SCC 124 - Introdução à Programação para Engenharias

Listas

python

Professor: André C. P. L. F. de Carvalho, ICMC-USP
Pos-doutorando: Isvani Frias-Blanco
Monitor: Henrique Bonini de Britto Menezes

Aula de hoje

- Introdução
- Sequências
- Classes e objetos
- Listas
  - Operações em listas
- Pilhas e filas

Introdução

- Estruturas de dados
  - Estruturas para o armazenamento de dados
- Existem outras estruturas de dados pré-construídas na linguagem Python
  - Range
  - Listas
  - Tuplas

  } Sequências
  - Dicionários
  - Conjuntos

Sequências especiais

- Armazenam coleção ordenada de valores relacionados
- Python possui 2 tipos de sequências especiais, manipulados como objetos
  - Tipo sequência de texto
    - Possui métodos para criar e manipular strings
  - Tipo sequência binária
    - Possui métodos para manipular tipos bytes e bytearray

Sequências

- Principais características
  - Teste de pertinência
    - Operadores *in* e *not in*
  - Operações de indexação
    - Permitem acesso direto a um item de uma sequência
      - Operador de subscrição ([posição do item])
  - Operação de fatiamento
    - Permite recuperar uma fatia de uma sequência

Listas

- Coleção ordenada de valores
  - Valores são chamados elementos ou itens
  - EX.: *minhaLista = ['Fisica', 'Programacao']*
- Valores podem ser de qualquer tipo e tipos podem estar misturados
  - Ex.: [3, 54, 2], ['casa', 'pe'], [3, 'b', 3, 2.1]
  - Lista sem elementos é uma lista vazia
    - Ex.: []

Listas

- Tipo de dado mutável (pode ser alterado)
  - Pode mudar conteúdo e tamanho
- Podem implementar vetores e matrizes
  - *Strings* são imutáveis
- Existem várias semelhanças entre *strings* e listas
  - Ex.: operações de indexação e fatiamento

Listas

- Inclui métodos para
  - Consultar (procurar) itens em uma lista
  - Alterar
    - Composição de uma lista
    - Ordem dos elementos em uma lista
  - Adicionar itens a uma lista
  - Remover itens de uma lista

Classes e objetos

- É um tema por si só
- Mas alguns conceitos básicos são importantes para entender listas
  - Uma lista é um exemplo de uso de classes e objetos
    - Tipos simples também
    - Ex.: comando *x = 6*

      cria um objeto *x* da classe (tipo) *int*

Classes e objetos

- Uma classe pode ter métodos (funções) relativos àquela classe apenas
  - Esses métodos podem ser usados apenas por objetos daquela classe
  - Ex.: Para a classe lista, Python possui o método *append*
    - Adiciona itens ao final da lista
    - Ex.: *minhaLista.append ("Calculo")*

Classes e objetos

```
>>> minhaLista = ['Fisica', 'Programacao']
>>> minhaLista

>>> minhaLista.append ("Calculo")
```


Classes e objetos

```
>>> minhaLista = ['Fisica', 'Programacao']
>>> minhaLista
['Fisica', 'Programacao']
>>> minhaLista.append ("Calculo")
['Fisica', 'Programacao', 'Calculo']
```

Classes e objetos

- Uma classe pode também conter campos
  - Variáveis definidas para aquela classe
  - Podem ser usadas apenas pelos objetos da classe
  - Ex.: *minhaLista.cursoAtual ("SCC 124")*

Variável local

Relembrando...

- Uma classe é como um tipo
  - Pode conter métodos
    - Que são funções
  - Pode conter campos
    - Equivale a variáveis
- Um objeto é como um valor de uma classe

Exemplo

```
# Minha lista de compras
listaCompras = ['abacate', 'manga', 'cenoura', 'banana']
print('Eu tenho', len(listaCompras), 'itens para comprar.')
print('Esses itens sao', end=' ')
for item in listaCompras:
    print(item, end=' ')
print('\n Eu tambem preciso comprar rucula.')
listaCompras.append('rucula')
print('Minha lista de compras agora eh ', listaCompras)
print('Vou ordenar minha lista por ordem alfabetica')
listaCompras.sort()
print('Minha lista de compras ordenada eh ', listaCompras)
print('O primeiro item que eu vou comprar eh ', listaCompras[0])
itemAntigo = listaCompras[0]
del listaCompras[0]
print('Eu comprei', itemAntigo)
print('Minha lista de compras agora eh ', listaCompras)
```


Exemplo

```
# Minha lista de compras
listaCompras = ['abacate', 'manga', 'cenoura', 'banana']
print('Eu tenho', len(listaCompras), 'itens para comprar.')
print('Esses itens sao', end=' ')
for item in listaCompras:
    print(item, end=' ')
```

Objeto

, end ' '
Ao invés de pular de linha, no próximo *print*, inclui espaço após o *string*

Saida:

Eu tenho 4 itens para comprar.
Esses itens sao abacate manga cenoura banana

listaCompras = ['abacate', 'manga', 'cenoura', 'banana']

