



Introdução à Programação para Sensoriamento Remoto

Aula 07 – Introdução à Programação com a Linguagem Python

**Gilberto Ribeiro de Queiroz
Thales Sehn Körting
Fabiano Morelli**



04 de Abril de 2018

Tópicos

- Sequências: strings, tuplas e listas.

Sequências: Strings, Tuplas e Listas

Sequências

- Uma sequência é um conjunto ordenado de n valores:

$$a_0, a_1, a_2, \dots, a_{n-1}$$

- Cada elemento de uma sequência é associado a um número: **índice** ou **posição**.
- O primeiro índice é o **zero**.
- Os três tipos básicos de sequências são:
 - **Strings**: sequência de caracteres.
 - **Tuplas**: sequência imutável de valores (ou itens).
 - **Listas**: sequência de valores (ou itens), que pode crescer, encolher, ou alterar elementos.

Sequências: Operações Comuns

Operador	Descrição
<code>x in s</code>	<code>True</code> se um item de <code>s</code> for igual a <code>x</code> , <code>False</code> caso contrário
<code>x not in s</code>	<code>False</code> se um item de <code>s</code> for igual a <code>x</code> , <code>True</code> caso contrário
<code>s + t</code>	Concatenação de <code>s</code> e <code>t</code>
<code>s[i]</code>	<i>i</i> -th item de <code>s</code>
<code>s[i:j]</code>	Parte da sequência no intervalo <code>[i, j)</code>
<code>len(s)</code>	Comprimento da sequência <code>s</code>
<code>min(s)</code>	Menor item da sequência <code>s</code>
<code>max(s)</code>	Maior item da sequência <code>s</code>
<code>s.index(x)</code>	Índice da primeira ocorrência de <code>x</code>
<code>s.count(x)</code>	Número total de ocorrências de <code>x</code> em <code>s</code>

Para maiores detalhes veja a [documentação oficial do Python](#).

Strings: Sequência de Caracteres

```
01 nome = "Gilberto Ribeiro"
02 letra = "b"

03 if letra in nome:
04     print("Nome contém a letra 'b'!")
05 else:
06     print("Nome não contém a letra 'b'!")

07 tamanho = len(nome)
08 print(tamanho)

09 pos = nome.index("t")
10 print(pos)
```

Tuplas

- As tuplas são expressas através de uma sequência cujos itens são separados por vírgula e, delimitados ou não por parênteses.



```
01 # Coordenadas do centróide da Cidade de São Paulo/Brasil
02 centroide_sp = (-46.7165, -23.6830)
03 print(centroide_sp)
04 print("longitude: {} latitude: {}".format(*centroide_sp))
05 print( len(centroide_sp) )
06 longitude = centroide_sp[0]
07 latitude = centroide_sp[1]
```

Tuplas

- As tuplas podem ser aninhadas (*nested*):

```
01 # retângulo envolvente mínimo (rem) da Cidade de São Paulo
02 rem_sp = ( (-46.8254, -24.0084), (-46.3648, -23.6830) )
03 canto_inferior_esquerdo = rem_sp[0]
04 canto_superior_direito = rem_sp[1]
05 print("xmin:", canto_inferior_esquerdo[0])
```



Tuplas

- Se tentarmos alterar um item de uma tupla, será lançada uma exceção:

```
>>> centroide_sp[1] = 45.0
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'tuple' object does not support item assignment

Listas

```
01 cidades = ["São Paulo", "Rio de Janeiro",  
02            "Belo Horizonte", "Ouro Preto"]
```

```
03 print(cidades)
```

Note que o **método sort** não cria uma nova lista, ele realiza a alteração in-place.

```
04 # Ordenando a lista
```

```
05 cidades.sort()
```

```
06 print(cidades)
```

```
07 # Gera uma nova lista "ordenada ao contrário"
```

```
08 nova_lista = sorted(cidades, reverse=True)
```

```
09 print(nova_lista)
```

```
10 print(cidades)
```

A **função sorted** cria uma nova lista a partir da lista de entrada.

Listas

```
01 cidades.append("São José dos Campos")
02 print(cidades)

03 del cidades[1]
04 print(cidades)

05 cidades.extend( [ "Ouro Preto", "Mariana" ] )
06 print(cidades)

07 cidades.reverse()
08 print(cidades)
```

Listas: outras formas de construção

```
01 l_letras = list( "Gilberto" )
02 print(l_letras)

03 primos = list( (1, 2, 3, 5, 7) )
04 print(primos)

05 seq1 = list( range(10) )
06 print(seq1)

07 seq2 = list( range(3, 10) )
08 print(seq2)

09 lista_vazia = []
10 print(lista_vazia)
```

Listas: List Comprehension

```
01 f_ident = [ x for x in range(0, 10) ]
02 print(f_ident)

03 f_quad = [ x**2 for x in range(0, 10) ]
04 print(f_quad)

05 f_exp = [ 2**x for x in range(0, 10) ]
06 print(f_exp)
```

Considerações Finais

Considerações Finais

- O tipo de dados lista é uma das estruturas mais utilizadas da linguagem Python.
- Cada item de uma lista é identificado por um índice.
- Listas são sequências mutáveis, isto é, podem encolher ou incluir novos elementos.
- Demos os primeiros passos para utilizar um idioma muito comum da programação em Python:
 - *List Comprehension*.

Referências Bibliográficas

Referências Bibliográficas

- [Common Sequence Operations](#). Acesso: Março, 2018.
- [Mutable Sequence Types](#). Acesso: Março, 2018.
- [Lists](#). Acesso: Março, 2018.

Exercícios