Aula 15

Listas – parte Deux

Vamos nos aprofundar mais nas listas e no que elas podem fazer.
As listas possuem diversos métodos.

list.extend(<iterable>)

Estende uma lista adicionando todos os itens do iterável.
Equivalente a fazer "lista[len(lista):] = iterable".

```
>>> numeros = [1,2,3,4]
>>> outros_numeros = [9,8,7,6]
>>> numeros
[1, 2, 3, 4]
>>>
>>> numeros.extend(outros_numeros)
>>> numeros
[1, 2, 3, 4, 9, 8, 7, 6]
>>>
```

```
>>> numeros
[1, 2, 3, 4, 9, 8, 7, 6]
>>> mais_numeros = [10,11,12]
>>> numeros[len(numeros):] = mais_numeros
>>> numeros
[1, 2, 3, 4, 9, 8, 7, 6, 10, 11, 12]
>>>
```

list.insert(i, x)

Insere um item em uma posição específica. O primeiro argumento ("i") é o índice onde o elemento será inserido. Logo, "list.insert(0, x)" vai inserir na primeira posição da lista, e "list.insert(len(lista), x)" é equivalente ao "list.append(x)".

```
>>> numeros = [1,2,3,4]
>>> numeros
[1, 2, 3, 4]
>>>
>>> numeros.insert(0,'f')
>>> numeros
['f', 1, 2, 3, 4]
>>>
>>> numeros.insert(len(numeros),'B')
>>> numeros
['f', 1, 2, 3, 4, 'B']
>>>
```

list.clear()

Remove todos os itens da lista. Equivalente ao "del lista[:]".

```
>>> numeros = [1,2,3,4]
>>> numeros
[1, 2, 3, 4]
>>>
>>> numeros.clear()
>>> numeros
[]
>>>
```

```
>>> numeros = [1,2,3,4]
>>> numeros
[1, 2, 3, 4]
>>>
>>> del numeros[:]
>>> numeros
[]
>>>
```

list.index(x[, start[, end]])

Retorna o índice da lista do primeiro item que o valor é igual ao "x". Vai gerar um "ValueError" se o item não estiver na lista.

O argumento opcional "start" e "end" são interpretados como uma notação de fatiamento e são usados para limitar a busca a uma subsequência da lista.

```
>>> planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte', 'Júpiter', 'Saturno']
>>> planetas
['Mercúrio', 'Vênus', 'Terra', 'Marte', 'Júpiter', 'Saturno']
>>>
>>> indice = planetas.index("Marte")
>>>
>>> indice
3
>>> planetas[indice]
'Marte'
>>>
```

list.count(x)

Retorna o número de vezes que o "x" aparece na lista.

```
>>> numeros = [1,2,3,4,5,6,1,5,6,9,5,6,9,7,5]
>>> numeros
[1, 2, 3, 4, 5, 6, 1, 5, 6, 9, 5, 6, 9, 7, 5]
>>> qtd = numeros.count(5)
>>>
>>> qtd
4
>>>
```

Tipo set

Em Python, um conjunto (set) é uma estrutura de dados que armazena elementos únicos e não ordenados. Isso significa que cada elemento aparece apenas uma vez em um conjunto e não há ordem definida para os elementos.

Os conjuntos em Python são implementados usando tabelas hash, o que significa que os elementos são armazenados em uma tabela onde a chave é derivada do próprio elemento. Isso permite que as operações de conjunto sejam realizadas com eficiência, mesmo para conjuntos grandes.

Para criar um conjunto em Python, você pode usar chaves {} ou a função set().

```
>>> # Criando um conjunto
>>> meu_set = {1, 2, 3, 4, 5}
>>>
>>> # Criando um conjunto vazio
>>> meu_set_vazio = set()
>>>
```

Também podemos criar um conjunto a partir de uma lista ou qualquer outro iterável usando a função set():

```
>>> # Criando um conjunto a partir de uma lista
>>> minha_lista = [1, 2, 3, 4, 5]
>>> minha_lista
[1, 2, 3, 4, 5]
>>>
>>> meu_set = set(minha_lista)
>>> meu_set
{1, 2, 3, 4, 5}
>>>
```

Os conjuntos em Python suportam várias operações úteis, como união, interseção e diferença de conjuntos. Além disso, é possível adicionar ou remover elementos de um conjunto usando os métodos add() e remove(), respectivamente.

set.union()

