INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Listas em Python

ALG – Algoritmos e Programação

Aula 13

Curso Técnico em Informática para Internet – Integrado ao Ensino Médio

INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Objetivos

- Introduzir o conceito de variáveis compostas.
- Apresentar o tipo de dado Lista (Python), usado para a construção de variáveis compostas.
- Desenvolver programas em Python usando Listas.

Introdução

- As variáveis de um programa podem ser simples ou compostas.
- Uma variável simples é capaz de armazenar apenas um valor em um dado momento.
- Exemplos de variáveis simples:
  - idade, salario, data_nascimento, nome_cliente, endereço, etc.
- Uma variável composta pode armazenar um conjunto de valores ao mesmo tempo.
- Por exemplo, uma variável composta é capaz de armazenar o nome, endereco, salario e o total_horas_trabalhadas de um funcionário de uma empresa.
  - Tudo em uma única variável!

O que é uma lista?

- Variável composta heterogênea que armazena um conjunto de valores (variáveis).
- Os valores que compõem uma lista são chamados de elementos.
- Por ser heterogênea, os elementos de uma lista podem ser de tipos diferentes (inclusive pode ser outra lista).
- Cada elemento em uma Lista é identificado por meio de um índice.

Definição de listas em Python

- Uma lista em Python é declarada da seguinte forma:
  - Nome_lista = [valor1, valor2, ..., valorN]

- Exemplo de lista:
  - L = [145, ‘Maria da Silva’, ‘Rua das Gardênias’, 38, 18562.60]
  - Lista_vazia=[ ]

- Para acessar um elemento qualquer da lista, usamos o **índice** que indica a posição do elemento na lista.

Acessando os elementos de uma lista

- Em Python, o primeiro elemento de uma lista está na posição **zero**.
- Por exemplo, na lista a seguir o elemento 145 encontra-se na posição de índice zero, enquanto que o elemento 38 está na posição de índice 3.
  - L = [145, 'Maria da Silva', 'Rua das Gardênias', 38, 18562.60]
- Para acessar um elemento qualquer de uma lista, usamos o nome da lista e o valor do índice (em que se encontra o elemento) entre colchetes.
- Exemplos:

```
>>> L = [145, 'Maria da Silva', 'Rua das Gardênias', 38, 18562.60]
>>> print(L[0])
145
>>> print(L[1])
Maria da Silva
>>> print(L[3] + 10)
48
```

Modificando os elementos de uma lista

- As listas em Python são mutáveis, podendo ser alteradas a qualquer momento.

- Por exemplo, se quisermos alterar o elemento da posição de índice 3 da lista L (slide anterior), podemos fazer a seguinte atribuição:
  - L[3] = ‘São Carlos’

- Ao imprimir a nova lista, o resultado será:

```
>>> L[3] = 'São Carlos'
>>> print(L)
[145, 'Maria da Silva', 'Rua das Gardênias', 'São Carlos', 18562.6]
```

Percorrendo uma lista

- Para percorrer uma lista utilizamos uma estrutura de repetição, por exemplo, a estrutura while:

```
cavaleiros = ['guerra', 'fome', 'peste', 'morte']
i = 0
while i < len(cavaleiros):
    print(cavaleiros[i])
    i = i+1
```

Outra forma de percorrer uma lista

- Outra maneira de percorrer uma lista é usar a estrutura de repetição for.

```
cavaleiros = ['guerra', 'fome', 'peste', 'morte']
for cavaleiro in cavaleiros:
    print(cavaleiro)
```

Principais operações com listas

- Verificar se um elemento pertence a uma lista
- Verificar o comprimento de uma lista
- Inserir um novo elemento em uma lista
- Remover um elemento de uma lista
- Concatenar listas

Elementos pertencentes a uma lista

- O operador lógico **in** é usado para verificar se um elemento pertence a uma lista.
  - O resultado é sempre verdadeiro (True) ou falso (False)

- Exemplo:

```
>>> cavaleiros = ['guerra', 'fome', 'peste', 'morte']
>>> 'fome' in cavaleiros
True
>>> 'guerrilheiros' in cavaleiros
False
```

Verificar o comprimento de uma lista

- A função **len()** retorna o comprimento de uma lista.
- Exemplo:

```
>>> cavaleiros = ['guerra', 'fome', 'peste', 'morte']
>>> len(cavaleiros)
4
```

- Pode ser usada como condição de parada em um laço de repetição, ao invés de uma constante. Exemplo:

```
cavaleiros = ['guerra', 'fome', 'peste', 'morte']
i = 0
while i < len(cavaleiros):
    print(cavaleiros[i])
    i = i+1
```

Inserir um novo elemento no final de uma lista

- A função **append()** adiciona um novo elemento no final de uma lista.

