



Dicionários em Python

Prof. Fábio Procópio

Prof. João Nascimento



2

Relembrando...

- Na [aula passada](#), estudamos uma estrutura de dados conhecida como **Lista**. Os principais métodos e funções associados a esse tipo de estrutura de dados que estudamos foram:
 - `append()` e `insert()`
 - `remove()` e `pop()`
 - `len()`, `min()`, `max()` e `sum()`
 - `index()`
 - `enumerate()`





Introdução

- **Dicionário** é uma coleção de dados não ordenada, modificável e indexada
 - Consiste no relacionamento entre uma chave e um valor específico (chave → valor), ou seja, é a definição de um mapeamento em memória
 - Diferentemente das listas, onde o índice é um número, os dicionários utilizam suas próprias chaves como sendo índices
- Para criar um dicionário, usa-se um par de chaves {}:

```
curros = {"Info": "Técnico em Informática",  
          "Meca": "Técnico em Mecatrônica"}  
print(curros)
```

- Outra alternativa para criar esse mesmo dicionário seria:

```
curros = {}  
curros["Info"] = "Técnico em Informática"  
curros["Meca"] = "Técnico em Mecatrônica"  
print(curros)
```



4

E aí: Listas x Dicionários?

- Imagine a situação em que queiramos armazenar os contatos do Instagram...
- Considerando que já conhecemos a estrutura de dados lista, uma solução seria definir duas variáveis desse tipo: uma contendo os nomes e outra contendo as páginas Insta.

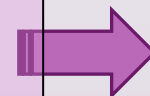
```
nomes = ["Camila Queiroz", "Paolla de Oliveira"]  
instagram = ["@camilaqueiroz", "@paollaoliveirareal"]
```

- Se desejássemos imprimir o contato de Paola, faríamos o seguinte:

```
ind = nomes.index("Paolla de Oliveira")  
print(instagram[ind])
```

- No entanto, o uso de dicionários deixaria a implementação muito mais simples:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
           "Paolla de Oliveira": "@paollaoliveirareal"}  
print(contato["Paolla de Oliveira"])
```



@paollaoliveirareal



5

Método get()

- Outra maneira de acessar um elemento de um dicionário é usando o método **get()**;
- Sintaxe básica:

```
nome_dicionario.get(<chave>)
```

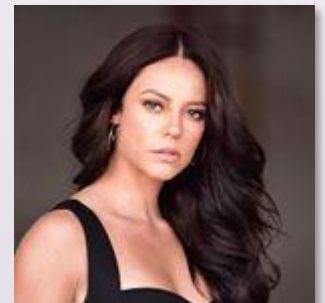
- Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
          "Paolla de Oliveira": "@paollaoliveira"}  
print(contato.get("Paolla de Oliveira"))
```

@paollaoliveira



Camila Queiroz



Paolla de Oliveira

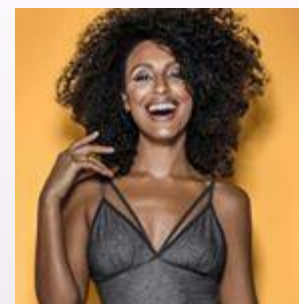
Método update()

- Usado para **inserir/atualizar** um par chave-valor de um dicionário especificado
 - Quando a chave já existe, o valor associado é alterado
 - Quando não, a chave e o valor são adicionados
- Sintaxe básica:

```
nome_dicionario.update({chave:valor})
```

Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",
           "Paolla de Oliveira": "@paollaoliveirareal"}
contato.update({"Paolla de Oliveira": "@paollaoliveira"})
contato.update({"Sheron Menezes": "@sheronmenezes"})
contato.update({"Bruna Marquezine": "@bruna_iconica"})
print(contato)
```



Sheron Menezes



Bruna Marquezine

Como Paola de Oliveira já é **cadastrada**, seu contato do Instagram será **atualizado**.

Sheron Menezes e Bruna Marquezine ainda não são **cadastradas**, portanto, seus registros serão **incluídos**.



Função len()

- Usado para determinar a **quantidade de itens** de um dicionário;
- Sintaxe básica:

