



**Instituto de  
Computação**

UNIVERSIDADE ESTADUAL DE CAMPINAS



## MC102 – Aula 07

# Dicionários

## Algoritmos e Programação de Computadores

---

Zanoni Dias

2020

Instituto de Computação

Dicionários

Exercícios

# Dicionários

---

# Dicionários

- Dicionários são estruturas de chave-valor, ou seja, os valores (dados) estão sempre associados a uma chave.
- Exemplos de declaração de um dicionário:

```
1 dicionario = {} # dicionario vazio.  
2 print(type(dicionario))  
3 # <class 'dict'>
```

```
1 localizacao = {  
2     "Lat": -22.817087,  
3     "Long": -47.069750  
4 }  
5 print(type(localizacao))  
6 # <class 'dict'>
```

- Podemos também declarar um dicionário de maneira explícita utilizando a função `dict`.

```
1 dicionario = dict({}) # dicionario vazio.  
2 print(type(dicionario))  
3 # <class 'dict'>
```

```
1 localizacao = dict(  
2     "Lat": -22.817087,  
3     "Long": -47.069750  
4 })  
5 print(type(localizacao))  
6 # <class 'dict'>
```

- Chaves e valores em um dicionário podem ser de diferentes tipos de dados (`int`, `float`, `bool`, entre outros).

```
1 dicionario = {  
2     1.2: True,  
3     123: "Um Dois Três",  
4     "cat": "dog",  
5     ("X", "Y"): (2, 3, 5)  
6 }  
7 print(dicionario)  
8 # {1.2: True, 123: 'Um Dois Três', 'cat': 'dog',  
9 #  ('X', 'Y'): (2, 3, 5)}
```

- Geralmente as chaves são mantidas na ordem em que o dicionário é criado ou alterado.

```
1 dicionario = {  
2     "Z": 1,  
3     "A": 2,  
4     "C": 3  
5 }  
6 print(dicionario)  
7 # {'Z': 1, 'A': 2, 'C': 3}
```

# Dicionários

- Podemos acessar um valor do dicionário da seguinte forma:

```
1 <dicionário>[<chave>]
```

- Essa operação retorna o valor associado à chave informada.
- Exemplo:

```
1 dicionario = {  
2     "Nome": "Ash Ketchum",  
3     "Idade": 10,  
4     "Profissão": "Treinador Pokémon"  
5 }  
6 print(dicionario["Nome"])  
7 # Ash Ketchum  
8 print(dicionario["Idade"])  
9 # 10  
10 print(dicionario["Profissão"])  
11 # Treinador Pokémon
```



# Dicionários

- E se informarmos uma chave que não está no dicionário? O que acontece?
- Exemplo:

```
1 dicionario = {  
2     "Nome": "Ash Ketchum",  
3     "Idade": 10,  
4     "Profissão": "Treinador Pokémon"  
5 }  
6 print(dicionario["Cidade"])  
7 # KeyError: 'Cidade'
```

- Um erro relacionado à chave é gerado.

# Dicionários

- Similar ao que vimos em listas, podemos verificar se uma chave está presente ou não em um dicionário utilizando o operador de inclusão `in`.
- Exemplo:

```
1 dicionario = {  
2     "Nome": "Ash Ketchum",  
3     "Idade": 10,  
4     "Profissão": "Treinador Pokémon"  
5 }  
6 print("Cidade" in dicionario)  
7 # False  
8 print("Nome" in dicionario)  
9 # True
```

# Dicionários

- O método `get` é outra forma para obter valores de um dicionário. Ele recebe como parâmetro a chave associada ao valor desejado.
- Caso a chave não seja encontrada no dicionário, será retornado `None` como resposta.
- Exemplo:

```
1 dicionario = {  
2     "Nome": "Ash Ketchum",  
3     "Idade": 10,  
4     "Profissão": "Treinador Pokémon"  
5 }  
6 print(dicionario.get("Nome"))  
7 # Ash Ketchum  
8 print(dicionario.get("Cidade"))  
9 # None
```

# Dicionários

- O tamanho de um dicionário também pode ser verificado através da função `len`.
- Cada conjunto de chave e valor corresponde a um elemento.
- Exemplo:

```
1 dicionario = {  
2     "Nome": "Ash Ketchum",  
3     "Idade": 10,  
4     "Profissão": "Treinador Pokémon"  
5 }  
6 print(len(dicionario))  
7 # 3
```

