Python - Dicionários de Loop

Ao contrário de uma lista, tupla ou string, o tipo de dados de dicionário em Python não é uma sequência, pois os itens não possuem um índice posicional. No entanto, percorrer um dicionário ainda é possível com diferentes técnicas.

Exemplo 1

A execução de um loop for simples sobre o objeto de dicionário percorre as chaves usadas nele.

```
numbers = {10:"Ten", 20:"Twenty", 30:"Thirty",40:"Forty"}
for x in numbers:
   print (x)
```

Ele produzirá a seguinte **saída** -

```
10
20
30
40
```

Exemplo 2

Assim que conseguirmos obter a chave, seu valor associado pode ser facilmente acessado usando o operador de colchetes ou com o método get().

```
numbers = {10:"Ten", 20:"Twenty", 30:"Thirty",40:"Forty"}
for x in numbers:
   print (x,":",numbers[x])
```

Ele produzirá a seguinte **saída** -

```
10 : Ten
20 : Twenty
30 : Thirty
40 : Forty
```

Os métodos items(), chaves() e valores() da classe dict retornam os objetos de visualiz dict_items, dict_keys e dict_values respectivamente. Esses objetos são iteradores e, po o, podemos executar um loop for sobre eles.

Exemplo 3

O objeto dict_items é uma lista de tuplas de valores-chave sobre as quais um loop for pode ser executado da seguinte maneira:

```
numbers = {10:"Ten", 20:"Twenty", 30:"Thirty",40:"Forty"}
for x in numbers.items():
   print (x)
```

Ele produzirá a seguinte **saída** -

```
(10, 'Ten')
(20, 'Twenty')
(30, 'Thirty')
(40, 'Forty')
```

Aqui, "x" é o elemento tupla do iterador dict_items. Podemos descompactar ainda mais esta tupla em duas variáveis diferentes.

Exemplo 4

```
numbers = {10:"Ten", 20:"Twenty", 30:"Thirty",40:"Forty"}
for x,y in numbers.items():
   print (x,":", y)
```

Ele produzirá a seguinte **saída** -

```
10 : Ten
20 : Twenty
30 : Thirty
40 : Forty
```

Exemplo 5

Da mesma forma, a coleção de chaves no objeto dict_keys pode ser iterada.

```
numbers = {10:"Ten", 20:"Twenty", 30:"Thirty",40:"Forty"}
for x in numbers.keys():
   print (x, ":", numbers[x])
```

As respectivas chaves e valores em dict_keys e dict_values estão no mesmo índice. No exemplo a seguir, temos um loop for que vai de 0 até o comprimento do dict e usa a variável de loop como índice e chave de impressão e seu valor correspondente.

Exemplo 6

```
numbers = {10:"Ten", 20:"Twenty", 30:"Thirty",40:"Forty"}
l = len(numbers)
for x in range(l):
   print (list(numbers.keys())[x], ":", list(numbers.values())[x])
```

Os dois trechos de código acima produzem **resultados** idênticos -

```
10 : Ten
20 : Twenty
30 : Thirty
40 : Forty
```