Python
Dicionários

Estrutura de dados

Dicionários em Python

O Python fornece outro tipo de dados composto chamado dicionário, que é semelhante a uma lista, pois é uma coleção de objetos.

Dicionários e listas compartilham as seguintes características:

Ambos são mutáveis.
Ambos são dinâmicos. Eles podem crescer e encolher conforme necessário.
Ambos podem ser aninhados. Uma lista pode conter outra lista. Um dicionário pode conter outro dicionário. Um dicionário também pode conter uma lista e vice-versa.
Os dicionários diferem das listas principalmente em como os elementos são acessados:

- Os elementos da lista são acessados por sua posição na lista, via indexação.
- Os elementos do dicionário são acessados por meio de chaves.

Definindo um dicionário

Dicionários são a implementação de uma estrutura de dados Python que é mais conhecida como uma matriz associativa. Um dicionário consiste em uma coleção de pares de valores-chave.

Cada par de valores-chave mapeia a chave para seu valor associado.

Você pode definir um dicionário colocando uma lista separada por vírgulas de pares de valores-chave em chaves ({}). Dois pontos (:) separam cada chave do seu valor associado:

Python

```
d = {
    <key>: <value>,
    <key>: <value>,
      .
      .
      .
    <key>: <value>
}
```

Definindo um dicionário

Python >>>

```
>>> MLB_team = {
...     'Colorado' : 'Rockies',
...     'Boston'   : 'Red Sox',
...     'Minnesota': 'Twins',
...     'Milwaukee': 'Brewers',
...     'Seattle'  : 'Mariners'
... }
```

Definindo um dicionário

Você também pode construir um dicionário com a função dict () incorporada. O argumento para dict () deve ser uma seqüência de pares de valores-chave. Uma lista de tuplas funciona bem para isso:

Python >>>

```
>>> MLB_team = dict([
...     ('Colorado', 'Rockies'),
...     ('Boston', 'Red Sox'),
...     ('Minnesota', 'Twins'),
...     ('Milwaukee', 'Brewers'),
...     ('Seattle', 'Mariners')
... ])
```

Definindo um dicionário

Se os valores-chave forem cadeias simples, eles podem ser especificados como argumentos nomeados. Então, aqui está mais uma maneira de definir o MLB_team:

Python >>>

```
>>> MLB_team = dict(
...     Colorado='Rockies',
...     Boston='Red Sox',
...     Minnesota='Twins',
...     Milwaukee='Brewers',
...     Seattle='Mariners'
... )
```

Acessando os valores do dicionário

Um valor é recuperado de um dicionário especificando sua **chave** correspondente entre colchetes ([]):

Python >>>

```
>>> MLB_team['Minnesota']
'Twins'
>>> MLB_team['Colorado']
'Rockies'
```

Acessando os valores do dicionário

Se você se referir a uma **chave que não está no dicionário,** o Python levanta uma exceção:

Python >>>

```
>>> MLB_team['Toronto']
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    MLB_team['Toronto']
KeyError: 'Toronto'
```

Adicionando valores em um dicionário

Adicionar **uma entrada a um dicionário existente** é simplesmente uma questão de atribuir uma nova chave.

Python >>>

```
>>> MLB_team['Kansas City'] = 'Royals'
>>> MLB_team
{'Colorado': 'Rockies', 'Boston': 'Red Sox', 'Minnesota': 'Twins',
'Milwaukee': 'Brewers', 'Seattle': 'Mariners', 'Kansas City': 'Royals'}
```

Atualizando valores em um dicionário

Se você quiser atualizar uma entrada, basta **atribuir um novo valor a uma chave existente:**

Python >>>

```
>>> MLB_team['Seattle'] = 'Seahawks'
>>> MLB_team
{'Colorado': 'Rockies', 'Boston': 'Red Sox', 'Minnesota': 'Twins',
'Milwaukee': 'Brewers', 'Seattle': 'Seahawks', 'Kansas City': 'Royals'}
```

Deletando valores em um dicionário

Para excluir uma entrada, use a instrução del, especificando a chave a ser excluída:

Python >>>

```
>>> del MLB_team['Seattle']
>>> MLB_team
{'Colorado': 'Rockies', 'Boston': 'Red Sox', 'Minnesota': 'Twins',
'Milwaukee': 'Brewers', 'Kansas City': 'Royals'}
```

Restrições em chaves de dicionário

Quase qualquer tipo de valor pode ser usado como uma chave de dicionário no Python. integer, float e objetos booleanos são usados como chaves:

Python >>>

```
>>> foo = {42: 'aaa', 2.78: 'bbb', True: 'ccc'}
>>> foo
{42: 'aaa', 2.78: 'bbb', True: 'ccc'}
```

You can even use built-in objects like types and functions:

Python >>>

```
>>> d = {int: 1, float: 2, bool: 3}
>>> d
{<class 'int'>: 1, <class 'float'>: 2, <class 'bool'>: 3}
>>> d[float]
2

>>> d = {bin: 1, hex: 2, oct: 3}
>>> d[oct]
3
```

Restrições em chaves de dicionário

Uma chave de dicionário deve ser de um tipo imutável.

Uma tupla também pode ser uma chave de dicionário, porque as tuplas são imutáveis:

A tuple can also be a dictionary key, because tuples are immutable:

Python >>>

```
>>> d = {(1, 1): 'a', (1, 2): 'b', (2, 1): 'c', (2, 2): 'd'}
>>> d[(1,1)]
'a'
>>> d[(2,1)]
'c'
```

Restrições em chaves de dicionário

No entanto, existem algumas restrições que as chaves do dicionário devem obedecer.

Primeiro, uma determinada chave pode aparecer em um dicionário **apenas uma vez**. Chaves duplicadas não são permitidas. Um dicionário mapeia cada chave para um valor correspondente, por isso não faz sentido mapear uma chave específica mais de uma vez.

Exercícios

Escreva um programa que funcione como uma agenda
O programa deve ter um menu inicial com as seguintes opções:

- Cadastrar contato
- Listar contatos
- Mostrar um contato específico
- Atualizar um contato
- Excluir um contato
- Sair

O programa deve ficar rodando até que o usuário escolha a opção sair