PeRSpectivAs

Programando o Futuro

Aumentando a base de dados sobre robôs

Informações adicionais

python™

EACH Escola de Artes, Ciências e Humanidades da Universidade de São Paulo

FUSP

PROGRAMA UNIFICADO DE BOLSAS DE ESTUDO PARA ESTUDANTES DE GRADUAÇÃO

PRCEU pró-reitoria de cultura e extensão universitária

USP

Santander Universidades

Aumentando a base de dados sobre robôs

Informações adicionais

Sobre o tipo de dado *list*

Esse tipo de dado, chamado de “listas” em português, é uma sequência de elementos que podem ou não ser do mesmo tipo. Há uma série de operações envolvendo listas que a linguagem de programação nos oferece. Alguns exemplo são:

```
# Cria uma lista sem nenhum elemento.
lista_vazia = []
```

```
# Cria uma lista de inteiros
lista_inteiros = [10, 20, 30, 40]
```

```
#Cria uma lista com diferentes tipos
lista_tipos_diferentes = ["George", "Orwell", 1984]
```

Observe que é possível criar:

- listas vazias
- listas com elementos de mesmo tipo
- listas com elementos de tipos diferentes.

Uma lista pode conter outra lista (que nesse caso é dita ser uma sublista), que por sua vez pode conter as próprias sublistas e assim por diante. Chamamos este recurso de “listas aninhadas”. Esse recurso pode ser usado para para representar estruturas muito úteis em programação, como as matrizes. Observe o exemplo:

Aumentando a base de dados sobre robôs

Informações adicionais

Nos tipos numéricos nós podemos armazenar números inteiros (*int*, de *integers*) e decimais (*floats*, de *floating-point*).

```
# Criando uma lista aninhada
lista_aninhada = [1, 2, ['a', 1], 3]
```

Uma característica muito importante do uso de listas é a capacidade de acessar um elemento específico nela armazenado. Para isso utilizamos o operador [ ], incluindo um número inteiro entre as chaves para indicar o índice (posição dentro da lista) do elemento que desejamos acessar. A contagem dos índices da lista sempre inicia em 0. Assim, se queremos acessar o primeiro elemento na lista "lista_inteiros", precisamos utilizar a seguinte operação:

```
# Acessando o primeiro elemento de uma lista
lista_inteiros[0]
```

Veja também um exemplo de como é o acesso a elementos de uma lista aninhada. Nesse caso, vamos acessar o terceiro índice da variável "lista_aninhada".

```
# Acessando o primeiro elemento de uma sublista
lista_aninhada[2][0]
```

```
'a'
```

Como o tipo de dado armazenado neste índice também é uma lista, também podemos utilizar o operador [] para acessar os seus elementos, e o primeiro elemento da sub-lista será mostrado ao executar o programa.

Aumentando a base de dados sobre robôs

Informações adicionais

Os índices de uma lista também podem ser acessados em ordem inversa. Por exemplo, para acessar o elemento de valor 40 na variável “lista_inteiros”, que se encontra na última posição da lista, podemos solicitar acesso na posição -1.

```
lista_inteiros[-1]
```

```
40
```

A possibilidade de acessos na ordem inversa pode causar um pouco de confusão inicialmente. Então, uma figura ilustrando como a ordem inversa deve ser interpretada em termos de índices pode ajudar a esclarecer esse conceito.

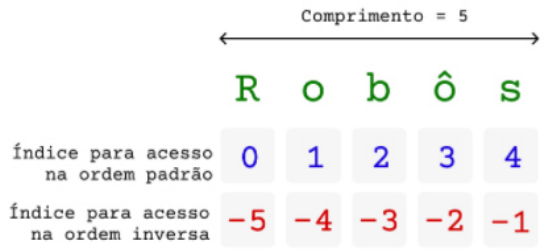

Figura 1: Ordem padrão (em azul) e ordem inversa (em vermelho) em uma lista

A tabela 1 mostra um resumo dos principais operadores para manipulação de listas em Python.

Operador	Descrição	Exemplo
+	concatena (conecta) duas listas	[1, 2, 3] + [4, 5] retorna [1, 2, 3, 4, 5]
*	repete uma lista múltiplas vezes	[0, 1] * 2 retorna [0, 1, 0, 1]
[i]	retorna o i-ésimo elemento da lista	[10, 20, 30][1] retorna 20
[i:j]	retorna a sublista que inicia no índice i e termina no índice j - 1	[1, 3, 5, 7, 9][1:3] retorna [3, 5]

Tabela 1: Principais operadores para manipulação de listas em Python

Aumentando a base de dados sobre robôs

Informações adicionais

Alguns destes operadores são iguais aos operadores algébricos também disponíveis para Python. Essa duplicidade de significados para um operador é possível porque a linguagem estabelece o significado de um operador a depender do tipo de dado com o qual o operador é usado. Se o operador + é usado com variáveis do tipo inteiro, por exemplo, o seu significado é automaticamente determinado para a soma de dois números. Se o operador + é usado com variáveis do tipo lista o seu significado é automaticamente determinado para a concatenação de duas listas.

Há diferentes maneiras de excluir um elemento de dentro de uma variável do tipo lista. A mais simples delas é usando o comando “del” e informando o índice em que se encontra o elemento que deve ser excluído. Veja:

```
#Excluindo o primeiro elemento da variável "minha_lista"
minha_lista = [1,2,3]
del minha_lista[0]
minha_lista
```

```
[2, 3]
```