Diferença entre Listas e Array

Slices, List Comprehension

Clenio Emidio

Senac
FACULDADE DE TECNOLOGIA E INOVAÇÃO

Introdução

A principal diferença entre listas e arrays em Python é que as listas são dinâmicas, enquanto os arrays têm um tamanho fixo.

Arrays: São coleções de elementos do mesmo tipo, com um tamanho fixo que é definido durante a declaração. Os elementos são acessados por meio de um índice numérico.

Listas: São coleções dinâmicas de elementos do mesmo tipo, cujo tamanho pode ser alterado dinamicamente durante a execução do programa.

As diferenças entre lista e array em Python são também na performance e nas funcionalidades disponíveis. O **array** usa menos memória do que a lista, é mais rápido nas operações de leitura e possibilita executar operações matemáticas sobre todos os itens de uma só vez. Além disso, no **array** todos os elementos têm que ter o mesmo tipo de dado, mas a **lista** pode ter elementos de tipos diferentes.

Outra diferença é que a lista é uma coleção embutida no Python e o array precisa ser importado do módulo array da biblioteca padrão, ou do pacote Numpy.

Para declarar uma lista, é só colocar os itens entre colchetes [].
Já o array precisa ser inicializado com uma das funções
array.array() ou numpy.array().

```python
#  Lista de strings
lista = ['x', 'y', 'z', '8',]

#  Lista de inteiros
lista = [1, 1, 2, 3, 5]

#  Lista vazia
lista = []

#  Lista com tipos de dados diferentes
lista = [False, 'False', 0]

#  Lista aninhada (lista dentro de lista)
lista = [1, 'Texto', [False, 0], []]


minha_lista = ['Olá', 123, 'mundo', 3.1]
minha_lista.append('Python')
print(minha_lista)

```

```
PROBLEMS 6  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

PS C:\Users\Clenio Emidio\Desktop\python> python -u "c:\Users\Clenio Emidi
p\python\testesaelista.py"
['Olá', 123, 'mundo', 3.1, 'Python']
PS C:\Users\Clenio Emidio\Desktop\python>
```

```python
import numpy as np

meu_array = np.array([1, 2, 3, 4, 5])
meu_array = meu_array * 2
print(meu_array)

```

```
PROBLEMS 6  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Clenio Emidio\Desktop\python> python -u "c:\U
p\python\array3.py"
[ 2  4  6  8 10]
PS C:\Users\Clenio Emidio\Desktop\python>
```

array4.py > ...

```python
from array import array

meu_array = array('i', [1, 2, 3, 4, 5])
print(meu_array)
```

```
PROBLEMS 7  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Clenio Emidio\Desktop\python> python -u "c:
p\python\array4.py"
array('i', [1, 2, 3, 4, 5])
PS C:\Users\Clenio Emidio\Desktop\python>
```

Qual estrutura de dados você deve usar, lista ou array?

Use uma lista para coleções que tenham poucos elementos, que precisem de muitas inclusões e exclusões, ou que tenham elementos com tipos de dados diferentes. Se a coleção tiver milhares de elementos, ou se precisar realizar operações matemáticas sobre os elementos, use um array.

É preferível usar o pacote Numpy em vez do módulo array, sempre que possível.

código usando o pacote Numpy é quase 50 vezes mais rápido que a versão que usa uma lista.

Slices

Slices em Python são uma forma de selecionar partes específicas de uma lista, matriz ou sequência. Essa funcionalidade é extremamente útil para a manipulação de dados em Python, permitindo que sejam selecionados elementos específicos de uma estrutura de dados de forma rápida e eficiente.

Slices

Slices são denotados pelo uso de colchetes **[] e dois pontos :** para especificar a faixa de elementos que você deseja selecionar. Como se fossemos selecionar um elemento da lista pelo seu índice, mas são usados dois pontos.