Podemos usar o método union() para combinar dois conjuntos em um novo conjunto:

```
>>> set_1 = {1, 2, 3}
>>> set_2 = {3, 4, 5}
>>>
>>> set_3 = set_1.union(set_2)
>>> set_3
{1, 2, 3, 4, 5}
>>>
```

set.intersection()

Da mesma forma, podemos usar o método intersection() para obter os elementos comuns a dois conjuntos:

```
>>> set_1 = {1, 2, 3}
>>> set_2 = {3, 4, 5}
>>>
>>> comum = set_1.intersection(set_2)
>>> comum
{3}
>>>
```

set.difference()

A diferença entre dois conjuntos é um novo conjunto que contém todos os elementos que estão presentes no primeiro conjunto, mas não no segundo conjunto.

```
>>> set_1 = {1, 2, 3}
>>> set_2 = {3, 4, 5}
>>>
>>> diferenca = set_1.difference(set_2)
>>> diferenca
{1, 2}
>>>
```

set.add()

O método add() é usado para adicionar um único elemento a um conjunto. Se o elemento já estiver presente no conjunto, a operação não terá nenhum efeito.

```
>>> meu_set = {1, 2, 3}
>>> meu_set
{1, 2, 3}
>>>
>>> meu_set.add(4)
>>> meu_set
{1, 2, 3, 4}
>>>
>>> meu_set.add(1)
>>> meu_set
{1, 2, 3, 4}
>>>
```

set.remove()

O método remove() é usado para remover um elemento específico de um conjunto. Se o elemento não estiver presente no conjunto, uma exceção do tipo "KeyError" será gerada.

```
>>> meu_set = {1, 2, 3}
>>> meu_set
{1, 2, 3}
>>>
>>> meu_set.remove(2)
>>> meu_set
{1, 3}
>>>
>>> meu_set.remove(4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 4
>>>
```

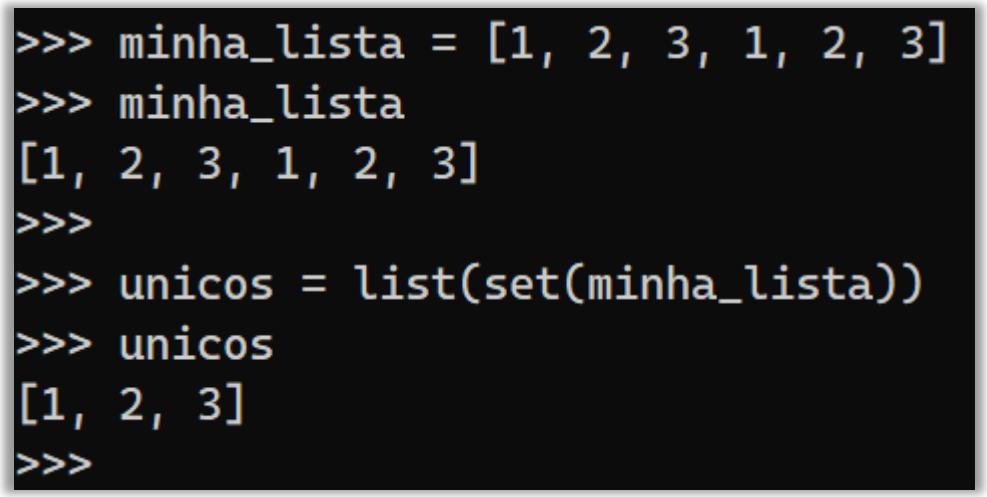
set.discard()

Além do método remove(), os conjuntos em Python também possuem o método discard(), que funciona de maneira semelhante, mas não gera uma exceção se o elemento não estiver presente no conjunto. Em vez disso, o método discard() simplesmente não faz nada.

```
>>> meu_set = {1, 2, 3}
>>> meu_set
{1, 2, 3}
>>>
>>> meu_set.discard(2)
>>> meu_set
{1, 3}
>>>
>>> meu_set.discard(4)
>>> meu_set
{1, 3}
>>>
```

Removendo Duplicados

Outro recurso interessante dos conjuntos em Python é a capacidade de remover elementos duplicados de uma lista. Você pode usar a função set() para converter uma lista em um conjunto, o que removerá automaticamente os elementos duplicados. Em seguida, você pode converter o conjunto novamente em uma lista usando a função list().



