Encapsulamento

Guilherme Arthur de Carvalho

Analista de sistemas

@decarvalhogui

Objetivo Geral

Entender o conceito de encapsulamento e como podemos aplicá-lo utilizando Python.

Pré-requisitos

- Conhecimento básico em Python.

Percurso

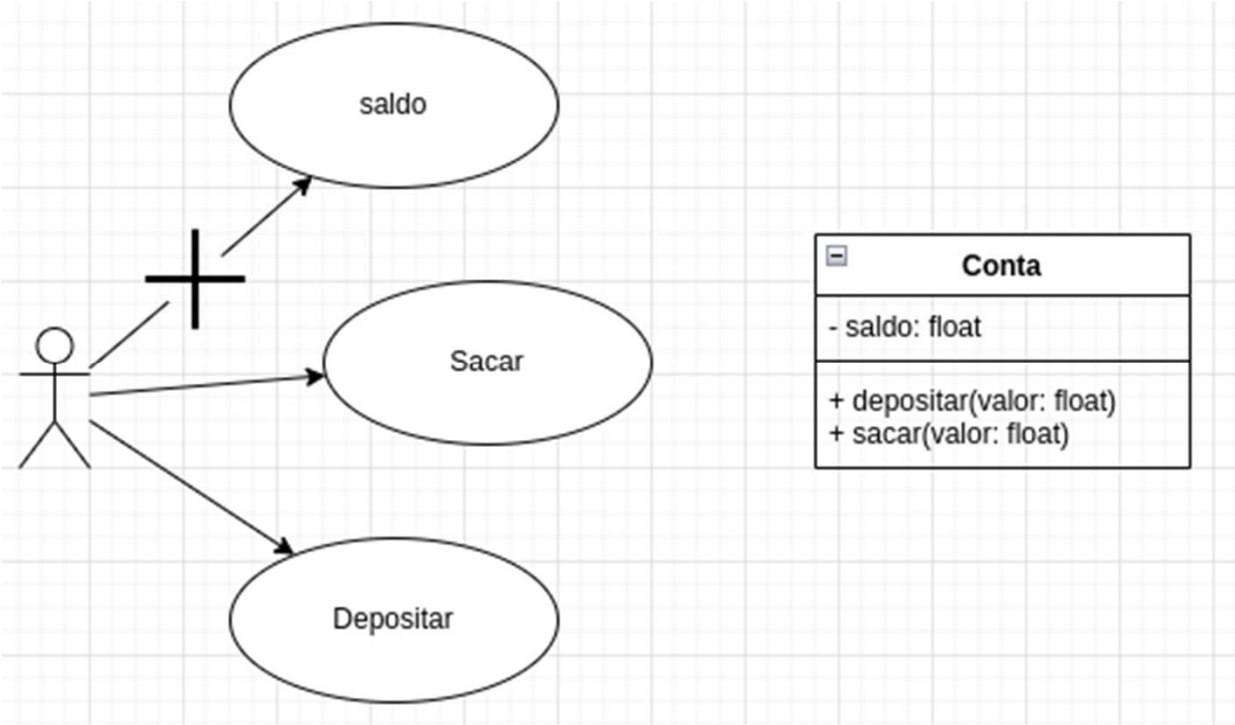
Etapa 1	**O que é encapsulamento?**
Etapa 2	Recursos públicos e privados
Etapa 3	Properties

Etapa 1

O que é encapsulamento?

Proteção de acesso

O encapsulamento é um dos conceitos fundamentais em programação orientada a objetos. Ele descreve a ideia de agrupar dados e os métodos que manipulam esses dados em uma unidade. Isso impõe restrições ao acesso direto a variáveis e métodos e pode evitar a modificação acidental de dados. Para evitar alterações acidentais, a variável de um objeto só pode ser alterada pelo método desse objeto.


saldo
Sacar
Depositar
Conta
- saldo: float
+ depositar(valor: float)
+ sacar(valor: float)

Percurso

~~Etapa 1~~	~~O que é encapsulamento?~~
Etapa 2	**Recursos públicos e privados**
Etapa 3	Properties

Etapa 2

Recursos públicos e privados

Modificadores de acesso

Em linguagens como Java e C++, existem palavras reservadas para definir o nível de acesso aos atributos e métodos da classe. Em Python não temos palavras reservadas, porém usamos convenções no nome do recurso, para definir se a variável é pública ou privada.

Definição

- Público: Pode ser acessado de fora da classe.
- Privado: Só pode ser acessado pela classe.

Público/Privado

Todos os recursos são públicos, a menos que o nome inicie com underline. Ou seja, o interpretador Python não irá garantir a proteção do recurso, mas por ser uma convenção amplamente adotada na comunidade, quando encontramos uma variável e/ou método com nome iniciado por underline, sabemos que não deveríamos manipular o seu valor diretamente, ou invocar o método fora do escopo da classe.

Exemplo

```
class Conta:
    def __init__(self, saldo=0):
        self._saldo = saldo

    def depositar(self,valor):
        pass

    def sacar(self,valor):
        pass
```

Percurso

~~Etapa 1~~ ~~O que é encapsulamento?~~

~~Etapa 2~~ ~~Recursos públicos e privados~~

Etapa 3 **Properties**

Etapa 3

Properties

Para que servem?

Com o property() do Python, você pode criar atributos gerenciados em suas classes. Você pode usar atributos gerenciados, também conhecidos como propriedades, quando precisar modificar sua implementação interna sem alterar a API pública da classe.

Exemplo

```
class Foo:
    def __init__(self, x=None):
        self._x = x

    @property
    def x(self):
        return self._x or 0

    @x.setter
    def x(self, value):
        _x = self._x or 0
        _value = value or 0
        self._x = _x + _value

    @x.deleter
    def x(self):
        self._x = -1

foo = Foo(10)
print(foo.x)
foo.x = 10
print(foo.x)
del foo.x
print(foo.x)
```

Percurso

~~Etapa 1~~ ~~O que é encapsulamento?~~

~~Etapa 2~~ ~~Recursos públicos e privados~~

~~Etapa 3~~ ~~Properties~~

Links Úteis

- https://github.com/digitalinnovationone/trilha-python-dio

Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)

SCAN ME