Técnico em Informática para Internet

INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Dicionários
– Algoritmos e Programação

Profas. Eloize e Silvana

INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Dicionários

- Os tipos de dados compostos estudados até agora são coleções sequenciais na memória, nas quais os elementos da coleção estão ordenados da esquerda para a direita e seus valores são acessados por meio de índices que são números inteiros.
  - Listas e *strings* são coleções sequenciais.

- **Dicionários** também são usados para representar uma coleção, porém a organização dos elementos de um dicionário não é sequencial.

Definindo Dicionários

- Um dicionário é uma estrutura de dados que implementa **mapeamentos**.

- Um mapeamento é uma coleção de associações entre **pares de valores**: o primeiro elemento do par é chamado de **chave** e o outro de **valor**.

- Um mapeamento é uma generalização da ideia de acessar dados por índices, exceto que, num mapeamento, os índices (**ou chaves**) podem ser de qualquer tipo **imutável**.
  - Geralmente são strings ou inteiros.

Definindo Dicionários (cont.)

- **Resumindo:**
  - Listas são indexadas por **inteiros**.
  - Dicionários são indexados por **chaves** (*keys*), que podem ser de qualquer tipo **imutável** (como strings e inteiros).

Dicionários – Exemplo de criação

- Um dicionário representa um mapeamento de **chaves** a **valores.**
  - O mapeamento de uma coleção é feito de forma associativa e desordenada.
- Como exemplo, vamos construir um dicionário de tradução que mapeia as palavras do português para o inglês:

```
>>> pt_ing = {} #cria um dicionário vazio
>>> pt_ing
Saída: {}
```

- Para representar um dicionário vazio, usamos chaves {}.
- Alternativamente, poderíamos escrever: `pt_ing = dict()`
- O tipo nativo ***dict*** também permite a criação de um dicionário vazio.

Inserindo Itens em um Dicionário

- Dicionário **são mutáveis.** Para acrescentarmos itens a um dicionário já criado, usamos **colchetes**. Nos exemplos a seguir, a **chave** é do tipo **string**.

```
>>> pt_ing['maça'] = 'apple'
>>> pt_ing['laranja'] = 'orange'
>>> pt_ing['morango'] = 'strawberry'
>>> pt_ing['uva'] = 'grape'
```

- A **chave** precisa ser **única**, ou seja, não pode haver chaves repetidas.
- A **chave** também deve ser **imutável**, ou seja, uma vez criada, não pode ser alterada.
- A associação de uma chave a um valor chama-se **par chave-valor** ou **item.**

Dicionários – Outro exemplo de criação

- Pode-se também criar um dicionário já com elementos:

```
estoque_frutas = {'banana': 50, 'laranja': 12}
print(estoque_frutas)
estoque_frutas["melancia"]=7
estoque_frutas["melão"]=9
print(estoque_frutas)
estoque_frutas["melancia"]=12
print(estoque_frutas)
```

Saídas:

```
{'banana': 50, 'laranja': 12}
{'banana': 50, 'laranja': 12, 'melancia': 7, 'melão': 9}
{'banana': 50, 'laranja': 12, 'melancia': 12, 'melão': 9}
```

Percurso em Dicionário

- Há três formas principais de se percorrer um dicionário:
  - Iterando sobre cada **chave** (método ***keys()***)
  - Iterando sobre cada **valor** (método ***values()***)
  - Iterando sobre cada **par chave-valor** (método ***items()***)

- **Obs:** A ordem dos pares na impressão pode não ser a que você esperava. O Python usa algoritmos complexos, projetados para acesso muito rápido, para determinar onde os pares chave-valor são armazenados em um dicionário.
- Para os nossos propósitos, podemos pensar que esta ordenação é imprevisível.

Percurso em Dicionário (cont.)

- Não importa em que ordem os pares chave-valor serão impressos, pois os valores em um dicionário são acessados por meio de chaves, e não com índices. Portanto, não há necessidade de se preocupar com a ordenação.

