Algoritmos e Estrutura de Dados

Bruno Feres de Souza
bferes@gmail.com

Universidade Federal do Maranhão
Bacharelado em Ciência e Tecnologia

1° semestre de 2016

Na aula anterior...

Dados e Tipos de Dados

- Um **dado** é uma informação que um algoritmo recebe ou manipula
- Exemplos de dados são nomes, datas, valores (preços, notas, etc.) e condições (verdadeiro e falso)
- Todo dado é de um certo **tipo** que define sua natureza (p. ex., um nome é diferente de um valor), identificando seu uso, e define as operações que podem ser realizadas com o dado
- Por exemplo, podemos somar dois valores numéricos, mas não podemos somar um número e uma frase

Dados e Tipos de Dados

- Em Python:
  - Tipos de dados **atômicos**:
    - **int e float: +, -, *, /, %, ****
    - **bool: and, or, not**
  - Tipos de dados de **coleção**:
    - **Listas**
    - Tuplas
    - String
    - Dicionários

Listas

Por que utilizar?

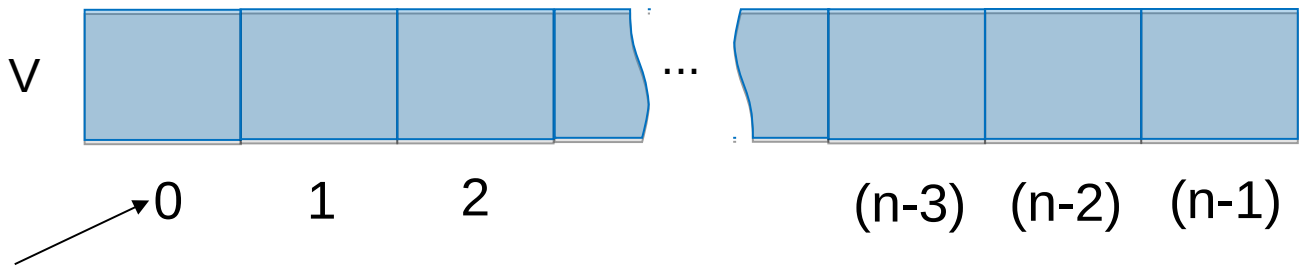
- Problema 1:
  - Dada uma relação de 5 notas, imprimir a nota de cada estudante se esta for maior que a média da classe.
  - Usar seis variáveis do tipo `float` é viável.
  - Faça o código em Python.

- Problema 2:
  - Problema 1 com uma turma do BCT.
  - Usar 61 variáveis do tipo `float` é inviável.
  - Use listas do Python.

Listas

Definição

- Uma **lista** é uma coleção ordenada de zero ou mais valores, de um mesmo tipo ou não.
- Cada valor pode ser acessado por um índice dentro do lista.
- Listas são dinâmicas: podem crescer ou diminuir com o tempo.



Listas

Criando listas

- Para criar uma lista, utilizam-se colchetes []
- Primeiro tipo: **criação explícita**
  - Lista vazia:
    ```
    L = []
    ```
  - Lista com elementos do mesmo tipo:
    ```
    L = [1,2,3]
    ```
  - Lista com elementos de tipos distintos:
    ```
    L = [True,2,'3']
    ```

Listas
Criando listas

- Segundo tipo: **criação implícita ou compreensão de listas**

1. L = [expressão for variável in sequência]
   - Calcular a expressão para cada elemento.

- Exemplo:

```
L = [x**2 for x in range(10)]
```

  - » Leia assim: “faça x ao quadrado para todo x de 0 a 9”.
  - Como seria utilizando o for?

Listas
Criando listas

- Segundo tipo: **criação implícita ou compreensão de listas**

2. L = [expressão for variável in sequência if condição]
   - Selecionar elementos que verificam uma condição.

• Exemplo:

```
L = [x**2 for x in range(10) if x%3==0]
```

» Leia assim: “faça x ao quadrado para todo x de 0 a 9 que seja múltiplo de 3”.

- Como seria utilizando o for?

