Introdução a

Dicionários

Mauricio Ferste

Dicionários

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

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

```

```
localizacao = {  "Lat": -22.817087,
"Long": -47.069750
}
print(type(localizacao))    # <class 'dict'>


```

Dicionários

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

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

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

Dicionários

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

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

Dicionários

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

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

Dicionário
s

- Podemos acessar um valor do dicionário da seguinte

```
<dicionário>[<chave>]
```

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

```
dicionario = {
  "Nome": "Ash Ketchum",  "Idade": 10,
  "Profissão": "Treinador Pokémon"
}
print(dicionario["Nome"])   # Ash Ketchum
print(dicionario["Idade"])    # 10
print (dicionario["Profissão"])    # Treinador Pokémon
```

Dicionários

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

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

- 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:

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

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",  "Idade": 10,
3   "Profissão": "Treinador Pokémon"
4 }
5 print(len(dicionario))    # 3
6
7
```

Dicionários

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

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

- Exemplo:

```
dicionario =
{
  "Nome": "Ash Ketchum",  "Idade":
  10,
  "Profissão": "Treinador Pokémon"
}
dicionario["Cidade"] = "Pallet"
print(dicionario)
# {'Nome': 'Ash Ketchum', 'Idade': 10,
'Profissão':
#  'Treinador Pokémon', 'Cidade': 'Pallet'}
```

Dicionários

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

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

- Exemplo:

```
dicionario = {
  "Nome": "Ash Ketchum",  "Idade": 10,
  "Profissão": "Treinador Pokémon",  "Cidade": "Pallet"
}
dicionario["Idade"] = 12  print(dicionario["Idade"])   # 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`:

```
dicionario = {
  "Nome": "Ash Ketchum",  "Idade":
  12,
  "Profissão": "Treinador Pokémon",
   "Cidade": "Pallet"
}
profissao = dicionario.pop("Profissão")
print(profissao)
# Treinador Pokémon
print(dicionario)
# {'Nome': 'Ash Ketchum', 'Idade': 12, 'Cidade':
'Pallet'}
```

Removendo Valores

- 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`:

```
dicionario = {
  "Nome": "Ash Ketchum",  "Idade":
  12,
  "Profissão": "Treinador Pokémon",
   "Cidade": "Pallet"
}
del dicionario["Profissão"]
print(dicionario)
# {'Nome': 'Ash Ketchum', 'Idade': 12, 'Cidade':
'Pallet'}
```

Removendo Valores

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

```
dicionario = {
  "Nome": "Ash Ketchum",  "Idade": 12,
  "Profissão": "Treinador Pokémon",  "Cidade": "Pallet"
}
cpf = dicionario.pop("CPF")    # KeyError: 'CPF'
del dicionario["CNH"]    # 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

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

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()))    # ['Lat', 'Long']
5
```

- 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()))    # [-22.817087, -47.06975]
5
```

Métodos Úteis

- 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:

```
dic = {
   "A": "Abacate",
   "B": "Banana",
   "C": "Caqui"
}
for letra in dic.keys():
   print("Letra:", letra)    # Letra: A
   # Letra: B    # Letra: C
```

Iterando Sobre Dicionários

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

```
dic = {
  "A": "Abacate",
  "B": "Banana",
  "C": "Caqui"
}
for fruta in dic.values():
  print("Fruta:", fruta)   # Fruta: Abacate
  # Fruta: Banana   # Fruta: Caqui
```

Iterando Sobre Dicionários

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

```
    "A": "Abacate",
    "B": "Banana",
    "C": "Caqui"
}
for (letra, fruta) in dic.items():
    print("Fruta com Letra", letra, ":", fruta)
# Fruta com Letra A: Abacate
# Fruta com Letra B: Banana
# Fruta com Letra C: Caqui
```

Exercício

Descrição

Escreva uma programa que dada a lista a seguir:

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

... imprime como resposta a seguinte saída:

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

Seu código deve iterar sobre a lista acessando cada dicionário.

Exercício

Resposta

Uma possível resposta para o exercício:

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

msg = "{0:02d}/{1:02d}/{2}: Temperatura: {3}C"
   print(msg.format(dic["dia"], dic["mes"],
                    dic["ano"], dic["temp"]))
```

Copiando Dicionários

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

```
dic_a = {"Nome": "João", "Idade": 18}
print(dic_a)
# {'Nome': 'João', 'Idade': 18}
dic_b = dic_a
dic_b["Nome"] = "Maria"
print(dic_b)
# {'Nome': 'Maria', 'Idade': 18}
print(dic_a)
# {'Nome': 'Maria', 'Idade': 18}
```

Copiando Dicionários

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

```
dic_a = {"Nome": "João", "Idade": 18}
print(dic_a)
# {'Nome': 'João', 'Idade': 18}
dic_b = dic_a.copy()
dic_b["Nome"] = "Maria"
print(dic_b)
# {'Nome': 'Maria', 'Idade': 18}
print(dic_a)
# {'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 uma 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

```
# Parte 1 - Lendo os dados    n = int(input())
pokemon = {}
max = 0

for i in range(n):
   (nome, tipo, ataque) = input().split()   ataque =
   int(ataque)
   pokemon[nome] = (tipo, ataque)

```

Exercício 1

Resposta

```
# Parte 2 - Obtendo e imprimindo a resposta

for (nome, atributos) in pokemon.items():
  if atributos[0] == "Fogo":
    if atributos[1] > max:  max = atributos[1]
      maxPokemon = nome

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

```
n = int(input())
cadastro = {}

for i in range(n):
  (nome, CPF, idade) = input().split()
   idade = int(idade)
  if not(CPF in cadastro):
    cadastro[CPF] = (nome, idade)

for (CPF, dados) in
  print(dados[0], CPF, dados[1])
```