Exercícios para Praticar

Exercícios de listas

1. Crie uma lista vazia A e adicione os elementos "a" e "b" a ela utilizando o método extend().
2. Crie uma lista A com os elementos "a", "b" e "c", e adicione os elementos "d" e "e" a ela utilizando o método extend().
3. Crie uma lista A com os elementos "a", "b" e "c", e insira o elemento "d" na posição 1 utilizando o método insert().
4. Crie uma lista A com os elementos "a", "b" e "c", e insira o elemento "d" na posição 3 utilizando o método insert().
5. Crie uma lista A com os elementos "a", "b" e "c", e insira o elemento "d" na posição -1 utilizando o método insert().
6. Crie uma lista A com os elementos "a", "b" e "c", e insira o elemento "d" na posição -3 utilizando o método insert().
7. Crie uma lista A com os elementos "a", "b" e "c", e insira o elemento "d" na posição 4 utilizando o método insert().
8. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e conte o número de ocorrências do elemento "a" utilizando o método count().
9. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e conte o número de ocorrências do elemento "d" utilizando o método count().
10. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e encontre o índice da primeira ocorrência do elemento "b" utilizando o método index().
11. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e encontre o índice da primeira ocorrência do elemento "d" utilizando o método index() (gerando uma exceção se o elemento não estiver presente).
12. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e encontre o índice da última ocorrência do elemento "b" utilizando o método index().
13. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e encontre o índice da última ocorrência do elemento "d" utilizando o método index().
14. Crie uma lista A com os elementos "a", "b" e "c", e crie uma lista B vazia e adicione os elementos de A à B utilizando o método extend().
15. Crie uma lista A com os elementos "a", "b" e "c", e crie uma lista B vazia e adicione os elementos de A à B utilizando um laço.
16. Crie uma lista A com os elementos "a", "b" e "c", e remova todos os elementos utilizando o método clear().
17. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e remova todas as ocorrências do elemento "a" utilizando um laço.
18. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e remova todas as ocorrências do elemento "d" utilizando um laço.
19. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e remova a primeira ocorrência do elemento "b" utilizando o método remove().
20. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e remova a primeira ocorrência do elemento "d" utilizando o método remove() (gerando uma exceção se o elemento não estiver presente).
21. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e remova todas as ocorrências do elemento "b" utilizando o método remove() e um laço.
22. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e substitua a primeira ocorrência do elemento "b" pelo elemento "d" utilizando o método insert() e o método remove().
23. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e substitua a primeira ocorrência do elemento "d" pelo elemento "e" utilizando o método insert() e o método remove() (gerando uma exceção se o elemento "d" não estiver presente).
24. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e crie uma nova lista B com os elementos de A invertidos utilizando o método reverse().
25. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e crie uma nova lista B com os elementos de A ordenados em ordem crescente utilizando o método sort().

26. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e crie uma nova lista B com os elementos de A ordenados em ordem decrescente utilizando o método sort() e o argumento reverse=True.
27. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e crie uma nova lista B com os elementos de A ordenados por ordem alfabética utilizando o método sort().
28. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e crie uma nova lista B com os elementos de A ordenados por ordem reversa alfabética utilizando o método sort() e o argumento reverse=True.
29. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e crie uma nova lista B com os elementos de A ordenados pela quantidade de ocorrências do elemento utilizando a função sorted() e o método count().
30. Crie uma lista A com os elementos "a", "b", "c", "b" e "a", e crie uma nova lista B com os elementos de A ordenados pela quantidade de ocorrências do elemento em ordem decrescente utilizando a função sorted(), o método count() e o argumento reverse=True.

