Introdução ao Python

Estrutura de dados - listas

Henrique Y. Shishido

Departamento de Computação
Universidade Tecnológica Federal do Paraná

Introdução

- A manipulação de conjunto de dados é uma necessidade comum em aplicações de inteligência artificial.
- Estrutura de dados podem auxiliar na organização, acesso e processamento de dados.
- O Python oferece quatro estruturas de dados para armazenar coleções de dados. São elas: **listas**, **conjuntos**, **tuplas** e **dicionários**.

Listas

- **Listas** são usadas para armazenar múltiplos elementos em uma única variável.
- A inicialização de uma lista pode ser realizada por meio da abertura e fechamento de colchetes `[` e `]`.
- As principais características das listas são a inserção **ordenada** de elementos, **mutabilidade** e possibilidade de **valores duplicados**

Inicialização de Lista

- Uma lista pode conter elementos de diversos tipos de dado como números, strings, objetos, boolean, etc.

Inicialização de uma lista

```
#Inicialização de uma lista de numérica
numberList = [5, 2, 10, 392, 0, 3, 5]

#Inicialização de uma lista de strings
stringList = ["Minas Gerais", "João Pessoa", "Manaus", "Campo Grande"]

#Inicialização de uma lista de objetos
p1 = Produto("Copo", 5.20)
p2 = Produto("Prato", 10.50)
objectList = [p1, p2]

#Lista com diferentes tipos de dados
p1 = Produto("Copo", 5.20)
myList = ["Jeferson", p1, True, 50.2, 1]
```

Propriedades de listas

Construtor `list()`

```
>>> minhaLista = list(("amarelo", "azul", "preto", "vermelho"))
>>> print(minhaLista)
['amarelo', 'azul', 'preto', 'vermelho']
```

Tipo de dado `list`

```
>>> numberList = [5, 2, 10, 392, 0, 3, 5]
>>> type(numberList)
<class 'list'>
```

Tamanho da lista

```
>>> minhaLista = ["abacate", "melão", "banana", "morango"]
>>> len(minhaLista)
4
```

Acesso a item de lista

- O acesso a um item de uma lista pode ser realizado por meio do índice numérico. O primeiro índice da lista é o 0 (zero).

```
minhaLista = [5, 10, 15, 20, 30]

print(minhaLista[0])
>>> 5

print(minhaLista[4])
>>> 30
```

- É possível utilizar índice negativo para referenciar elementos em ordem inversa:

```
minhaLista = [5, 10, 15, 20, 30]

print(minhaLista[-1])
>>> 30

print(minhaLista[-2])
>>> 20
```

Intervalo de lista

- É possível especificar um intervalo de uma lista passando os índices de início de fim do invervalo:

```
minhaLista = [5, 10, 15, 20, 30]

print(minhaLista[1:3])
>>>[10, 15]

print(minhaLista[:4])
>>>[5, 10, 15, 20]

print(minhaLista[2:])
>>>[15, 20, 30]
```

Checando a existência de um item

- Para verificar se um item específico existe na lista é possível usar a palavra-chave `in`:

```
minhaLista = [5, 10, 15, 20, 30]

if 5 in minhaLista:
    print("Existe o número 5 na lista")

>>>Existe o número 5 na lista
```

Mudança de item(ns) da lista

- Para modificar um item na lista, pode-se referenciá-lo por meio de seu índice:

```
minhaLista = [5, 10, 15, 20, 30]

minhaLista[0] = 1000

print(minhaLista)
>>>[1000, 10, 15, 20, 30]
```

- Para modificar um intervalo de valores, é preciso passar os índices do intervalo:

```
minhaLista = [5, 10, 15, 20, 30]

minhaLista[2:4] = [3000, 6000]

print(minhaLista)
>>>[5, 10, 3000, 6000, 30]
```

Mudança de item(ns) da lista - Parte 2

- Para modificar um intervalo de uma lista com um mesmo valor, basta definir os índices do intervalo e passar um único valor na atribuição:

```
minhaLista = [5, 10, 15, 20, 30]

minhaLista[1:4] = 1000

print(minhaLista)
>>>[5, 1000, 1000, 1000, 30]
```