- Veja no exemplo:

```
estoque_frutas = {'banana': 50, 'laranja': 12,
'melancia': 25}

print(f"Quantidade de melancias em estoque:
{estoque_frutas['melancia']}")
```

Saída:

```
Quantidade de melancias em estoque: 25
```

Iterando sobre as chaves de um Dicionário

- Podemos iterar sobre as **chaves** de um dicionário utilizando o método ***keys()*** da seguinte forma:

```
estoque_frutas = {'banana': 50, 'laranja': 12, 'melancia': 25}
for chave in estoque_frutas.keys():
    valor = estoque_frutas[chave]
    print(f"Temos {valor} itens do produto {chave} em estoque")
```

Saídas:

Temos 50 itens do produto banana em estoque.

Temos 12 itens do produto laranja em estoque.

Temos 25 itens do produto melancia em estoque.

Iterando sobre os valores de um Dicionário

- É possível também iterar sobre os valores de um dicionário utilizando o método ***values().***
- Por exemplo, para saber quantas frutas há no estoque podemos fazer:

```
estoque_frutas = {'banana': 50, 'laranja': 12, 'melancia': 25}
total = 0
for valor in estoque_frutas.values():
    total += valor
print(f"Temos um total de {total} frutas no estoque.")
```

Saída: Temos um total de 87 frutas no estoque.

Iterando sobre os pares chave-valor de um Dicionário

- É possível também iterar sobre os **pares chave-valor** de um dicionário utilizando o método ***items().***
- Por exemplo, para imprimir todos os itens de um dicionário, podemos fazer como no código a seguir:

```
estoque_frutas = {'banana': 50, 'laranja': 12, 'melancia': 25}
for k in estoque_frutas.items():
      print("Par Chave-Valor:", k)
```

Saídas:

Par Chave-Valor: ('banana', 50)
Par Chave-Valor: ('laranja', 12)
Par Chave-Valor: ('melancia', 25)

Obtendo o valor associado à uma chave

- Há duas maneiras de se obter o valor associado a determinada chave:
  - Por meio do método ***get*** ou através do **operador colchetes []**
  - Exemplos:

```
estoque_frutas = {'banana': 50, 'laranja': 12,
'melancia': 25}
print(estoque_frutas['melancia']) #imprime 25
print(estoque_frutas.get('melancia')) #imprime 25
print(estoque_frutas.get('pinha','Chave não existe'))
#imprime Chave não existe
```

- A diferença entre eles é que, se a chave não estiver presente no dicionário, o operador **[]** irá causar um erro de execução, enquanto que o **get** retornará **None.**

Criando uma lista com as chaves de um Dicionário

- O método *keys* pode ser usado para se obter uma lista com as chaves de um dicionário. Exemplo:

```
estoque_frutas = {'banana': 50, 'laranja': 12, 'melancia': 25}
Lista_keys = list(estoque_frutas.keys())
print("Lista de frutas:", Lista_keys)
Lista_keys.append('kiwi')
for k in Lista_keys:
    print(k, ": ", estoque_frutas.get(k, "Chave Inexistente"))
```

```
Saídas: Lista de frutas: ['banana', 'laranja', 'melancia']
        banana :  50
        laranja :  12
        melancia :  25
        kiwi :  Chave Inexistente
```

Criando uma lista com os valores de um Dicionário

- De forma similar, podemos criar listas de valores de um dicionário.

- Exemplos:

```
estoque_frutas = {'banana': 50, 'laranja': 12, 'melancia': 25}
print(list(estoque_frutas.values()))
```

Saídas: [50, 12, 25]

Programa Exemplo

- O programa a seguir lê uma frase e conta quantas vezes cada letra ocorreu na frase, usando dicionários. Caracteres em branco, pontuação, tabulação e o caractere de pular linha (\n) são ignorados na contagem.

```
def calcular_frequencia(frase):
    IGNORE = ' .,;:?!\t\n'
    dic_letras = {}
    for c in frase:
        if c not in IGNORE:
            if c in dic_letras:
                dic_letras[c] += 1
            else:
                dic_letras[c] = 1
    return dic_letras
```

