Listas, tuplas, set, dicionário em Python

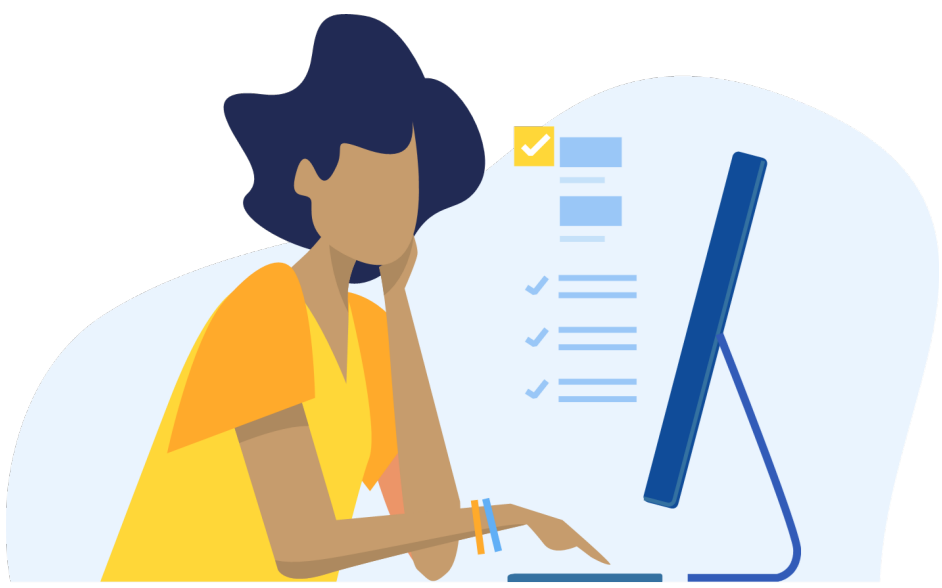
Em Python, é permitido a utilização de vários tipos de estruturas de dados. As principais estruturas são as listas, tuplas, set e dicionário. Nesta webaula vamos conhecer cada uma delas.

Listas

Lista é uma estrutura de dados do tipo sequencial que possui como principal característica ser mutável. Ou seja, novos valores podem ser adicionados ou removidos da sequência.

Na lista, os dados são posicionais e sequenciais, conforme a figura a seguir.

Fonte: elaborada pela autora.



Fonte: Shutterstock.

Observe que o primeiro valor ocupa a posição zero da lista, já o último ocupa a posição n – 1, em que *n* é a quantidade de elementos que a lista é capaz de armazenar.

Como criar uma lista em Python

Em Python, uma das maneiras de criar uma lista é colocando os valores entre colchetes, conforme código a seguir:

```
vogais = ['a', 'e', 'i', 'o', 'u']
```

A lista pode ser criada sem nenhum elemento, e a inserção pode ser feita posteriormente:

```
vogais = []
vogais.append('a')
vogais.append('e')
vogais.append('i')
vogais.append('o')
vogais.append('u')
```

Para acessar o valor guardado em uma lista, basta indicar o nome da variável e, entre colchetes, a posição do elemento, ou a fatia (slice) de valores que se deseja:

```
vogais[3]
vogais[3:]
```

List comprehension

Uma maneira muito elegante de criar uma lista é usando a *list comprehension*. Também chamada de *listcomp*, é uma forma pythônica de criar uma lista com uso de um objeto iterável.

Esse tipo de técnica é utilizado quando, dada uma sequência, deseja-se criar uma nova sequência, porém com as informações originais transformadas ou filtradas por um critério.

Fonte: Shutterstock.

Sua sintaxe básica é:
[item **for** item **in** lista]

Os comandos **for-in** são obrigatórios.

As variáveis **item** e **lista** dependem do nome dado no programa. Veja um exemplo de sintaxe utilizando a listcomp:

```
[2*x for x in range(10)]
```

Tuplas

As **tuplas** também são estruturas de dados do grupo de objetos do tipo sequência.

A grande diferença entre listas e tuplas é que, com as primeiras, uma vez que são mutáveis, conseguimos fazer atribuições a posições específicas (por exemplo, lista[2] = 'maça'), o que, nas segundas, não é possível, pois estas são objetos imutáveis.

Fonte: Shutterstock.

Em Python, uma das maneiras de criar uma tupla é colocando os valores entre parênteses, conforme código a seguir:

```
vogais = ('a', 'e', 'i', 'o', 'u')
```

Ao contrário da lista, uma tupla não permite a inserção posterior de dados, mas os dados podem ser acessados pela sua posição na sequência.

Sets

Um objeto do tipo **set** habilita operações matemáticas de conjuntos, tais como: união, intersecção, diferença, etc. Esse tipo de estrutura pode ser usado em testes de associação e remoção de valores duplicados de uma sequência.

Fonte: Shutterstock.

Em Python, uma das maneiras de se criar um objeto do tipo **set** é colocando os valores entre chaves, conforme código a seguir:

```
vogais = {'a', 'e', 'i', 'o', 'u'}
```

Um set permite a inserção de valores posteriores à sua criação, com a função add(), mas não permite acessar valores pela sua posição.

Dicionários

As estruturas de dados que possuem um mapeamento entre uma chave e um valor são consideradas objetos do tipo *mapping*. Em Python, o objeto que possui essa propriedade é o dict (dicionário). Tal objeto é mutável, ou seja, com ele conseguimos atribuir um novo valor a uma chave já existente.

Fonte: Shutterstock.

Em Python, uma das maneiras de criar um objeto do tipo dicionário é colocando as chaves e os valores entre estas, conforme código a seguir:

```
cadastro = {'nome': 'João', 'idade': 30, 'cidade': 'São Paulo'}
```

Para acessar um valor em um dicionário, basta digitar:

```
nome_dicionario[chave]
```

E, para atribuir um novo valor, use:

```
nome_dicionario[chave] = novo_valor
```

Pesquise mais

Dicionários são estruturas de dados não sequenciais que permitem o acesso a um valor por meio de uma chave. O tipo dicionário apresenta diversos métodos.

Leia, para aprofundamento, o Capítulo 6 (p. 152) da seguinte obra e veja um quadro com todos os métodos disponíveis para o tipo dict.:

BANIN, S. L. **Python 3** – conceitos e aplicações: uma abordagem didática. São Paulo: Érica, 2018.

Fonte: Shutterstock.