**What is Class Method in Python?**

The @classmethod decorator is a built-in function decorator that is an expression that gets evaluated after your function is defined. The result of that evaluation shadows your function definition. A class method receives the class as an implicit first argument, just like an instance method receives the instance

Syntax Python Class Method:

class C(object):

@classmethod

def fun(cls, arg1, arg2, ...):

....

fun: function that needs to be converted into a class method

returns: a class method for function.

A class method is a method that is bound to the class and not the object of the class.

They have the access to the state of the class as it takes a class parameter that points to the class and not the object instance.

It can modify a class state that would apply across all the instances of the class. For example, it can modify a class variable that will be applicable to all the instances.

**What is the Static Method in Python?**

A static method does not receive an implicit first argument. A static method is also a method that is bound to the class and not the object of the class. This method can’t access or modify the class state. It is present in a class because it makes sense for the method to be present in class.

Syntax Python Static Method:

class C(object):

@staticmethod

def fun(arg1, arg2, ...):

...

returns: a static method for function fun.

**The difference between the Class method and the static method is:**

* A class method takes cls as the first parameter while a static method needs no specific parameters.
* A class method can access or modify the class state while a static method can’t access or modify it.

In general, static methods know nothing about the class state. They are utility-type methods that take some parameters and work upon those parameters. On the other hand class methods must have class as a parameter.

We use @classmethod decorator in python to create a class method and we use @staticmethod decorator to create a static method in python.

**When to use the class or static method?**

* We generally use the class method to create factory methods. Factory methods return class objects ( similar to a constructor ) for different use cases.
* We generally use static methods to create utility functions.

Example:

**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))

Output:

30

30