Python - O básico de listas

O básico de lista em Python

Abaixo temos um exemplo de uma lista:

```
animals = ["pangolin", "cassowary", "sloth", "dog"];
animals[0]   # 'pangolin'
animals[1]   # 'cassowary'
animals[2]   # 'sloth'
animals[3]   # 'dog'
```

Contamos o índice da lista a partir do zero, veja:

```
+-------------+-------------+-------------+-------------+-------------+
| "pangolin"  | "cassowary" |   "sloth"   |    "dog"    |lanca um erro|
+-------------+-------------+-------------+-------------+-------------+
      0             1             2             3            4
```

Ao acessar um índice inexistente recebemos um erro.

```
animals[4]   # IndexError: list index out of range
```

Podemos fatiar (leia-se acessar) um lista de diversas formas, veja a matéria sobre fatiamento de sequencias (/python/sequencias-fatiamento/).

Para criar uma lista

Podemos criar uma lista vazia.

```
lis = []
```

Podemos criar uma lista com alguns itens separados por vírgula.

```
lis = ['a', 'b', 'c']
```

Podemos criar uma lista através do “list comprehension”

```
[x for x in iterable]
```

Ou também podemos utilizar a função interna (built in) list() (/python/built-in/list/).

```
lis = list() # equivalente a l = []
lis = list(['a', 'b', 'c']) # equivalente a l = ['a', 'b, 'c']
```

Percorrendo a lista (list looping)

Abaixo vemos um exemplo de como percorrer um lista na sua forma mas simples.

```
myList = [1, 2, 3, 4]
for number in myList:
    print number * 2
# 2
# 4
# 6
# 8
```

Abaixo incrementamos a implementação com a função interna enumarate() (/python/built-in/enumerate). ela irá numerar a lista.

```
choices = ['pizza', 'pasta', 'salad', 'nachos']

print 'Your choices are:'
for index, item in enumerate(choices):
    print index, item

"""
Your choices are:
0 pizza
1 pasta
2 salad
3 nachos
"""
```

Copiando listas

Se quisermos copiar uma lista podemos realizar uma simples atribuição, mas manteremos uma relação entre as duas, ou melhor, ambas apontam para o mesmo objeto e as alterações em uma afetará a outra.

```
list_a = [6, 7, 8, 9]
list_b = lista_a
```

Temos uma cópia mas ambas apontam para o mesmo objeto.

Clonando listas

Se quisermos copiar uma lista sem manter a referência entre elas, podemos utilizar o operador de fatia `[:]`.

```
list_a = [6, 7, 8, 9]
list_b = lista_a[:]
```

Temos cópias independentes, um clone.

Juntando listas (join lists)

```
m = [1, 2, 3]
n = [4, 5, 6]
o = m + n
print(o) # [1, 2, 3, 4, 5, 6]

o += [7, 8, 9]
print(o) # [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Funções nativas para listas

append()

```
nums = ["um"]
print(nums) # ['um']

nums.append("dois")
nums.append("tres")
nums.append("quatro")
print(nums) # ['um', 'dois', 'tres', 'quatro']
```

index()

A função `index()` retorna o index de determinado elemento.

```
animals = ["ant", "bat", "cat"]
print(animals.index("bat")) # 1
```

Se você procurar por um item que não existe um erro será lançado.

```
print(animals.index("dog")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: 'dog' is not in list
```

Por tanto, se o seu objetivo é saber se um item pertence a lista utilize o operador de teste de inlclusão `in` .

```
>>> animals = ["ant", "bat", "cat"]
>>> 'dog' in animals
False
>>> 'cat' in animals
True
```

insert()

```
animals = ["ant", "bat", "cat"]
animals.insert(1, "dog")
print(animals) # ["ant", "dog", "bat", "cat"]
```

remove()

Remove através do valor

```
animals = ["ant", "bat", "cat"]
animals.remove("ant")
print(animals) # ["bat", "cat"]
```

pop()

Remove através do índice (retorna o valor removido).

```
animals = ["ant", "bat", "cat"]
animals.pop(0) # 'ant'
print(animals) # ["bat", "cat"]
```

Semelhante a utilização de `del()`:

```
animals = ["ant", "bat", "cat"]
del(animals[0])
print(animals) # ["bat", "cat"]
```

sort()

```
lista = ["c", "b", "a"]
print(lista) # ['c', 'b', 'a']

lista.sort()
print(lista) # ['a', 'b', 'c']
```

Para ordenar uma lista também é possível utilizar a função interna sorted() (https://docs.python.org/3.4/library/functions.html#sorted) , exemplo:

```
sorted([5, 2, 3, 1, 4]) # [1, 2, 3, 4, 5]
```

Uma observação importante é que a função `sort()` não retorna a lista, então…

```
lista = ["c", "b", "a"]
print(lista.sort()) # None
```

… já a função `sorted()` retorna a lista `['a', 'b', 'c']` veja:

```
lista = ["c", "b", "a"]
print(sorted(lista))
# ['a', 'b', 'c']
```

g

(https://github.com/devfuria)

www.devfuria.com.br (/)

desde 2012