Classes bases abstratas e métodos abstratos

O que é uma classe abstrata?

Uma classe abstrata (ABC) é uma classe que não pode ser usada para criar objetos e sua utilização é para **definir interfaces**, ou seja, listar métodos e propriedades que as classes que herdam a classe base abstrata devem fornecer (Programming in Python 3_ A Complete Introduction to the Python Language).

Qual é a utilidade da classe abstrata? É útil pois obrigamos as classes que herdam ABC a declararem os métodos e propriedades que a classe abstrata contém.

Qual é a estrutura da classe abstrata? As classes bases abstratas precisam definir ao menos um método ou propriedade abstrata, os quais não precisam necessariamente de uma implementação nesse cenário. A implemetação concreta será construída nas subclasses. Outro ponto importante é que métodos abstratos que possuem implementações concretas (mesmo que seja apenas pass), a classe derivada pode simplesmente usar super() para usar a versão do ABC.

Implementação

```
from abc import ABC, abstractclassmethod
```

```
class Base(ABC):
    def __init__(self) -> None:
        print('Criando uma classe base')

    def metodo1(self):
        print('Chamando método 1 da Classe Base')

    def metodo2(self):
        print("Chamando metodo2 da Classe Base")
```

```
    @abstractclassmethod
    def metodo_abstrato(self):
        pass
```

Como a classe Base é uma classe abstrata, não conseguimos criar um objeto com ela, ou seja, o código a seguir apresentará erro

```
# objeto1 = Base()
```

Além disso, a subclasse que herdar Base e **não implementar o `metodo_abstrato`**, automaticamente essa subclasse não consegue criar novos objetos. Vejamos

```
class Derivada(Base):
    def metodo1(self):
        print('Chamando método1 de Derivada')
```

```
# objeto3 = Derivada()
```

Assim, quando criamos uma classe base com um método abstrato, nós **necessariamente** precisamos implementar tal método na classe base