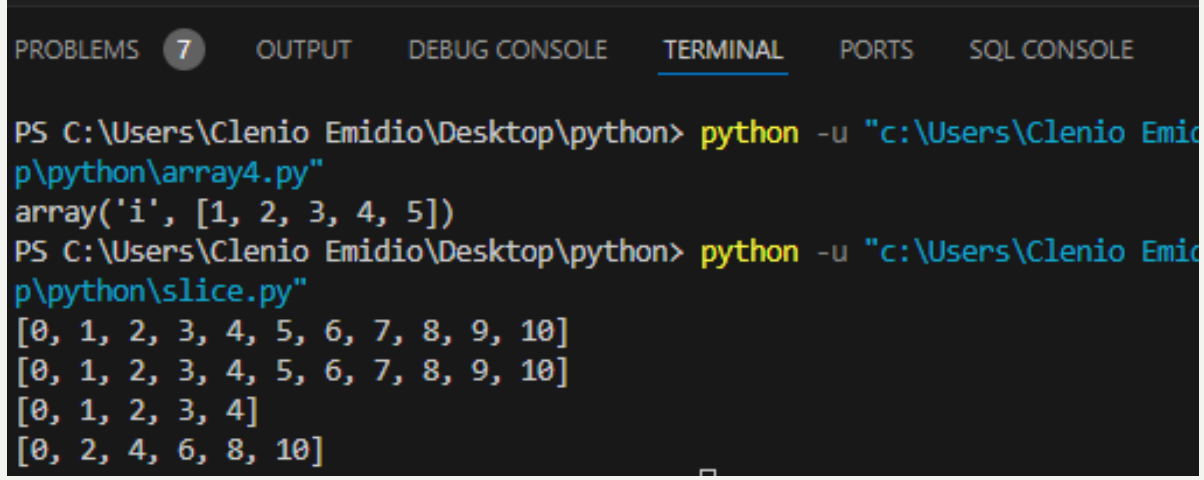
A sintaxe geral é a seguinte: **objeto[inicio:fim:passo],** onde início é o índice inicial da seleção, fim é o índice final (exclusivo) da seleção e passo é a quantidade de elementos a serem pulados. Veja o exemplo abaixo:

```python
sequencia_numerica = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

primeiros_cinco = sequencia_numerica[:5]

pares = sequencia_numerica[::2]

print(sequencia_numerica)
print(sequencia_numerica[::])
print(primeiros_cinco)
print(pares)
```



No código ao lado, vemos a variável **sequencia numérica**, que recebe uma lista de inteiros de 0 a 10, e abaixo mais duas variáveis, também listas, derivadas da primeira que utilizam o slicing para gerar novas listas a partir de uma original. Seguindo a sintaxe descrita, temos a lista **primeiros_cinco**, que recebe uma slice de sequencia numérica que vai da posição inicial 0 (omitida) até a posição final 5 (explícita) com o passo de 1 (não pula nenhum índice).

Já variável **pares** recebe uma slice onde tanto o início quanto o fim estão omitidos, ou seja, do começo da lista até o fim dela, e possui um passo de 2 (seleciona os índices de 2 em 2)

List Comprehension

List comprehensions permitem criar listas de uma forma mais concisa e legível, tornando o código mais fácil de ser entendido e mantido. Essa funcionalidade permite criar uma nova lista a partir de uma lista existente ou outra estrutura de dados, aplicando uma expressão a cada elemento.

Sintaxe:

```
[<expressao> for item in list]
```

Aplique expressão para todo item da lista list.

List Comprehension

Expressao é qualquer sentença da linguagem que retorne um valor. Por exemplo, elevar ao quadrado retorna o resultado da potência;

item representa cada elemento da lista a ser formada/manipulada. A expressão precisa ser compatível com o elemento.

list é a lista a ser criada/manipulada. Por mais que a lista já exista, essa manipulação criará uma nova lista, pois estamos copiando a lista original e modificando cada elemento de acordo com a expressão

```python
lista = []

for el in range(1, 10):
    lista.append(el * 10)
print(lista)

#PODE SER REESCRITA ASSIM:

lista2 = [el*10 for el in range(1, 10)]
print(lista2)
```

```
PROBLEMS 7  OUTPUT  DEBUG CONSOLE  TERMINAL  PORT

PS C:\Users\Clenio Emidio\Desktop\python> python -u "c
o\python\lsitac.py"
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
PS C:\Users\Clenio Emidio\Desktop\python>
```

Você pode, ainda, inserir condicionais na forma de instruções if em list comprehensions para limitar seu set de resultados. Por exemplo:

```python
pares = [el for el in range(10) if el % 2 == 0]

print(pares)
```

```
PROBLEMS 7  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

PS C:\Users\Clenio Emidio\Desktop\python> python -u "c:\Users\Clenio
o\python\LISTAC1.PY"
[0, 2, 4, 6, 8]
PS C:\Users\Clenio Emidio\Desktop\python>
```

https://dev.to/iugstav/python-manipulacao-de-listas-e-matrizes-4bpj