Set ou conjuntos

quinta-feira, 18 de agosto de 2022 20:37

A sintaxe para criação de um set é parecido com o dicionário ou usando o set():

```
sets = set()
sets2 = {}
```

Para se adicionar valor dentro do set usamos a função add e so podemos adicionar valores 1 por vez ou então listas, tuplas etc.

```
sets = set()
sets2 = {}

sets.add(1)
sets.add(2)
sets.add(3)
```

para remover usamos a função discard

```
sets.discard(2)
print(sets)
```

função update também é usada para adicionar ao set porém se usada uma string será feito uma iteração sem ordem.

```
sets.update("geronimo")
print(sets)
```

Resultado:

```
{1, 3, 'r', 'o', 'i', 'n', 'm', 'e', 'g'}

Process finished with exit code 0
```

Quando adicionamos valores dentro de um set ele não adiciona valores duplicados.

```
sets = set()
sets2 = {}

sets.update([1,2,3,5,4],{4,1,2,3,5})
print(sets)
```

Resultado:

```
{1, 2, 3, 4, 5}

Process finished with exit code 0
```

Podemos usar o cast para transformar uma lista em um set para não repetir valores e depois podemos retornar a lista novamente

```
lista = [1,2,3,3,2,2,1,43,5,4,1,2,1,2,1,3]
lista = set(lista)
print(lista)
```

Resultado:

```
{1, 2, 3, 4, 5, 43}

Process finished with exit code 0
```

Podemos unir 2 sets porém os valores duplicados não se repetirão

```
sets = {1,2,3,4,5}
sets2 = {1,2,3,4,5,6}

set3 = sets.union(sets,sets2)
print(set3)
```

Podemos usar a função union ou então usar o pipe (|) para digitar o pipe é alt + 124

```
primeiro = {1,2,1,1,4,4}
segundo = {1,2,1,1,4,4}

terceiro = primeiro | segundo
```

Quando usamos o simbolo do & pegamos apenas os elementos presentes nos 2 sets

```
primeiro = {1,2,1,1,4,4}
segundo = {1,2,1,1,4,4}

terceiro = primeiro & segundo
print(terceiro)
```

Resultado

```
{1, 2, 4}

Process finished with exit code 0
```

quando usamos o sinal de - pegamos apenas o elemento que tem de diferente no elemento que está subtraindo :

```
primeiro = {1,2,1,1,4,4,3}
segundo = {1,2,1,1,4,4}

terceiro = primeiro - segundo

print(terceiro)
```

Resultado:

```
{3}

Process finished with exit code 0
```

e usamos o ^ para pegar apenas os elementos que não se duplicam nos sets:

```
primeiro = {1,2,1,1,4,4,3}
segundo = {1,2,1,1,4,4,8}

terceiro = primeiro - segundo

print(terceiro)
```

Resultado:

```
{3, 8}

Process finished with exit code 0
```

Exercicio:

"""
-> É uma lista de listas de números inteiros
-> As listas internas tem o tamanho de 10 elementos
-> As listas internas contém números entre 1 a 10 e eles podem ser duplicados

Exercício

-> Crie uma função que encontra o primeiro duplicado considerando o segundo
número como a duplicação. Retorne a duplicação considerada.
Requisitos:
A ordem do número duplicado é considerada a partir da segunda
ocorrência do número, ou seja, o número duplicado em si.
Exemplo:
[1, 2, 3, ->3<-, 2, 1] -> 1, 2 e 3 são duplicados (retorne 3)
[1, 2, 3, 4, 5, 6] -> Retorne -1 (não tem duplicados)

Se não encontrar duplicados na lista, retorne -1
"""

```python
def avaliar(lista_numeros):  lista_numeros: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    numeros_checados = set()  numeros_checados: {1, 2, 3}
    primeiro_duplicado = -1  primeiro_duplicado: -1
    for n in lista_numeros:  n: 3
        if n in numeros_checados:
                primeiro_duplicado = n
                break
        numeros_checados.add(n)
    return primeiro_duplicado


lista_de_listas_de_inteiros = [
        [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        [9, 1, 8, 9, 9, 7, 2, 1, 6, 8],
        [1, 3, 2, 2, 8, 6, 5, 9, 6, 7],
        [3, 8, 2, 8, 6, 7, 7, 3, 1, 9],
        [4, 8, 8, 8, 5, 1, 10, 3, 1, 7],
        [1, 3, 7, 2, 2, 1, 5, 1, 9, 9],
        [10, 2, 2, 1, 3, 5, 10, 5, 10, 1],
        [1, 6, 1, 5, 1, 1, 1, 4, 7, 3],
        [1, 3, 7, 1, 10, 5, 9, 2, 5, 7],
        [4, 7, 6, 5, 2, 9, 2, 1, 2, 1],
        [5, 3, 1, 8, 5, 7, 1, 8, 8, 7],
        [10, 9, 8, 7, 6, 5, 4, 3, 2, 1],
    ]
for lista_nova in lista_de_listas_de_inteiros:
        print(avaliar(lista_nova))
```