# Dicionários

- Novos valores podem ser adicionados dinamicamente em um dicionário informando o novo par chave-valor.

```
1 <dicionário>[<nova_chave>] = <novo_valor>
```

- Exemplo:

```
1 dicionario = {  
2     "Nome": "Ash Ketchum",  
3     "Idade": 10,  
4     "Profissão": "Treinador Pokémon"  
5 }  
6 dicionario["Cidade"] = "Pallet"  
7 print(dicionario)  
8 # {'Nome': 'Ash Ketchum', 'Idade': 10, 'Profissão':  
9 #  'Treinador Pokémon', 'Cidade': 'Pallet'}
```

# Dicionários

- Para atualizar um valor já existente em um dicionário, basta atribuir à chave o valor atualizado.

```
1 <dicionário>[<chave>] = <novo_valor>
```

- Exemplo:

```
1 dicionario = {  
2     "Nome": "Ash Ketchum",  
3     "Idade": 10,  
4     "Profissão": "Treinador Pokémon",  
5     "Cidade": "Pallet"  
6 }  
7 dicionario["Idade"] = 12  
8 print(dicionario["Idade"])  
9 # 12
```

# Removendo Valores

- Para remover um valor (e chave associada) em um dicionário podemos utilizar o método **pop**.
- O método **pop** recebe como parâmetro a chave que está associada ao valor que deve ser removido.
- Como resposta o método retorna o valor que foi removido do dicionário.
- Caso a chave informada como parâmetro não esteja no dicionário, um erro será gerado.

# Removendo Valores

- Removendo um valor de um dicionário utilizando o método `pop`:

```
1 dicionario = {  
2     "Nome": "Ash Ketchum",  
3     "Idade": 12,  
4     "Profissão": "Treinador Pokémon",  
5     "Cidade": "Pallet"  
6 }  
7 profissao = dicionario.pop("Profissão")  
8 print(profissao)  
9 # Treinador Pokémon  
10 print(dicionario)  
11 # {'Nome': 'Ash Ketchum', 'Idade': 12, 'Cidade': 'Pallet'}
```



- Outra forma de remover um valor (e chave associada) em um dicionário é utilizando a declaração `del`.
- A declaração `del` pode ser utilizada da seguinte forma:

```
1 del <dicionario>[<chave>]
```

- Caso a chave informada como parâmetro não esteja no dicionário, um erro será gerado.

## Removendo Valores

- Removendo um valor de um dicionário utilizando a declaração `del`:

```
1 dicionario = {  
2     "Nome": "Ash Ketchum",  
3     "Idade": 12,  
4     "Profissão": "Treinador Pokémon",  
5     "Cidade": "Pallet"  
6 }  
7 del dicionario["Profissão"]  
8 print(dicionario)  
9 # {'Nome': 'Ash Ketchum', 'Idade': 12, 'Cidade': 'Pallet'}
```

# Removendo Valores

- Erro ao tentar remover um valor cujo a chave não está no dicionário:

```
1 dicionario = {  
2     "Nome": "Ash Ketchum",  
3     "Idade": 12,  
4     "Profissão": "Treinador Pokémon",  
5     "Cidade": "Pallet"  
6 }  
7 cpf = dicionario.pop("CPF")  
8 # KeyError: 'CPF'  
9 del dicionario["CNH"]  
10 # KeyError: 'CNH'
```

## Removendo Valores

- O método `popitem` remove sempre o último par (chave, valor) de um dicionário.
- Como resposta o método retorna o par (chave, valor) removido, em formato de tupla.
- Caso o dicionário esteja vazio, um erro será gerado.
- Exemplo:

```
1 carro = {"Modelo": "Gol", "Ano": 2019}
2 print(carro)
3 # {'Modelo': 'Gol', 'Ano': 2019}
4 print(carro.popitem())
5 # ('Ano', 2019)
6 print(carro.popitem())
7 # ('Modelo', 'Gol')
8 print(carro)
9 # {}
10 print(carro.popitem())
11 # KeyError: 'popitem(): dictionary is empty'
```

# Métodos Úteis

- O método **update** pode ser utilizado para atualizar um dicionário.
- Esse método recebe como parâmetro um outro dicionário.
- O dicionário original é modificado com base no dicionário informado como parâmetro, de tal forma que os valores das chaves previamente existentes no primeiro são atualizados e novos valores são adicionados para as novas chaves.

```
1 dic_a = {"A": "Avião", "B": "Barco"}
2 dic_b = {"B": "Balão", "C": "Carro"}
3 dic_a.update(dic_b)
4 print(dic_a)
5 # {'A': 'Avião', 'B': 'Balão', 'C': 'Carro'}
```

