>> UNIVERSIDADE FEDERAL DO PARÁ <<
>> INSTITUTO DE TECNOLOGIA <<
>> RAMO ESTUDANTIL IEEE UFPA <<
>> WOMEN IN ENGINEERING UFPA <<

>> GRUPO DE ESTUDOS EM PYTHON <<
#WIECODEPYTHON

>> SEMANA #4 <<
>> LISTAS <<
>> APOSTILA DE REFERÊNCIA <<

>> Semana #4 - Listas

Listas são estruturas de dados capazes de armazenar múltiplos elementos.

Declaração

Para a criação de uma lista, basta colocar os elementos separados por vírgulas dentro de colchetes [ ], como no exemplo abaixo:

```
>>> nomes_frutas = ["maçã", "banana", "abacaxi"]
>>> nomes_frutas
['maçã', 'banana', 'abacaxi']
>>>
>>> numeros = [2, 13, 16, 47]
>>> numeros
[2, 13, 16, 47]
```

A lista pode conter elementos de tipos diferentes:

```
>>> ['lorem ipsum', 150, 1.3, [-1, -2]]
['lorem ipsum', 150, 1.3, [-1, -2]]
>>>
>>> vazia = []
>>> vazia
[]
```

Índices

Assim como nas strings, é possível acessar separadamente cada item de uma lista a partir de seu índice:

```
>>> lista = [100, 200, 300, 400, 500]
>>> lista[0] # os indices sempre começam em zero
100
>>> lista[2]
300
>>> lista[4]
500
>>> lista[-1]
500
>>> #outra maneira de acessar o ultimo elemento
```

Conforme visto anteriormente, ao utilizar um índice negativo os elementos são acessados de trás pra frente, a partir do final da lista:

```
>>> lista[-2] #penultimo elemento
400
>>> lista[-4] #terceiro
200
>>> lista[-2] #penultimo elemento
400
>>> lista[-3] #terceiro
300
>>> lista[-4] #segundo
200
>>> lista[-5] #primeiro
100
```

Ou pode-se acessar através de ***slices*** *(fatias)*, como nas *strings*:

```
>>> lista[2:4] # da posição 2 ate a 4 (nao inclusa)
[300, 400]
>>> lista[:3] # ate a posição 3 (nao inclusa)
[100, 200, 300]
>>> lista[2:] # da posição 2 ate o final
[300, 400, 500]
>>> lista[:] # do começo ate o fim
[100, 200, 300, 400, 500]
```

Tentar acessar uma posição inválida de uma lista causa um erro:

```
>>> lista[10]
Traceback (most recent call last):
  File "<pyshell#43>", line 1, in <module>
    lista[10]
IndexError: list index out of range
>>> lista[-10]
Traceback (most recent call last):
  File "<pyshell#44>", line 1, in <module>
    lista[-10]
IndexError: list index out of range
```

Podemos avaliar se os elementos estão na lista com a palavra ***in***:

```
>>> lista_estranha = ['duas palavras', 42, True, ['batman', 'robin'], -0.87, 'hi
pofise']
>>> 42 in lista_estranha
True
>>> 'duas palavras' in lista_estranha
True
>>> 'domino' in lista_estranha
False
```

É possível obter o tamanho da lista utilizando o método len( ):

```
>>> len(lista)
5
>>> len(lista_estranha)
6
```

Removendo itens da lista

Devido à lista ser uma estrutura mutável, é possível remover seus elementos utilizando o comando `del`:

```
>>> lista_estranha
['duas palavras', 42, True, ['batman', 'robin'], -0.87, 'hipofise']
>>> del lista_estranha[2]
>>> lista_estranha
['duas palavras', 42, ['batman', 'robin'], -0.87, 'hipofise']
>>> del lista_estranha[-1]
>>> lista_estranha
['duas palavras', 42, ['batman', 'robin'], -0.87]
```

Trabalhando com listas

O operador `+` concatena listas:

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> c
[1, 2, 3, 4, 5, 6]
```

O operador `*` repete a lista dado um número de vezes:

```
>>> [0] * 3
[0, 0, 0]
>>> [1, 2, 3] * 2
[1, 2, 3, 1, 2, 3]
```

O método `append( )` adiciona um elemento ao final da lista:

```
>>> lista = ['a', 'b', 'c']
>>> lista
['a', 'b', 'c']
>>> lista.append('e')
>>> lista
['a', 'b', 'c', 'e']
```

Temos também o `insert( )`, que insere um elemento na posição especificada e move os demais elementos para direita:

```
>>> lista.insert(3, 'd')
>>> lista
['a', 'b', 'c', 'd', 'e']
```

extend( ) recebe uma lista como argumento e adiciona todos seus elementos a outra:

```
>>> lista1 = ['a', 'b', 'c']
>>> lista2 = ['d', 'e']
>>> lista1
['a', 'b', 'c']
>>>
>>> lista2
['d', 'e']
>>> lista1.extend(lista2)
>>> lista1
['a', 'b', 'c', 'd', 'e']
```

lista2 não é modificado:

```
>>> lista2
['d', 'e']
```

O método sort( ) ordena os elementos da lista em ordem ascendente:

```
>>> lista_desordenada = ['b', 'z', 'k', 'a', 'h']
>>> lista_desordenada
['b', 'z', 'k', 'a', 'h']
>>> lista_desordenada.sort()
>>> lista_desordenada
['a', 'b', 'h', 'k', 'z']
```

Para fazer uma cópia de uma lista, devemos usar o método copy( ):

```
>>> lista1 = ['a', 'b', 'c']
>>> lista2 = lista1.copy()
>>> lista1
['a', 'b', 'c']
```

```
>>> lista2
['a', 'b', 'c']
>>> lista2.append('d')
>>> lista1
['a', 'b', 'c']
>>> lista2
['a', 'b', 'c', 'd']
```

Se não usarmos o copy( ), acontece algo bem estranho:

```
>>> lista1 = ['a', 'b', 'c']
>>> lista2 = lista1
>>> lista1
['a', 'b', 'c']
>>> lista2
['a', 'b', 'c']
>>> lista2.append('d')
>>> lista1
['a', 'b', 'c', 'd']
>>> lista2
['a', 'b', 'c', 'd']
```

Tudo o que for feito com **lista2** nesse exemplo também altera **lista1** e vice-versa.

Função range( )

Aprendemos a adicionar itens a uma lista mas, e se fosse necessário produzir uma lista com os números de 1 até 200?

```
>>> lista_grande = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
>>> lista_grande
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
```

Em Python existe a função embutida `range( )`, com ela é possível produzir uma lista extensa de uma maneira bem simples:

```
>>> print(list(range(1, 200)))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 10
2, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 11
8, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 13
4, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 15
0, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 16
6, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 18
2, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 19
8, 199]
```

Além disso, o `range( )` também oferece algumas coisas interessantes. Por exemplo, imprimir os números espaçados de 5 em 5, entre 0 e 30:

```
>>> print(list(range(0, 30, 5)))
[0, 5, 10, 15, 20, 25]
```

Mas, como na maior parte das vezes apenas queremos uma lista começando em 0 e indo até o número desejado, a função range( ) também funciona da seguinte maneira:

```
>>> print(list(range(10)))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Nota: O intervalo do range( ) é aberto, ou seja, quando passamos o valor 10, ele vai até o 9 (n - 1). Caso deseje criar a lista até o 10 de fato, deve-se passar o valor 11.

Mas por que precisamos transformar o range( ) em list? O que acontece se não fizermos isso?

```
>>> print(range(0, 200))
range(0, 200)
```

Mas o que é que essa função retorna?

```
>>> type(range(200))
<class 'range'>
```

A função range( ) retorna algo do tipo range, por isso precisamos transformar em uma lista para vermos todos os números no print( ).