Estrutura de dados

Prof. Dr. Alex Sandro Roschildt Pinto

Mateus Manoel Pereira

Estrutura de dados

- Python implementa por padrão seis estruturas de dados.
  - Lista
  - Tupla
  - Dicionário
  - String
  - Conjunto
  - Conjunto Imutável

Lista (List)

- Lista é a estrutura mais genérica do python, são mutáveis e os elementos podem ser inseridos em qualquer posição.

```
>>> listaFilmes = ["Vingadores", "Toy Story", "O Plano Imperfeito"]
>>> listaFilmes[2]
>>> O Plano Imperfeito
```

Lista (List)

- Uma lista tem vários métodos que permite inserir, remover e manusear seus elementos. Os principais são:

  - append( )
  - extend( )
  - pop( )
  - insert( )
  - remove( )
  - sort( )

Método: append(obj)

- Anexa um **objeto** ao final da lista.

```
>>> sequencia = [1, 2, 3, 4]
>>> sequencia.append(5)
>>> sequencia
>>> [1, 2, 3, 4, 5]
```

- Atenção: O método append( ) anexa um objeto na lista e não cada elemento um a um, ou seja, se utilizar append( ) para anexar uma outra lista, por exemplo, a lista será completamente anexada.

Método: append(obj)

- Atenção:

```
>>> sequencia = [1, 2, 3, 4]
>>> sequencia.append([5, 6, 7])
>>> sequencia
>>> [1, 2, 3, 4, [5 , 6, 7]]
```

- Na maioria das vezes esse não é o resultado desejado. Para adicionar elementos de uma lista em outra lista é melhor utilizar o método extend( ).

Método: extend(obj iterável)

- Anexa **cada elemento** de um objeto iterável na lista.

```
>>> sequencia = [1, 2, 3, 4]
>>> sequencia.extend([5, 6, 7])

>>> sequencia
>>> [1, 2, 3, 4, 5 , 6, 7]
```

Método: pop( )

- **Remove** e **retorna** o último elemento da lista.

```
>>> sequencia = [1, 2, 3, 4]
>>> sequencia.pop( )
>>> 4
>>> sequencia
>>> [1, 2, 3]
```

Método: insert(posição, elemento)

- **Insere** um elemento na posição desejada da lista.

```
>>> sequencia = [1, 2, 3, 4]
>>> sequencia.insert(3, 5)
>>> sequencia
>>> [1, 2, 3, 5, 4]
```

Método: remove(elemento)

- **Remove** a primeira ocorrência de um elemento da lista.

```
>>> sequencia = [1, 2, 3, 4]
>>> sequencia.remove(3)
>>> sequencia
>>> [1, 2, 4]
```

Método: sort( )

- Ordena os elementos de uma lista em ordem crescente.

```
>>> sequencia = [4, 3, 2, 1]
>>> sequencia.sort( )
>>> sequencia
>>> [1, 2, 3, 4]
```

- Pode ordernar em ordem reversa se passado como parâmetro.

```
>>> sequencia.sort(reverse=True)
```

Tupla

- Estrutura similar a lista, com a principal diferença de que as tuplas são imutáveis, tornando sua manipulação muito mais rápida que uma lista.

- Tuplas tem apenas dois métodos disponíveis:
  - count( )
  - index( )

Método: count(elemento)

- Retorna o número de ocorrências de um elemento.

```
>>> sequencia = (1, 1, 3, 4)
>>>
sequencia.count(1)
>>> 2
```

Método: index(elemento)

- Retorna a posição da primeira ocorrência de um elemento da tupla.

```
>>> sequencia = (1, 1, 3, 4)
>>> sequencia.count(1)
>>> 2
```

- Se a tupla não conter o elemento, uma exceção é gerada.