```
len(nome_dicionário)
```

- Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
          "Paolla de Oliveira": "@paollaoliveira"}  
print(len(contato))
```



8

Palavra-chave in

- Usada para verificar se um **chave existe** em um dicionário. Se existir, retorna True e, caso contrário, retorna False;
- Sintaxe básica:

```
<chave> in <nome_dicionário>
```

- Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
          "Paolla de Oliveira": "@paollaoliveira"}  
  
if "Paola de Oliveira" in contato:  
    print("Insta de Paola é {}".format(contato.get("Paolla de Oliveira")))
```




Método keys()

- Usado para **retornar as chaves** do dicionário como uma lista;
- Sintaxe básica:

```
nome_dicionario.keys()
```

- Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
          "Paolla de Oliveira": "@paollaoliveira",  
          "Sheron Menezes": "@sheronmenezes",  
          "Bruna Marquezine": "@bruna_iconica"}  
for nome in contato.keys():  
    print(nome)
```



```
Camila Queiroz  
Paola de Oliveira  
Sheron Menezes  
Bruna Marquezine
```



Método values()


➤ Usado para **retornar os valores** do dicionário, como uma lista;

➤ Sintaxe básica:

```
nome_dicionario.values()
```

➤ Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
           "Paolla de Oliveira": "@paollaoliveira",  
           "Sheron Menezes": "@sheronmenezes",  
           "Bruna Marquezine": "@bruna_iconica"}  
for insta in contato.values():  
    print(insta)
```



```
@camilaqueiroz  
@paollaoliveira  
@sheronmenezes  
@bruna_iconica
```



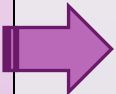
Método items()

- Usado para **retornar** um objeto contendo os **pares chave-valor** do dicionário, como uma tupla em uma lista;
- Sintaxe básica:

```
nome_dicionario.items()
```

- Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
           "Paolla de Oliveira": "@paollaoliveira",  
           "Sheron Menezes": "@sheronmenezes",  
           "Bruna Marquezine": "@bruna_iconica"}  
for nome, insta in contato.items():  
    print("{} --> {}".format(nome, insta))
```



```
Camila Queiroz --> @camilaqueiroz  
Paola de Oliveira --> @paollaoliveira  
Sheron Menezes --> @sheronmenezes  
Bruna Marquezine --> @bruna_iconica
```



Função sorted() – 1 de 3

- Usado para retornar uma **lista ordenada**, com base nas chaves, de um dicionário informado;

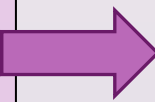
- Sintaxe básica:

```
sorted(nome_dicionario)
```

- Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
          "Paolla de Oliveira": "@paollaoliveira",  
          "Sheron Menezes": "@sheronmenezes",  
          "Bruna Marquezine": "@bruna_iconica"}
```

```
for nome, insta in sorted(contato.items()):  
    print("{} --> {}".format(nome, insta))
```



```
Bruna Marquezine --> @bruna_iconica  
Camila Queiroz --> @camilaqueiroz  
Paola de Oliveira --> @paollaoliveira  
Sheron Menezes --> @sheronmenezes
```



Função sorted() – 2 de 3

- Caso seja necessário ordenar um dicionário com base nos valores (e não com base nas chaves), pode-se utilizar a função **itemgetter()** dentro da função **sorted()**;

- Sintaxe básica:

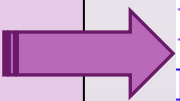
```
sorted(nome_dicionario.items(), key=itemgetter(1))
```

- Exemplo:

```
from operator import itemgetter
```

```
contato = {"Camila Queiroz": 1.77,  
           "Paolla de Oliveira": 1.70,  
           "Sheron Menezes": 1.67,  
           "Bruna Marquezine": 1.70}
```

```
for nome, estatura in sorted(contato.items(), key=itemgetter(1)):  
    print("{} --> {:.2f}m".format(nome, estatura))
```



```
Sheron Menezes --> 1.67m  
Paolla de Oliveira --> 1.70m  
Bruna Marquezine --> 1.70m  
Camila Queiroz --> 1.77m
```



Função sorted() – 3 de 3

- No entanto, se a ordenação do dicionário for com base nos valores e de forma decrescente, utiliza-se o parâmetro **reverse** com o valor **True** (como vimos em Listas);

- Sintaxe básica:

