
OOP
Methods

Adding Methods to Class

Previously we talked about attributes. Now, Let's talk about methods.

Attributes are the entity that an object has i.e. It represents the characteristics of an object.

Methods are the entity that tells what work an object can do. i.e. The things that an object can perform.

```
class Car:
    def enter_race_mode():
        self.seats = 2
```

TALKING OF METHODS

Unlike functions, methods always needs to have a “self” parameter as the first parameter. This basically means that whenever that particular method is called, it knows the object that called it.

Methods In Python

Let’s start with an example

main.py

Safe mode, limited functionality. Trust the project to access full IDE functionality.

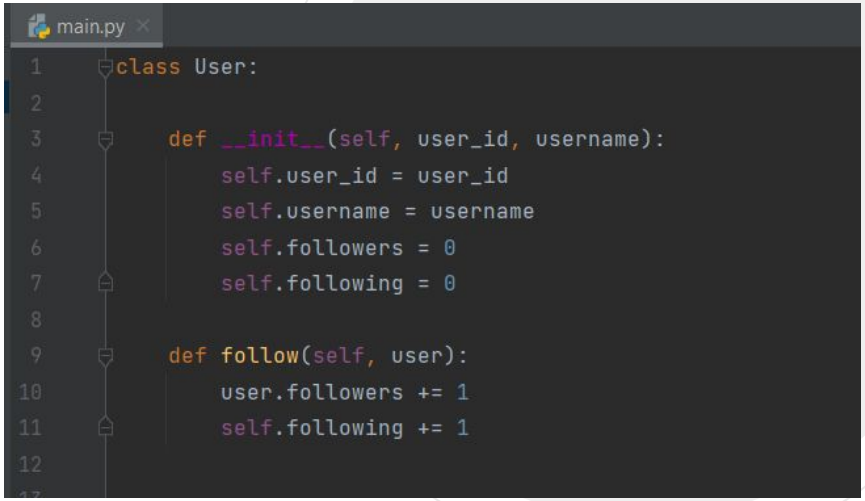
```python
class User:

    def __init__(self, user_id, username):
        self.user_id = user_id
        self.username = username
        self.followers = 0
        self.following = 0

    def follow(self, user):
        user.followers += 1
        self.following += 1


user_1 = User("007", "James Bond")
user_2 = User("10", "Messi")


user_1.follow(user_2)
print(user_1.followers)
print(user_1.following)
print(user_2.followers)
print(user_2.following)

```

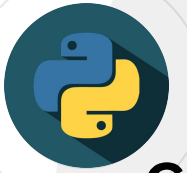
Explanation of example

First, we created a class and called it “User”.

We used constructor function to initialize variables.

We also defined a method called “follow”.





While creating objects, Our class will take two arguments. First one is user_id and the second one is username. Also, when the object is created, the followers and following of object is initialized at 0.

Explanation of example

Now, we created two objects called “user_1” and “user_2” from class “User” with some parameters.

```
user_1 = User("007", "James Bond")
user_2 = User("10", "Messi")
```

We already know that “follower” and “following” attributes of both “user_1” and “user_2” are initialized at zero.

Output of the code in RHS

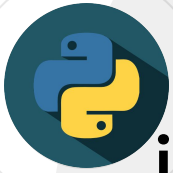
```
007
James Bond
0
0
```

```
print(user_1.user_id)
print(user_1.username)
print(user_1.followers)
print(user_1.following)
```

Output of the code in RHS

```
10
Messi
0
0
```

```
print(user_2.user_id)
print(user_2.username)
print(user_2.followers)
print(user_2.following)
```



We can see that objects are initialized with certain values.

Assume that user_1 followed user_2. Now, the follower and following needs to be updated. But how can we achieve that?

Explanation of example

For that, we have created a method called “follow”.

We can see that the method “follow” have two parameters: “self” and “user” where self is the object that calls the method “follow”and user is another object passed as parameter.

```
user_1 = User("007", "James Bond")
user_2 = User("10", "Messi")

user_1.follow(user_2)
```

Let’s say that when an object follows another object, “follow”method is called by following object.

Explanation of example

In the code, we can see the line “self.following += 1 & user.follower +=1”.

It means that one object followed another object represented by “user”. Now, the method is called by the follower object so, self.following is updated by 1. Similarly, object represented by “user” is being followed. So, user.follower is updated by 1.

```
def follow(self, user):
    user.followers += 1
    self.following += 1
```

Explanation of example

Let’s look at line “user_1.follow(user_2)”.

This line means that the “user_1” has called method “follow” with “user_2” as parameter. i.e. “user_1.following” & “user_2.follower” both should increase respectively by 1.

```
user_1.follow(user_2)
```

Let’s look at output!

Output of the code in RHS

```
C:\Python310\python.exe "D:/100/OOP - II/main.py"
0
1
1
0
```

```
print(user_1.followers)
print(user_1.following)
print(user_2.followers)
print(user_2.following)
```

Thank You!!!