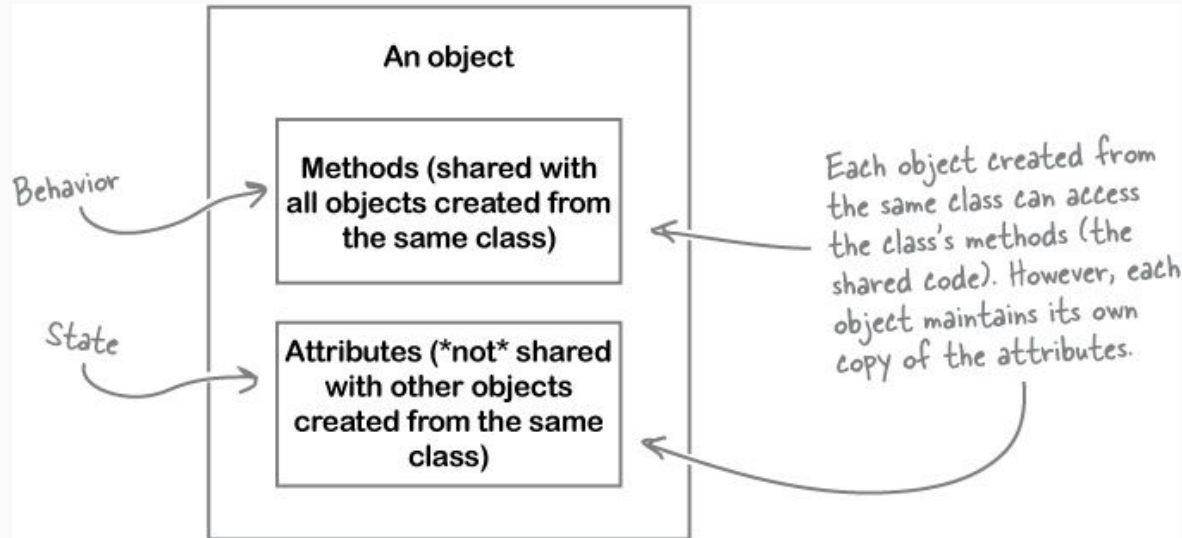
The diagram illustrates the syntax for defining a class in Python. It shows the code `>>> class CountFromBy:  
pass` with several handwritten annotations and arrows pointing to specific parts of the code:

- An arrow points from the text "Classes start with the 'class' keyword." to the `class` keyword.
- An arrow points from the text "Here's the class suite." to the `pass` statement.
- An arrow points from the text "The name of the class" to the `CountFromBy` identifier.
- An arrow points from the text "Don't forget the colon." to the colon at the end of the `class` line.



# Definición

- Los **objetos** de una misma **clase**, comparten el comportamiento pero no el estado.
- Cada **objeto** mantiene su propio estado.

