Aula 06 - POO em Python: Métodos de Classe e Métodos Estáticos

Introdução ao tema

Na programação orientada a objetos , usamos métodos de instância e métodos de classe. No nível de classe, usamos métodos de classe e métodos estáticos.

- **Método de classe**: Usado para acessar ou modificar o estado da classe. Na implementação do método, utilizamos variáveis de classe. O método de classe tem um `cls` parâmetro que se refere à classe.
- **Método estático**: É um método de utilidade geral que realiza uma tarefa isoladamente. Dentro deste método, não usamos variáveis de instância ou classe, pois o método estático não recebe parâmetros como `self` e `cls`.

Introdução ao tema

Nessa aula veremos:

- Como criar e usar métodos de classe em Python.
- Criar um método de classe usando o `@classmethod` decorador e `classmethod()` a função.
- Como criar e usar métodos estáticos em Python.
- Crie `staticmethod` usando o `@staticmethod` decorador e `staticmethod()` a função.

Métodos de Classe em Python

Métodos de classe são métodos que são chamados na própria classe , não em uma instância de objeto específica. Portanto, ele pertence a um nível de classe, e todas as instâncias de classe compartilham um método de classe.

O método de classe pode ser chamado usando `ClassName.method_name()` ou usando um ***objeto da classe***.

Métodos de Classe em Python: Decorator @classmethod

Adicione `@classmethod` um decorador antes da definição do método e adicione-o `cls` como o primeiro parâmetro do método, para definir um `método de classe`.

Exemplo Real

```
class Student:
    school_name = 'RaroLabs' # Variável de Classe

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

    # parametro CLS faz referência para classe
    @classmethod
    def change_school(cls, name):
        Student.school_name = name
        print(Student.school_name) # acessando a variável de classe
```

Métodos de Classe em Python: Decorator @classmethod

Exemplo Real

```
objeto_estudante = Student('Alberto Silva', 17)
Student.change_school('Raro Academy - Python') # Chamada método de classe
```

Saída

```
Raro Academy - Python
```

Métodos de Classe em Python: Função classmethod()

Além de um ***decorador***, a função interna `classmethod()` é usada para converter um ***método normal*** em um ***método de classe***. A `classmethod()` é uma função interna em Python, que retorna um método de classe para uma função dada.

Sintaxe

```
classmethod(function)
```

Onde `function` é o nome do método que você deseja converter como um método de classe, que será retornado como **método de classe** convertido.

Nota: O método que você deseja converter como um método de classe deve aceitar `class(cls)` como o primeiro argumento, assim como um método de instância recebe o `instance(self)`.

Métodos de Classe em Python: Função classmethod()

Exemplo Real

```
class School:
    name = 'Raro Academy'

    def school_name(cls):
        print('O nome da Escola é :', cls.name)

# Criando um método de classe
School.school_name = classmethod(School.school_name)

# Chamando um método de classe
School.school_name()
```

Saída

```
Raro Academy
```

Métodos Estátiscos em Python

Um ***método estático*** é um método utilitário geral que executa uma tarefa isoladamente.

Um ***método estático*** é vinculado à ***classe*** e não ao ***objeto da classe***. Portanto, podemos chamá-lo usando o ***nome da classe***.

Um ***método estático*** não tem acesso às ***variáveis de classe*** e ***instância*** porque não recebe um primeiro argumento implícito como `self` e `cls`. Portanto, não pode modificar o estado do objeto ou classe.

O método de classe pode ser chamado usando `ClassName.method_name()` ou usando um ***objeto da classe***.

Métodos Estátiscos em Python: Decorator @staticmethod

Adicione `@staticmethod` um decorador antes da definição do método.

O `@staticmethod` decorator é um decorador de função embutido em Python para declarar um método como um método estático.

Sintaxe

```
class C:
    @staticmethod
    def f(arg1, arg2, ...): ...
```

Métodos Estátiscos em Python: Decorator @staticmethod

Exemplo Real

```
class Employee(object):

    def __init__(self, name, project_name):
        self.name = name
        self.project_name = project_name

    # Método Estático
    @staticmethod
    def gather_requirement(project_name):
        if project_name == 'Migração de dados':
            requirement = ['tarefa_1', 'tarefa_2', 'tarefa_3']
        else:
            requirement = ['tarefa_1']
        return requirement


    ...
```

Métodos Estátiscos em Python: Decorator @staticmethod

Exemplo Real

```
    ...

    # Método de Instância
    def work(self):
        # Chamando um método estático dentro de um método de instância
        requirement = self.gather_requirement(self.project_name)
        for task in requirement:
            print('Finalizada ', task)

emp = Employee('Kelly', 'Machine Learning')
emp.work()
```

Saída

```
Finalizada  tarefa 1
```

Métodos Estátiscos em Python: Decorator @staticmethod

Exemplo Real

```
    ...

emp_2 = Employee('Luiz', 'Migração de dados')
emp_2.work()
```

Saída

```
Finalizada  tarefa 1
Finalizada  tarefa 1
Finalizada  tarefa 2
Finalizada  tarefa 3
```

Métodos Estátiscos em Python: Função staticmethod()

Você só deve usar `staticmethod()` function para definir ***método estático*** se tiver que dar *suporte a versões mais antigas do* `Python 2.2` *e* `Python 2.3`. Caso contrário, é recomendado usar o `@staticmethod` decorator.

Sintaxe

```
staticmethod(function)
```

É retornado o método estático convertido.

Métodos Estátiscos em Python: Função staticmethod()

Exemplo Real

```
class Employee:
    def sample(x):
        print('Dentro do método convertido para estático -- ', x)

# Converte o método para estático
Employee.sample = staticmethod(Employee.sample)

# chamando método estático
Employee.sample(34)
```

Saída

```
Dentro do método convertido para estático -- 34
```