# Métodos Úteis

- O método `keys` retorna uma estrutura com as chaves do dicionário, que pode ser convertida para uma lista.

```
1 lugar = {"Lat": -22.817087, "Long": -47.069750}
2 print(lugar.keys())
3 # dict_keys(['Lat', 'Long'])
4 print(list(lugar.keys()))
5 # ['Lat', 'Long']
```

- O método `values` retorna uma estrutura com os valores do dicionário, que pode ser convertida para uma lista.

```
1 lugar = {"Lat": -22.817087, "Long": -47.069750}
2 print(lugar.values())
3 # dict_values([-22.817087, -47.06975])
4 print(list(lugar.values()))
5 # [-22.817087, -47.06975]
```

- O método `items` retorna uma estrutura que pode ser convertida para uma lista de tuplas, onde cada tupla é composta pelo par (chave, valor).

```
1 lugar = {"Lat": -22.817087, "Long": -47.069750}
2 print(lugar.items())
3 # dict_items([('Lat', -22.817087), ('Long', -47.06975)])
4 print(list(lugar.items()))
5 # [('Lat', -22.817087), ('Long', -47.06975)]
```

# Iterando Sobre Dicionários

- Podemos iterar sobre uma lista de chaves utilizando o método `keys`.
- Exemplo:

```
1 dic = {  
2     "A": "Abacate",  
3     "B": "Banana",  
4     "C": "Caqui"  
5 }  
6 for letra in dic.keys():  
7     print("Letra:", letra)  
8 # Letra: A  
9 # Letra: B  
10 # Letra: C
```



# Iterando Sobre Dicionários

- Podemos iterar sobre uma lista de valores utilizando o método `values`.
- Exemplo:

```
1 dic = {  
2     "A": "Abacate",  
3     "B": "Banana",  
4     "C": "Caqui"  
5 }  
6 for fruta in dic.values():  
7     print("Fruta:", fruta)  
8 # Fruta: Abacate  
9 # Fruta: Banana  
10 # Fruta: Caqui
```

# Iterando Sobre Dicionários

- Podemos também iterar sobre uma lista de tuplas contendo as chaves e os valores utilizando o método `items`.
- Exemplo:

```
1 dic = {  
2     "A": "Abacate",  
3     "B": "Banana",  
4     "C": "Caqui"  
5 }  
6 for (letra, fruta) in dic.items():  
7     print("Fruta com Letra", letra, ":", fruta)  
8 # Fruta com Letra A: Abacate  
9 # Fruta com Letra B: Banana  
10 # Fruta com Letra C: Caqui
```

# Exercício

## Descrição

Escreva um programa que dada a lista a seguir:

```
1 dados = [  
2   {"dia": 12, "mes": 2, "ano": 2019, "temp": 30.5},  
3   {"dia": 18, "mes": 3, "ano": 2019, "temp": 29.1},  
4   {"dia": 22, "mes": 4, "ano": 2019, "temp": 28.5},  
5   {"dia": 17, "mes": 5, "ano": 2019, "temp": 26.4}  
6 ]
```

... imprime como resposta a seguinte saída:

```
1 # 12/02/2019: Temperatura: 30.5C  
2 # 18/03/2019: Temperatura: 29.1C  
3 # 22/04/2019: Temperatura: 28.5C  
4 # 17/05/2019: Temperatura: 26.4C
```

Seu código deve iterar sobre a lista acessando cada dicionário.

## Resposta

Uma possível resposta para o exercício:

```
1 dados = [  
2     {"dia": 12, "mes": 2, "ano": 2019, "temp": 30.5},  
3     {"dia": 18, "mes": 3, "ano": 2019, "temp": 29.1},  
4     {"dia": 22, "mes": 4, "ano": 2019, "temp": 28.5},  
5     {"dia": 17, "mes": 5, "ano": 2019, "temp": 26.4}  
6 ]  
7  
8 msg = "{0:02d}/{1:02d}/{2}: Temperatura: {3}C"  
9 for dic in dados:  
10     print(msg.format(dic["dia"], dic["mes"],  
11                     dic["ano"], dic["temp"]))
```

