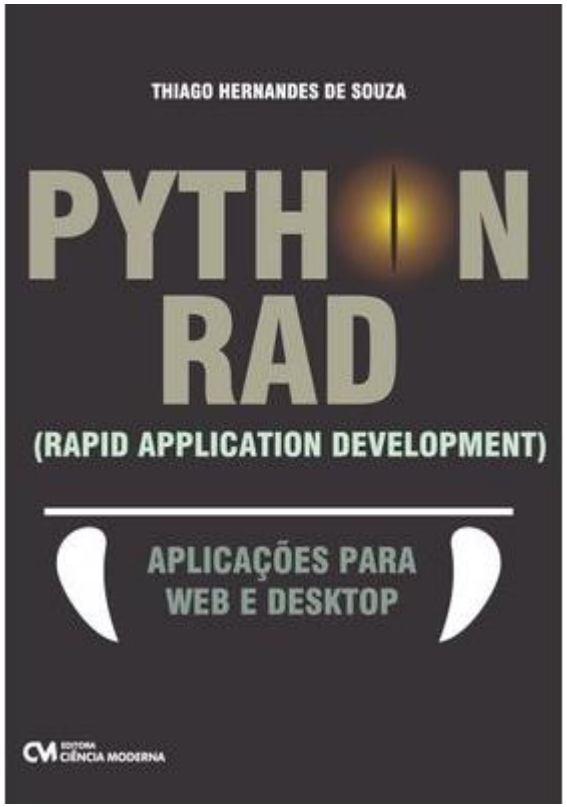
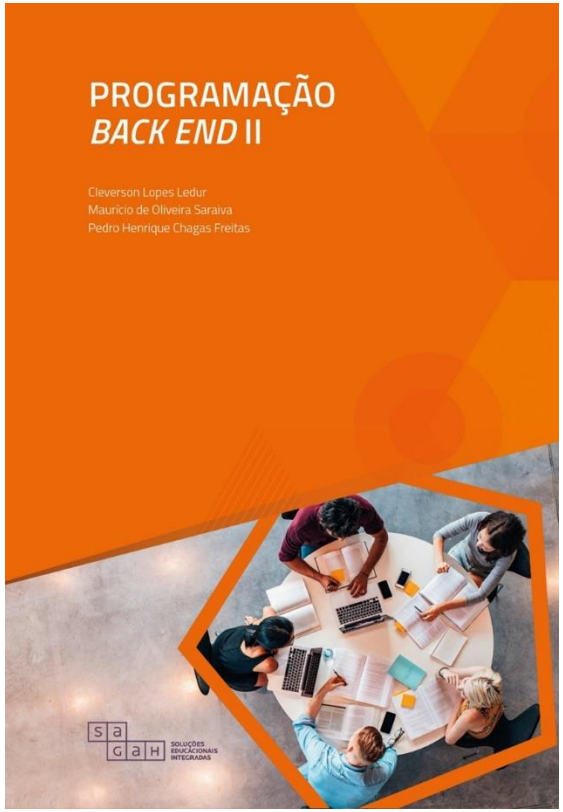
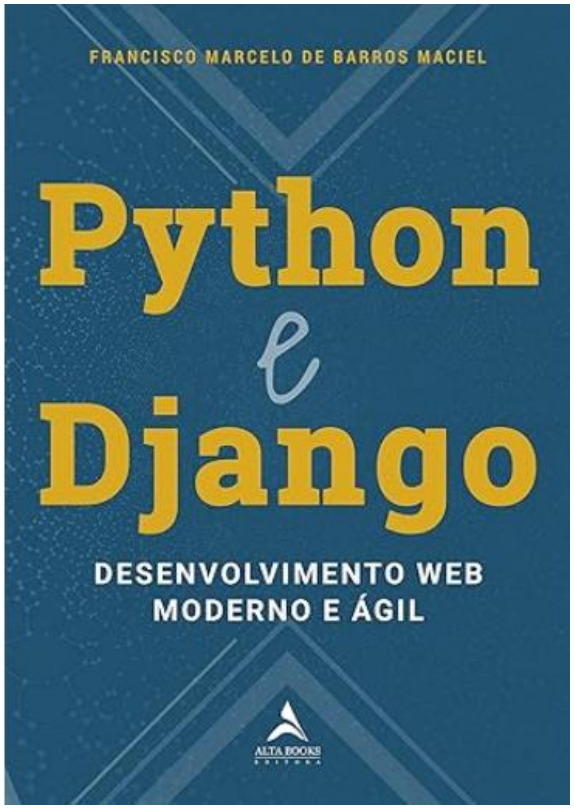
PROGRAMAÇÃO BACK-END







Python (Prática)

Conjuntos

Conjuntos

Conjuntos são usados para armazenar vários itens em uma única variável.