```
sorted(nome_dicionario.items(), key=itemgetter(1), reverse = [False | True])
```

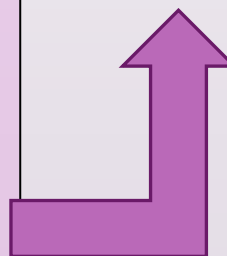
- Exemplo:

```
from operator import itemgetter
```

```
contato = {"Camila Queiroz": 1.77,  
           "Paolla de Oliveira": 1.70,  
           "Sheron Menezes": 1.67,  
           "Bruna Marquezine": 1.70}
```

```
for nome, estatura in sorted(contato.items(), key=itemgetter(1), reverse = True):  
    print("{} --> {:.2f}m".format(nome, estatura))
```

```
Camila Queiroz --> 1.77m  
Paolla de Oliveira --> 1.70m  
Bruna Marquezine --> 1.70m  
Sheron Menezes --> 1.67m
```





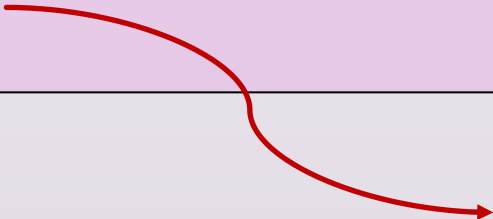
Método copy()

- Usado para **retornar uma cópia** do dicionário especificado;
- Sintaxe básica:

```
nome_dicionario.copy()
```

- Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
          "Paolla de Oliveira": "@paollaoliveira",  
          "Sheron Menezes": "@sheronmenezes",  
          "Bruna Marquezine": "@bruna_iconica"}  
backup = contato.copy()  
print(backup)
```



```
{'Camila Queiroz': '@camilaqueiroz',  
 'Paola de Oliveira': '@paollaoliveira',  
 'Sheron Menezes': '@sheronmenezes',  
 'Bruna Marquezine': '@bruna_iconica'}
```



Método pop()

- Usado para **remover o item** do dicionário a partir de um chave especificada;
- Sintaxe básica:

```
nome_dicionario.pop(<chave>)
```

- Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
          "Paolla de Oliveira": "@paollaoliveira",  
          "Sheron Menezes": "@sheronmenezzes",  
          "Bruna Marquezine": "@bruna_iconica"}  
contato.pop("Paolla de Oliveira")  
print(contato)
```

Paolla de Oliveira foi removida.

```
{'Camila Queiroz': '@camilaqueiroz',  
'Sheron Menezes': '@sheronmenezzes',  
'Bruna Marquezine': '@bruna_iconica'}
```




Método popitem()

- A partir da versão 3.7, tem sido usado para **excluir o último elemento** incluído no dicionário. Nas versões anteriores, removia um elemento aleatoriamente;

- Sintaxe básica:

```
nome_dicionario.popitem()
```

- Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
          "Paolla de Oliveira": "@paollaoliveira",  
          "Sheron Menezes": "@sheronmenezes",  
          "Bruna Marquezine": "@bruna_iconica"}  
contato.popitem()  
print(contato)
```

Bruna Marquezine foi removida.

```
{'Camila Queiroz': '@camilaqueiroz',  
 'Paola de Oliveira': '@paollaoliveira',  
 'Sheron Menezes': '@sheronmenezes'}
```



Método clear()

- Usado para **esvaziar** um dicionário;
- Sintaxe básica:

```
nome_dicionario.clear()
```

- Exemplo:

```
contato = {"Camila Queiroz": "@camilaqueiroz",  
           "Paolla de Oliveira": "@paollaoliveira",  
           "Sheron Menezes": "@sheronmenezes",  
           "Bruna Marquezine": "@bruna_iconica"}  
contato.clear()  
print(contato)
```



Exercício Resolvido 1 – 1 de 2

1. Construa um programa que utilize um dicionário cujas chaves são os códigos do produto e os valores são o nome do produto, o preço unitário e a quantidade comprada, como no exemplo abaixo.