Adicionando novo elemento

- Para acrescentar um novo elemento na lista:

```
minhaLista = [5, 10, 15, 20, 30]

#Acrescentando o valor 9000 na lista
minhaLista.append(9000)
print(minhaLista)
>>>[5, 10, 15, 20, 30, 9000]
```

Inserção de um novo elemento na lista

- Para inserir um novo elemento na lista sem substituir valores existentes, pode-se utilizar o método `insert()`:

```
minhaLista = [5, 10, 15, 20, 30]

#Inserindo o valor 7000 na posição 1:
minhaLista.insert(1, 7000)
print(minhaLista)
>>>[5, 7000, 10, 15, 20, 30]
```

Extendendo a lista

- Para acrescentar elementos de outra lista na atual, pode-se utilizar o método `extend()`:

```
minhaLista = [10, 20, 30]
outraLista = [40, 50, 60]

minhaLista.extend(outraLista)
print(minhaLista)
>>>[10, 20, 30, 40, 50, 60]
```

Removendo elemento da lista

- O método `remove()` retira um elemento especificado como parâmetro (se existir na lista):

```
meusObjetos = ["alicate", "chave", "martelo"]

meusObjetos.remove("chave")
print(meusObjetos)
>>>["alicate", "martelo"]
```

- O método `pop()` remove um elemento de um determinado índice:

```
meusObjetos = ["alicate", "chave", "martelo"]

meusObjetos.pop(0)
print(meusObjetos)
>>>["chave", "martelo"]
```

Limpando a lista

- O método `clear()` esvazia toda a lista:

```
meusObjetos = ["alicate", "chave", "martelo"]

meusObjetos.clear()
print(meusObjetos)
>>> []
```

Loop em listas

- Para iterar sobre uma lista de elementos, pode-se usar o comando `for`:

```
meusObjetos = ["alicate", "chave", "martelo"]

for obj in meusObjetos:
    print(obj)
>>>"alicate"
>>>"chave"
>>>"martelo"
```

- Pode-se iterar em uma lista por meio dos índices numéricos:

```
meusObjetos = ["alicate", "chave", "martelo"]

for i in range(len(meusObjetos)):
    print(meusObjetos[i])
>>>"alicate"
>>>"chave"
>>>"martelo"
```

Ordenação de listas

- Objetos do tipo `list` possuem o método `sort()` que ordena a lista alfabeticamente, de maneira ascendente (por padrão):

```
#Ordenação alfabética
meusObjetos = ["onça", "chave", "martelo", "mamute"]
meusObjetos.sort()
print(meusObjetos)
>>>["chave", "mamute", "martelo", "onça"]
```

- Para ordenar de maneira descendente, usa-se o argumento `reverse = True`:

```
#Ordenação inversa
meusObjetos = ["onça", "chave", "martelo", "mamute"]
meusObjetos.sort(reverse = True)
print(meusObjetos)
>>>["onça", "martelo", "mamute", "chave"]
```

Cópia de lista

- Não é possível copiar uma lista usando `lista2 = lista1`, pois qualquer alteração em `lista1` irá refletir nos valores de `lista2`.
- Logo, para copiar valores de uma lista para outra, pode-se usar o método `copy()`:

```
lista1 = [10, 20, 30]
lista2 = lista1.copy()

print(lista2)
>>>[10, 20, 30]
```

- Uma outra forma de copiar os valores de uma lista a outra, é por meio do método `list()`:

```
meusNumeros = [10, 20, 30]
novaLista = list(meusNumeros)

print(novaLista)
>>>[10, 20, 30]
```

Contagem da incidência um elemento na lista

- Para retornar o número de vezes que um elemento aparece em uma lista, pode-se usar o método `count()`:

```
meusNumeros = [10, 20, 30, 20, 20, 50, 60, 70]

print(meusNumeros.count(20))
>>>3
```

Contagem da incidência um elemento na lista

- Para retornar o número de vezes que um elemento aparece em uma lista, pode-se usar o método `count()`:

```
meusNumeros = [10, 20, 30, 20, 20, 50, 60, 70]

print(meusNumeros.count(20))
>>>3
```