Dicionário

- Conjunto de dados **não ordenado**, no qual cada dado é um par composto por uma chave e um valor. As chaves e valores podem ser acessados separadamente.

```
>>> dicionario = {1: 'Victor', 2: 'Bob', 3:
    'Enzo'}
>>> dicionario.keys())
>>> [1, 2, 3]
>>> dicionario.values()
>>> ['Victor', 'Bob', 'Enzo']
```

Dicionário

- **Adicionando** um novo elemento em um dicionário.

```
>>> dicionario = {1: 'Lux'}
>>> dicionario[2] = 'Ezreal'
>>> dicionario
>>> {1: 'Lux', 2: 'Ezreal'}
```

Método: has_key(chave)

- Verifica a existência de uma determinada chave no dicionário.

```
>>> dicionario.has_key(2)
>>> True
```

Método: get(chave)

- Retorna o valor de uma determinada chave.

```
>>> dicionario.get(2)
>>> 'Ezreal'
```

Método: pop(chave)

- **Remove** e retorna o elemento relacionado a chave passada como parâmetro.

```
>>> dicionario.pop(1)
>>> 'Lux'
    >>> dicionario.items()
    >>> [(2: 'Ezreal')]
```

Percorrendo um dicionário

- Podemos percorrer um dicionário pela chaves e valores ao mesmo tempo.

```
>>> for k, v in dicionario.items():
...     print('chave: {} elemento: {}'.format(k, v))
```

- Atenção: Dicionários **não** tem conceito de ordem, isso significa que nenhuma ordem é garatida ao imprimir os elementos dessa forma.

String

- Strings em python são sequências de caracteres **imutáveis.**

```
>>> string1 = 'String'
>>> string2 = "String com 'aspas' simples dentro"
>>> string3 = """String 'com' "aspas" dentro"""
```

String

- Diferente de outras linguagens, em python, os operadores >, >=, ==, <=, < e != sobrecarregam as operações de comparação entre strings.

```
>>> 'batata' == 'lux'
>>> False
    >>> 'acd' > 'abc'
    >>> True
```

String

- Como as strings são imutáveis nenhum método a modifica, mas alguns podem retornar uma nova string com a modificação exigida.
- Strings tem muitos métodos então vamos apenas citar os principais.

  - count()
  - title()
  - lower()
  - strip()
  - split()

Método: count(caractere)

- Retorna a **quantidade** de ocorrências do caractere passado como parâmetro.

```
>>> string = 'Remember, this is a bandit country.'
>>> string.count('b')
>>> 2
```

Método: title()

- Retorna uma versão em título da string, ou seja, com as letras iniciais de cada palavra **maiúsculas**.

```
>>> string = 'shoot everything that moves'
>>> string.title()
>>> 'Shoot Everything That Moves'
```

Método: lower()

- Retorna uma versão em **caixa baixa** da string.

```
>>> string = 'My gIRL waNTs to Party all THe tiMe'
>>> string.lower()
>>> 'my girl wants to party all the time'
```

Método: strip()

- Retorna uma versão sem **espaços desnecessários** na string.

```
>>> string = '    Today   '
>>> string.strip()
>>> 'Today'
```

Método: split(caractere)

- Separa uma string de acordo com um caractere, se nenhum parâmetro for passado a string é separada pelo caractere espaço.

```
>>> string = 'Surprise of my life'
>>> string.split('of')
>>> ['Surprise ', ' my life']
```

Conjunto

- Conjunto é uma estrutura de dados que funciona igualmente como a definição de conjunto na matemática, ou seja, é possível fazer a união, intersecção e diferença simétrica de conjuntos além de poder verificar se um conjunto é subconjunto de outro ou se são diferentes.

```
>>> a = set([1, 2, 3, 4])
>>> b = set([3, 4, 5, 6])
>>> a & b  #{3, 4}
>>> a < b #False
>>> a – b #{1, 2}
>>> a ^ b #{1, 2, 5, 6}
```

Conjunto Imutável

- Conjuntos imutáveis são como conjuntos, entretanto não podem ser modificados, assim é possível que haja Conjuntos de Conjuntos, algo que não é possível em Conjutos normais.

```
>>> a = frozenset([1, 2, 3, 4])
>>> b = frozenset([3, 4, 5, 6])
>>> a & b #{3, 4}
>>> a < b #False
>>> a – b #{1, 2}
>>> a ^ b #{1, 2, 5, 6}
```

Documentação do Python

- Nem todos os métodos de cada estrutura foram citados, para uma lista completa de métodos acesse a documentação de estrutra de dados em Python.

https://docs.python.org/3/tutorial/datastructures.html