N_ALG PROG_A8 - Texto de apoio

Site: EAD Mackenzie
Tema: ALGORITMOS E PROGRAMAÇÃO I {TURMA 01B} 2021/2
Livro: N_ALG PROG_A8 - Texto de apoio

Impresso por: ANDRE SOUZA OCLECIANO .
Data: terça, 28 set 2021, 20:13

Índice

1. LISTA EM PYTHON

Nós utilizamos listas frequentemente em nosso dia a dia: lista de compras, lista de convidados, lista de tarefas etc. É uma maneira de organizar os dados. Abaixo, temos um exemplo de uma lista de compras. Note que seu primeiro elemento é Cereal, o segundo é Leite, o terceiro é Banana etc.

Lista de compras
0 Cereal
1 Leite
2 Banana
3 Maçã
4 Iogurte

Fonte: Elaborado pela autora

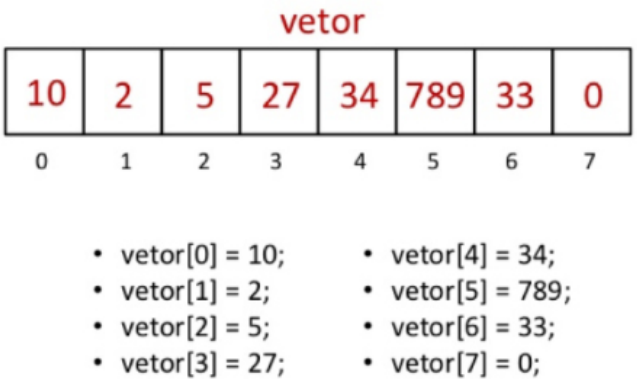
A ordem em que os elementos aparecem na lista pode ser a ordem em que o autor planeja pegar os itens no mercado, ou pode ser simplesmente a ordem em que ele anotou os itens à medida que foi se lembrando.

Na programação de computadores, é usual numerar os índices a partir do 0, por isso, nossa contagem na lista de compras começa nele.

Na programação de computadores, o conceito de lista é similar ao do nosso dia a dia.

Uma lista é uma estrutura de dados que organiza seus elementos de forma linear, ou seja, dentro dela, há o primeiro elemento, o segundo, o terceiro e assim por diante. Uma lista é também uma estrutura de dados linear que mantém a posição de seus elementos.

Em Python, utiliza-se o nome lista; em outras linguagens, utiliza-se o nome vetor.



Fonte: Elaborado pela autora.

2. APLICANDO O CONCEITO

Uma lista em Python é uma estrutura de dados linear, mutável (ou seja, pode ser modificada), de comprimento variável, que permite tipos diferentes de elementos. Ela é representada por uma lista de elementos separados por vírgula dentro de colchetes.

Exemplos de listas:

[1, 10, 3]

['cereal', 'banana', 'maçã', 'melão', 'iogurte']

[5, 'maçã', True, 10]

[ ]

Na última linha desses exemplos temos somente um par de colchetes ([ ]), o que representa uma lista vazia.

Podemos inicializar uma variável com uma lista fazendo uso do operador atribuição.

Exemplo:

lst = [1, 10, 3]

Após a atribuição, o conteúdo da lista pode ser acessado utilizando a variável.

Se quisermos imprimir todos os elementos da lista que é acessada por meio desta variável, basta utilizar a função print:

Exemplo:

```
lista_compras = ['cereal', 'leite', 'banana', 'maçã', 'iogurte']
print(lista_compras)
```

Para saber o número de itens em uma lista, usamos a função len:

Exemplo:

```
lista_compras = ['cereal', 'leite', 'banana', 'maçã', 'iogurte']
n = len(lista_compras)
print('Número de itens: ', n)
```

Este programa apresentará a saída:

```
Número de itens: 5
```

3. OPERAÇÕES EM UMA LISTA

As operações que normalmente são realizadas em uma lista são: recuperar, substituir, inserir, remover e acrescentar.

