

Notas de aula para OBI

Programação em Python

Encontro 7 - Dicionários

Prof. Louis Augusto

`louis.augusto@ifsc.edu.br`



**INSTITUTO FEDERAL
SANTA CATARINA**

Instituto Federal de Santa Catarina
Campus São José

- 1 Apresentação
 - Dicionários e Vetores
 - Inicialização de dicionário e vetor
 - Principais métodos para dicionários
- 2 Exercícios
 - Exercícios da Beecrowd
 - Contador de palavras
- 3 Arquivos json
 - Arquivos json

- 1 **Apresentação**
 - **Dicionários e Vetores**
 - Inicialização de dicionário e vetor
 - Principais métodos para dicionários
- 2 **Exercícios**
 - Exercícios da Beecrowd
 - Contador de palavras
- 3 **Arquivos json**
 - Arquivos json

Dicionários - Introdução

Dicionários são como vetores, mas a diferença é que dicionários podem usar qualquer tipo imutável de dados como índice e não apenas números inteiros.

```
# -*- coding : utf -8 -*-  
Preco = [None]*3 # Inicializa um vetor chamado Preco,  
                com 3 elementos  
Preco[0] = 6  
Preco[1] = 4  
Preco[2] = 10  
print (Preco[1])  
print (Preco[2])
```

Suponha que os preços 6, 4 e 10 correspondam ao preço do kg de manga, banana e maçã. Para fazer sentido é necessário um outro vetor de strings que faça esta referência, por exemplo:

```
Produto = [None]*3  
Produto[0] = "manga"  
Produto[1] = "banana"  
Produto[2] = "maçã"
```

Agora temos uma conexão do produto e seu preço.

Dicionários - Introdução

Dicionários são como vetores, mas a diferença é que dicionários podem usar qualquer tipo imutável de dados como índice e não apenas números inteiros.

```
# -*- coding : utf -8 -*-  
Preco = [None]*3 # Inicializa um vetor chamado Preco,  
               com 3 elementos  
Preco[0] = 6  
Preco[1] = 4  
Preco[2] = 10  
print (Preco[1])  
print (Preco[2])
```

Suponha que os preços 6, 4 e 10 correspondam ao preço do kg de manga, banana e maçã. Para fazer sentido é necessário um outro vetor de strings que faça esta referência, por exemplo:

```
Produto = [None]*3  
Produto[0] = "manga"  
Produto[1] = "banana"  
Produto[2] = "maçã"
```

Agora temos uma conexão do produto e seu preço.

Dicionários - Introdução

Dicionários são como vetores, mas a diferença é que dicionários podem usar qualquer tipo imutável de dados como índice e não apenas números inteiros.

```
# -*- coding : utf -8 -*-  
Preco = [None]*3 # Inicializa um vetor chamado Preco,  
               com 3 elementos  
Preco[0] = 6  
Preco[1] = 4  
Preco[2] = 10  
print (Preco[1])  
print (Preco[2])
```

Suponha que os preços 6, 4 e 10 correspondam ao preço do kg de manga, banana e maçã. Para fazer sentido é necessário um outro vetor de strings que faça esta referência, por exemplo:

```
Produto = [None]*3  
Produto[0] = "manga"  
Produto[1] = "banana"  
Produto[2] = "maçã"
```

Agora temos uma conexão do produto e seu preço.

Dicionários - Introdução

Dicionários são como vetores, mas a diferença é que dicionários podem usar qualquer tipo imutável de dados como índice e não apenas números inteiros.

```
# -*- coding : utf -8 -*-  
Preco = [None]*3 # Inicializa um vetor chamado Preco,  
               com 3 elementos  
Preco[0] = 6  
Preco[1] = 4  
Preco[2] = 10  
print (Preco[1])  
print (Preco[2])
```

Suponha que os preços 6, 4 e 10 correspondam ao preço do kg de manga, banana e maçã. Para fazer sentido é necessário um outro vetor de strings que faça esta referência, por exemplo:

```
Produto = [None]*3  
Produto[0] = "manga"  
Produto[1] = "banana"  
Produto[2] = "maçã"
```

Agora temos uma conexão do produto e seu preço.

Dicionários - Introdução

Dicionários são como vetores, mas a diferença é que dicionários podem usar qualquer tipo imutável de dados como índice e não apenas números inteiros.

```
# -*- coding : utf -8 -*-  
Preco = [None]*3 # Inicializa um vetor chamado Preco,  
               com 3 elementos  
Preco[0] = 6  
Preco[1] = 4  
Preco[2] = 10  
print (Preco[1])  
print (Preco[2])
```

Suponha que os preços 6, 4 e 10 correspondam ao preço do kg de manga, banana e maçã. Para fazer sentido é necessário um outro vetor de strings que faça esta referência, por exemplo:

```
Produto = [None]*3  
Produto[0] = "manga"  
Produto[1] = "banana"  
Produto[2] = "maçã"
```

Agora temos uma conexão do produto e seu preço.

Dicionários - Introdução

O dicionários são estruturas que incorporam as duas informações anteriores.

O dicionário possui uma chave, que equivale à posição do elemento do vetor, e um valor.