- Similar ao que vimos em listas, podemos atribuir um dicionário para diferentes variáveis, mas as variáveis estarão relacionadas ao mesmo dicionário (objeto).
- Exemplo:

```
1 dic_a = {"Nome": "João", "Idade": 18}
2 print(dic_a)
3 # {'Nome': 'João', 'Idade': 18}
4 dic_b = dic_a
5 dic_b["Nome"] = "Maria"
6 print(dic_b)
7 # {'Nome': 'Maria', 'Idade': 18}
8 print(dic_a)
9 # {'Nome': 'Maria', 'Idade': 18}
```

- Similar ao que vimos em listas, se quisermos criar uma cópia independente de um dicionário devemos utilizar o método `copy`.
- Exemplo:

```
1 dic_a = {"Nome": "João", "Idade": 18}
2 print(dic_a)
3 # {'Nome': 'João', 'Idade': 18}
4 dic_b = dic_a.copy()
5 dic_b["Nome"] = "Maria"
6 print(dic_b)
7 # {'Nome': 'Maria', 'Idade': 18}
8 print(dic_a)
9 # {'Nome': 'João', 'Idade': 18}
```

## Criando Dicionários com Listas

- É possível criar um dicionário a partir de duas listas com o auxílio da função `zip`.
- A função `zip` recebe dois parâmetros, o primeiro é lista contendo as chaves desejadas para o dicionário, enquanto o segundo é uma lista contendo os respectivos valores.
- Exemplo:

```
1 pessoas = ["Alice", "Beatriz", "Carlos"]
2 telefones = ["99999-0000", "99999-1111", "99999-2222"]
3 contatos = dict(zip(pessoas, telefones))
4 print(contatos)
5 # {'Alice': '99999-0000', 'Beatriz': '99999-1111',
6 #  'Carlos': '99999-2222'}
```

# Exercícios

---



# Exercício 1

## Descrição

Escreva um programa que recebe como entrada um número inteiro  $n$ . Em seguida, seu programa deve receber as informações de  $n$  Pokémon (nome, tipo e ataque). Para cada Pokémon seu programa deve armazenar as informações utilizando uma estrutura de dicionário. No fim, seu programa deve imprimir o nome do Pokémon do tipo “Fogo” com maior ataque. Você pode assumir que os valores de ataque são inteiros positivos distintos e que pelo menos um Pokémon do tipo “Fogo” será fornecido.

# Exercício 1

## Exemplo

Entrada:

4

Bulbasaur Planta 78

Charmander Fogo 83

Squirtle Água 87

Vulpix Fogo 72

Resposta:

Charmander

# Exercício 1

## Resposta

```
1 # Parte 1 - Lendo os dados
2 n = int(input())
3 pokemon = {}
4 max = 0
5
6 for i in range(n):
7     (nome, tipo, ataque) = input().split()
8     ataque = int(ataque)
9     pokemon[nome] = (tipo, ataque)
```

# Exercício 1

## Resposta

```
1 # Parte 2 - Obtendo e imprimindo a resposta
2
3 for (nome, atributos) in pokemon.items():
4     if atributos[0] == "Fogo":
5         if atributos[1] > max:
6             max = atributos[1]
7             maxPokemon = nome
8
9 print(maxPokemon)
```

## Exercício 2

### Descrição

Escreva um programa que recebe como entrada um número inteiro  $n$ . Em seguida, seu programa deve receber as informações de  $n$  pessoas (nome, CPF e idade). Para cada pessoa seu programa deve armazenar as informações utilizando uma estrutura de dicionário. Infelizmente, algumas entradas do cadastro podem estar repetidas e você deve removê-las (utilize a chave CPF para isso). No fim, seu programa deve imprimir a lista de pessoas, sem repetições. Ao remover as repetições mantenha sempre o primeiro registro lido da pessoa.

## Exercício 2

### Exemplo

Entrada:

6

José 999.999.999-99 19

Maria 888.888.888-88 18

José 999.999.999-99 20

Bob 777.777.777-77 21

Josué 999.999.999-99 20

Bob 777.777.777-77 20

Resposta:

José 999.999.999-99 19

Maria 888.888.888-88 18

Bob 777.777.777-77 21

## Exercício 2

### Resposta

```
1 n = int(input())
2 cadastro = {}
3
4 for i in range(n):
5     (nome, CPF, idade) = input().split()
6     idade = int(idade)
7     if not(CPF in cadastro):
8         cadastro[CPF] = (nome, idade)
9
10 for (CPF, dados) in cadastro.items():
11     print(dados[0], CPF, dados[1])
```