10

Para saber mais: Dictionary

Você aguenta aprender mais um pouco sobre coleções? Então vamos lá :)

Vamos lembrar rapidamente do último exemplo no vídeo quando misturamos as listas e tuples. Primeiro criamos pessoas com nome e idade usando `tuple` s:

```
pessoa1 = ("Nico", 39)
pessoa2 = ("Flavio", 37)
pessoa3 = ("Marcos", 30)
```

Depois guardamos as tuples dentro de uma lista:

```
instrutores = [pessoa1, pessoa2, pessoa3]
```

Se imprimimos `instrutores` recebemos:

```
[('Nico', 39), ('Flavio', 37), ('Marcos', 30)]
```

Para saber a idade do `Flavio` , devemos usar os índices (primeiro da lista, depois da tuple). Lembrando que acessamos o índice com os `[]` :

```
instrutores[1][1]
37
```

Agora vem o problema: E se não sabemos a posição do `Flavio` ? Em geral, como podemos descobrir a idade de um instrutor sem saber a posição dele?

Instrutor pelo nome

Repare que temos, na nossa coleção, sempre um **nome** *associado* com a **idade**. Sempre temos um par de valores, aqui é:

```
nome : idade
```

Invés de usar a posição gostaria de descobrir a idade através do nome, algo assim:

```
instrutores['Flavio']
```

Só que isso não vai funcionar com lista, nem com tuple, nem combinando os dois :(É preciso usar uma nova estrutura de dados, o **Dictionary**!

Conhecendo o dictionary

Para criar um dicionário devemos inicializar os instrutores de um modo um pouco diferente. Veja o código:

```
instrutores = {'Nico' : 39, 'Flavio': 37, 'Marcos' : 30}
```

Repare que usamos as chaves `{}` (como se fosse um set), mas sempre tem em pares. O primeiro par é `'Nico' : 39` , o segundo é `'Flavio': 37` , etc.

Nesse par, temos no lado esquerdo a **chave** e no lado direito o `valor` . Isso é importante pois usamos a chave para recuperar um valor e assim resolvemos nosso problema:

```
instrutores['Flavio']
```

Imprime:

```
37
```

Isso foi apenas uma introdução, mas não se preocupe pois veremos ainda mais sobre dicionários dentro da carreira Python 3.