Exemplo

```
# Minha lista de compras
...
print('\n Eu tambem preciso comprar rucula.')
listaCompras.append('rucula')
print('Minha lista de compras agora eh ', listaCompras)
print('Vou ordenar minha lista por ordem alfabetica')
listaCompras.sort()
print('Minha lista de compras ordenada eh ', listaCompras)
```

Saida:

Eu tambem preciso comprar rucula
Minha lista de compras agora eh abacate manga cenoura banana rucula
Vou ordenar minha lista por ordem alfabetica
Minha lista de compras ordenada eh abacate banana cenoura manga rucula

listaCompras = ['abacate', 'banana', 'cenoura', 'manga', 'rucula']

Exemplo

```
# Minha lista de compras
...
print('O primeiro item que eu vou comprar eh ', listaCompras[0])
itemAntigo = listaCompras[0]
del listaCompras[0]
print('Eu comprei', itemAntigo)
print('Minha lista de compras agora eh ', listaCompras)
```

Saida:

O primeiro item que eu vou comprar eh abacate
Eu comprei abacate
Minha lista de compras agora eh ['banana', 'cenoura', 'manga', 'rucula']

listaCompras = ['banana', 'cenoura', 'manga', 'rucula']

Métodos para listas

- Supor um objeto do tipo (classe) lista, denominado *lta*
  - Classe lista possui vários métodos para:
    - Consultar objetos de uma lista
    - Alterar objetos de uma lista
    - Expandir uma lista
    - Reduzir uma lista

Métodos para consultar listas

- *lta.count (x)*
  - Retorna quantas vezes valor *x* aparece na lista *lta*
- *lta.index (x)*
  - Retorna posição em que o valor *x* aparece em *lta*
- *lta[i]*
  - Retorna item na posição i da lista *lta*
- *len(lta)*
  - Retorna quantos itens existem na lista *lta*

Exemplo

```
>>> lta = [66.25, 333, -1, 333, 1, 1234.5, 333, 0, 1, 2]
>>> print lta.count(333), lta.count(66.25), lta.count('x')

>>> lta.index(333)

>>> a = lta[2]
>>> a

>>> len(lta)
```


Exemplo

```
>>> lta = [66.25, 333, -1, 333, 1, 1234.5, 333, 0, 1, 2]
>>> print lta.count(333), lta.count(66.25), lta.count('x')
3 1 0
>>> lta.index(333)
1
>>> a = lta[2]
>>> a
-1
>>> len(lta)
10
```


Métodos para alterar listas

- *lta.sort ()*
  - Ordena os elementos da lista em ordem crescente
- *lta.reverse ()*
  - Inverte a ordem dos elementos da lista *lta*
- *lta[i] = x*
  - Armazena valor de x na lista *lta*

Exemplo

```
>>> lta

>>> lta.reverse()
>>> lta

>>> lta.sort()
>>> lta

>>> lta[2] = 3
>>> lta
```

Exemplo

```
>>> lta
[66.25, -1, 333, 1, 1234.5, 333, 0, 1]
>>> lta.reverse()
>>> lta
[1, 0, 333, 1234.5, 1, 333, -1, 66.25]
>>> lta.sort()
>>> lta
[-1, 0, 1, 1, 66.25, 333, 333, 1234.5]
>>> lta[2] = 3
>>> lta
[-1, 0, 3, 1, 66.25, 333, 333, 1234.5]
```


Métodos para expandir listas

- *lta.append* (*x*)
  - Anexa item *x* ao final da lista *lta*
    - *lta.append (x)* equivale a *lta[len(lta):] = [x]*
- *lta.extend (ltb)*
  - Anexa lista de elementos *ltb* ao final da lista *lta*
    - *lta.extend (ltb)* equivale a *lta[len(lta):] = ltb*
- *lta.insert (i, x)*
  - Insere item *x* na i-ésima posiçao da lista *lta*
    - *lta.insert (len(lta), x)* equivale a *lta.append (x)*

Exemplo 1

```
>>> lta = [66.25, 333, 333, 1, 1234.5]
>>> lta

>>> lta.insert(2, -1)
>>> lta.append(333)
>>> lta

>>> ltb = [0, 1, 2]
>>> lta.extend (ltb)
>>> lta
```


Exemplo 1

```
>>> lta = [66.25, 333, 333, 1, 1234.5]
>>> lta
[66.25, 333, 333, 1, 1234.5]
>>> lta.insert(2, -1)
>>> lta.append(333)
>>> lta
[66.25, 333, -1, 333, 1, 1234.5, 333]
>>> ltb = [0, 1, 2]
>>> lta.extend (ltb)
>>> lta
[66.25, 333, -1, 333, 1, 1234.5, 333, 0, 1, 2]
```


Exemplo 2

```
>>> def fun (a, L=[]):
...         L.append(a)
...         return L

>>> print (fun(1)))

>>> print ((fun(2))

>>> print (fun(3))
```