- Exemplo:

```
cavaleiros = ['guerra', 'fome', 'peste', 'morte']
cavaleiros.append('guerrilheiros')
print(cavaleiros)
```

Inserir um novo elemento em uma lista

- A função **insert()** adiciona um novo elemento em uma posição qualquer de uma lista.

- Por exemplo, para inserir a string ‘morte’ na posição de índice 1 da lista a seguir, faça:

```
>>> cavaleiros = ['guerra', 'fome', 'peste']
>>> cavaleiros.insert(1,'morte')
>>> cavaleiros
['guerra', 'morte', 'fome', 'peste']
```

Remover um elemento de uma lista

- O operador **del** remove um elemento de uma lista.

- Exemplo:

```
>>> cavaleiros = ['guerra', 'fome', 'peste', 'morte']
>>> del cavaleiros[2]
>>> print(cavaleiros)
['guerra', 'fome', 'morte']
```

Concatenação de listas

- O operador + concatena (soma) duas listas.

- Exemplo:

```
>>> L1 = [1,2,3]
>>> L2 = [4,5,6]
>>> L1+L2
[1, 2, 3, 4, 5, 6]
```

Encontrando a posição de um elemento qualquer de uma lista

- A função index() é usada para encontrar a posição de um elemento qualquer em uma lista.

- Por exemplo, para encontrar a posição do elemento ‘peste’ na lista a seguir, faça:

```
>>> cavaleiros = ['guerra', 'fome', 'peste','morte']
>>> cavaleiros.index('peste')
2
```

Criando listas de inteiros

- Listas que contém números inteiros consecutivos são bastante comuns.
- O operador **range** fornece uma maneira simples de criar uma lista de inteiros.
- Exemplos:

```
>>> L= list(range(5,13))
>>> print(L)
[5, 6, 7, 8, 9, 10, 11, 12]
```

- O operador **range** pega dois argumentos (dois inteiros) e devolve uma lista que contém todos os inteiros do primeiro até o segundo, incluindo o primeiro mas não incluindo o segundo!

Criando listas de inteiros (continua)

- Com um argumento apenas, **range** cria uma lista que inicia em 0.
- Exemplo:

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

- Se houver um terceiro argumento, ele especifica o "espaço" entre os elementos sucessivos. Exemplo:

```
>>> L= list(range(2,23,3))
>>> print(L)
[2, 5, 8, 11, 14, 17, 20]
```

Exercícios

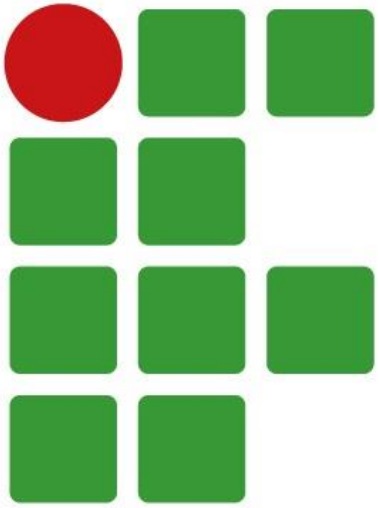
Exercícios

1. Dada a lista L = [8,0,97,105,21,303], faça um programa que imprima:
   - O maior elemento da lista
   - O menor elemento da lista
   - A soma de todos os elementos da lista
   - Os elementos ímpares
   - Os elementos maiores do que 18
2. Gere uma lista contendo os múltiplos de 3 entre 1 e 150.
3. Crie um programa que leia inicialmente uma sequência de N notas de alunos fornecidas pelo usuário e ao final mostre a sequência e sua média aritmética.
4. Crie um programa que leia inicialmente uma sequencia de N números inteiros e mostre ao final 2 listas: uma sem repetição e outra dos elementos repetidos.

Exercícios

5. Crie um programa que leia inicialmente uma sequência de N números inteiros e ao seu final mostre a sequência original, a soma de seus elementos que forem pares e a multiplicação dos elementos que forem impares.
6. Crie um programa que leia inicialmente uma sequência de N números inteiros fornecidos pelo usuário e mostre ao final da leitura a sequência original e a sequência invertida.
7. Crie um programa que leia inicialmente duas sequências de N elementos cada uma e ao final mostre as duas sequências originais e a sequência resultante da soma de seus elementos. Exemplo:

   a=[5, 9, 0] b=[12, 34, 101] soma=[17, 43, 101]



INSTITUTO FEDERAL

São Paulo

Câmpus São Carlos