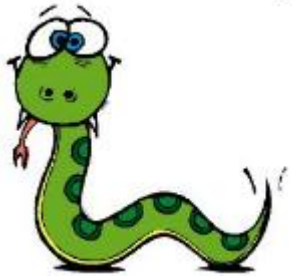
UFRPE
Universidade Federal Rural de Pernambuco



BSI Sistemas de Informação UFRPE

Listas

Prof. Wylliams Barbosa Santos
wylliamss@gmail.com
Introdução à Programação

Crédito de Conteúdo: Professora Ceça Moraes

UFRPE
Universidade Federal Rural de Pernambuco

Agenda

BSI Sistemas de Informação UFRPE

- Listas
  - Conceitos
  - Operações
  - Métodos
  - Exercícios

Listas

- Listas são coleções **heterogêneas** de objetos, que podem ser de qualquer tipo, inclusive outras listas.

- As listas no Python são **mutáveis**, podendo ser alteradas a qualquer momento
  - é possível fazer atribuições a itens da lista

- Listas podem ser **"fatiadas"** da mesma forma que as ***strings***

Listas

- Uma lista é na verdade um objeto da classe chamada **`list`**

- Na verdade, pode ser vista como uma implementação de ***arrays***
  - Acesso **seqüencial** e **direto** através de índices

Listas

- Listas são **variações de seqüências** assim como strings e portanto têm **APIs semelhantes**
  - Podem ser indexadas e fatiadas
  - Podem ser **concatenadas (+)** e **repetidas**

Operações

- # Uma nova lista: grupos dos anos 70

```
>>> progs = ['Yes', 'Genesis', 'Pink Floyd', 'ELP']
```

- # Varrendo a lista inteira

```
>>> for p in progs:
        print p


Yes
Genesis
Pink Floyd
ELP
```

Atribuições


```
>>> lista = [2, 28, 9, 'ceca', 78, 12]
>>> lista[0]= 33
>>> lista
[33, 28, 9, 'ceca', 78, 12]
>>> lista[-1] = 'teste'
>>> lista
[33, 28, 9, 'ceca', 78, 'teste']
>>> lista[3] = 99
>>> lista
[33, 28, 9, 99, 78, 'teste']
```

Operações

- Trocando elementos

```
>>> progs[-1] = 'King Crimson'

>>> progs[2] = 'Rolling Stones'
>>> for p in progs:
        print p


Yes
Genesis
Rolling Stones
King Crimson
```

Operações

- Incluindo elementos

```
King Crimson
>>> progs.append('Camel')
>>> for p in progs:
        print p


Yes
Genesis
Rolling Stones
King Crimson
Camel
```

Operações

- Removendo elementos (por valor)

```
>>> progs.remove('Camel')
>>> for p in progs:
        print p


Yes
Genesis
Rolling Stones
King Crimson
```

Operações

- Removendo elementos (por posição)

```
>>> del progs[2]
>>> progs
['Yes', 'Genesis', 'King Krimson']
```

Operações

Ordenando a lista:

```
>>> progs.sort()

>>> for p in progs:
        print p


Genesis
King Crimson
Rolling Stones
Yes
```

Invertendo a lista:

```
>>> progs.reverse()

>>> for p in progs:
        print p


Yes
Rolling Stones
King Crimson
Genesis
```

Operações

- “fatiando”

```
>>> print progs[1:]
['Rolling Stones', 'King Crimson', 'Genesis']
>>> print progs[:2]
['Yes', 'Rolling Stones']
>>> print progs[1:3:2]
['Rolling Stones']
>>> print progs[0:3:2]
['Yes', 'King Crimson']
```

Observações

- A operações de ordenação (***sort***) e inversão (***reverse***) são realizadas na própria lista, alterando-a

Mais Operações


```
>>> elemento = [1,2,3,4,5]
>>> sum(elemento)
15
>>> len(elemento)
5
>>> max(elemento)
5
>>> min(elemento)
1
```

Métodos

- **`extend(lista2)`**
  - Acrescenta os elementos de **`lista2`** ao **final da lista**
  - Altera a lista original

```
>>> lista=[1,2]
>>> lista.extend([3,4])
>>> lista
[1, 2, 3, 4]
```

Métodos

- `count(elemento)`
  - Retorna **quantas vezes** o elemento aparece na lista

```
>>> lista2 = [1,2,3,1,8,12,7]
>>> lista2.count(1)
2
```

Métodos

- **`index(elemento)`**
  - Retorna o índice da **primeira ocorrência** de **`elemento`** na lista
  - Um **erro** ocorre se **`elemento`** não consta da lista

```
>>> lista3 = [9,8,33,12]
>>> lista3.index(33)
2
>>> lista3.index(22)

Traceback (most recent call last):
  File "<pyshell#49>", line 1, in <module>
    lista3.index(22)
ValueError: list.index(x): x not in list
```

Métodos

- **`insert(índice, elemento)`**
  - Insere **`elemento`** na lista na posição indicada por **`índice`**
  - Altera a lista original

```
>>> lista4 = [0,1,2,3]
>>> lista4.insert(1,'dois')
>>> lista4
[0, 'dois', 1, 2, 3]
```

Métodos

- `pop(índice)`
  - **Remove** da lista o elemento na posição **índice** e o **retorna**
  - Se índice **não for mencionado**, é assumido o **último**

```
>>> lista6 = [1,2,3,4]
>>> lista6.pop(1)
2
>>> lista6
[1, 3, 4]
>>> lista6.pop()
4
>>> lista6
[1, 3]
```

String: método `split()`

- Separa uma string em uma lista de strings menores
- Recebe como parâmetro um caractere separador e um número máximo de pedaços (opcional)
- Retorna uma lista de strings, são os pedaços da string original divididos pelo separador.
- Não altera a string original.

String: método split()

```
>>> 'www.eupodiatamatando.com'.split('.')
['www', 'eupodiatamatando', 'com']
>>> '19:16:23'.split(':')
['19', '16', '23']
>>> hora, minuto, segundos = '19:16:23'.split(':')
>>> hora
'19'
>>> minuto
'16'
>>> segundos
'23'
```

EXERCÍCIO

Exercícios

1. Criar uma agenda telefônica para armazenar e remover valores itens.

Bibliografia

- Livro “Como pensar como um Cientista de Computação usando Python” – Capítulo 8
  - http://pensarpython.incubadora.fapesp.br/portal
- Python Tutorial
  - http://www.python.org/doc/current/tut/tut.html
- Dive into Python
  - http://www.diveintopython.org/
- Python Brasil
  - http://www.pythonbrasil.com.br/moin.cgi/DocumentacaoPython#head5a7ba2746c5191e7703830e02d0f5328346bcaac