Vetores são inicializados com [], já dicionários com { }, mas a chamada de ambos é feita com [].

```
Preco={} #Dicionário que irá guardar preços de produtos
Preco ["manga"] = 6
Preco ["banana"] = 4
Preco [ "maçã" ] = 10
print ( Preco [ "manga" ])
print ( Preco [ "maçã" ])
```

Dicionários não são estruturas fundamentais, e nem existem em todas as linguagens de programação, mas eles facilitam bastante a vida em algumas situações.

Dicionários - Introdução

O dicionários são estruturas que incorporam as duas informações anteriores. O dicionário possui uma chave, que equivale à posição do elemento do vetor, e um valor.

Vetores são inicializados com [], já dicionários com { }, mas a chamada de ambos é feita com [].

```
Preco={} #Dicionário que irá guardar preços de produtos
Preco ["manga"] = 6
Preco ["banana"] = 4
Preco [ "maçã" ] = 10
print ( Preco [ "manga" ])
print ( Preco [ "maçã" ])
```

Dicionários não são estruturas fundamentais, e nem existem em todas as linguagens de programação, mas eles facilitam bastante a vida em algumas situações.

Dicionários - Introdução

O dicionários são estruturas que incorporam as duas informações anteriores.

O dicionário possui uma chave, que equivale à posição do elemento do vetor, e um valor.

Vetores são inicializados com [], já dicionários com { }, mas a chamada de ambos é feita com [].

```
Preco={} #Dicionário que irá guardar preços de produtos
Preco ["manga"] = 6
Preco ["banana"] = 4
Preco [ "maçã" ] = 10
print ( Preco [ "manga" ])
print ( Preco [ "maçã" ])
```

Dicionários não são estruturas fundamentais, e nem existem em todas as linguagens de programação, mas eles facilitam bastante a vida em algumas situações.

Dicionários - Introdução

O dicionários são estruturas que incorporam as duas informações anteriores. O dicionário possui uma chave, que equivale à posição do elemento do vetor, e um valor.

Vetores são inicializados com [], já dicionários com { }, mas a chamada de ambos é feita com [].

```
Preco={} #Dicionário que irá guardar preços de produtos
Preco ["manga"] = 6
Preco ["banana"] = 4
Preco [ "maçã" ] = 10
print ( Preco [ "manga" ])
print ( Preco [ "maçã" ])
```

Dicionários não são estruturas fundamentais, e nem existem em todas as linguagens de programação, mas eles facilitam bastante a vida em algumas situações.

Dicionários - Introdução

O dicionários são estruturas que incorporam as duas informações anteriores.

O dicionário possui uma chave, que equivale à posição do elemento do vetor, e um valor.

Vetores são inicializados com [], já dicionários com { }, mas a chamada de ambos é feita com [].

```
Preco={} #Dicionário que irá guardar preços de produtos
Preco ["manga"] = 6
Preco ["banana"] = 4
Preco [ "maçã" ] = 10
print ( Preco [ "manga" ])
print ( Preco [ "maçã" ])
```

Dicionários não são estruturas fundamentais, e nem existem em todas as linguagens de programação, mas eles facilitam bastante a vida em algumas situações.

- 1 **Apresentação**
 - Dicionários e Vetores
 - Inicialização de dicionário e vetor
 - Principais métodos para dicionários
- 2 **Exercícios**
 - Exercícios da Beecrowd
 - Contador de palavras
- 3 **Arquivos json**
 - Arquivos json

Dicionários - Inicialização

Você deve ter percebido que um elemento de um dicionário possui uma chave (identifica o elemento e precisa ser imutável) e um valor.

Observe a diferença no código anterior referente aos dicionários e vetores:

- Definimos um vetor de 3 itens: `v = [None]*3`
Se quisermos inserir um novo dado, usamos o método `append(a)` (insere `a` no final da lista) ou `insert(a,b)` (insere `b` na posição `a` da lista):

```
v.append(3)
v.insert(2,-5)
for i in v:
    print(v[i])
```
- Já para um dicionário basta indicar a chave e o valor a inserir:

```
Preco [ "milho" ] = 7
for i in Preco.items():
    print("Preço de {0}: {1}".format(i[0], i[1]))
```

O preço a ser pago é que nos vetores temos garantia de ordem, enquanto no dicionário não, pode vir em qualquer ordem.

Dicionários - Inicialização

Você deve ter percebido que um elemento de um dicionário possui uma chave (identifica o elemento e precisa ser imutável) e um valor.

Observe a diferença no código anterior referente aos dicionários e vetores:

- Definimos um vetor de 3 itens: `v = [None]*3`

Se quisermos inserir um novo dado, usamos o método `append(a)` (insere `a` no final da lista) ou `insert(a,b)` (insere `b` na posição `a` da lista:

```
v.append(3)
v.insert(2,-5)
for i in v:
```

```
    print(v[i])
```

- Já para um dicionário basta indicar a chave e o valor a inserir:

```
Preco [ "milho" ] = 7
for i in Preco.items():
    print("Preço de {0}: {1}".format(i[0], i[1]))
```

O preço a ser pago é que nos vetores temos garantia de ordem, enquanto no dicionário não, pode vir em qualquer ordem.

Dicionários - Inicialização

Você deve ter percebido que um elemento de um dicionário possui uma chave (identifica o elemento e precisa ser imutável) e um valor.

