

Aula 11 - Coleções Parte 3: Dicionário

Para quem já tem experiência com programação, principalmente com a linguagem JavaScript, vai sentir essa aula familiar.

Um **dicionário** no Python é um tipo de coleção similar às listas, a diferença é que agora, para cada valor do dicionário, há uma **chave** associada àquele valor. Pense em um dicionário na vida real, você tem a palavra e tem o significado dela. Funciona de forma parecida. Assim, você encontra um valor pelo elemento correspondente — diferentemente de encontrar por um índice como em uma lista. O dicionário é uma forma de organização similar a um de banco de dados, mais completa e abrangente para diversas situações. Um dicionário grande e bem estruturado lembra muito um dataset, com os dados em um formato matricial, de tabela. Isso facilita o controle e o mapeamento desses dados. Uma forma fácil de visualizar um dicionário é uma lista telefônica. Nesse caso, as chaves são os nomes, elementos que demarcam a individualidade; ao passo que os números são os valores, elementos identificadores.

Para os programadores Front-End vai ficar mais fácil de explicar, já que no JavaScript há uma estrutura quase idêntica: o **JavaScript Object Notation (Objeto de Notação JavaScript)**, também conhecido como **JSON**. A estrutura é praticamente idêntica, funcionando quase como que um "JSON do Python" mesmo. Enquanto nas listas são usados os colchetes (`[]`) e nas tuplas os parênteses (`()`), nos dicionários são usados as chaves (`{}`). Veja abaixo:

- `lista = ['Fulano', 'Cicrano', 'Beltrano']` é uma lista.
- `tupla = ('Fulano', 'Cicrano', 'Beltrano')` é uma tupla.
- `dicionario = {'nome': 'Fulano', 'idade': 40, 'profissao': 'Programador'}` é um dicionário.

Segue abaixo um exemplo de código-fonte utilizando dicionário:

```
In [ ]: # dicionário
        pessoa = {
            'nome': 'Alex Machado',
            'idade': 39,
            'profissao': 'programador',
            'empresa': 'SENAI'
        }

        print(pessoa)
```

```
{'nome': 'Alex Machado', 'idade': 39, 'profissao': 'programador', 'empresa': 'SENAI'}
```

Diferenciando os tipos de coleções

Assim como os tipos de variáveis, você pode exibir qual tipo de lista você está lidando através do comando `type()` :

```
In [ ]: # coleções
colecao1 = ['Fulano', 'Cicrano', 'Beltrano']
colecao2 = ('Brasília', 'São Paulo', 'Rio de Janeiro')
colecao3 = {'Nome': 'Fulano de Tal', 'Profissao': 'Programador', 'Genero': 'Masculi

# verificando os tipos de coleções
print(type(colecao1))
print(type(colecao2))
print(type(colecao3))

<class 'list'>
<class 'tuple'>
<class 'dict'>
```

Acessando os dados do dicionário

É claro que também posso acessar as chaves do dicionário de forma isolada, assim como nas listas, tuplas, e também no JSON (para quem veio do JS). No Python, há duas formas de se fazer isso:

Forma 1

Chamando o dicionário como se fosse uma lista, mas colocando o nome da chave dentro dos colchetes ao invés do índice:

```
In [ ]: pessoa = {
    'nome': 'Alex Machado',
    'idade': 39,
    'profissao': 'programador',
    'empresa': 'SENAI'
}

# exibindo os dados do dicionário
print(pessoa['nome'])
print(pessoa['idade'])
print(pessoa['profissao'])
print(pessoa['empresa'])
```

```
Alex Machado
39
programador
SENAI
```

Este método é bastante simples e fácil de ser implementado. Porém, ele tem uma desvantagem: se a chave informada não existir, então o código retornará um erro, e não irá executar o código. Para evitar o erro usando essa forma de acesso, será necessário usar o `try...except` :

```
In [ ]: pessoa = {
    'nome': 'Alex Machado',
    'idade': 39,
```

```
'profissao': 'programador',  
'empresa': 'SENAI'  
}  
  
try:  
    print(pessoa['cidade'])  
except:  
    print('Dados inexistentes')
```