1 – Recuperar (retrieve)

É comum querermos recuperar o item que está em determinada posição na lista. Na nossa lista de compras, o item que está na posição de índice 2 é a Banana.

Lista de compras
0 Cereal
1 Leite
2 Banana
3 Maçã
4 Iogurte

Fonte: Elaborado pela autora.

Para recuperar (retrieve) elementos, nós acessamos a lista pelo índice, que é colocado entre colchetes logo após o nome da variável.

A contagem dos índices em uma lista em Python inicia-se em 0 (zero), e não em 1. Assim, se uma lista possui cinco itens, o primeiro elemento é acessado utilizando o índice 0, o último elemento é acessado utilizando-se o índice 4, e não há elemento com índice 5.

Exemplo:

```
lista_compras = ['cereal', 'leite', 'banana', 'maçã', 'iogurte']

print('Elemento na posição de índice 1:', lista_compras[1])
```

Este programa apresentará a saída:

Elemento na posição de índice 1: leite

Outro exemplo:

```
lista_compras = ['cereal', 'leite', 'banana', 'maçã', 'iogurte']

print('Primeiro item:', lista_compras[0])

print('Último item:', lista_compras[4])
```

Este programa apresentará a saída:

Primeiro item: cereal

Último item: iogurte

Podemos acessar vários elementos da lista em uma única linha de código e fazer operações com eles.

Exemplo:

```
lista_notas = [8.0, 7.5, 7.0, 9.9]

soma = lista_notas[0] + lista_notas[1] + lista_notas[2] + lista_notas[3]

print('Soma:', soma)
```

Este programa irá apresentar a saída:

Soma: 32.4

2 – Substituir (replace)

Se trocarmos o item na posição 3 da lista por Mamão, realizaremos uma substituição.

Antes	Depois
0 – Cereal	0 –Cereal
1 – Leite	1 – Leite
2 – Banana	2 – Banana
3 – Maça	**3 – Mamão**
4 – Iogurte	4 - Iogurte

Fonte: Elaborado pela autora.

Para substituir (replace), deve-se fazer a atribuição de um novo valor para uma posição específica da lista.

Exemplo:

```
lista_compras = ['cereal', 'leite', 'banana', 'maçã', 'iogurte']

print('Antes:', lista_compras)

lista_compras[3] = 'mamão'

print('Depois:', lista_compras)
```

Este programa apresentará a saída:

Antes: ['cereal', 'leite', 'banana', 'maçã', 'iogurte']

Depois: ['cereal', 'leite', 'banana', 'mamão', 'iogurte']

3 – Inserir (insert)

Suponha que o autor da lista se lembrou de que ele deve pegar mais uma fruta (Melão) antes de ir para a seção de iogurtes. Neste caso, ele inserirá na posição 4 da lista o novo item.

Antes	Depois
0 – Cereal	0 –Cereal
1 – Leite	1 – Leite
2 – Banana	2 – Banana
3 – Maça	3 – Maça
4 – Iogurte	4 – **Melão**
	5 – Iogurte

Fonte: Elaborado pela autora.

Note que o item que antes estava na posição número 4 (Iogurte) passou agora para a posição de número 5, e o número total de itens aumentou de cinco para seis.

A operação de inserir (insert) um elemento é realizada utilizando o método *insert* da lista. Para executá-lo, deve-se fazer a chamada utilizando a notação do ponto. Para isso, escreve-se:

- o nome da variável que acessa a lista seguido por um ponto (.);
- o nome do método (insert);
- entre parênteses, o número do índice da posição onde deve ser realizada a inserção e o valor a ser inserido.

Exemplo:

```
lista_compras = ['cereal', 'leite', 'banana', 'mamão', 'iogurte']

print('Antes:', lista_compras)

lista_compras.insert(4, 'melão')

print('Depois:', lista_compras)
```

Este programa apresentará a saída:

Antes: ['cereal', 'leite', 'banana', 'mamão', 'iogurte']

Depois: ['cereal', 'leite', 'banana', 'mamão', 'melão', 'iogurte']

4 – Remover (remove)

Se o autor da lista não precisar mais de Leite, ele pode remover o item da posição número 1.

