Dicionários

Os dicionários são usados para armazenar valores de dados em pares **chave**:**valor**.

Um dicionário é uma coleção que é ordenada*, modificável e não permite duplicatas.

Os dicionários são escritos com chaves e têm chaves e valores:

my_dict = {chave1:valor1, chave2:valor2, …, chaveN:valorN}

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

```
meus_alunos = {
'2022-2-10':'Maria Joaquina',
'2022-2-11':'Francisco Carvalho',
'2022-2-12':'Antônio da Silva'
}
print(meus_alunos)
```

```
{'2022-2-10': 'Maria Joaquina', '2022-2-11': 'Francisco Carvalho', '2022-2-12': 'Antônio da Silva'}
```

Itens de um dicionário

Os itens do dicionário são ordenados, alteráveis e não permitem duplicatas.

Os itens do dicionário são apresentados em pares **chave**:**valor** e podem ser referidos usando o nome da chave.

my_dict = {chave1:valor1, chave2:valor2, …, chaveN:valorN}
item1 = chave1:valor1
item2 = chave2:valor2
…
itemN = chaveN:valorN

my_dict[chave1] = valor1
my_dict[chave2] = valor2
…
my_dict[chaveN] = valorN

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964}
print(thisdict["brand"])
print(thisdict["model"])
print(thisdict["year"])
```

```
Ford
Mustang
1964
```

```
meus_alunos = {
'2022-2-10':'Maria Joaquina',
'2022-2-11':'Francisco Carvalho',
'2022-2-12':'Antônio da Silva'
}
print(meus_alunos['2022-2-10'])
print(meus_alunos['2022-2-11'])
print(meus_alunos['2022-2-12'])
```

```
Maria Joaquina
Francisco Carvalho
Antônio da Silva
```

Ordem e acesso aos itens do dicionário

Quando dizemos que os dicionários são ordenados, significa que os itens possuem uma ordem definida, e essa ordem não será alterada.

Não ordenado significa que os itens não possuem uma ordem definida, você não pode se referir a um item usando um índice.

my_dict = {chave1:valor1, chave2:valor2, …, chaveN:valorN}
item1 = chave1:valor1
item2 = chave2:valor2
…
itemN = chaveN:valorN

Ordem dos itens: item1, item2, …, itemN

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964}

for item in thisdict.items():
    print(item)
```

```
('brand', 'Ford')
('model', 'Mustang')
('year', 1964)
```

Mutabilidade do dicionário

Os dicionários são "alteráveis", ou seja, podemos alterar, adicionar ou remover itens após a criação do dicionário.

my_dict = {chave1:valor1, chave2:valor2}

Adicionando um novo item (chave3, valor3)
my_dict[chave3] = valor3

my_dict = {chave1:valor1, chave2:valor2, chave3:valor3}

Removendo o item1 (chave1, valor1)
my_dict.pop(chave1)

my_dict = {chave2:valor2, chave3:valor3}

Os dicionários não podem ter dois itens com a mesma chave.

O tamanho do dicionário corresponde a quantidade de itens

len(my_dict) = 2

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
# Adicionando um novo item
# item = ("plate", "PZ65 BYV")
thisdict["plate"] = "PZ65 BYV"

for item in thisdict.items():
    print(item)

# Tamanho do dicionario
print(len(thisdict))
```

```
('brand', 'Ford')
('model', 'Mustang')
('year', 1964)
('plate', 'PZ65 BYV')
4
```

Tipos suportados para itens de um dicionário

String, int, booleano e tipos de dados de lista.

Da perspectiva do Python, os dicionários são definidos como objetos com o tipo de dados 'dict'

my_dict = {chave1:valor1, chave2:valor2, …, chaveN:valorN}

type(my_dict) retorna class 'dict'