Dados inexistentes

Forma 2

Há, porém, uma forma de evitar isso. É possível usar a função `get()` em conjunto com o nome do dicionário para o mesmo fim. Este método tem a vantagem de que, caso a chave não exista, ele não irá retornar um erro, mas sim `none`. De qualquer forma, o código será executado normalmente:

```
In [ ]: pessoa = {  
        'nome': 'Alex Machado',  
        'idade': 39,  
        'profissao': 'programador',  
        'empresa': 'SENAI'  
    }  
  
    print(pessoa.get('nome'))  
    print(pessoa.get('idade'))  
    print(pessoa.get('profissao'))  
    print(pessoa.get('empresa'))  
    print(pessoa.get('cidade'))
```

Alex Machado
39
programador
SENAI
None

Forma 3

Você pode também, caso deseje, criar uma **tupla** com o nome das chaves desejadas. Assim, você protege o dicionário contra alterações das chaves, caso precise, além de poder exibir os dados do dicionário em um laço **for**:

```
In [ ]: # tupla das chaves  
        chaves = ('Nome', 'Idade', 'Profissão', 'Empresa')  
  
        # dicionário  
        pessoa = {  
            chaves[0]: 'Alex Machado',  
            chaves[1]: 39,  
            chaves[2]: 'Programador',  
            chaves[3]: 'SENAI'  
        }  
  
        # Laço
```

```
for chave in chaves:  
    print(f'{chave}: {pessoa.get(chave)}')
```

Nome: Alex Machado
Idade: 39
Profissão: Programador
Empresa: SENAI

Adicionando uma nova chave ao dicionário

Veja que tentamos acessar uma chave que não existe no nosso dicionário. Pelo menos não ainda, porque podemos adicioná-la. O processo é bastante simples:

```
In [ ]: pessoa = {  
        'nome': 'Alex Machado',  
        'idade': 39,  
        'profissao': 'programador',  
        'empresa': 'SENAI'  
    }  
  
    # adicionando uma nova chave ao dicionário  
    pessoa['cidade'] = 'Brasília'  
  
    # exibindo os dados do dicionário  
    print(pessoa.get('nome'))  
    print(pessoa.get('idade'))  
    print(pessoa.get('profissao'))  
    print(pessoa.get('empresa'))  
    print(pessoa.get('cidade'))
```

Alex Machado
39
programador
SENAI
Brasília

Também podemos adicionar um novo valor através do `input()` :

```
In [ ]: pessoa = {  
        'nome': 'Alex Machado',  
        'idade': 39,  
        'profissao': 'programador',  
        'empresa': 'SENAI'  
    }  
  
    # adicionando uma nova chave ao dicionário  
    pessoa['cidade'] = input('Informe o nome da cidade: ')  
  
    # exibindo os dados do dicionário  
    print(pessoa.get('nome'))  
    print(pessoa.get('idade'))  
    print(pessoa.get('profissao'))  
    print(pessoa.get('empresa'))  
    print(pessoa.get('cidade'))
```

Alex Machado
39
programador
SENAI
Taguatinga

Removendo uma chave do dicionário

Além de podermos adicionar nova chave, o contrário também acontece. Podemos remover uma chave, caso ela exista dentro do dicionário:

Forma 1

```
In [ ]: pessoa = {  
    'nome': 'Alex Machado',  
    'idade': 39,  
    'profissao': 'programador',  
    'empresa': 'SENAI'  
}  
  
pessoa.pop(input('Informe a chave que deseja excluir: '), None)  
  
# exibe os novos dados na tela  
print(pessoa)
```

```
{'nome': 'Alex Machado', 'idade': 39, 'profissao': 'programador'}
```

Obs: o `None` serve para o programa não retornar um erro, caso o usuário informe uma chave inexistente, e para esse caso, é uma opção que substitui de forma mais satisfatória o `try...except`.