Observe a diferença no código anterior referente aos dicionários e vetores:

- Definimos um vetor de 3 itens: `v = [None]*3`

Se quisermos inserir um novo dado, usamos o método `append(a)` (insere `a` no final da lista) ou `insert(a, b)` (insere `b` na posição `a` da lista):

```
v.append(3)
v.insert(2, -5)
for i in v:
    print(v[i])
```

- Já para um dicionário basta indicar a chave e o valor a inserir:

```
Preco [ "milho" ] = 3
for i in Preco.items():
    print("Preço de {0}: {1}".format(i[0], i[1]))
```

O preço a ser pago é que nos vetores temos garantia de ordem, enquanto no dicionário não, pode vir em qualquer ordem.

Dicionários - Inicialização

Você deve ter percebido que um elemento de um dicionário possui uma chave (identifica o elemento e precisa ser imutável) e um valor.

Observe a diferença no código anterior referente aos dicionários e vetores:

- Definimos um vetor de 3 itens: `v = [None]*3`

Se quisermos inserir um novo dado, usamos o método `append(a)` (insere `a` no final da lista) ou `insert(a, b)` (insere `b` na posição `a` da lista):

```
v.append(3)
v.insert(2, -5)
for i in v:
    print(v[i])
```

- Já para um dicionário basta indicar a chave e o valor a inserir:

```
Preco [ "milho" ] = 3
for i in Preco.items():
    print("Preço de {0}: {1}".format(i[0], i[1]))
```

O preço a ser pago é que nos vetores temos garantia de ordem, enquanto no dicionário não, pode vir em qualquer ordem.

Dicionários - Inicialização

Você deve ter percebido que um elemento de um dicionário possui uma chave (identifica o elemento e precisa ser imutável) e um valor.

Observe a diferença no código anterior referente aos dicionários e vetores:

- Definimos um vetor de 3 itens: `v = [None]*3`

Se quisermos inserir um novo dado, usamos o método `append(a)` (insere `a` no final da lista) ou `insert(a, b)` (insere `b` na posição `a` da lista:

```
v.append(3)
v.insert(2, -5)
for i in v:
    print(v[i])
```

- Já para um dicionário basta indicar a chave e o valor a inserir:

```
Preco [ "milho" ] = 7
for i in Preco.items():
    print("Preço de {0}: {1}".format(i[0], i[1]))
```

O preço a ser pago é que nos vetores temos garantia de ordem, enquanto no dicionário não, pode vir em qualquer ordem.

Dicionários - Inicialização

Você deve ter percebido que um elemento de um dicionário possui uma chave (identifica o elemento e precisa ser imutável) e um valor.

Observe a diferença no código anterior referente aos dicionários e vetores:

- Definimos um vetor de 3 itens: `v = [None]*3`

Se quisermos inserir um novo dado, usamos o método `append(a)` (insere `a` no final da lista) ou `insert(a, b)` (insere `b` na posição `a` da lista:

```
v.append(3)
v.insert(2, -5)
for i in v:
    print(v[i])
```

- Já para um dicionário basta indicar a chave e o valor a inserir:

```
Preco [ "milho" ] = 7
for i in Preco.items():
    print("Preço de {0}: {1}".format(i[0], i[1]))
```

O preço a ser pago é que nos vetores temos garantia de ordem, enquanto no dicionário não, pode vir em qualquer ordem.

Dicionários - Inicialização

Você deve ter percebido que um elemento de um dicionário possui uma chave (identifica o elemento e precisa ser imutável) e um valor.

Observe a diferença no código anterior referente aos dicionários e vetores:

- Definimos um vetor de 3 itens: `v = [None]*3`
Se quisermos inserir um novo dado, usamos o método `append(a)` (insere `a` no final da lista) ou `insert(a,b)` (insere `b` na posição `a` da lista):

```
v.append(3)
v.insert(2,-5)
for i in v:
    print(v[i])
```
- Já para um dicionário basta indicar a chave e o valor a inserir:

```
Preco [ "milho" ] = 7
for i in Preco.items():
    print("Preço de {0}: {1}".format(i[0], i[1]))
```

O preço a ser pago é que nos vetores temos garantia de ordem, enquanto no dicionário não, pode vir em qualquer ordem.

Dicionários - Inicialização

Você deve ter percebido que um elemento de um dicionário possui uma chave (identifica o elemento e precisa ser imutável) e um valor.

Observe a diferença no código anterior referente aos dicionários e vetores:

- Definimos um vetor de 3 itens: `v = [None]*3`

Se quisermos inserir um novo dado, usamos o método `append(a)` (insere `a` no final da lista) ou `insert(a, b)` (insere `b` na posição `a` da lista:

```
v.append(3)
v.insert(2, -5)
for i in v:
    print(v[i])
```

- Já para um dicionário basta indicar a chave e o valor a inserir:

```
Preco [ "milho" ] = 7
for i in Preco.items():
    print("Preço de {0}: {1}".format(i[0], i[1]))
```

O preço a ser pago é que nos vetores temos garantia de ordem, enquanto no dicionário não, pode vir em qualquer ordem.

Dicionários - Inicialização

Da mesma forma que podemos inicializar um vetor já com vários termos, igualmente isto pode ser feito com um dicionário.

```
#Inicialização de vetor:  
vetor = [2,5,-90,"carro"]
```

```
#Inicialização do dicionário:  
dic = {"carro":"Toyota", 1:2, -90:"Marte"}
```

Identifique no dicionário acima quais são as chaves e os valores, e lembre-se de que Python é uma linguagem não tipada.