Antes	Depois
0 – Cereal	0 –Cereal
1 – Leite	1 – Banana
2 – Banana	2 – Maça
3 – Maça	3 – Melão
4 – Melão	4 – Iogurte
5 - Iogurte	

Fonte: Elaborado pela autora.

Note que todos os itens que estavam abaixo do Leite na lista "subiram" uma posição.

Em Python, há algumas funções que podemos usar para remover um elemento, por exemplo, o comando *del* seguido do nome da variável e o índice do elemento que será removido.

Exemplo:

```
lista_compras = ['cereal', 'leite', 'banana', 'mamão', 'melão', 'iogurte']

print('Antes:', lista_compras)

del lista_compras[1]

print('Depois:', lista_compras)
```

Este programa apresentará a saída:

Antes: ['cereal', 'leite', 'banana', 'mamão', 'melão', 'iogurte']

Depois: ['cereal', 'banana', 'mamão', ' melão', 'iogurte']

5 – Acrescentar (append)

Caso o autor da lista queira apenas adicionar mais um item à lista (Mel) sem que isto tenha que ser feito em uma posição específica, o mais fácil é acrescentá-lo no final da lista.

Antes	Depois
0 –Cereal	0 –Cereal
1 – Banana	1 – Banana
2 – Maça	2 – Maça
3 – Melão	3 – Melão
4 – Iogurte	4 – Iogurte
	5 – Mel

Fonte: Elaborado pela autora.

A operação de acrescentar (*append*) um elemento no final da lista é feita utilizando o método append, informando qual elemento deve ser adicionado.

Exemplo:

```
lista_compras = ['cereal', 'banana', 'mamão', 'melão', 'iogurte']

print('Antes:', lista_compras)

lista_compras.append('mel')

print('Depois:', lista_compras)
```

Este programa apresentará a saída:

```
Antes: ['cereal', 'banana', 'mamão', 'melão', 'iogurte']

Depois: ['cereal', 'banana', 'mamão', 'melão', 'iogurte', 'mel']
```

4. OUTRAS OPERAÇÕES NAS LISTAS EM PYTHON

Além das operações recuperar, substituir, inserir, remover e acrescentar vistas anteriormente, em Python, temos à disposição ordenar crescente e decrescente.

6 – Ordenar crescente (sort)

O método *sort* é utilizado para alterar a posição dos elementos na lista de forma que fiquem em ordem crescente:

Exemplo:

```
lista_compras = ['cereal', 'banana', 'mamão', 'melão', 'iogurte', 'mel']

print('Antes:', lista_compras)

lista_compras.sort()

print('Depois:', lista_compras)
```

Este programa apresentará a saída:

Antes: ['cereal', 'banana', 'mamão', 'melão', 'iogurte', 'mel']

Depois: ['banana', 'cereal', 'iogurte', 'mamão', 'mel', 'melão']

Neste caso, como os elementos da lista são strings, o método sort colocou os elementos em ordem alfabética.

7 – Ordenar decrescente (reverse)

Há também o método reverse, que inverte a posição de todos os elementos.

Exemplo:

```
lista_compras = ['cereal', 'banana', 'mamão', 'melão', 'iogurte', 'mel']

print('Antes:', lista_compras)

lista_compras.reverse()

print('Depois:', lista_compras)
```

Este programa apresentará a saída:

Antes: ['cereal', 'banana', 'mamão', 'melão', 'iogurte', 'mel']

Depois: ['mel', 'iogurte', 'melão', 'mamão', 'banana', 'cereal']

5. REFERÊNCIAS

DIERBACH, C. *Introduction to Computer Science Using Python*: A Computational Problem Solving Focus. New York: Wiley, 2012.