```
thisdict = {
  1: "6Fa 0ufAA5 AH da5067",
  'license':True,
  "brand": "Ford",
  "electric": False,
  "year": 1964,
  "colors": ["red", "white", "blue"]
}

for item in thisdict.items():
    print(item)
```

```
(1, '6Fa 0ufAA5 AH da5067')
('license', True)
('brand', 'Ford')
('electric', False)
('year', 1964)
('colors', ['red', 'white', 'blue'])
```

O constructor dict()

Também é possível usar o construtor **dict()** para criar um dicionário.

Por exemplo: para criar um dicionário vazio:

my_dict = {}
ou
my_dict = dict()

my_dict2 = dict(chave1=valor1, chave2=valo2,...chaveN=valorN)

```
meus_carros = {}
print(f'meus_carros: {meus_carros}')
meus_alunos = dict()
print(f'meus_alunos: {meus_alunos}')
```

```
meus_carros: {}
meus_alunos: {}
```

```
thisdict = dict(name = "John", age = 36, country = "Norway")
print(thisdict)
```

```
{'name': 'John', 'age': 36, 'country': 'Norway'}
```

Acessando itens de um dicionário

Você pode acessar os itens de um dicionário referindo-se ao seu nome de chave, entre colchetes:

my_dict = {chave1:valor1, chave2:valor2, …, chaveN:valorN}

my_dict[chave1] = valor1
my_dict[chave2] = valor2
…
my_dict[chaveN] = valorN

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
valor1 = thisdict["brand"]
valor2 = thisdict["model"]
valor3 = thisdict["year"]

print(f'thisdict["brand"]: {valor1}')
print(f'thisdict["model"]: {valor2}')
print(f'thisdict["year"]: {valor3}')
```

```
thisdict["brand"]: Ford
thisdict["model"]: Mustang
thisdict["year"]: 1964
```

Acessando as chaves dos itens de um dicionário

Você pode acessar os as chaves de cada item de um dicionário chamando o método **keys()**

my_dict = {chave1:valor1, chave2:valor2, …, chaveN:valorN}

my_dict.keys()
retorna um objeto iterable
dict_keys([chave1, chave2, …, chaveN])

dict_keys é um Iterable, ou seja, pode ser percorrido em um **for..in**

```
for each in my_dict.keys():
  print(each)
```

retorna
chave1
chave2
…
chaveN

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

chaves = thisdict.keys()
print(chaves)
print(type(chaves))

# Chaves é um iterable
for chave in chaves:
    print(chave)
```

```
dict_keys(['brand', 'model', 'year'])
<class 'dict_keys'>
brand
model
year
```

Acessando os valores dos itens de um dicionário

Você pode acessar os valores de cada item de um dicionário chamando o método **values()**

my_dict = {chave1:valor1, chave2:valor2, …, chaveN:valorN}

my_dict.values()
retorna um objeto iterable
dict_values([valor1, valor2, …, valorN])

dict_values é um Iterable, ou seja, pode ser percorrido em um **for..in**

```
for each in my_dict.values():
  print(each)
```

retorna

chave1
chave2
…
chaveN

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

valores = thisdict.values()
print(valores)
print(type(valores))

# Valores é um iterable
for valor in valores:
    print(valor)
```

```
dict_values(['Ford', 'Mustang', 1964])
<class 'dict_values'>
Ford
Mustang
1964
```

Acessando os itens de um dicionário

Você também pode acessar os itens de um dicionário chamando o método **items()**

my_dict = {chave1:valor1, chave2:valor2, …, chaveN:valorN}

my_dict.items()
retorna um objeto iterable
dict_items[(chave1, valor1), (chave2, valor2), …, (chaveN, valorN)]

Corresponde a um lista de tuplas de itens que pode ser percorrido em um **for..in**

```
for each in my_dict.items():
  print(each)
```

retorna
(chave1, valor1)
(chave2, valor2)
…
(chaveN, valorN)

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

itens = thisdict.items()
print(itens)
print(type(itens))

# Valores é um iterable
for each in itens:
    print(each)
```

```
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
<class 'dict_items'>
('brand', 'Ford')
('model', 'Mustang')
('year', 1964)
```

Verificando se uma chave existe no dicionário

Para determinar se uma chave especificada está presente em um dicionário, use a palavra-chave **in**:

my_dict = {chave1:valor1, chave2:valor2, …, chaveN:valorN}