Forma 2

Podemos usar também o comando `del`, que tem também a vantagem de não precisarmos fazer um tratamento de exceções caso a chave informada não exista:

```
In [ ]: pessoa = {  
    'nome': 'Alex Machado',  
    'idade': 39,  
    'profissao': 'programador',  
    'empresa': 'SENAI'  
}  
  
del pessoa[input('Informe a chave que deseja excluir: ')]  
  
# exibe os novos dados na tela  
print(pessoa)
```

```
{'nome': 'Alex Machado', 'idade': 39, 'empresa': 'SENAI'}
```

Lista de dicionários

É possível também montar uma lista de dicionários. Dessa forma, é possível montar um mini-banco de dados não-relacional.

```
In [ ]: # lista de dicionários
pessoas = [
    {
        'nome': 'Fulano',
        'idade': 20,
        'profissao': 'programador'
    },
    {
        'nome': 'Cicrano',
        'idade': 25,
        'profissao': 'cientista'
    },
    {
        'nome': 'Beltrano',
        'idade': 30,
        'profissao': 'gerente de projetos'
    }
]

# imprimindo a lista
print(pessoas)
```

```
[{'nome': 'Fulano', 'idade': 20, 'profissao': 'programador'}, {'nome': 'Cicrano', 'idade': 25, 'profissao': 'cientista'}, {'nome': 'Beltrano', 'idade': 30, 'profissao': 'gerente de projetos'}]
```

Exibindo os dados de apenas uma das chaves

```
In [ ]: pessoas = [
    {
        'nome': 'Fulano',
        'idade': 20,
        'profissao': 'programador'
    },
    {
        'nome': 'Cicrano',
        'idade': 25,
        'profissao': 'cientista'
    },
    {
        'nome': 'Beltrano',
        'idade': 30,
        'profissao': 'gerente de projetos'
    }
]

# listando apenas os nomes
for pessoa in pessoas:
    print(pessoa['nome'])
```

```
Fulano
Cicrano
Beltrano
```

Exibindo todos os dados de todos da lista de forma organizada

```
In [ ]: pessoas = [
    {
        'nome': 'Fulano',
        'idade': 20,
        'profissao': 'programador'
    },
    {
        'nome': 'Cicrano',
        'idade': 25,
        'profissao': 'cientista'
    },
    {
        'nome': 'Beltrano',
        'idade': 30,
        'profissao': 'gerente de projetos'
    }
]

# listando como um banco de dados
for i in range(len(pessoas)):
    print(f'Índice: {i + 1}:')
    print(f'Nome: {pessoas[i]['nome']}')
    print(f'Idade: {pessoas[i]['idade']}')
    print(f'Profissão: {pessoas[i]['profissao']}\n')
```

Índice: 1:
Nome: Fulano
Idade: 20
Profissão: programador

Índice: 2:
Nome: Cicrano
Idade: 25
Profissão: cientista

Índice: 3:
Nome: Beltrano
Idade: 30
Profissão: gerente de projetos

Pesquisando por um dicionário dentro de uma lista

```
In [ ]: pessoas = [
    {
        'nome': 'Fulano',
        'idade': 20,
        'profissao': 'programador'
    },
    {
        'nome': 'Cicrano',
        'idade': 25,
        'profissao': 'cientista'
    },
    {

```

```

        'nome': 'Beltrano',
        'idade': 30,
        'profissao': 'gerente de projetos'
    }
]

# usuário informa o nome que deseja procurar
nome = input('Informe o nome que deseja procurar: ')

# programa retorna o resultado
for i in range(len(pessoas)):
    if nome in pessoas[i]['nome']:
        print(f'Índice: {i + 1}')
        print(f'Nome: {pessoas[i]['nome']}')
        print(f'Idade: {pessoas[i]['idade']}')
        print(f'Profissão: {pessoas[i]['profissao']}')
    else:
        continue