Exemplo 2

```
>>> def fun (a, L=[]):
...         L.append(a)
...         return L

>>> print (fun(1))
[1]
>>> print ((fun(2))
[1,2]
>>> print (fun(3))
[1,2,3]
```

Exercício

- Escrever um módulo em Python que:
  - Recebe como entrada uma lista de números
  - Envia a lista para uma função que soma os valores presentes na lista
  - Retorna soma a quem chamou a função
  - Imprime valor da soma

Métodos para reduzir listas

- *lta.remove (x)*
  - Remove primeiro item da lista *lta* cujo valor é igual a *x*
    - Se lista não possui elemento com valor *x*, ocorre um erro
- *lta.pop ([i])*
  - Remove e retorna item de *lta* na posição *i*
  - Colchete indica que parâmetro é opcional
    - Se posição não é especificada, é removido e retornado último elemento da lista *lta*

Exemplo

```
>>> lta = [66.25, 333, -1, 333, 1, 1234.5, 333, 0, 1, 2]
>>> lta.remove(333)
>>> lta

>>> lta.pop ()
>>> lta
```


Exemplo

```
>>> lta = [66.25, 333, -1, 333, 1, 1234.5, 333, 0, 1, 2]
>>> lta.remove(333)
>>> lta
[66.25, -1, 333, 1, 1234.5, 333, 0, 1, 2]
>>> lta.pop ()
>>> lta
[66.25, -1, 333, 1, 1234.5, 333, 0, 1]
```


Exercício

- Dada a lista do exercício anterior, escrever um módulo em Python que:
  - Retira da lista todo objeto cujo valor for menor que 20% da soma dos valores
  - Envia a nova lista para uma função que soma os valores presentes na lista
  - Retorna soma a quem chamou a função
  - Imprime valor da soma

Pilhas e filas

- Elementos podem seguir uma ordem para entrar ou sair de listas
  - Pilhas
    - Último elemento adicionado é o primeiro a ser removido (LIFO)
  - Filas
    - Primeiro elemento adicionado é o primeiro a ser removido (FIFO)

Pilhas

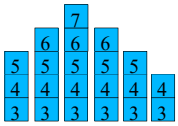
- Métodos do tipo listas facilitam seu uso como uma pilha
  - Para incluir um item, usar método o *append()*
  - Para recuperar um item, usar o método *pop()*

Exemplo

```
>>> pilha = [3, 4, 5]
>>> pilha.append(6)
>>> pilha.append(7)
>>> pilha

>>> pilha.pop()

>>> pilha

>>> pilha.pop()

>>> pilha.pop()

>>> pilha
```

Exemplo

```
>>> pilha = [3, 4, 5]
>>> pilha.append(6)
>>> pilha.append(7)
>>> pilha
[3, 4, 5, 6, 7]
>>> pilha.pop()
7
>>> pilha
[3, 4, 5, 6]
>>> pilha.pop()
6
>>> pilha.pop()
5
>>> pilha
[3, 4]
```



Filas

- Métodos de listas listas também facilitam uso de filas
  - Para incluir item, usar o método *append()*
  - Para recuperar item, usar o método *pop(0)*

Exemplo

```
>>> fila = ["Pedro", "Joao", "Jose"]
>>> fila.append("Luiz")          # Luiz chegou
>>> fila.append("Mario")         # Mario chegou
>>> fila.pop(0)

>>> fila.pop(0)

>>> fila
```


Exemplo

```
>>> fila = ["Pedro", "Joao", "Jose"]
>>> fila.append("Luiz")          # Luiz chegou
>>> fila.append("Mario")         # Mario chegou
>>> fila.pop(0)
'Pedro'
>>> fila.pop(0)
'Joao'
>>> fila
['Jose', 'Luiz', 'Mario']
```

Exemplo

```
L = [1,2,3]
for i in range(0,len(L)):
    print (i)    # imprime o indice

L = [1,2,3]
for i in range(0,len(L)):
    print (L[i])    # imprime o elemento da lista de indice i
```


Conclusão

- Tipos sequência
- Listas
- Classes e objetos
- Inclusão de elementos em listas
- Remoção de elementos em listas
- Consulta de elementos de listas
- Pilhas e filas

Perguntas

Exercício

- Suponha que você esta montando uma fábrica
  - Precisa comprar itens nas quantidades e preços:
    - 10 máquinas empacotadoras, a 1500,00 cada
    - 100 kits de ferramentas, a 130,00 cada
    - 80 vestimentas a 120,00 cada
    - 40 mesas a 600,00 cada
  - Você tem 30.000,00 para gastar

Exercício

- Implementar um módulo em Python para:
  - Receber os itens, com quantidades e valores e armazenar em uma lista
  - Definir que itens vai comprar com os recursos que você tem
  - Gerar uma nova lista apenas com os recursos que você vai comprar
  - Imprimir a lista, o total gasto e o valor que sobrou
  - Usando pilha e depois usando fila