Listas

Criando listas

- Python cria uma nova lista sempre que [] for usado.

```
A = []
B = []
C = D = [] #C e D são a mesma lista!
C = []; D=[] #C e D são independentes!
```

Listas

Acessando listas

- `len` retorna o número de itens da lista `L`.

```
L = [3,2,7,4,1]
n = len(L)
```

- `L[i]` retorna o iésimo iten da lista `L`.

```
L = [3,2,7,4,1]
item = L[4]
```

- `L[inicio:fim]` retorno os elementos do início ao fim de `L`. Isto chama-se **fatiamento** de listas.

```
L = [3,2,7,4,1]
seq = L[1:4]
```

Listas

Acessando listas

- L[inicio:fim:n] retorna os elementos de inicio a fim em passos de n.

```
L = [3,2,7,4,1]
seq = L[::2] #elemento sim, elemento não.
seq = L[1::2] #elemento sim, elemento não, a partir de 1.
```

- **Observações**:
  - Se inicio faltar, usa a primeira posicao da lista.
  - Se fim faltar, usa o tamanho da lista.
  - Se n faltar, usa 1 para incrementar.
  - Se um índice i for negativo, o Python faz len(L)-i.

```
L = [3,2,7,4,1]
L[-1] #Acessa o último elemento da lista.
```

  - Se o passo n for negativo, é mais claro utilizar explicitamente o inicio maior que o fim.

```
L = [3,2,7,4,1]
L[4:1:-1] #Faz o fatiamento [1,4,7]
L[::-1] #Faz o fatiamento [1,4,7,2,3]
```

Listas

Iterando listas

- Utilizando o for-in: iterar pelos itens

```
L = [3,2,7,4,1]
for item in L:
    print item
```

- Utilizando o for-in: iterar pelos índices

```
L = [3,2,7,4,1]
for index in range(len(L)):
    print index
```

- Utilizando o for-in: iterar pelos itens e pelos índices

```
L = [3,2,7,4,1]
for index, item in enumerate(L):
    print index, item
```

Listas

Modificando listas

- Modificando o iésimo item da lista L

```
L = [3,2,7,4,1]
L[4] = 8
```

- Modificando uma sequência de itens da lista L

```
L = [3,2,7,4,1]
L[1:3] = ['a','b']
L[3:]  = ['z']
```

- append: adicionar um item ao final da lista L

```
L = [3,2,7,4,1]
L.append(9)
L = L + [8]
```

Listas

Modificando listas

- **Atenção**: M e L representam a mesma lista!

```
L = []
M = L
L.append(3)
```

- **Atenção**: M e L não representam a mesma lista!

```
L = []
M = L[:]
L.append(3)
```

Listas

Modificando listas

- extend: adiciona uma sequência ao final da lista L

```
L = [3,2,7,4,1]
L.extend(['a','b','c'])
```

Observação: como o comando acima difere do comando abaixo?

```
L.append(['a','b','c'])
```

- insert: insere um item em uma posição da lista L

```
L = [3,2,7,4,1]
L.insert(1,9)
```

Listas

Modificando listas

- **del**: remove da lista **L** um item ou todos os itens de um fatiamento

```
L = [3,2,7,4,1]
del L[3]
del L[1:4]
```

- **pop**: remove e retorna um item individual da lista **L**.

```
L = [3,2,7,4,1]
Item = L.pop()
Item = L.pop(0)
Item = L.pop(3)
```

Listas

Modificando listas

- reverse: inverte os itens de uma lista L.

```
L = [3,2,7,4,1]
L.reverse()
```

Listas

Buscando em listas

- O operador **in** verifica se um determinado valor está na lista **L**.

```
L = [3,2,7,4,1]
if 7 in L:
    print('O numero estah na lista.')
else:
    print('O numero nao estah na lista.')
```

Listas

Buscando em listas

- **index**: retorna o índice da primeira ocorrência de um valor na lista L.
  - L = [3,2,7,4,1,7]

```
i = L.index(7) #leia “busque 7”.
print(i)
i = L.index(7,2) #leia “busque 7 a partir da posicao 2”.
print(i)
```

- E se o item não estiver na lista?

```
L = [3,2,7,4,1,7]
try:
    i = L.index(8)
except ValueError:
    i = -1
```

Listas

Buscando em listas

- **count**: retorna quantas vezes um dado valor aparece na lista L.

```
L = [3,2,7,4,1,7]
i = L.count(7)
print(i)
```

- **max** e **min**: retorna os valores máximo e mínimo de uma lista L.

```
L = [3,2,7,4,1,7]
m = max(L)
n = min(L)
```

Listas

Ordenando listas

- sort: ordena os elementos de uma lista L.

```
L = [3,2,7,4,1,7]
L.sort()
print(L)
```

Observação: sort modifica a lista L!

