**SUMÁRIO**

[1. INTRODUÇÃO 2](#_Toc52044306)

[2. LISTA 2](#_Toc52044307)

[2.1. Métodos para lista 2](#_Toc52044308)

[3. TUPLA 2](#_Toc52044309)

[3.1. Métodos para tupla 2](#_Toc52044310)

[4. DICIONÁRIO 2](#_Toc52044311)

[4.1. Métodos para dicionário 2](#_Toc52044312)

[5. CONJUNTO 2](#_Toc52044313)

[5.1. Métodos para conjunto 2](#_Toc52044314)

[6. REFERÊNCIAS BIBLIOGRÁFICAS 2](#_Toc52044315)

1. INTRODUÇÃO

Estruturas de dados são uma parte fundamental da Computação e de qualquer projeto de software. São responsáveis pelo armazenamento e organização de dados, com esses dados podendo ser manipulados através de métodos diversos para a inserção, exclusão localização e classificação, entre outras possibilidades.

As estruturas de dados se dividem entre estruturas lineares, nas quais os dados são organizados pela estrutura de forma sequencial, onde cada dados possui um índice de posição especifico, o qual é utilizado para na implementação de métodos para obtenção de um certo resultado, e as estruturas não lineares, que não possui nenhum tipo de organização sequencial dos itens inseridos na estrutura.

Em Python temos as estruturas lineares chamadas de tuplas e listas e as estruturas não lineares chamadas de dicionários e coleções. Podemos também implementar outros tipos de estruturas não lineares, mais complexas, como arvores e dataframe, entre outros através das estruturas padrão do Python.

1. LISTA

A lista é estrutura de dados que possui um armazenamento sequencial de dados, onde para cada dado é atribuída uma posição de índice que começa no numero zero, e é incrementada a cada novo dado adicionado. Os dados armazenados dentro de uma lista podem ser alterados a qualquer momento.

* 1. Métodos para lista

A lista em Python oferece os métodos abaixo para manipulação dos dados armazenados:

list.append(x) - Adiciona o item “x” ao fim da lista, ou seja, na posição índice seguinte ao último item armazenado na lista.

list.extend(iterable) - Adiciona ao fim da lista todos os elementos do argumento “iterable” passado como parâmetro.

Exemplo:

list.insert(i, x) - Insere o item “i” na posição índice informada como parâmetro “x”.

list.remove(x) - Remove o primeiro item encontrado na lista cujo valor é igual a “x”. Se não existir valor igual, uma exceção “ValueError” é informada.

list.pop([i]) - Remove um item em uma dada posição “i” na lista e o retorna. Se nenhum índice é especificado, o método remove e devolve o último item da lista.

Os colchetes ao redor do “i” na demonstração do método indica que o parâmetro é opcional, e não que é necessário escrever estes colchetes ao chamar o método. Este tipo de notação é frequentemente utilizado na Biblioteca de Referência Python.

list.clear() - Remove todos os itens de uma lista.

list.index(x[, start[, end]]) - Devolve o índice base-zero do primeiro item cujo valor é igual a “x”, levantando ValueError se este valor não existe.

Os argumentos start e end são opcionais, e são usados para limitar a busca para uma fatia específica da lista.

list.count(x) - Devolve o número de vezes em que “x” aparece na lista.

list.sort(key=None, reverse=False) - Ordena os itens na lista. Os argumentos podem ser usados para personalizar a ordenação, veja a função sorted() para maiores explicações).

list.reverse() - Inverte a ordem dos elementos na lista.

list.copy() - Devolve uma cópia da lista.

1. TUPLA

A tupla é uma estrutura muito parecida com uma lista, também possuindo índices de posição sequencial, porém os dados armazenados em uma tupla são imutáveis, ou seja, não podem ser alterados. As tuplas possuem uma certa vantagem em relação as listas em certas aplicações, sendo mais eficientes em iterações, devido ao fato dos seus dados serem imutáveis, ser mais segura para armazenar dados que não devem ser modificados, pode ser utilizada para armazenar dados de diferentes tipos e os seus dados podem ser utilizados como chaves de dicionários, coisas que as listas não podem.

* 1. Métodos para tupla

A tupla em Python oferece apenas os 2 métodos abaixo para manipulação dos dados armazenados:

t.count(x): retorna o número de vezes que o objeto “x” ocorre na tupla.

t.index(x): retorna a posição de índice da primeira ocorrência igual a “x” a partir da esquerda para a direita. Se não existir valor igual, uma exceção “ValueError” é informada.

