Atributos e Métodos de Classes

Programação Orientada a Objetos

Python

Prof. Diógenes Furlan

Sumário

- Atributo de Classe
- Método de Classe
- Método Estático
- Classe Abstrata
- Método Abstrato
- Classe Final
- Método Final
- Objeto Constante

Atributo de Classe

- **Atributo de classe** é aquele que é compartilhado por todos os objetos pertencentes àquela classe.
  - Variável ÚNICA dentro da classe.

Método de Classe

- **Método de classe** é um método que se comporta como uma função regular, mas pertence a uma classe.
  - recebe a classe como o primeiro argumento.
  - consegue manipular atributos da classe
  - o método não tem acesso à instância (self) na qual é chamado.

Exemplo 1

```
class Bebida:
    nome = "Bebida"     # atributo de classe

    @classmethod
    def imprimir_nome(cls):
        print(f"Esta bebida é do tipo {cls.nome}")

class Café(Bebida):
    nome = "Café"

class Chá(Bebida):
    nome = "Chá"

# main
Bebida.imprimir_nome()
Café.imprimir_nome()
Chá.imprimir_nome()
```

Exemplo 2

```python
class Informativo:
    mensagem1 = 'Primeira aula'

    @classmethod
    def print(cls):
        print(f'Informativo: {cls.mensagem1}')

# main
obj1 = Informativo()
obj2 = Informativo()
obj3 = Informativo()

obj1.print()
obj2.print()
obj3.print()

Informativo.mensagem1 = 'Segunda aula'

obj1.print()
obj2.print()
obj3.print()
```

Statics Concepts

Método Estático

- **Método estático** é um método que se comporta como uma função regular, mas pertence a uma classe.
  - o método não tem acesso nem à classe (cls)
  - o método não tem acesso à instância (self) na qual é chamado.
  - não pode modificar o estado da instância do objeto nem o estado da classe.

Exemplo 3

```
class Calculadora:
    @staticmethod
    def somar(x, y):
        return x + y

    @staticmethod
    def subtrair(x, y):
        return x - y

# main
print(Calculadora.somar(5, 3))
print(Calculadora.subtrair(60, 15))
```

Exemplo 4

```
class Classe1:
    def metodo_objeto(self):
        return 'método de instância chamado', self

    @classmethod
    def metodo_classe(cls):
        return 'método de classe chamado', cls

    @staticmethod
    def metodo_estatico():
        return 'método estático chamado'

obj1 = Classe1()
print( obj1.metodo_objeto() )

print( Classe1.metodo_classe() )
print( Classe1.metodo_estatico() )
```

Abstracts Concepts

Conceitos

- **Classe Abstrata**
  - Não pode ser instanciada
  - Só pode servir para herança
- **Método Abstrato**
  - Sem implementação
- **Classe Final**
  - Não pode ser herdada por outra classe
  - Obrigatoriamente folha
- **Método Final**
  - Não pode ser sobrescrito pelas suas subclasses
  - Obrigatoriamente herdado

Objeto Constante

- Um **objeto constante** não pode ser alterado após o uso do construtor.
  - o compilador proíbe a ele o acesso a qualquer método, pois não consegue identificar quais métodos alteram os seus dados.
  - Colocar a palavra **const** após os parênteses que envolvem a lista de parâmetros libera o método para uso

```
void mostra() const {
    printf("\n%02d/%02d/%04d",dia,mes,ano);
}
```

ABC

- abc – abstract base class
- Este módulo fornece a infraestrutura para definir classes base abstratas

```
from abc import ABC, abstractmethod

class MinhaClasse(ABC):
    @abstractmethod
    def metodo1(self):
        pass

# main
obj1 = MinhaClasse()
```

Exemplo 5

```
from abc import ABC, abstractmethod

class Veículo(ABC):
    @abstractmethod
    def Número_portas(self):
        pass

    def Motor(self):
        print("Flex")

class Carro(Veículo):
    def Número_portas(self):
        print("Quatro")

class Moto(Veículo):
    def Número_portas(self):
        print("Zero")
```

```
# main
carro1 = Carro()
carro1.Motor()
carro1.Número_portas()

moto1 = Moto()
moto1.Motor()
moto1.Número_portas()
```

EXERCÍCIOS

Exercícios

1) Para a hierarquia de classes Animal, crie contadores para:
   - contar o número de objetos criados
   - contar o número de vezes que os animais se alimentaram

2) Crie um atributo ID para os objetos da classe Animal. Cada ID deve ser um número único e fornecido pela própria classe.

Exercícios

3) Crie uma versão estática do método Bissexto(int ano), na classe Data.

4) Para a classe Conta, com nome do cliente e valor.
   - Com os métodos saldo(), deposito() e saque()
   - Crie um atributo de classe para contar o número de contas ativas.
   - E um método de classe para mostrar o número de contas ativas.