>> continuação

Programa Exemplo (continuação)

```
def imprimir_dicionario(dicionario):
    print("Frequencia de cada letra na frase:")
    for k in dicionario.keys():
        print(f"{k} = {dicionario[k]}")

def main():
    frequencia_letras = {} # criando o dicionário
    frase = input("Digite uma frase: ")
    frequencia_letras = calcular_frequencia(frase)
    imprimir_dicionario(frequencia_letras)

main()
```

Operações com Dicionários

- O comando **del** remove um **par chave-valor** de um dicionário
- Exemplo: caso alguém compre todas as peras, pode-se remover a entrada do dicionário

```
estoque_frutas = {'kiwis': 430, 'bananas': 312,
'laranjas': 525, 'peras': 217}

print(estoque_frutas)

del estoque_frutas['peras']

print(estoque_frutas)
```

Operações com Dicionários

- Os operadores **in** e **not** in podem testar se uma chave está no dicionário:

```
estoque_frutas = {'kiwis': 430, 'bananas': 312, 'laranjas': 525, 'peras': 217}

print('maça' in estoque_frutas) # imprime False

if 'uvas' not in estoque_frutas:
      print("Não temos uvas no estoque!")
```

Operações com Dicionários

- A função **len** também é compatível com dicionários.

- Exemplo:

```
estoque_frutas = {'kiwis': 430, 'bananas': 312, 'laranjas':
525, 'peras': 217}
print(len(estoque_frutas)) # imprime 4
```

Copiando um Dicionário

- Se a intenção for modificar um dicionário e manter uma cópia do original, deve-se utilizar método **copy.**

- Exemplo:

```
dic1 = {"Joao": 10, "Maria":20}
dic2 = dic1.copy()
dic2["Pedro"] = 30
dic1["Joao"] = 40
print(dic1) # saída: {"Joao": 40,"Maria": 20}
print(dic2) # saída: {"Pedro": 30,"Joao":
10,"Maria": 20}
```

Exercícios

1. A seguinte tabela contém tradução de algumas palavras em inglês para a língua dos piratas. Escreva um programa que pergunta ao usuário uma frase em inglês e imprime a tradução da frase para a língua dos piratas.

Inglês	Pirata
sir	matey
hotel	fleabag inn
student	swabbie
boy	matey
madam	proud beauty
professor	foul blaggart
restaurant	galley
your	yer
excuse	arr
students	swabbies
are	be
lawyer	foul blaggart
the	th'
restroom	head
my	me
hello	avast
is	be
man	matey

Exercícios

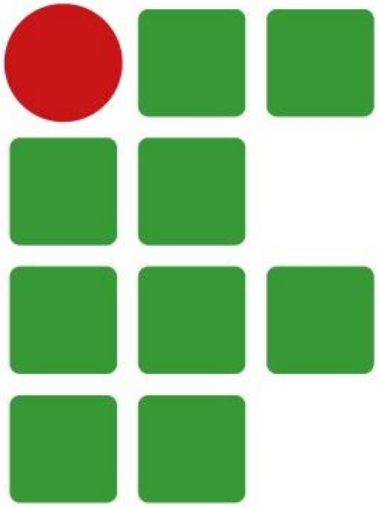
2. Elabore um programa para armazenar uma agenda de telefones em um dicionário. Cada pessoa pode ter um ou mais telefones e a chave do dicionário é o nome da pessoa. O programa deve ter as seguintes funções:

   a) incluirNovoNome() – essa função acrescenta um novo nome na agenda, com um ou mais telefones. Ela deve receber como argumentos o nome e os telefones;

   b) incluirTelefone() – essa função acrescenta um telefone em um nome existente na agenda. Caso o nome não exista na agenda, você deve perguntar se a pessoa deseja incluí-lo. Caso a resposta seja afirmativa, use a função anterior para incluir o novo nome;

   c) excluirTelefone() – essa função exclui um telefone de uma pessoa que já está na agenda. Se a pessoa tiver apenas um telefone, ela deve ser excluída da agenda;

   d)excluirNome() – essa função exclui uma pessoa da agenda. consultarTelefone – essa função retorna os telefones de uma pessoa na agenda.



INSTITUTO FEDERAL

São Paulo

Câmpus São Carlos