1. DICIONÁRIO

O Dicionário é uma estrutura de dados não linear associativa, ou seja, não possui nenhuma organização sequencial de armazenamento, não possui os índices de posição como nas listas e tuplas, ao invés disso, o dicionário utiliza chaves de identificação atreladas ao item armazenado, associando assim cada dado à uma chave correspondente única.

* 1. Métodos para dicionário

O dicionário em Python oferece os métodos abaixo para manipulação dos dados armazenados:

fromkeys(i, x) - Retorna um novo dicionário cujas chaves são os elementos de “i” e cujos valores são todos iguais a “x”. Se “x” não for especificado, o default é “none”.

get(i, [x]) - Obtém o valor associado à chave “i”. O argumento “x” é opcional, e é usado para retornar uma mensagem ou valor, caso nenhum valor esteja associado à chave “i”. Se o argumento “x” não for utilizado, será retornado o default “none”.

d.items() - Retorna uma lista com todos as associações chave-valor do dicionário no formato de uma tupla.

d.keys() - Retorna uma lista com todas as chaves do dicionário.

d.values() - Retornar uma lista com todos os valores do dicionário.

d.pop(i) - Remove a associação chave-valor do elemento correspondente à chave “i”. Caso a chave “i” não exista, retornará o erro “KeyError”.

d. popitem() - Remove a associação chave-valor do último elemento aleatório do dicionário.

d['i'] = x - Adiciona um elemento ao dicionário utilizando a sua chave “i” e atribuindo o conteúdo “x” a ela.

d.update(x) – Adiciona a um dicionário, elementos de outro dicionário “x”.

1. CONJUNTO

A estrutura de dados Conjunto é uma estrutura não linear assim como o dicionário, não possuindo assim uma organização sequencial dos elementos armazenados, porém não é associativa, pois não possui chaves de associação. Além disso o Conjunto não aceita o armazenamento de itens duplicados, sendo muito utilizada para eliminar elementos duplicados de outras estruturas.

* 1. Métodos para conjunto

O Conjunto em Python oferece apenas os métodos abaixo para manipulação dos dados armazenados:

d.add(x) – Adiciona o elemento “x” ao conjunto.

d.remove(x) – Remove o elemento “x” do conjunto.

d.discard(x) – Remove o elemento “x” do conjunto. Caso o elemento “x” não esteja presente no conjunto, retornará o erro “KeyError”.

d.clear – Remove todos os elementos do conjunto.

d.copy – Retorna uma cópia do conjunto.

d.update(x) – Adiciona uma outra estrutura “x” ao conjunto.

d.union(x) – Retorna um conjunto a partir da união do conjunto “d” e do conjunto “x”.

d.difference(x) – Retorna um conjunto com os elementos que existem no conjunto “d” e não existem no conjunto “x”.

d.difference\_update(x) – Remove os elementos do conjunto que também estão presentes no conjunto “x”.

d.intersection(x) – Retorna um conjunto com os elementos que estão presentes em ambos os conjuntos “d” e “x”.

d.intersection\_update(x) – Remove os elementos do conjunto que estão presentes em ambos os conjuntos “d” e “x”.

d.isdisjoint(x) – Retorna “True” caso não existam elementos que estejam presentes em ambos os conjuntos “d” e “x”, e False caso existam.

d.issubset(x) – Retorna “True” caso todos os elementos do conjunto “x” estejam presentes no conjunto “d”, e False caso não estejam.

d.issuperset(x) – Retorn “True” caso todos os elementos do conjunto “d” estejam presentes no conjunto “x”, e False caso não estejam.

d.pop() – Remove um elemento aleatório do conjunto.

d.symmetric\_difference(x) – Retorna um conjunto que contém todos os elementos de ambos os conjuntos “d” e “x”, removendo os itens que existam em ambos os conjuntos.

d.symmetric\_difference\_update(x) – Adiciona ao conjunto todos os elementos do conjunto “x”, removendo os itens que existam em ambos os conjuntos.

1. REFERÊNCIAS BIBLIOGRÁFICAS

<http://calhau.dca.fee.unicamp.br/wiki/images/0/01/EstruturasDados.pdf>

<https://docs.python.org/pt-br/3/tutorial/datastructures.html>

<https://www.treinaweb.com.br/blog/principais-estruturas-de-dados-no-python/>

<https://www.w3schools.com/python/python_sets.asp>

<https://www.w3schools.com/python/python_tuples.asp>

<https://www.w3schools.com/python/python_lists.asp>

<https://www.w3schools.com/python/python_dictionaries.asp>