Métodos de impressão de vetor e dicionário.

```
print("Vetor de entrada")  
for i in vetor:  
    print(i)  
  
print("Dicionario de entrada")  
for chave,valor in dic.items():  
    print(chave,valor)
```

Dicionários - Inicialização

Da mesma forma que podemos inicializar um vetor já com vários termos, igualmente isto pode ser feito com um dicionário.

```
#Inicialização de vetor:
```

```
vetor = [2,5,-90,"carro"]
```

```
#Inicialização do dicionário:
```

```
dic = {"carro":"Toyota", 1:2, -90:"Marte"}
```

Identifique no dicionário acima quais são as chaves e os valores, e lembre-se de que Python é uma linguagem não tipada.

Métodos de impressão de vetor e dicionário.

```
print("Vetor de entrada") print("Dicionario de entrada")
for i in vetor:           for chave,valor in dic.items():
    print(i)              print(chave,valor)
```


Dicionários - Inicialização

Da mesma forma que podemos inicializar um vetor já com vários termos, igualmente isto pode ser feito com um dicionário.

```
#Inicialização de vetor:
```

```
vetor = [2,5,-90,"carro"]
```

```
#Inicialização do dicionário:
```

```
dic = {"carro":"Toyota", 1:2, -90:"Marte"}
```

Identifique no dicionário acima quais são as chaves e os valores, e lembre-se de que Python é uma linguagem não tipada.

Métodos de impressão de vetor e dicionário.

```
print("Vetor de entrada") print("Dicionario de entrada")
for i in vetor:           for chave,valor in dic.items():
    print(i)               print(chave,valor)
```

Dicionários - Inicialização

Da mesma forma que podemos inicializar um vetor já com vários termos, igualmente isto pode ser feito com um dicionário.

```
#Inicialização de vetor:
```

```
vetor = [2,5,-90,"carro"]
```

```
#Inicialização do dicionário:
```

```
dic = {"carro":"Toyota", 1:2, -90:"Marte"}
```

Identifique no dicionário acima quais são as chaves e os valores, e lembre-se de que Python é uma linguagem não tipada.

Métodos de impressão de vetor e dicionário.

```
print("Vetor de entrada") print("Dicionario de entrada")
for i in vetor:             for chave,valor in dic.items():
    print(i)                 print(chave,valor)
```

Dicionários - Inicialização

Da mesma forma que podemos inicializar um vetor já com vários termos, igualmente isto pode ser feito com um dicionário.

```
#Inicialização de vetor:  
vetor = [2,5,-90,"carro"]
```

```
#Inicialização do dicionário:  
dic = {"carro":"Toyota", 1:2, -90:"Marte"}
```

Identifique no dicionário acima quais são as chaves e os valores, e lembre-se de que Python é uma linguagem não tipada.

Métodos de impressão de vetor e dicionário.

```
print("Vetor de entrada")  
for i in vetor:  
    print(i)  
  
print("Dicionario de entrada")  
for chave,valor in dic.items():  
    print(chave,valor)
```

Dicionários - Inicialização

Da mesma forma que podemos inicializar um vetor já com vários termos, igualmente isto pode ser feito com um dicionário.

```
#Inicialização de vetor:  
vetor = [2,5,-90,"carro"]
```

```
#Inicialização do dicionário:  
dic = {"carro":"Toyota", 1:2, -90:"Marte"}
```

Identifique no dicionário acima quais são as chaves e os valores, e lembre-se de que Python é uma linguagem não tipada.

Métodos de impressão de vetor e dicionário.

```
print("Vetor de entrada")  
for i in vetor:  
    print(i)  
  
print("Dicionario de entrada")  
for chave,valor in dic.items():  
    print(chave,valor)
```

Trabalhando com dicionários

Assim como vetores, dicionários aceitam a função embutida `len()` (que retorna a quantidade de termos do dicionário) e a função `del()`, que permite remover um elemento do dicionário. Compare o vetor e o dicionário:

```
del(Preco["maçã"])
del(v[2])
print("Vetor e dicionario apos remocoes: ")
for i in range(len(v)):
    print("v[{0}] = {1}".format(i, v[i]))
for chave,valor in Preco.items():
    print("Preço de {0}: {1}".format(chave, valor))
```

Para imprimir os valores de um dicionário `dic` (nome do dicionário) temos 3 métodos principais:

- `dic.items()` Retorna os pares (chave, valor).
- `dic.keys()` Retorna as chaves que estão no dicionário.
- `dic.values()` Retorna os valores que estão no dicionário.

Trabalhando com dicionários

Assim como vetores, dicionários aceitam a função embutida `len()` (que retorna a quantidade de termos do dicionário) e a função `del()`, que permite remover um elemento do dicionário. Compare o vetor e o dicionário:

```
del(Preco["maçã"])
del(v[2])
print("Vetor e dicionario apos remocoes: ")
for i in range(len(v)):
    print("v[{0}] = {1}".format(i, v[i]))
for chave,valor in Preco.items():
    print("Preço de {0}: {1}".format(chave, valor))
```

Para imprimir os valores de um dicionário `dic` (nome do dicionário) temos 3 métodos principais:

- `dic.items()` Retorna os pares (chave, valor).
- `dic.keys()` Retorna as chaves que estão no dicionário.
- `dic.values()` Retorna os valores que estão no dicionário.