Exercícios de sets

1. Crie dois conjuntos A e B com elementos inteiros, e retorne a união deles.
2. Crie dois conjuntos A e B com elementos inteiros, e retorne a interseção deles.
3. Crie dois conjuntos A e B com elementos inteiros, e retorne a diferença entre eles (elementos presentes em A, mas não em B).
4. Crie dois conjuntos A e B com elementos inteiros, e retorne a diferença entre eles (elementos presentes em B, mas não em A).
5. Crie dois conjuntos A e B com elementos inteiros, e retorne a união deles sem elementos duplicados.
6. Crie dois conjuntos A e B com elementos inteiros, e retorne a interseção deles sem elementos duplicados.
7. Crie dois conjuntos A e B com elementos inteiros, e retorne a diferença simétrica (elementos presentes em A ou B, mas não em ambos).
8. Crie dois conjuntos A e B com elementos inteiros, e verifique se eles têm elementos em comum.
9. Crie dois conjuntos A e B com elementos inteiros, e verifique se A é um subconjunto de B.
10. Crie dois conjuntos A e B com elementos inteiros, e verifique se B é um subconjunto de A.
11. Crie um conjunto A com elementos inteiros e um conjunto B com elementos de outras classes (como strings), e retorne a união deles.
12. Crie um conjunto A com elementos inteiros e um conjunto B com elementos de outras classes (como strings), e retorne a interseção deles.
13. Crie um conjunto A com elementos inteiros e um conjunto B com elementos de outras classes (como strings), e retorne a diferença entre eles.
14. Crie um conjunto A com elementos inteiros e um conjunto B com elementos de outras classes (como strings), e verifique se eles têm elementos em comum.
15. Crie um conjunto A com elementos inteiros e um conjunto B com elementos de outras classes (como strings), e verifique se A é um subconjunto de B.
16. Crie um conjunto A com elementos inteiros e um conjunto B com elementos de outras classes (como strings), e verifique se B é um subconjunto de A.
17. Crie um conjunto A com elementos inteiros e remova todos os elementos de um conjunto B (que contém alguns elementos de A) de A.
18. Crie um conjunto A com elementos inteiros e adicione todos os elementos de um conjunto B (que contém alguns elementos que não estão em A) a A.
19. Crie um conjunto A com elementos inteiros e verifique se um elemento específico está presente em A.
20. Crie um conjunto A com elementos inteiros e remova um elemento específico de A (gerando uma exceção se o elemento não estiver presente).
21. Crie um conjunto A vazio e adicione o elemento "a" a ele.
22. Crie um conjunto A com os elementos "a", "b" e "c", e remova o elemento "a".
23. Crie um conjunto A com os elementos "a", "b" e "c", e remova o elemento "d" (gerando uma exceção se o elemento não estiver presente).
24. Crie um conjunto A com os elementos "a", "b" e "c", e remova o elemento "d" (sem gerar exceção, mesmo que o elemento não esteja presente).
25. Crie um conjunto A com os elementos "a", "b" e "c", e adicione o elemento "d" a ele.
26. Crie um conjunto A com os elementos "a", "b" e "c", e adicione os elementos "d" e "e" a ele.

27. Crie um conjunto A com os elementos "a", "b" e "c", e adicione os elementos "d" e "e" a ele utilizando um laço.
28. Crie um conjunto A com os elementos "a", "b" e "c", e remova todos os elementos utilizando um laço.
29. Crie um conjunto A com os elementos "a", "b" e "c", e verifique se o elemento "a" está presente no conjunto.
30. Crie um conjunto A com os elementos "a", "b" e "c", e verifique se o elemento "d" está presente no conjunto.
31. Crie um conjunto A com os elementos "a", "b" e "c", e remova um elemento aleatório utilizando o método pop().
32. Crie um conjunto A com os elementos "a", "b" e "c", e remova todos os elementos utilizando o método clear().
33. Crie um conjunto A com os elementos "a", "b" e "c", e crie um novo conjunto B igual a A e adicione o elemento "d" a ele.
34. Crie um conjunto A com os elementos "a", "b" e "c", e crie um novo conjunto B igual a A e remova o elemento "a" apenas do conjunto B.
35. Crie um conjunto A com os elementos "a", "b" e "c", e crie um novo conjunto B igual a A e remova o elemento "d" apenas do conjunto B (sem gerar exceção, mesmo que o elemento não esteja presente).
36. Crie um conjunto A com os elementos "a", "b" e "c", e crie um novo conjunto B igual a A e remova o elemento "d" apenas do conjunto B (gerando uma exceção se o elemento não estiver presente).
37. Crie um conjunto A com os elementos "a", "b" e "c", e crie um novo conjunto B igual a A e adicione os elementos "d" e "e" apenas ao conjunto B.
38. Crie um conjunto A com os elementos "a", "b" e "c", e crie um novo conjunto B igual a A e adicione os elementos "d" e "e" apenas ao conjunto B utilizando um laço.
39. Crie um conjunto A com os elementos "a", "b" e "c", e crie um novo conjunto B igual a A e remova todos os elementos apenas do conjunto B.
40. Crie um conjunto A com os elementos "a", "b" e "c", e crie um novo conjunto B igual a A e verifique se o elemento "a" está presente em ambos os conjuntos. Em seguida, remova o elemento "a" apenas do conjunto B e verifique novamente se ele está presente em ambos os conjuntos.