- sorted: retorna uma nova lista com os valores ordenados da lista L.

```
L = [3,2,7,4,1,7]
L2 = sorted(L)
print(L)
print(L2)
```

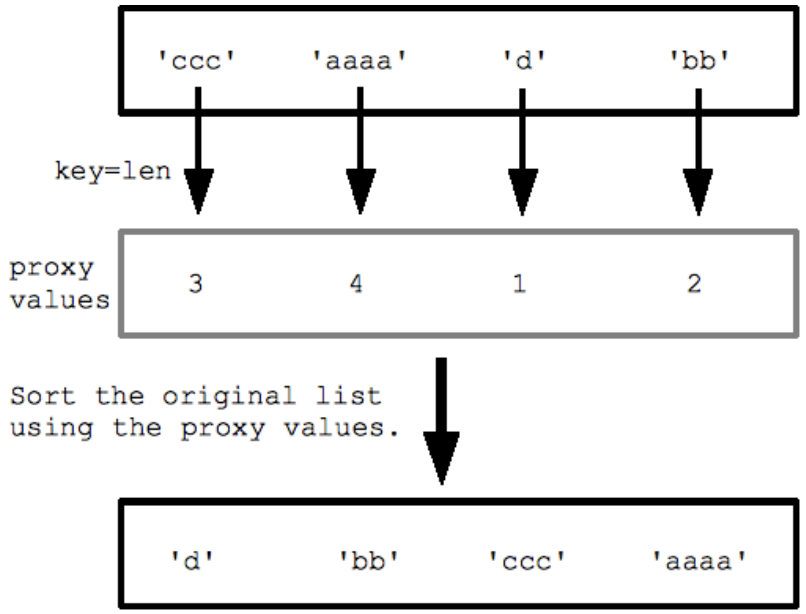
- **Observação**: sorted não modifica a lista L!

Listas

Ordenando listas

- Ordenação personalizada usando **key**: aplica uma função “key” a cada iten da lista **L** e depois ordena pelos itens transformados.

```
L = ['ccc', 'aaaa', 'd', 'bb']
L2 = sorted(L, key=len)
print(L2)
```



Listas
Ordenando listas

- Ordenação personalizada usando `key`: podemos escrever nossa própria função `key`.

```
def minhaKey(nome):
    return nome[-1]

L = ['ccc', 'aaaa', 'd', 'bb']
L2 = sorted(L, key=minhaKey)
print(L2)
```

Listas

Imprimindo listas

- **print**: imprime uma string contendo a representação textual da lista **L**.

```
L = [3,2,7,4,1]
print(L)
```

- **join**: retorna uma lista com os itens da lista **L** unidos por um separador especificado.

```
L1 = ['a','bb','ccc','dddd','eeee'] #os itens são strings!
print( '-'.join(L1) )
print( '@'.join(L1) )
L2 = [3,2,7,4,1] #os itens são números
print( '-'.join(L2) ) #Errado!
print( '-'.join(str(v) for v in L2) )#Converte-se cada item para string e usa-se compreensão de listas.
```

Listas

Exercícios em laboratório

- Dada uma lista L de números inteiros, escreva uma função que retorne uma lista P contendo os números pares de L. Observação: utilize criação implícita de listas.

- Dada uma lista L de números inteiros, escreva uma função que imprima todos os números de índices múltiplos de 3. Observação: utilize fatiamento de listas.

Listas

Exercícios em laboratório

- Dada uma lista L de números inteiros, escreva uma função retorne uma lista P contendo todos os itens de L acrescidos em 10%.

- Dada uma lista L de números inteiros, escreva uma função que imprima os itens de L de trás para frente, sem utilizar reverse().

Listas

Exercícios em laboratório

- Dada uma lista L de números inteiros, escreva uma função que retorne uma lista P contendo os itens de L sem repetição

- Dada uma lista L de números inteiros, escreva uma função que imprima os índices de todas as ocorrências de um número especificado.

Listas

Exercícios em laboratório

- Dada uma lista L de números inteiros, escreva uma função que retorne uma lista P com os valores ordenados de P, agrupados em números pares primeiro e ímpares depois.

Material complementar

http://effbot.org/zone/python-list.htm

https://developers.google.com/edu/python/lists

https://pythonhelp.wordpress.com/2013/06/26/brincando-com-listas/

Dúvidas?