Trabalhando com dicionários

Assim como vetores, dicionários aceitam a função embutida `len()` (que retorna a quantidade de termos do dicionário) e a função `del()`, que permite remover um elemento do dicionário. Compare o vetor e o dicionário:

```
del(Preco["maçã"])
del(v[2])

print("Vetor e dicionario apos remocoes: ")
for i in range(len(v)):
    print("v[{0}] = {1}".format(i, v[i]))
for chave,valor in Preco.items():
    print("Preço de {0}: {1}".format(chave, valor))
```

Para imprimir os valores de um dicionário `dic` (nome do dicionário) temos 3 métodos principais:

- `dic.items()` Retorna os pares (chave, valor).
- `dic.keys()` Retorna as chaves que estão no dicionário.
- `dic.values()` Retorna os valores que estão no dicionário.

Trabalhando com dicionários

Assim como vetores, dicionários aceitam a função embutida `len()` (que retorna a quantidade de termos do dicionário) e a função `del()`, que permite remover um elemento do dicionário. Compare o vetor e o dicionário:

```
del(Preco["maçã"])
del(v[2])
print("Vetor e dicionario apos remocoes: ")
for i in range(len(v)):
    print("v[{0}] = {1}".format(i, v[i]))
for chave,valor in Preco.items():
    print("Preço de {0}: {1}".format(chave, valor))
```

Para imprimir os valores de um dicionário `dic` (nome do dicionário) temos 3 métodos principais:

- `dic.items()` Retorna os pares (chave, valor).
- `dic.keys()` Retorna as chaves que estão no dicionário.
- `dic.values()` Retorna os valores que estão no dicionário.

Trabalhando com dicionários

Assim como vetores, dicionários aceitam a função embutida `len()` (que retorna a quantidade de termos do dicionário) e a função `del()`, que permite remover um elemento do dicionário. Compare o vetor e o dicionário:

```
del(Preco["maçã"])
del(v[2])
print("Vetor e dicionario apos remocoes: ")
for i in range(len(v)):
    print("v[{0}] = {1}".format(i, v[i]))
for chave,valor in Preco.items():
    print("Preço de {0}: {1}".format(chave, valor))
```

Para imprimir os valores de um dicionário `dic` (nome do dicionário) temos 3 métodos principais:

`dic.items()` Retorna os pares (chave, valor).

`dic.keys()` Retorna as chaves que estão no dicionário.

`dic.values()` Retorna os valores que estão no dicionário.

Trabalhando com dicionários

Assim como vetores, dicionários aceitam a função embutida `len()` (que retorna a quantidade de termos do dicionário) e a função `del()`, que permite remover um elemento do dicionário. Compare o vetor e o dicionário:

```
del(Preco["maçã"])
del(v[2])
print("Vetor e dicionario apos remocoes: ")
for i in range(len(v)):
    print("v[{0}] = {1}".format(i, v[i]))
for chave, valor in Preco.items():
    print("Preço de {0}: {1}".format(chave, valor))
```

Para imprimir os valores de um dicionário `dic` (nome do dicionário) temos 3 métodos principais:

`dic.items()` Retorna os pares (chave, valor).

`dic.keys()` Retorna as chaves que estão no dicionário.

`dic.values()` Retorna os valores que estão no dicionário.

Trabalhando com dicionários

Assim como vetores, dicionários aceitam a função embutida `len()` (que retorna a quantidade de termos do dicionário) e a função `del()`, que permite remover um elemento do dicionário. Compare o vetor e o dicionário:

```
del(Preco["maçã"])
del(v[2])
print("Vetor e dicionario apos remocoes: ")
for i in range(len(v)):
    print("v[{0}] = {1}".format(i, v[i]))
for chave,valor in Preco.items():
    print("Preço de {0}: {1}".format(chave, valor))
```

Para imprimir os valores de um dicionário `dic` (nome do dicionário) temos 3 métodos principais:

`dic.items()` Retorna os pares (chave, valor).

`dic.keys()` Retorna as chaves que estão no dicionário.

`dic.values()` Retorna os valores que estão no dicionário.

- 1 Apresentação
 - Dicionários e Vetores
 - Inicialização de dicionário e vetor
 - Principais métodos para dicionários
- 2 Exercícios
 - Exercícios da Beecrowd
 - Contador de palavras
- 3 Arquivos json
 - Arquivos json

Métodos para dicionários

Temos uma relação dos principais métodos para dicionários, considere um dicionário chamado `d` (inicializa `d = { }`) :

`d.clear()` Remove todos itens do dicionário `d`.

`d.copy()` Retorna uma cópia rasa do dicionário `d`.

`d.get(k)` Retorna uma chave do valor associado de `k`, ou `None`, se `k` não estiver em `d`.

`d.items()` Retorna uma visualização de todos os pares (chave,valor) em `d`.

`d.keys()` Retorna uma visualização de todas as chaves em `d`.

`d.values()` Retorna uma visualização de todos os valores em `d`.

`d.pop(k)` Retorna uma chave do valor associado de `k` e remove o item no qual a chave é `k`, ou lança uma exceção *KeyError*, caso `k` não esteja em `d`.

Métodos para dicionários

Temos uma relação dos principais métodos para dicionários, considere um dicionário chamado `d` (inicializa `d = { }`) :

`d.clear()` Remove todos itens do dicionário `d`.

`d.copy()` Retorna uma cópia rasa do dicionário `d`.

