Black drawing chalks - Life is a big holiday for us (2009)

Live de Python #68

Programação orientada a objetos #5

Roteiro

- Interfaces
- Protocolos
- Codar um pouquinho, mas bem pouco

Referências:

- Fluent Python
- http://masnun.rocks/2017/04/15/interfaces-in-python-protocols-and-abcs/

Interfaces

Vamos usar um pouco de abstração para tentar entender o que é uma interface.

- Boca
- Ouvido
- Macaneta
- Controle remoto (Dusty Phillips)

interfaces

“Interfaces são métodos ou atributos públicos que outros objetos possam usar para se **comunicar** com outros objetos”

Dusty Phillips

Interfaces (Definição informal)

“O subconjunto de métodos públicos de um objeto que lhe permitem desempenhar um papel específico em um sistema”

Ramalho

Interface (exemplo)

- Interface: Controle remoto
- Cada botão é um método
- Não me importo como isso funciona
- Eu sou um objeto
- Interagindo com outro objeto

Interfaces (problemas)

imagine uma pizza, ela não pode ter uma interface como adicionar frango (Ghiridhar)

Interfaces em Python

Interfaces (no sentido das outras linguagens) **não existem em python**. Pronto falei. Pois Python existe Herança Múltipla.

Tipagem pato (duck-typing)

> “Se um objeto quacha como um pato e voa como um pato, então é um pato”

Isso é um padrão levantado pelo que vamos chamar de protocolos, já que não temos “interfaces”

Tipagem pato (duck-typing)

```python
class Time:
    def __init__(self, membros):
        self.__membros = membros

    def __len__(self):
        return len(self.__membros)

    def __contains__(self, member):
        return member in self.__membros
```

```python
class Trem:
    def __init__(self, vagões):
        self.__vagões = vagões

    def __len__(self):
        return len(self.__vagões)

    def __contains__(self, vagão):
        return vagão in self.__vagões
```

Segregação de interfaces (Protocolos)

Na cabeça de um pythonista **protocolos e interfaces são sinônimos**. (Ramalho). Apenas, de eu mesmo, achar protocolo uma palavra bem mais elegante.

Minha definição: Protocolo de x, explica como x deve se comportar.

protocolos

Na biblioteca padrão existem diversos exemplos de protocolos, em:

- collections.abc
- numbers

Você pode se basear neles para tentar entender comportamentos “pré definidos”

protocolos

Porém, há um detalhe muito importante. Não devemos fazer fazer ‘classes’ e sim para metaclasses (sim, ainda vamos falar sobre isso em outra live). Mas o importante é entender isso, são métodos “padrões” para todo mundo que implementa o mesmo protocolo. Vamos pensar nas sequências.

(code)

collections.abc [0]

ABC - Abstract Base Classes (Classes bases abstratas). São classes “virtuais” que não herdam de nenhuma classe builtin mas fornecem as interfaces (ou métodos) necessários para que seja possível emular o comportamento dos objetos nativos do python (Sequências, Iteráveis, Containers, Corrotinas, Invocáveis, ...).

collections.abc [1]

Por exemplo, se preciso desenvolver o comportamento de um container (ou seja, usar ‘in’, será necessário implementar o método ‘__contains__’.

```
'a' in 'eduardo'  # True
'eduardo'.__contains__('a')  # True
```

collections.abc [2]

Então nosso objeto deve implementar as interfaces necessárias para que seja possível aplicar o operador ‘in’.

```
class Container:
    def __init__(self, container):
        self.data = container

    def __contains__(self, value):
        return value in self.data
```

collections.abc [3]

Porém, o Python já proporciona uma abstração necessária para esses casos, collections.abc.Container.

```
from collections.abc import Container


class MyContainer(Container):
    pass
```

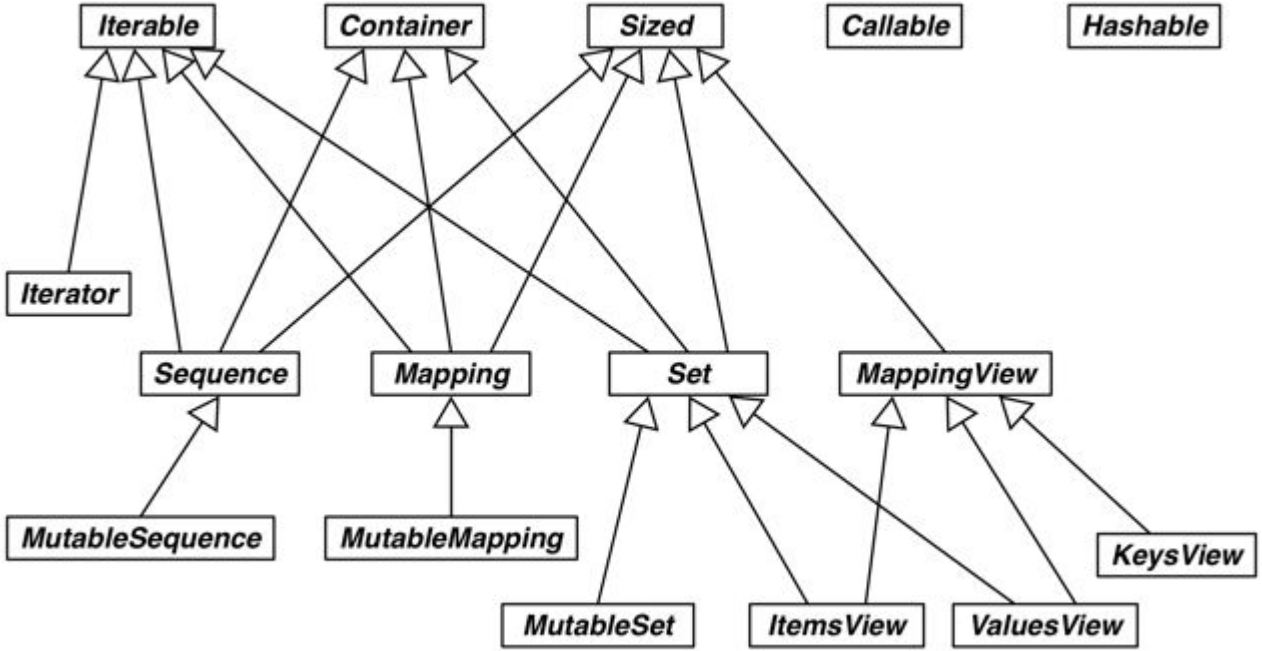
collections.abc [4]

E ao ser instanciado, nos cobra (usando abstractmethod) que essa implementação seja feita

```
In [1]: a = MyContainer()
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-7d547f1c1f50> in <module>()
----> 1 a = MyContainer()

TypeError: Can't instantiate abstract class MyContainer with abstract me
thods __contains__
```

Classes de collections.abc



FONTE: FLUENT PYTHON - Luciano Ramalho - O'Reilly