


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

# Métodos con parámetros

## Resumen

- Veremos cómo se declaran métodos con parámetros formales a mayores que `self`
- Escribiremos dos nuevos métodos en `chatbot.py`
- Los invocaremos desde `run_class_example.py`

## Descripción

Nuestro primer método *declara* `self`:

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

...pero este *parámetro* **no** se pasa al invocar el método:

```
my_object.test_hello()
```

## ¿Pero se pueden pasar parámetros?

¡Sí!

Escribe un método nuevo en `chatbot.py` y llámalo `test_one_param`. Esperará un *número* y hará un `print`:

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

    def test_one_param(self, number):
        print("Tu número es: " + str(number))
```

Ahora, desde `run_class_example.py` vamos a *invocarlo* con un valor de ejemplo:

```
if __name__ == '__main__':
    my_object = ChatBot()
    my_object.test_hello()
    # El siguiente método recibe un parámetro
    my_object.test_one_param(49)
```

Ejecuta `run_class_example.py`. Funciona, ¿verdad?

## ¿Y más de uno?

Por supuesto.

## Completemos la tarea

- **Añade** otro método a `ChatBot`
  - Se llamará `subtract_numbers`
  - Declarará `self` y otros dos parámetros: `number1` y `number2`
  - Hará un `print` de `number2 - number1`
- **Invoca** ese método desde tu *instancia* de `ChatBot` **en** `run_class_example.py`, pasando dos valores cualesquiera.

## 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