`d.get(k)` Retorna uma chave do valor associado de `k`, ou `None`, se `k` não estiver em `d`.

`d.items()` Retorna uma visualização de todos os pares (chave,valor) em `d`.

`d.keys()` Retorna uma visualização de todas as chaves em `d`.

`d.values()` Retorna uma visualização de todos os valores em `d`.

`d.pop(k)` Retorna uma chave do valor associado de `k` e remove o item no qual a chave é `k`, ou lança uma exceção `KeyError`, caso `k` não esteja em `d`.

Métodos para dicionários

Temos uma relação dos principais métodos para dicionários, considere um dicionário chamado `d` (inicializa `d = { }`) :

`d.clear()` Remove todos itens do dicionário `d`.

`d.copy()` Retorna uma cópia rasa do dicionário `d`.

`d.get(k)` Retorna uma chave do valor associado de `k`, ou `None`, se `k` não estiver em `d`.

`d.items()` Retorna uma visualização de todos os pares (chave,valor) em `d`.

`d.keys()` Retorna uma visualização de todas as chaves em `d`.

`d.values()` Retorna uma visualização de todos os valores em `d`.

`d.pop(k)` Retorna uma chave do valor associado de `k` e remove o item no qual a chave é `k`, ou lança uma exceção *KeyError*, caso `k` não esteja em `d`.

Métodos para dicionários

Temos uma relação dos principais métodos para dicionários, considere um dicionário chamado `d` (inicializa `d = { }`) :

`d.clear()` Remove todos itens do dicionário `d`.

`d.copy()` Retorna uma cópia rasa do dicionário `d`.

`d.get(k)` Retorna uma chave do valor associado de `k`, ou `None`, se `k` não estiver em `d`.

`d.items()` Retorna uma visualização de todos os pares (chave,valor) em `d`.

`d.keys()` Retorna uma visualização de todas as chaves em `d`.

`d.values()` Retorna uma visualização de todos os valores em `d`.

`d.pop(k)` Retorna uma chave do valor associado de `k` e remove o item no qual a chave é `k`, ou lança uma exceção *KeyError*, caso `k` não esteja em `d`.

Métodos para dicionários

Temos uma relação dos principais métodos para dicionários, considere um dicionário chamado `d` (inicializa `d = { }`) :

`d.clear()` Remove todos itens do dicionário `d`.

`d.copy()` Retorna uma cópia rasa do dicionário `d`.

`d.get(k)` Retorna uma chave do valor associado de `k`, ou `None`, se `k` não estiver em `d`.

`d.items()` Retorna uma visualização de todos os pares (chave,valor) em `d`.

`d.keys()` Retorna uma visualização de todas as chaves em `d`.

`d.values()` Retorna uma visualização de todos os valores em `d`.

`d.pop(k)` Retorna uma chave do valor associado de `k` e remove o item no qual a chave é `k`, ou lança uma exceção *KeyError*, caso `k` não esteja em `d`.

Métodos para dicionários

Temos uma relação dos principais métodos para dicionários, considere um dicionário chamado `d` (inicializa `d = { }`) :

`d.clear()` Remove todos itens do dicionário `d`.

`d.copy()` Retorna uma cópia rasa do dicionário `d`.

`d.get(k)` Retorna uma chave do valor associado de `k`, ou `None`, se `k` não estiver em `d`.

`d.items()` Retorna uma visualização de todos os pares (chave,valor) em `d`.

`d.keys()` Retorna uma visualização de todas as chaves em `d`.

`d.values()` Retorna uma visualização de todos os valores em `d`.

`d.pop(k)` Retorna uma chave do valor associado de `k` e remove o item no qual a chave é `k`, ou lança uma exceção `KeyError`, caso `k` não esteja em `d`.

Métodos para dicionários

Temos uma relação dos principais métodos para dicionários, considere um dicionário chamado `d` (inicializa `d = { }`) :

- `d.clear()` Remove todos itens do dicionário `d`.
- `d.copy()` Retorna uma cópia rasa do dicionário `d`.
- `d.get(k)` Retorna uma chave do valor associado de `k`, ou `None`, se `k` não estiver em `d`.
- `d.items()` Retorna uma visualização de todos os pares (chave,valor) em `d`.
- `d.keys()` Retorna uma visualização de todas as chaves em `d`.
- `d.values()` Retorna uma visualização de todos os valores em `d`.
- `d.pop(k)` Retorna uma chave do valor associado de `k` e remove o item no qual a chave é `k`, ou lança uma exceção *KeyError*, caso `k` não esteja em `d`.

- 1 Apresentação
 - Dicionários e Vetores
 - Inicialização de dicionário e vetor
 - Principais métodos para dicionários
- 2 Exercícios
 - Exercícios da Beecrowd
 - Contador de palavras
- 3 Arquivos json
 - Arquivos json

Exercício 1052

beecrowd | 1052



Mês

Adaptado por Neilor Tonin, URI  Brasil

Timelimit: 1

Leia um valor inteiro entre 1 e 12, inclusive. Correspondente a este valor, deve ser apresentado como resposta o mês do ano por extenso, em inglês, com a primeira letra maiúscula.

Entrada

A entrada contém um único valor inteiro.

Saída

Imprima por extenso o nome do mês correspondente ao número existente na entrada, com a primeira letra em maiúscula.