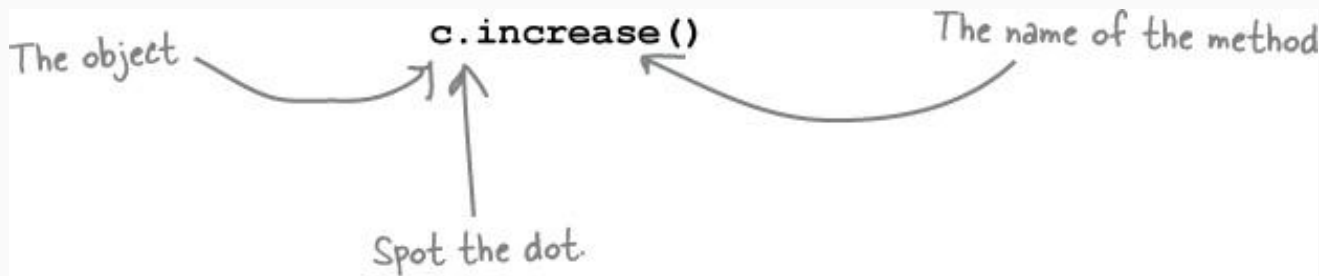


# Ejemplos

Creación de la representación de  
un auto

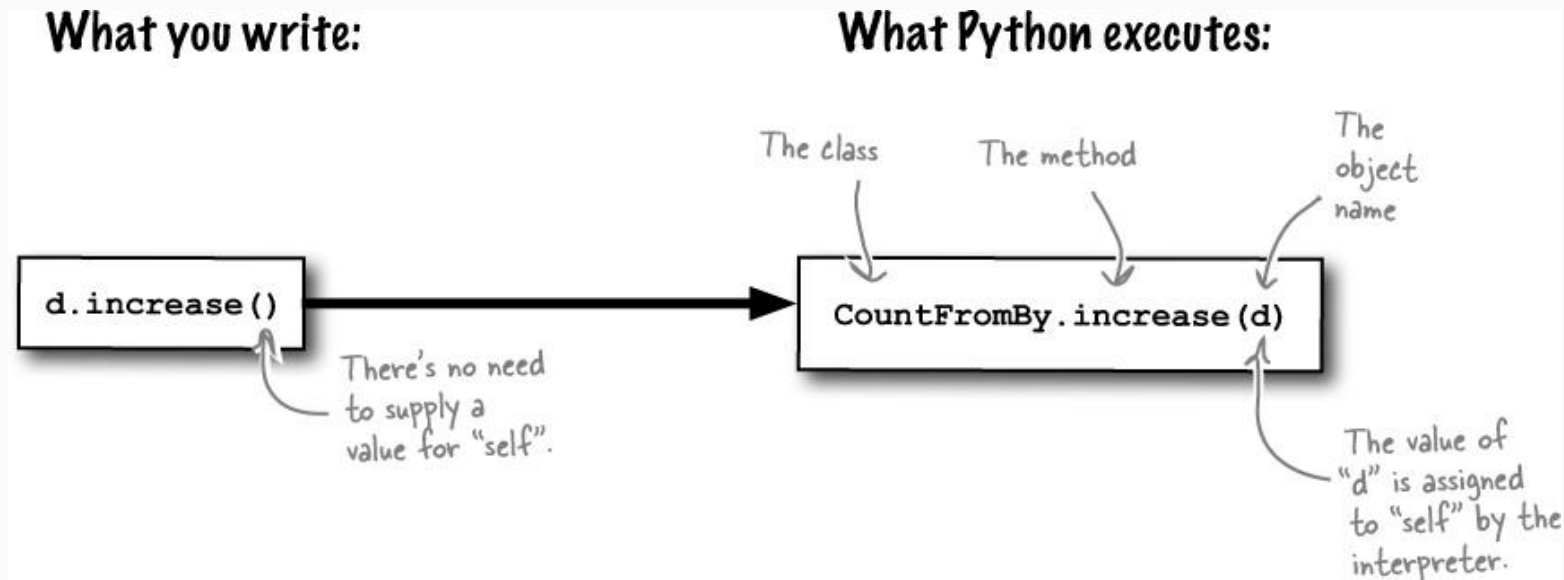
# Métodos

- Es una función definida dentro de una clase.
- El primer argumento es la instancia del objeto invocado (**self**).



# Métodos

Interpretación de una llamada al método de una clase.



# Métodos - Argumento Self

Methods are just like functions, so are defined with "def".

```
class CountFromBy:
```

```
    def increase(self) -> None:
```

As with the other functions in this book, we provide an annotation for the return value.

The first argument to every method is always "self", and its value is supplied by the interpreter.

```
class CountFromBy:
```

```
    def increase(self) -> None:
```

```
        self.val += self.incr
```

What's the deal with using "self" within the method's suite?

# Ejemplos


Creación de un método a la clase  
auto

# Atributos

- Las variables definidas dentro una **función** existente mientras se ejecuta la función.
- Los valores de los atributos continúan existiendo después de la ejecución de los métodos.

# Atributos

```
class CountFromBy:  
    def increase(self) -> None:  
        self.val += self.incr
```



This is much better, as "val" and "incr" are now associated with the object thanks to the use of "self".




# Atributos - Iniciación

- El método constructor (`__init__`) define qué sucede cuando se crea un objeto.
- Existe constructor por defecto.

```
def __init__(self):
```


Despite the strange-looking name, dunder "init" is a method like any other. Remember: you must pass "self" as its first argument.



# Atributos - Iniciación

```
class CountFromBy:  
    def __init__(self) -> None:  
        pass  
    def increase(self) -> None:  
        self.val += self.incr
```

At the moment, this dunder "init" doesn't do anything. However, the use of "self" as its first argument is a **BIG CLUE** that dunder "init" is a method.



# Ejemplos

Creación de método constructor  
de la clase auto

# Importación de clases

# Herencia

# Ejercicio

## Restaurante

1. Crear una clase llamada Restaurante. El método `__init__()` debe almacenar dos atributos: un `restaurant_name` y `cuisine_type`.
2. Haga un método llamado `describe_restaurant()` que imprima estos dos datos, y un método llamado `open_restaurant()` que imprima un mensaje que indica que el restaurante está abierto.
3. Haga una instancia llamada `restaurant` de su clase e imprimir los dos atributos individualmente y luego llame a ambos métodos.

# Ejercicio

## Restaurante

1. Agregar el atributo llamado **numero\_served** con un valor predeterminado de 0. Imprimir el número de clientes que el restaurante ha servido, y luego cambie este valor e imprímalo nuevamente.
2. Agregar el método llamado **set\_number\_served()** que le permite establecer el número de clientes que han sido atendidos.
3. Agregar un método llamado **increment\_number\_served()** que le permite incrementar el número de clientes que han sido atendidos. Llame a este método con cualquier número que desee que podría representar cuántos clientes fueron atendidos, por ejemplo, en un día de trabajo.

# Resumen

- ¿En qué se diferencian **función y método**?
- ¿Para qué **sirven** una clase?
- ¿Qué **elementos** tiene en clase?
- ¿Para qué sirve la importación?