Programação Orientada a Objetos

Aula 02 – Listas e Tuplas

Paulo Viniccius Vieira
paulo.vieira@faculdadeimpacta.com.br

Sequências

- Sequências são estruturas que armazenam dados
- Em Python há três estruturas básicas que permitem armazenar dados:
  - Lista
  - Tupla
  - Dicionário

Listas

- **Lista:**
  - é uma estrutura que armazena dados, organizados sequencialmente

- **Item** ou **Elemento**:
  - É o nome utilizado para identificar cada dado armazenado na lista

- **Formato**:
  - Itens agrupados entre colchetes

```
lista = [item1, item2, etc]
```

Listas

- As listas são estruturas **heterogêneas:**
  - Pode conter diferentes tipos de dados

```
lista1 = [2, 3, 4, 5]
lista2 = ['Molly', 'Steven', 'Alicia']
lista3 = ['Alicia', 27, 1550.87]
```

- As listas são estruturas **mutáveis:**
  - Seus itens podem ser alterados

```
lista = [2, 3, 4, 5]
lista[0] = 10
lista.append(8)
print(lista)              #[10, 3, 4, 5, 8]
```

Listas

- **Índice:**
  - É a posição de um item na lista
  - O índice do primeiro elemento sempre é zero
  - O índice do último elemento é n-1
  - Permite acessar os itens individualmente

```
lista = [2, 3, 4, 5]
lista[0] = 10    # altera item do índice 0
print(lista[2]) # imprime item no índice 2
```

Listas

- **Índice:**
  - Índices negativos podem ser utilizados para identificar posições relativas ao final da lista
    - Índice -1 identifica o último elemento,
    - Índice -2 identifica o penúltimo elemento,
    - Etc.

```
lista = [2, 3, 4, 5]
lista[-1] = 10    # altera último item
print(lista[-2]) # imprime penúltimo item
```

Listas

- Percorrer os elementos de uma lista
  - Pode ser utilizada a repetição **`for`**
  - A cada repetição é acessado um item da lista

```
lista = [2, 5, 8, 6]
for item in lista:
    print(item)        #imprime cada elemento
```

Listas

- Preencher lista com valores digitados
  - Exemplo: Preencher lista com 10 números digitados pelo usuário

```
lista = []
for i in range(10):
    n = int(input("Número: "))
    lista.append(n)
```

Listas - Principais Funções

`print(lista)`

- Imprime a lista

```
lista = [1, 10, 2, 6]
print(lista)              # [1, 10, 2, 6]
```

Métodos e Funções Úteis

`append(item)`

- Insere um item no final da lista

```
lista = [1, 2, 3]
lista.append(4)
lista.append(10)
print(lista)          # [1, 2, 3, 4, 10]
```

Listas - Principais Funções

`len(lista)`

- Retorna o tamanho de uma lista (quantidade de itens)

```
lista = [1, 10, 2, 10, 3, 10, 4]
tamanho = len(lista)
print(tamanho)                  # 7
```

Listas - Principais Funções

`count(item)`

- Contar quantas vezes um item aparece na lista

```
lista = [1, 10, 2, 10, 3, 10, 4, 5, 6]
quantidade = lista.count(10)
print(quantidade)                  # 3
```

Listas - Principais Funções

`index(item)`

- Retorna o índice da primeira ocorrência de um item
  - Se o elemento não for encontrado na lista, retorna um erro

```
lista = [1, 10, 2, 10, 3, 10, 4, 5, 6]
n = lista.index(10)
print(n)                    # 1
```

Listas - Principais Funções

`insert(indice, item)`

- insere item em um índice específico

```
lista = [4, 10, 5]
lista.insert(1, 'dois')
print(lista)      # [4, 'dois', 10, 5]
```

Listas - Principais Funções

`pop()`

- Remove o último item da lista

```
lista = [1, 2, 3, 4]
lista.pop()
print(lista)              # [1, 2, 3]
```

`pop(indice)`

- Remove o item de um índice específico

```
lista = [1, 2, 3, 4]
lista.pop(1)
print(lista)              # [1, 3, 4]
```

Listas - Principais Funções

`remove(item)`

- Remove a primeira ocorrência do item
  - Se item não existe, um erro é gerado

```
lista = ['oi', 'alo', 'ola', 'alo']
lista.remove('alo')
print(lista)      # ['oi', 'ola', 'alo']
```

Listas - Principais Funções

`sort()`

- Ordena os itens da lista.

```
lista = [9, 8, 7, 1, 4, 2]
lista.sort()
print(lista)      # [1, 2, 4, 7, 8, 9]
```

- Para ordenar de forma decrescente:

```
lista = [9, 8, 7, 1, 4, 2]
lista.sort(reverse=True)
print(lista)      # [9, 8, 7, 4, 2, 1]
```

Listas - Principais Funções

`min()` e **`max()`**

- Retorna o menor e o maior item da lista

`sum()`

- Retorna o somatório da lista

```
lista = [1, 2, 9, 3, 4]
menor = min(lista)
print(menor)                    # 1
maior = max(lista)
print(maior)                    # 9
soma = sum(lista)
print(soma)                     # 19
```