É um dos 4 tipos de dados integrados em Python usados para armazenar coleções de dados.

Um conjunto é uma coleção não ordenada, imutável e não indexada.

Os Conjuntos são criadas usando chaves { }.

```
teste69.py > ...
1    cores = {"vermelho", "azul", "verde", "amarelo"}
2    print(cores)
```

Observação:

- **Os itens do conjunto não podem ser alterados, mas você pode remover itens e adicionar novos itens.**

- **Os conjuntos não estão ordenados, portanto você não pode ter certeza da ordem em que os itens aparecerão.**

Itens duplicados não são permitidos.

Os conjuntos não podem ter dois itens com o mesmo valor.

teste70.py > ...

```
1    cores = {"vermelho", "azul", "verde", "amarelo", "azul"}
2    print(cores)
```

True e 1 é considerado o mesmo valor.

teste71.py > ...

```
1    cores = {"vermelho", "azul", "verde", "amarelo", True, 1}
2    print(cores)
```

False e 0 é considerado o mesmo valor.

teste72.py > ...

```
1    cores = {"vermelho", "azul", "verde", "amarelo", False, 0}
2    print(cores)
```

Obter o número de itens em um conjunto.

teste73.py > ...

```
1    cores = {"vermelho", "azul", "verde", "amarelo", "verde"}
2    print(len(cores))
```

Como fazer um loop pelo conjunto e imprimir os valores.

teste74.py > ...

```
1   cores = {"vermelho", "azul", "verde", "amarelo", "verde"}
2   for x in cores:
3     print(x)
```

Verifique se um item está presente no conjunto.

teste75.py > ...

```
1   cores = {"vermelho", "azul", "verde", "amarelo"}
2   print("verde" in cores)
```

Adicione um item a um conjunto usando o método add().

teste76.py > ...

```
1   cores = {"vermelho", "azul", "verde", "amarelo"}
2   cores.add("Laranja")
3   print(cores)
```

Para adicionar itens de outro conjunto ao conjunto atual, use o método update().

teste77.py > ...

```
1   cores = {"vermelho", "azul", "verde", "amarelo"}
2   coresnovas = {"Laranja","Pink"}
3   cores.update(coresnovas)
4   print(cores)
```

PROGRAMAÇÃO BACK-END

Adicionar elementos de uma lista ao conjunto.

lteste78.py > ...

```
1    cores = {"vermelho", "azul", "verde", "amarelo"}
2    listacores = ["Laranja","Pink"]
3    cores.update(listacores)
4    print(cores)
```

Remova um item usando o método remove()

teste78.py > ...

```
1    cores = {"vermelho", "azul", "verde", "amarelo"}
2    cores.remove("vermelho")
3    print(cores)
```

Remova um item usando o método discard().

teste79.py > ...

```python
cores = {"vermelho", "azul", "verde", "amarelo"}
cores.discard("vermelho")
print(cores)
```

Remova um item aleatório usando o método pop().

teste80.py > ...

```python
cores = {"vermelho", "azul", "verde", "amarelo"}
x = cores.pop()
print(x)
print(cores)
```

O método clear() esvazia o conjunto

```
teste81.py > ...
1    cores = {"vermelho", "azul", "verde", "amarelo"}
2    cores.clear()
3    print(cores)
```

A palavra-chave del excluirá o conjunto.

```
teste81.py > ...
1    cores = {"vermelho", "azul", "verde", "amarelo"}
2    del cores
3    print(cores)
```

Pode-se fazer um loop usando for pelo conjunto e imprimir os seus itens.

teste82.py > ...

```
1   cores = {"vermelho", "azul", "verde", "amarelo"}
2   for x in cores:
3       print(x)
```

O método union() retorna um novo conjunto com todos os itens de ambos os conjuntos,

teste83.py > ...

```
1   cores1 = {"vermelho", "azul", "verde", }
2   cores2 = {"laranja", "azul", "verde", "amarelo"}
3   cores = cores1.union(cores2)
4   print(cores)
```

Métodos integrados para usar em conjuntos

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not

Métodos integrados para usar em conjuntos

pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others