```
if chave1 in my_dict:
  print("Sim, a chave1 é uma chave do dicionário my_dict")
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

itens = thisdict.items()
print(itens)
print(type(itens))

# Valores é um iterable
for each in itens:
    print(each)
```

```
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
<class 'dict_items'>
('brand', 'Ford')
('model', 'Mustang')
('year', 1964)
```

Alterando itens de um dicionário

Você pode alterar o valor de um item específico referindo-se ao seu nome de chave.

Você pode usar o método **update()** para atualizar o dicionário com os itens do argumento fornecido.

O argumento deve ser um dicionário ou um objeto iterável com pares **chave:valor**.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

print(f'original: {thisdict}')

# Alterando usando a chave
thisdict["year"] = 2018

print(f'Item com chave year alterado para 2018: {thisdict}')

# Alterando passando o item
thisdict.update({"year": 2020})

print(f'Update do item ("year":2020) {thisdict}')
```

```
original: {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
Item com chave year alterado para 2018: {'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
Update do item ("year":2020) {'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

Adicionando itens de um dicionário

Adicionar um item ao dicionário é feito usando uma nova chave de índice e atribuindo um valor a ela.

O método **update()** atualizará o dicionário com os itens de um determinado argumento. Se o item não existir, o item será adicionado.

O argumento deve ser um dicionário ou um objeto iterável com pares **chave:valor**.

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

print(f'Original: {thisdict}')

# Adicionando um novo item usando a chave
thisdict["color"] = "red"

print(f'Novo item com chave "color" e valor "red {thisdict}"')

thisdict.update({"plate": "BXE PZ24"})

print(f'Novo item ("plate", "BXE PZ24") {thisdict}')
```

```
Original: {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
Novo item com chave "color" e valor "red {'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}"
Novo item ("plate", "BXE PZ24") {'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red', 'plate': 'BXE PZ24'}
```

Removendo itens de um dicionário

Existem vários métodos para remover itens de um dicionário.

O método **pop()** remove o item com o nome de chave especificado

O método **popitem()** remove o último item inserido.

A palavra-chave **del** remove o item com o nome de chave especificado.

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(f'Original: {thisdict}')

thisdict.pop("model")
print(f'Dicionário após a remoção de "model": {thisdict}')

thisdict.popitem()
print(f'Dicionário sem o último elemento: {thisdict}')
```

```
Original: {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
Dicionário após a remoção de "model": {'brand': 'Ford', 'year': 1964}
Dicionário sem o último elemento: {'brand': 'Ford'}
```

Copiando os itens de um dicionário para outro dicionário

Você não pode copiar um dicionário simplesmente digitando dict2 = dict1, porque: dict2 será apenas uma referência a dict1, e as alterações feitas em dict1 também serão feitas automaticamente em dict2.

Existem maneiras de fazer uma cópia, uma delas é usar o método interno **copy()** do Dictionary.

Outra maneira de fazer uma cópia é usar a função interna **dict()**

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(f'Original: {thisdict}')

mydict1 = thisdict.copy()
print(f'Cópia 1: {mydict1}')

mydict2 = dict(thisdict)
print(f'Cópia 2: {mydict2}')
```

```
Original: {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
Cópia 1: {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
Cópia 2: {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Dicionários aninhados (*Nested dicionaries*)

Um dicionário pode conter dicionários, isso é chamado de dicionários aninhados.

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}

for item in myfamily.items():
    print(item)
```

```
('child1', {'name': 'Emil', 'year': 2004})
('child2', {'name': 'Tobias', 'year': 2007})
('child3', {'name': 'Linus', 'year': 2011})
```

Adicionando dicionários dentro de um dicionário

Crie três dicionários e, em seguida, crie um dicionário que conterá os outros três dicionários.

```
child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
child3 = {
  "name" : "Linus",
  "year" : 2011
}

myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}

for item in myfamily.items():
    print(item)
```

```
('child1', {'name': 'Emil', 'year': 2004})
('child2', {'name': 'Tobias', 'year': 2007})
('child3', {'name': 'Linus', 'year': 2011})
```

Métodos mais comuns em dicionários

Python tem um conjunto de métodos integrados que você pode usar em dicionários.

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary