Curso livre de programação
Programação em Python

Encontro 7 - Dicionários

Prof. Louis Augusto
louis.augusto@ifsc.edu.br

Instituto Federal de Santa Catarina
Campus São José

Índice

Sumário

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 no vetor ao índice do elemento**, 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 no vetor ao índice do elemento**, 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 no vetor ao índice do elemento**, 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 no vetor ao índice do elemento**, 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 no vetor ao índice do elemento**, 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.

Sumário

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

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"}
pessoa = dict(nome="Carol", idade = 18, altura = 1.8)
```

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"}
pessoa = dict(nome="Carol", idade = 18, altura = 1.8)
```

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"}
pessoa = dict(nome="Carol", idade = 18, altura = 1.8)
```

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"}
pessoa = dict(nome="Carol", idade = 18, altura = 1.8)
```

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"}
pessoa = dict(nome="Carol", idade = 18, altura = 1.8)
```

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"}
pessoa = dict(nome="Carol", idade = 18, altura = 1.8)
```

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

No último caso `item` é uma tupla com a chave e o valor.
Para imprimir os valores de um dicionário dic (nome do dicionário) temos 3 métodos principais:

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

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

No último caso `item` é uma tupla com a chave e o valor.
Para imprimir os valores de um dicionário dic (nome do dicionário) temos 3 métodos principais:

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

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

No último caso `item` é uma tupla com a chave e o valor.

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

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

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

No último caso `item` é uma tupla com a chave e o valor.
Para imprimir os valores de um dicionário dic (nome do dicionário) temos 3 métodos principais:

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

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

No último caso `item` é uma tupla com a chave e o valor.
Para imprimir os valores de um dicionário dic (nome do dicionário) temos 3 métodos principais:

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

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

No último caso `item` é uma tupla com a chave e o valor.
Para imprimir os valores de um dicionário dic (nome do dicionário) temos 3 métodos principais:

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

Sumário

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 o valor de uma chave *k*, se o dicionário tiver esta chave.
- 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 o valor de uma chave *k*, se o dicionário tiver esta chave.
- 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 o valor de uma chave *k*, se o dicionário tiver esta chave.
- 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 o valor de uma chave *k*, se o dicionário tiver esta chave.
- 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 o valor de uma chave *k*, se o dicionário tiver esta chave.
- 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 o valor de uma chave *k*, se o dicionário tiver esta chave.
- 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 o valor de uma chave *k*, se o dicionário tiver esta chave.
- 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

Vamos a alguns exemplos de como aplicar os métodos anteriores.

1. Suponha que você é um dev que vai fazer um código para uma revenda de carros e está fazendo um teste simples usando dicionário para o código posterior usando banco de dados. Uma forma de fazer isto pode ser:

```
carro12512 = {"marca": "Ford", "modelo": "EcoSport",
    "ano": 2009}
```

E você não se lembra se colocou ou não o preço do carro. Bastaria fazer:

```
x = carro12512.get("preco")
```

Se *x* for `None` o programador esqueceu-se de colocar a chave `preco` no dicionário `carro12512`, caso contrário *x* recebe o preço. É importante salientar que para ter todos os carros da loja na memória bastaria fazer um vetor de dicionários, com cada elemento representando um carro.

```
carros = []
carros.append(carro12512)
```

Métodos para dicionários

Vamos a alguns exemplos de como aplicar os métodos anteriores.

1. Suponha que você é um dev que vai fazer um código para uma revenda de carros e está fazendo um teste simples usando dicionário para o código posterior usando banco de dados. Uma forma de fazer isto pode ser:

```
carro12512 = {"marca": "Ford", "modelo": "EcoSport",
    "ano": 2009}
```

E você não se lembra se colocou ou não o preço do carro. Bastaria fazer:

```
x = carro12512.get("preco")
```

Se *x* for `None` o programador esqueceu-se de colocar a chave `preco` no dicionário `carro12512`, caso contrário *x* recebe o preço. É importante salientar que para ter todos os carros da loja na memória bastaria fazer um vetor de dicionários, com cada elemento representando um carro.

```
carros = []
carros.append(carro12512)
```

Métodos para dicionários

Vamos a alguns exemplos de como aplicar os métodos anteriores.

1. Suponha que você é um dev que vai fazer um código para uma revenda de carros e está fazendo um teste simples usando dicionário para o código posterior usando banco de dados. Uma forma de fazer isto pode ser:

```
carro12512 = {"marca": "Ford", "modelo": "EcoSport",
    "ano": 2009}
```

E você não se lembra se colocou ou não o preço do carro. Bastaria fazer:

```
x = carro12512.get("preco")
```

Se *x* for `None` o programador esqueceu-se de colocar a chave `preco` no dicionário `carro12512`, caso contrário *x* recebe o preço. É importante salientar que para ter todos os carros da loja na memória bastaria fazer um vetor de dicionários, com cada elemento representando um carro.

```
carros = []
carros.append(carro12512)
```

Métodos para dicionários

Vamos a alguns exemplos de como aplicar os métodos anteriores.

1. Suponha que você é um dev que vai fazer um código para uma revenda de carros e está fazendo um teste simples usando dicionário para o código posterior usando banco de dados. Uma forma de fazer isto pode ser:

```
carro12512 = {"marca": "Ford", "modelo": "EcoSport",
    "ano": 2009}
```

E você não se lembra se colocou ou não o preço do carro. Bastaria fazer:

```
x = carro12512.get("preco")
```

Se x for `None` o programador esqueceu-se de colocar a chave `preco` no dicionário `carro12512`, caso contrário x recebe o preço. É importante salientar que para ter todos os carros da loja na memória bastaria fazer um vetor de dicionários, com cada elemento representando um carro.

```
carros = []
carros.append(carro12512)
```

Métodos para dicionários

Vamos a alguns exemplos de como aplicar os métodos anteriores.

1. Suponha que você é um dev que vai fazer um código para uma revenda de carros e está fazendo um teste simples usando dicionário para o código posterior usando banco de dados. Uma forma de fazer isto pode ser:

```
carro12512 = {"marca": "Ford", "modelo": "EcoSport",
    "ano": 2009}
```

E você não se lembra se colocou ou não o preço do carro. Bastaria fazer:

```
x = carro12512.get("preco")
```

Se *x* for `None` o programador esqueceu-se de colocar a chave `preco` no dicionário `carro12512`, caso contrário *x* recebe o preço. É importante salientar que para ter todos os carros da loja na memória bastaria fazer um vetor de dicionários, com cada elemento representando um carro.

```
carros = []
carros.append(carro12512)
```

Métodos para dicionários

Vamos a alguns exemplos de como aplicar os métodos anteriores.

1. Suponha que você é um dev que vai fazer um código para uma revenda de carros e está fazendo um teste simples usando dicionário para o código posterior usando banco de dados. Uma forma de fazer isto pode ser:

```
carro12512 = {"marca": "Ford", "modelo": "EcoSport",
    "ano": 2009}
```

E você não se lembra se colocou ou não o preço do carro. Bastaria fazer:

```
x = carro12512.get("preco")
```

Se *x* for `None` o programador esqueceu-se de colocar a chave `preco` no dicionário `carro12512`, caso contrário *x* recebe o preço. É importante salientar que para ter todos os carros da loja na memória bastaria fazer um vetor de dicionários, com cada elemento representando um carro.

```
carros = []
carros.append(carro12512)
```

Métodos para dicionários

Vamos a alguns exemplos de como aplicar os métodos anteriores.

1. Suponha que você é um dev que vai fazer um código para uma revenda de carros e está fazendo um teste simples usando dicionário para o código posterior usando banco de dados. Uma forma de fazer isto pode ser:

```
carro12512 = {"marca": "Ford", "modelo": "EcoSport",
    "ano": 2009}
```

E você não se lembra se colocou ou não o preço do carro. Bastaria fazer:

```
x = carro12512.get("preco")
```

Se *x* for `None` o programador esqueceu-se de colocar a chave `preco` no dicionário `carro12512`, caso contrário *x* recebe o preço. É importante salientar que para ter todos os carros da loja na memória bastaria fazer um vetor de dicionários, com cada elemento representando um carro.

```
carros = []
carros.append(carro12512)
```

Métodos para dicionários

Vamos a alguns exemplos de como aplicar os métodos anteriores.

1. Suponha que você é um dev que vai fazer um código para uma revenda de carros e está fazendo um teste simples usando dicionário para o código posterior usando banco de dados. Uma forma de fazer isto pode ser:

```
carro12512 = {"marca": "Ford", "modelo": "EcoSport",
    "ano": 2009}
```

E você não se lembra se colocou ou não o preço do carro. Bastaria fazer:

```
x = carro12512.get("preco")
```

Se *x* for `None` o programador esqueceu-se de colocar a chave `preco` no dicionário `carro12512`, caso contrário *x* recebe o preço. É importante salientar que para ter todos os carros da loja na memória bastaria fazer um vetor de dicionários, com cada elemento representando um carro.

```
carros = []
carros.append(carro12512)
```

Sumário

Compreensão de dicionários

Além de compreensão de listas, temos compreensão de dicionários.
Para compreensão de listas, temos

```
[expressão for membro in iterável]
```

Já para compreensão de dicionários, temos

```
{chave:expressão for membro in iterável}
```

Observe o código:

```
print({i:i*2 for i in range(5)})
```

Teremos um dicionário com as chaves de 0 a 4 e os respectivos valores das chaves entre 0 e 8 (dobro das chaves).
Porém:

```
print({i**2:i*2 for i in range(5)})
```

gera as chaves `{0,1,4,9,16}` e valores `{0,2,4,6,8}`.

Compreensão de dicionários

Preenchendo um dicionário tendo uma lista como base.

```
frutas = ["maçã", 'pera', 'uva', 'melancia', 'mamão']
precos = ["12", '10', '10', '8', '11']
print({frutas[i]:precos[i] for i in range(len(frutas))})
```

e obtemos:

```
{'mamão': '11', 'maçã': '12', 'melancia': '8', 'pera': '10', 'uva': '10'}
```

Interessante também perceber que podemos ter um aninhamento com compreensão de listas:

```
print({fruta:[i for i in range(5)] for fruta in frutas})
```

com a saída:

```
{'mamão': [0, 1, 2, 3, 4],
 'maçã': [0, 1, 2, 3, 4],
 'melancia': [0, 1, 2, 3, 4],
 'pera': [0, 1, 2, 3, 4],
 'uva': [0, 1, 2, 3, 4]}
```

Compreensão de dicionários

Se quisermos um vetor aleatório ao invés de um vetor constante:

```
import random as rn
print({fruta:[rn.randint(100,1000) for _ in range(5)] for fruta in frutas})
```

com a saída:

```
{'mamão': [986, 281, 796, 861, 673],
 'maçã': [532, 277, 665, 656, 792],
 'melancia': [345, 346, 135, 893, 339],
 'pera': [266, 513, 495, 839, 410],
 'uva': [507, 764, 551, 644, 604]}
```

Sumário

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

Exercício 2850

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 direita nenhuma as duas	ingles frances portugues caiu

Exercício 1281

beecrowd | 1281

Ida à Feira

Por Neilor Tonin, URI Brasil

Timelimit: 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	Exemplo de Saída
2 4 mamao 2.19 cebola 3.10 tomate 2.80 uva 2.73 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	R$ 15.37 R$ 15.73

Sumário

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!!!!

Sumário

Compreensão de dicionários

Problema 1

Usando a lista seguir como base:

```
sorteios = ['sorteio1','sorteio2','sorteio3']
participantes = ['joel','jessica','maria','cris','Larissa',
    'rafael','marcus', 'john']
```

crie a seguir, selecionando o ganhador aleatóriamente, um nomes da lista de participantes. A ideia é simular quem irá ganhar cada sorteio, sua lista deve gerar a seguinte estrutura (porém o nome pode vir a ser diferente, já que estamos selecionando os nomes aleatóriamente)

```
{
    sorteio1: 'cris',
    sorteio2: 'rafael',
    sorteio3: 'marcus',
}
```

Compreensão de dicionários

Problema 2

Precisamos gerar 5 valores de 1 a 100 aleatóriamente. E estes valores precisam ser gerados para cada grupo na lista abaixo

```
grupos = ['grupo 1', 'grupo 2', 'grupo 3']
```

O resultado esperado é o dicionário com a estrutura a seguir(os valores entre contindos dentro da lista estarão diferentes, uma vez que os valores abaixo foram geradores aleatóriamente)

```
{
 'grupo 1': [93, 97, 63, 36, 34],
 'grupo 2': [81, 24, 22, 46, 52],
 'grupo 3': [5, 35, 6, 86, 37]
}
```

Compreensão de dicionários

Soluções

```
import random as rn
from pprint import *
sorteios = ['sorteio1','sorteio2','sorteio3']
participantes = ['joel','jessica','maria','cris','Larissa', 'rafael',
    'marcus', 'john']
dic1 = {sorteio: participantes[rn.randint(0,7)]  for sorteio in sorteios}
pprint(dic1)
grupos = ['grupo 1', 'grupo 2', 'grupo 3']
dic2 = {grupo: [rn.randint(1,100) for _ in range(5)] for grupo in grupos}
print(dic2)
```