Exemplo de Entrada

Exemplo de Saída

4

April



beecrowd | 2850



Papagaio Poliglota

Por Felipe C. Ochial, URI  Brazil

Timelimit: 1

Humberto tem um papagaio muito esperto. Quando está com as duas pernas no chão, o papagaio fala em português. Quando levanta a perna esquerda, fala em inglês. Por fim, quando levanta a direita fala em francês. Nico, amigo de Humberto, ficou fascinado com o animal. Em sua emoção perguntou: "E quando ele levanta as duas?". Antes que Humberto pudesse responder, o papagaio gritou: "Ai eu caio, idiota!".

Entrada

A entrada consiste de diversos casos de teste. Cada caso de teste consiste uma string informando qual a situação de levantamento de pernas do papagaio.

Saída

Para cada condição de levantamento de pernas do papagaio, imprima a linguagem que ele utilizará. Caso ele levante as duas pernas, imprima "caiu". Quebre uma linha a cada caso de teste.

Exemplo de Entrada	Exemplo de Saída
esquerda	ingles
direita	frances
nenhuma	portugues
as duas	caiu



Exercício 1281

beecrowd | 1281

Ida à Feira

Por Neilor Tonin, URI  Brasil

Time limit: 1

Dona Parcinova costuma ir regularmente à feira para comprar frutas e legumes. Ela pediu então à sua filha, Mangojata, que a ajudasse com as contas e que fizesse um programa que calculasse o valor que precisa levar para poder comprar tudo que está em sua lista de compras, considerando a quantidade de cada tipo de fruta ou legume e os preços destes itens.



Entrada

A primeira linha de entrada contém um inteiro **N** que indica a quantidade de idas à feira de dona Parcinova (que nada mais é do que o número de casos de teste que vem a seguir). Cada caso de teste inicia com um inteiro **M** que indica a quantidade de produtos que estão disponíveis para venda na feira. Seguem os **M** produtos com seus preços respectivos por unidade ou Kg. A próxima linha de entrada contém um inteiro **P** ($1 \leq P \leq M$) que indica a quantidade de diferentes produtos que dona Parcinova deseja comprar. Seguem **P** linhas contendo cada uma delas um texto (com até 50 caracteres) e um valor inteiro, que indicam respectivamente o nome de cada produto e a quantidade deste produto.

Saída

Para cada caso de teste, imprima o valor que será gasto por dona Parcinova no seguinte formato: R\$ seguido de um espaço e seguido do valor, com 2 casas decimais, conforme o exemplo abaixo.

Exemplo de Entrada

```
2
4
mamao 2.19
cebola 3.10
tomate 2.80
uva 2.75
3
mamao 2
tomate 1
uva 3
5
morango 6.70
repolho 1.12
brocolis 1.71
tomate 2.80
cebola 2.81
4
brocolis 2
tomate 1
cebola 1
morango 1
```

Exemplo de Saída

```
R$ 19.37
R$ 15.75
```

- 1 Apresentação
 - Dicionários e Vetores
 - Inicialização de dicionário e vetor
 - Principais métodos para dicionários
- 2 Exercícios
 - Exercícios da Beecrowd
 - Contador de palavras
- 3 Arquivos json
 - Arquivos json

Contador de palavras

Dicionários podem ser usados para manter a conta de itens únicos.

Um bom exemplo é fazer um programa que abra um arquivo de texto e conte a ocorrência de cada palavra.

Como tarefa, abra um arquivo de texto, leia todas as palavras, exceto números e espaços e as que tiverem somente um caracter.

A função built-in do python `strip()` remove a ocorrência de um determinado caracter da string, na verdade de um grupo de caracteres. Faça o script abaixo e observe:

```
texto = "          banana          "
print("De todas as frutas, ", texto, " é minha favorita." )
x = texto.strip() #Remove os espaços
print("De todas as frutas, ", x, " é minha favorita." )

texto2 = ",,,,,rrttgg.. ...goiaba.. ..rrr"
print(texto2)
x2 = texto2.strip(",. grt") #Remove espaço e os 5 caracteres.
print("Eu não gosto de ",x2)
```

Agora é só se divertir!!!!

Contador de palavras

Dicionários podem ser usados para manter a conta de itens únicos.

Um bom exemplo é fazer um programa que abra um arquivo de texto e conte a ocorrência de cada palavra.

Como tarefa, abra um arquivo de texto, leia todas as palavras, exceto números e espaços e as que tiverem somente um caracter.

A função built-in do python `strip()` remove a ocorrência de um determinado caracter da string, na verdade de um grupo de caracteres. Faça o script abaixo e observe:

```
texto = "          banana          "
print("De todas as frutas, ", texto, " é minha favorita." )
x = texto.strip() #Remove os espaços
print("De todas as frutas, ", x, " é minha favorita." )

texto2 = ",,,,,rrttgg.. ...goiaba.. ..rrr"
print(texto2)
x2 = texto2.strip(",. grt") #Remove espaço e os 5 caracteres.
print("Eu não gosto de ",x2)
```

Agora é só se divertir!!!!

Contador de palavras

Dicionários podem ser usados para manter a conta de itens únicos.

Um bom exemplo é fazer um programa que abra um arquivo de texto e conte a ocorrência de cada palavra.

Como tarefa, abra um arquivo de texto, leia todas as palavras, exceto números e espaços e as que tiverem somente um caracter.

