Methods in Python – A Key Concept of Object Oriented Programming

Types of Methods in Python

There are basically three types of methods in Python:

- Instance Method
- Class Method
- Static Method

1. Instance Methods

The purpose of instance methods **is to set or get details about instances (objects), and that is why they're known as instance methods.** They are the most common type of methods used in a Python class.

```
class My_class:
  def instance_method(self):
    return "This is an instance method."
obj = My_class()
obj.instance_method()
```

2. Class Methods

The purpose of the class methods is to **set or get the details (status) of the class.** That is why they are known as class methods. They **can't access or modify specific instance data**. They are **bound to the class instead of their objects**. Two important things about class methods:

- In order to define a class method, you have to specify that it is a class method with the help of the **@classmethod decorator**
- Class methods also take one default parameter- **cls,** which points to the class. Again, this not mandatory to name the default parameter **cls.** But it is always better to go with the conventions

```
class My_class:

  @classmethod
  def class_method(cls):
    return "This is a class method."
obj = My_class()
obj.class_method()
```

3. Static Methods

Static methods cannot access the class data. In other words, **they do not need to access the class data**. They are **self-sufficient and can work on their own**. Since **they are not attached to any class attribute**, they cannot get or set the instance state or class state.

In order to define a static method, we can use the **@staticmethod decorator** (in a similar way we used @classmethod decorator). Unlike instance methods and class methods, **we do not need to pass any special or default parameters**

```
class My_class:

  @staticmethod
  def static_method():
    return "This is a static method."
obj = My_class()
obj.static_method()
```

Class Method – The most common use of the class methods is for creating **factory methods**. Factory methods are those methods **that return a class object (like a constructor) for different use cases**.

Static Method – They are used for **creating utility functions**. For accomplishing routine programming tasks we use utility functions

Python classmethod()

```
class geeks:
    course = 'DSA'

    def purchase(obj):
        print("Purchase course : ", obj.course)


geeks.purchase = classmethod(geeks.purchase)
geeks.purchase()
```

```python
# Python program to demonstrate
# use of a class method and static method.
from datetime import date

class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    # a class method to create a
    # Person object by birth year.
    @classmethod
    def fromBirthYear(cls, name, year):
        return cls(name, date.today().year - year)

    def display(self):
        print("Name : ", self.name, "Age : ", self.age)

person = Person('mayank', 21)
person.display()
```

```python
# Python program to demonstrate
# use of a class method and static method.
from datetime import date

class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    # a class method to create a
    # Person object by birth year.
    @classmethod
    def fromBirthYear(cls, name, year):
        return cls(name, date.today().year - year)

    # a static method to check if a
    # Person is adult or not.
    @staticmethod
    def isAdult(age):
        return age > 18

person1 = Person('mayank', 21)
person2 = Person.fromBirthYear('mayank', 1996)

print(person1.age)
print(person2.age)

# print the result
print(Person.isAdult(22))
```

```python
class MyClass:
    def __init__(self, value):
        self.value = value

    @staticmethod
    def get_max_value(x, y):
        return max(x, y)

# Create an instance of MyClass
obj = MyClass(10)

print(MyClass.get_max_value(20, 30))

print(obj.get_max_value(20, 30))
```