ال class بيكون فيه 3 انواع من ال methods اتكلمنا عن ال object method في الدروس اللى فاتت ودي اللى بيستخدمها ال object اللى معمول من ال class، في عندنا بقا ال **class method** و ال **static method** ودول اللى هنتكلم عنهم فى الدرس دا ونعرف الفرق بين ال 3.

اول حاجه ال **class method** ودي زي ال attribute method بتكون خاصه بال class مش بال object. في ال method العاديه كنت بحط اول parameter اسمه self، في ال class method هحط اول parameter اسمه **cls** وهيكون فى الحاله دي بيشاور على اسم ال class مش ال object ، بس حاسس بتناقض صح ؟ لأننا كنا قلنا ان self اللى بتشاور على ال object ممكن تتسمى اي اسم تاني، فأه كدا لسه ممكن يبقا object method عادي وتكون cls دي بتشاور على ال object، فعلشان نأكدله ان دا class method بحط decorator قبل ال method اسمه **@classmethod** ، كدا بقا class method رسمي.

طيب بستخدمه ف ايه ال class method ؟ لما اكون هنفذ task على ال attributes بتاعة ال class نفسه، يعني مش محتاج اني اعمل object من ال class لا دانا عندي attributes خاصه بال class نفسه بعمل بيها task.

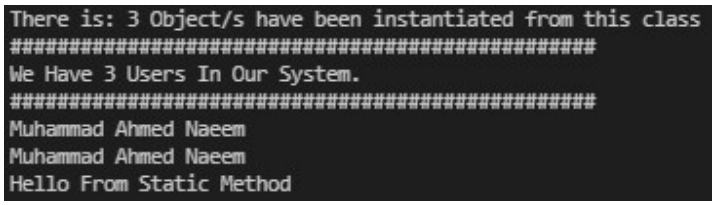
بعد كدا عندي ال **static method** ودي لا بتشاور لل class ولا لل object ومش بتستخدم اى attributes من جوا ال class تنفذ عليه task فبالتالى مش محتاج اضيفلها parameter زي self او cls (مع ذلك ممكن هضيف parameters تانيه بس هتكون كأنها parameters الفانكشن العاديه علشان استخدمها جواها واليوزر يدخلها ك argument لما يجي يستدعي الفانكشن)، هي بتكون مجرد فانكشن عاديه بس بكون عارف انها موجوده فى ال class دا فلما بحتاج انفذها بكتب اسم ال class وجنبه اسم ال method.

علشان اعرفها ان هى static بحط قبلها decorator اسمه **@staticmethod** .

```
# ---------------------------------------------------------------------
# -- Object Oriented Programming => Class Methods & Static Methods --
# ---------------------------------------------------------------------
# Class Methods:
# - Marked With @classmethod Decorator To Flag It As Class Method
# - It Take Cls Parameter Not Self To Point To The Class not The Instance
# - It Doesn't Require Creation of a Class Instance
# - Used When You Want To Do Something With The Class Itself
# Static Methods:
# - It Takes No Parameters
# - Its Bound To The Class Not Instance
# - Used When Doing Something Doesnt Have Access To Object Or Class But Related To Cl
ass
# ---------------------------------------------------------------

class Member:
    not_allowed_names = ["Hell", "Shit", "Baloot"]

    users_num = 0

    @classmethod
    def show_users_count(cls):

        print(f"We Have {cls.users_num} Users In Our System.")
```

```python
    @staticmethod
    def say_hello():

        print("Hello From Static Method")

    def __init__(self, first_name, middle_name, last_name, gender):

        self.fname = first_name

        self.mname = middle_name

        self.lname = last_name

        self.gender = gender

        Member.users_num += 1  # Member.users_num = Member.users_num + 1

    def full_name(self):

        if self.fname in Member.not_allowed_names:

            raise ValueError("Name Not Allowed")

        else:

            return f"{self.fname} {self.mname} {self.lname}"

member_one = Member("Muhammad", "Ahmed", "Naeem", "Male")
member_two = Member("Ahmed", "Ali", "Mahmoud", "Male")
member_three = Member("Mona", "Ali", "Mahmoud", "Female")

print(f"There is: {Member.users_num} Object/s have been instantiated from this class"
)

print("#" * 50)

Member.show_users_count()

print("#" * 50)

print(member_one.full_name()) # It's the same as below
print(Member.full_name(member_one)) # It's the same as above

Member.say_hello()
```

CODE


There is: 3 Object/s have been instantiated from this class
##
We Have 3 Users In Our System.
##
Muhammad Ahmed Naeem
Muhammad Ahmed Naeem
Hello From Static Method

OUTPUT