Listas - Principais Funções

Operador **`in`**

- Permite verificar se determinado item pertence a uma lista

```
lista = [1, 'a', 'bc’]
if 1 in lista:
    print('Numero 1 está na lista')
else:
    print('Numero 1 não está na lista')
```

Listas

Concatenação

- O operador + pode ser usado para concatenação

```
lista1 = [10, 2, 100]
lista2 = [2, 5, 6]
lista3 = lista1 + lista2
print(lista3)      # [10, 2, 100, 2, 5, 6]
```

Representação de Listas em Memória

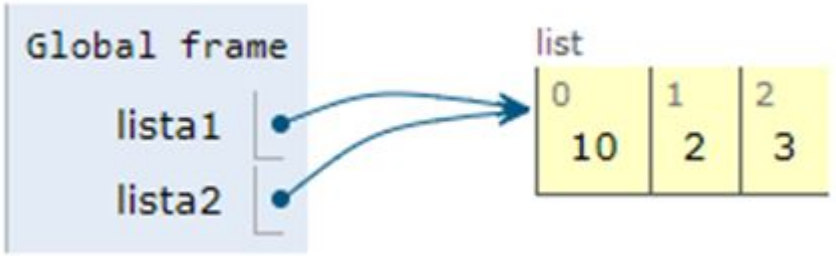
- O valor de uma variável de lista é um endereço de memória
- Ao copiar uma lista para outra, o que é feito é copiar o valor do endereço de memória

```
lista1 = [1, 2, 3]
lista2 = lista1
print(lista2)   #[1,2,3]
```

Representação de Listas em Memória

- Ambas passam a apontar para o mesmo endereço de memória, portanto o que for modificado em uma lista também será modificado na outra

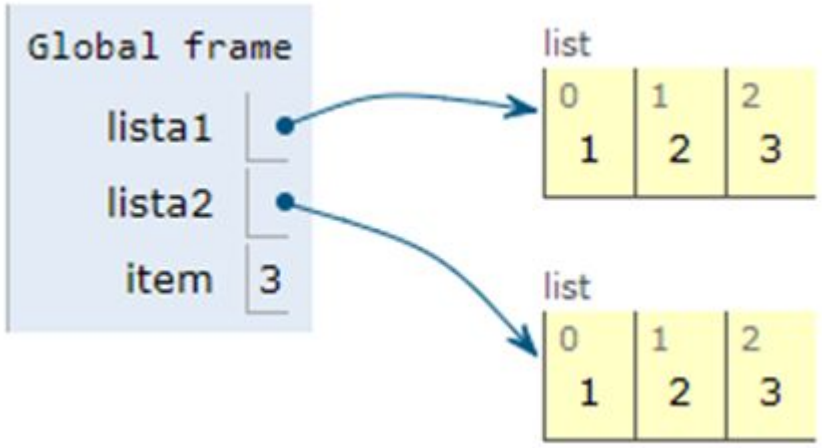
```
lista1 = [1, 2, 3]
lista2 = lista1
lista1[0] = 10
print(lista1)  #[10,2,3]
print(lista2)  #[10,2,3]
```



Representação de Listas em Memória

- Como evitar isso?
  - Copiar cada uns dos valores:

```
lista1 = [1, 2, 3]
lista2 = []
for item in lista1:
    lista2.append(item)
print(lista1)   #[1,2,3]
print(lista2)   #[1,2,3]
```



Tuplas

- Tupla é uma sequências de itens, semelhante à uma lista. Mas, existem diferenças...
  - Tuplas são **imutáveis**: os itens da tupla não podem ser alterados
  - Os itens da tupla são agrupados com parênteses

```
lista = [1, 2, 3, 4]
tupla = (1, 2, 3, 4)
```

Tuplas

- Tupla vazia

```
tupla = ()
```

- Tupla com um único elemento
  - note a necessidade da vírgula ao final, em tuplas de um único elemento

```
tupla = (1,)      # isso é uma tupla
tupla = (1)       # isso não é uma tupla
```

Acesso aos itens de uma Tupla

- Acesso é feito pelo índice, da mesma forma que nas listas

```
tupla = ("Maria", "Joao", "Carlos")
print(tupla[0])       # Maria
```

Atualização de Tuplas

- Tuplas são **imutáveis**, portanto não é permitido atualizar os valores de uma tupla

```
tupla = ("Maria", "Joao", "Carlos")
tupla[0] = "Ana"
TypeError: 'tuple' object does not support
item assignment
```

```
tupla = ("Maria", "Joao", "Carlos")
tupla.Append("Ana")
TypeError: 'tuple' object has no attribute
'Append'
```

Concatenação de Tuplas

- Tuplas podem ser concatenadas
  - Só é possível concatenar tuplas com outras tuplas

```
tupla1 = ('Maria', 'Joao')
tupla2 = ('Pedro', 'Ana')

tupla3 = tupla1 + tupla2          # OK!
print(tupla3)
#('Maria','Joao','Pedro','Ana')

tupla3 = tupla3 + 'Antonio'       # Erro!
```

Exemplos