A função built-in do python `strip()` remove a ocorrência de um determinado caracter da string, na verdade de um grupo de caracteres. Faça o script abaixo e observe:

```
texto = "          banana          "
print("De todas as frutas, ", texto, " é minha favorita." )
x = texto.strip() #Remove os espaços
print("De todas as frutas, ", x, " é minha favorita." )

texto2 = ",,,,,rrttgg.. ...goiaba.. ..rrr"
print(texto2)
x2 = texto2.strip(",. grt") #Remove espaço e os 5 caracteres.
print("Eu não gosto de ",x2)
```

Agora é só se divertir!!!!

Contador de palavras

Dicionários podem ser usados para manter a conta de itens únicos.

Um bom exemplo é fazer um programa que abra um arquivo de texto e conte a ocorrência de cada palavra.

Como tarefa, abra um arquivo de texto, leia todas as palavras, exceto números e espaços e as que tiverem somente um caracter.

A função built-in do python `strip()` remove a ocorrência de um determinado caracter da string, na verdade de um grupo de caracteres. Faça o script abaixo e observe:

```
texto = "          banana          "
print("De todas as frutas, ", texto, " é minha favorita." )
x = texto.strip() #Remove os espaços
print("De todas as frutas, ", x, " é minha favorita." )

texto2 = ",,,,,rrttgg.. ...goiaba.. ..rrr"
print(texto2)
x2 = texto2.strip(",. grt") #Remove espaço e os 5 caracteres.
print("Eu não gosto de ",x2)
```

Agora é só se divertir!!!!

Contador de palavras

Dicionários podem ser usados para manter a conta de itens únicos.

Um bom exemplo é fazer um programa que abra um arquivo de texto e conte a ocorrência de cada palavra.

Como tarefa, abra um arquivo de texto, leia todas as palavras, exceto números e espaços e as que tiverem somente um caracter.

A função built-in do python `strip()` remove a ocorrência de um determinado caracter da string, na verdade de um grupo de caracteres. Faça o script abaixo e observe:

```
texto = "          banana          "
print("De todas as frutas, ", texto, " é minha favorita." )
x = texto.strip() #Remove os espaços
print("De todas as frutas, ", x, " é minha favorita." )

texto2 = ",,,,,rrttgg.. ...goiaba.. ..rrr"
print(texto2)
x2 = texto2.strip(",. grt") #Remove espaço e os 5 caracteres.
print("Eu não gosto de ",x2)
```

Agora é só se divertir!!!!

- 1 Apresentação
 - Dicionários e Vetores
 - Inicialização de dicionário e vetor
 - Principais métodos para dicionários
- 2 Exercícios
 - Exercícios da Beecrowd
 - Contador de palavras
- 3 Arquivos json
 - Arquivos json

Arquivos json

Arquivos json¹ são bastante comuns para guardar dados, e basicamente são vetores de dicionários.

Abra o site <https://www.mercadobitcoin.com.br/api> e verifique como o site libera informações para o usuário.

Baixe os valores de uma moeda, pode ser bitcoin, entre dois momentos da era unix próximos da atual, e verifique o arquivo.

Importe o arquivo usando a biblioteca requests e faça alguns cálculos que ache importante.

Salve o arquivo usando a biblioteca json.

¹JSON é um acrônimo para javascript object notation.

Arquivos json

Arquivos json¹ são bastante comuns para guardar dados, e basicamente são vetores de dicionários.

Abra o site `https://www.mercadobitcoin.com.br/api` e verifique como o site libera informações para o usuário.

Baixe os valores de uma moeda, pode ser bitcoin, entre dois momentos da era unix próximos da atual, e verifique o arquivo.

Importe o arquivo usando a biblioteca requests e faça alguns cálculos que ache importante.

Salve o arquivo usando a biblioteca json.

¹JSON é um acrônimo para javascript object notation.

Arquivos json

Arquivos json¹ são bastante comuns para guardar dados, e basicamente são vetores de dicionários.

Abra o site `https://www.mercadobitcoin.com.br/api` e verifique como o site libera informações para o usuário.

Baixe os valores de uma moeda, pode ser bitcoin, entre dois momentos da era unix próximos da atual, e verifique o arquivo.

Importe o arquivo usando a biblioteca requests e faça alguns cálculos que ache importante.

Salve o arquivo usando a biblioteca json.

¹JSON é um acrônimo para javascript object notation.

Arquivos json

Arquivos json¹ são bastante comuns para guardar dados, e basicamente são vetores de dicionários.

Abra o site `https://www.mercadobitcoin.com.br/api` e verifique como o site libera informações para o usuário.

Baixe os valores de uma moeda, pode ser bitcoin, entre dois momentos da era unix próximos da atual, e verifique o arquivo.

Importe o arquivo usando a biblioteca requests e faça alguns cálculos que ache importante.

Salve o arquivo usando a biblioteca json.

¹JSON é um acrônimo para javascript object notation.

Arquivos json¹ são bastante comuns para guardar dados, e basicamente são vetores de dicionários.

Abra o site `https://www.mercadobitcoin.com.br/api` e verifique como o site libera informações para o usuário.

Baixe os valores de uma moeda, pode ser bitcoin, entre dois momentos da era unix próximos da atual, e verifique o arquivo.

Importe o arquivo usando a biblioteca requests e faça alguns cálculos que ache importante.

Salve o arquivo usando a biblioteca json.

¹JSON é um acrônimo para javascript object notation.