CHAVE Código	VALOR		
	Nome	Preço Unitário	Comprada
1	Monitor LED 24"	599,99	1
2	Teclado wireless	49,26	1
3	Mouse wireless	19,90	1
4	Cartucho colorido	54,00	2

A partir do dicionário, o programa deve imprimir os itens da compra em ordem crescente e calcular o subtotal de cada um deles, ou seja, quantidade * preço unitário. Por fim, o programa deve apresentar o valor total da compra.



Exercício Resolvido 1 – 2 de 2

```
produtos = {1: ['Monitor LED 24"', 599.99, 1],  
            2: ['Teclado wireless', 49.26, 1],  
            3: ['Mouse wireless', 19.9, 1],  
            4: ['Cartucho colorido', 54, 2]}  
  
total = 0  
for cod, prod in sorted(produtos.items()):  
    subtotal = produtos[cod][1] * produtos[cod][2]  
    print(prod[cod][0] + ": R$ {:.2f}".format(subtotal))  
    total += subtotal  
print(10 * "-")  
print("Total: R$ {:.2f}".format(total))
```



Exercício Resolvido 2 – 1 de 2

2. No exercício anterior, nós definimos as chaves e os valores do dicionário. Mas, se quiséssemos que o próprio usuário informasse esses dados? Veja o exemplo abaixo.



Exercício Resolvido 2 – 2 de 2

```
produtos = {}  
while True:  
    cod = int(input("Código: "))  
    nome = input("Nome: ")  
    preco = float(input("R$: "))  
    qtde = int(input("Qtde: "))  
    prod = []  
    prod.append(nome)  
    prod.append(preco)  
    prod.append(qtde)  
    produtos.update({cod: prod})  
    resp = input("Deseja continuar [S | N]? ")  
    if resp == "N" or resp == "n":  
        break  
total = 0  
for cod, prod in sorted(produtos.items()):  
    subtotal = produtos[cod][1] * produtos[cod][2]  
    print(prod[0] + ": R$ {:.2f}".format(subtotal))  
    total += subtotal  
print(10 * "-")  
print("Total: R$ {:.2f}".format(total))
```



Material complementar

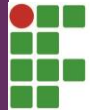
► Para complementar este material, assista às minhas vídeo-aulas no canal Procópio na Rede:

- 1) [Exemplo 1](#)
- 2) [Exemplo 2](#)
- 3) [Exemplo 3](#)
- 4) [Exemplo 4](#)
- 5) [Exemplo 5](#)



Exercícios de Fixação

- 1) Crie um dicionário para cadastrar os dados de funcionários, tais como nome, telefone, sexo, e-mail e salário. A quantidade de funcionários a serem cadastrados é indeterminada. Ao fim do cadastro, imprima um relatório contendo os dados dos funcionários ordenados alfabeticamente.
- 2) Baseado no cadastro anterior, crie um programa para que o usuário informe o nome de um funcionário e o sistema retorne os seus dados. Caso o nome informado não exista no sistema, deve ser informada uma mensagem para alertar o usuário da inexistência do cadastro.
- 3) Ainda baseado no cadastro de funcionários, crie um programa para aumentar o salário das funcionárias, em 10%, da empresa cujo salário é inferior a R\$ 3.000,00. Ao fim, imprima o novo cadastro de funcionários ordenado alfabeticamente.



Referências

- 1) Esperança, Cláudio. **Python: Dicionários.** Disponível em: http://orion.lcg.ufrj.br/python/_06%20-%20Programando%20em%20Python%20-%20Dicionarios.pdf. Acessado em: 31 jan. 2019.
- 2) Filho, Cláudio R. C.. **Funções dos Dicionários no Python.** Disponível em: <http://excript.com/python/funcoes-dicionarios.html>. Acessado em: 31 jan. 2019.
- 3) EXCRIPT. **Funções dos dicionários no Python.** Disponível em: <http://excript.com/python/funcoes-dicionarios.html>. Acessado em: 31 jan. 2019.
- 4) ALIENRETRÔ. **O que é lambda.** Disponível em: <http://blog.alienretro.com/entendendo-python-lambda/>. Acessado em: 31 jan. 2019.