```

Índice: 1
 Nome: Fulano
 Idade: 20
 Profissão: programador

Alterando dados de um dicionário dentro de uma lista

```

In [ ]: pessoas = [
    {
        'nome': 'Fulano',
        'idade': 20,
        'profissao': 'programador'
    },
    {
        'nome': 'Cicrano',
        'idade': 25,
        'profissao': 'cientista'
    },
    {
        'nome': 'Beltrano',
        'idade': 30,
        'profissao': 'gerente de projetos'
    }
]

indice = int(input('Informe o índice a ser alterado: '))
campo = input('Informe o nome do campo a ser alterado: ')

indice -= 1

# usuário altera o dado do campo desejado
try:
    pessoas[indice][campo] = input(f'Informe o novo valor do campo {campo}: ')
except:
    print('Não foi possível alterar.')

# novos dados são exibidos na lista
print(f'Novos dados do índice {indice + 1}:\n')
print(f'Nome: {pessoas[indice]['nome']}')

```



```
print(f'Idade: {pessoas[indice]['idade']}')  
print(f'Profissão: {pessoas[indice]['profissao']}')
```

Novos dados do índice 1:

Nome: Fulano

Idade: 20

Profissão: atendente

Inserindo um novo dicionário da lista

```
In [ ]: pessoas = [  
    {  
        'nome': 'Fulano',  
        'idade': 20,  
        'profissao': 'programador'  
    },  
    {  
        'nome': 'Cicrano',  
        'idade': 25,  
        'profissao': 'cientista'  
    },  
    {  
        'nome': 'Beltrano',  
        'idade': 30,  
        'profissao': 'gerente de projetos'  
    }  
]  
  
# usuário informa os dados a serem inseridos  
pessoa['nome'] = input('Informe o nome: ')  
pessoa['idade'] = int(input('Informe a idade: '))  
pessoa['profissao'] = input('Informe a profissão: ')  
  
# novos dados são inseridos  
pessoas.append(pessoa)  
  
# novos dados exibidos na lista  
for i in range(len(pessoas)):  
    print(f'Índice: {i + 1}:')  
    print(f'Nome: {pessoas[i]['nome']}')  
    print(f'Idade: {pessoas[i]['idade']}')  
    print(f'Profissão: {pessoas[i]['profissao']}\n')
```

Índice: 1:
Nome: Fulano
Idade: 20
Profissão: programador

Índice: 2:
Nome: Cicrano
Idade: 25
Profissão: cientista

Índice: 3:
Nome: Beltrano
Idade: 30
Profissão: gerente de projetos

Índice: 4:
Nome: Alex
Idade: 39
Profissão: programador

Deletando um dicionário da lista

```
In [ ]: pessoas = [
    {
        'nome': 'Fulano',
        'idade': 20,
        'profissao': 'programador'
    },
    {
        'nome': 'Cicrano',
        'idade': 25,
        'profissao': 'cientista'
    },
    {
        'nome': 'Beltrano',
        'idade': 30,
        'profissao': 'gerente de projetos'
    }
]

# usuário informa o índice a ser excluído
indice = int(input('Informe o índice que deseja deletar da lista de dicionários:'))

indice -= 1

# índice é deletado
try:
    del[pessoas[indice]]
except:
    print('Não foi possível deletar o índice.')

# nova lista é exibida
for i in range(len(pessoas)):
    print(f'Índice: {i + 1}:')
    print(f'Nome: {pessoas[i]['nome']}')
    print(f'Idade: {pessoas[i]['idade']}')
    print(f'Profissão: {pessoas[i]['profissao']}\n')
```

Índice: 1:
Nome: Cicrano
Idade: 25
Profissão: cientista

Índice: 2:
Nome: Beltrano
Idade: 30
Profissão: gerente de projetos

Exercícios

1. Em programação, chamamos de sistema **CRUD** um sistema que faça 4 operações: **Create**, **Read**, **Update** e **Delete**, que significam, respectivamente: cadastrar, pesquisar, atualizar e excluir. Crie um **CRUD**, ou seja, um sistema que faça essas 4 operações em uma lista de dicionário.