


Open · Opened 4 days ago by  **Rubén Montero**

Una clase con un método

Resumen

- Veremos cómo se declara una nueva clase en Python
- Escribiremos una clase con un método en un fichero nuevo `chatbot.py`
- Lo instanciamos

Descripción

Hemos trabajado con Python y hemos visto muchas de sus bondades y sus desafíos. Pero hay una característica que no hemos comentado.

Es un lenguaje de programación...

...multiparadigma

Esto significa que [soporta varios paradigmas](#) de programación.

¿Qué es un [paradigma de programación](#)?

Es un *estilo* de programación. Podríamos decir, una *filosofía*.

Ya ves que Python y Java son diferentes. Cada lenguaje tiene su *estilo*.

Principales paradigmas de programación

Son:

- Orientación a objetos
- Procedimental
- Declarativo
 - Funcional

En Python, hasta ahora hemos trabajado de una forma **procedimental**.

¡Detengámonos un momento para usar **orientación a objetos** (como en Java)!

¿Y cómo?

Definiendo clases con métodos y atributos.

Instanciándolas. Usándolas.

¿Y cómo?

Una nueva clase se define con la palabra reservada `class`:

- No hay que especificar `public` ó `private`
- No existe la restricción de *una clase por fichero* (aunque puede ser aconsejable)
- No existe la restricción de *el nombre del fichero debe coincidir con el nombre de la clase*

Pongamos en marcha un ejemplo...

Añade un nuevo fichero a `advanced/`. **Llámallo** `chatbot.py`.

Dentro, vamos a definir una clase llamada `ChatBot`. Así:

```
class ChatBot:
```

Debe tener contenido, así que vamos a añadir **nuestro primer método**:

```
class ChatBot:
    def test_hello(self):
        print("¡Hola!")
```

¡Nuestra primera clase válida!

¡Qué emoción!

Uhm... ¿qué es eso de **self**?

Has observado que el **método** `test_hello` declara **1** parámetro formal:

```
def test_hello(self):
```

Se trata de **la propia instancia**

¿Eh?

`self` equivale al `this` de Java

¿Pero por qué se declara?

Se trata de una [cuestión del lenguaje](#). Estamos **obligados**.

En otras palabras, algo como esto:

```
class ChatBot:
    def test_hello(): # MAL
        print("¡Hola!")
```

...**no** es correcto.

¿Y hay que *pasarlo* cuando invoquemos el método?

No.

Eso sucede **automáticamente**.

Eh... Explicación, por favor

Veámoslo en funcionamiento.

Añade un nuevo fichero `run_class_example.py` dentro de `advanced/`.

Desde ahí, *importa* la *clase*. Ya conocemos la sintaxis:

```
from chatbot import ChatBot
# import chatbot # También vale, aunque habrá que usar el prefijo
```

Añadiremos también la línea de *main*:

```
from chatbot import ChatBot

if __name__ == '__main__':
    # ...
```

Instanciar una clase en Python

Se hace como en Java, indicando el *nombre de la clase* y *paréntesis* `()` directamente, pero **no** se usa `new`:

```
from chatbot import ChatBot

if __name__ == '__main__':
    my_object = ChatBot()
```

Invocar un método

Igual que en Java, se usa el *punto* (.) seguido del nombre del método y *paréntesis*.

```
from chatbot import ChatBot

if __name__ == '__main__':
    my_object = ChatBot()
    my_object.test_hello()
```

Tal y como dijimos arriba, **no** hay que *pasar* *self*. Eso sucede automáticamente.

¡Enhorabuena! Has creado tu primera clase con un método en Python.

Por último

Verifica que tu código pasa el *test* asociado a la tarea.

Haz *commit* y *push* para subir los cambios al repositorio.



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 3](#) 4 days ago