Programação I

Licenciatura em Engenharia Informática

2015–2016

Vitor Beires Nogueira

Escola de Ciências e Tecnologia
Universidade de Évora

Introdução

- Um *dicionário* é semelhante a uma lista
  - Numa lista os índices são inteiros
  - Num dicionários os índices pode ser de *quase* qualquer tipo

Introdução

- Um *dicionário* é semelhante a uma lista
  - Numa lista os índices são inteiros
  - Num dicionários os índices pode ser de *quase* qualquer tipo
- Um dicionário é uma atribuição (*mapping*) entre chaves (*keys*) e um conjunto de valores (*values*). Uma associação *key value* é por vezes designada por *item*

Introdução

- Um *dicionário* é semelhante a uma lista
  - Numa lista os índices são inteiros
  - Num dicionários os índices pode ser de *quase* qualquer tipo
- Um dicionário é uma atribuição (*mapping*) entre chaves (*keys*) e um conjunto de valores (*values*). Uma associação *key value* é por vezes designada por *item*
- A função `dict` cria um novo dicionário sem *items*

Exemplo

```
>>> eng2sp = dict()
>>> eng2sp
    {}
```

Introdução

- Um *dicionário* é semelhante a uma lista
  - Numa lista os índices são inteiros
  - Num dicionários os índices pode ser de *quase* qualquer tipo
- Um dicionário é uma atribuição (*mapping*) entre chaves (*keys*) e um conjunto de valores (*values*). Uma associação *key value* é por vezes designada por *item*
- A função `dict` cria um novo dicionário sem *items*

Exemplo

```
>>> eng2sp = dict()
>>> eng2sp
    {}
>>> eng2sp['one'] = 'uno'
>>> eng2sp
```

Introdução

- Um *dicionário* é semelhante a uma lista
  - Numa lista os índices são inteiros
  - Num dicionários os índices pode ser de *quase* qualquer tipo
- Um dicionário é uma atribuição (*mapping*) entre chaves (*keys*) e um conjunto de valores (*values*). Uma associação *key value* é por vezes designada por *item*
- A função `dict` cria um novo dicionário sem *items*

Exemplo

```
>>> eng2sp = dict()
>>> eng2sp
    {}
>>> eng2sp['one'] = 'uno'
>>> eng2sp
    { 'one':  'uno' }
```

Dicionário (simplista) de Inglês - Espanhol

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> eng2sp
```

Dicionário (simplista) de Inglês - Espanhol

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> eng2sp
    {'three':  'tres', 'two':  'dos', 'one':  'uno'}
```

Dicionário (simplista) de Inglês - Espanhol

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> eng2sp
    {'three':  'tres', 'two':  'dos', 'one':  'uno'}
>>> eng2sp['one']
    uno
```

Dicionário (simplista) de Inglês - Espanhol

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> eng2sp
    {'three':  'tres', 'two':  'dos', 'one':  'uno'}
>>> eng2sp['one']
    uno
>>> eng2sp['four']
    KeyError:  'four'
```

Dicionário (simplista) de Inglês - Espanhol

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> eng2sp
    {'three':  'tres', 'two':  'dos', 'one':  'uno'}
>>> eng2sp['one']
    uno
>>> eng2sp['four']
    KeyError:  'four'
>>> len(eng2sp)
    3
```

Dicionário (simplista) de Inglês - Espanhol

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> eng2sp
    {'three':  'tres', 'two':  'dos', 'one':  'uno'}
>>> eng2sp['one']
    uno
>>> eng2sp['four']
    KeyError:  'four'
>>> len(eng2sp)
    3
>>> 'one' in eng2sp
    True
```

Dicionário (simplista) de Inglês - Espanhol

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> eng2sp
    {'three':  'tres', 'two':  'dos', 'one':  'uno'}
>>> eng2sp['one']
    uno
>>> eng2sp['four']
    KeyError:  'four'
>>> len(eng2sp)
    3
>>> 'one' in eng2sp
    True
>>> 'uno' in eng2sp
```

Dicionário (simplista) de Inglês - Espanhol

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> eng2sp
    {'three':  'tres', 'two':  'dos', 'one':  'uno'}
>>> eng2sp['one']
    uno
>>> eng2sp['four']
    KeyError:  'four'
>>> len(eng2sp)
    3
>>> 'one' in eng2sp
    True
>>> 'uno' in eng2sp
    False
```

Valores de um dicionário

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> vals = eng2sp.values()
>>> 'uno' in vals
```

Valores de um dicionário

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> vals = eng2sp.values()
>>> 'uno' in vals
    True
```

Valores de um dicionário

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> vals = eng2sp.values()
>>> 'uno' in vals
    True
>>> vals
```

Valores de um dicionário

Exemplo

```
>>> eng2sp = {'one':'uno', 'two':'dos',
    'three':'tres'}
>>> vals = eng2sp.values()
>>> 'uno' in vals
    True
>>> vals
    dict_values(['tres', 'dos', 'uno'])
```

Dicionários como um conjunto de contadores

Considere que pretendemos fazer uma função que recebe uma string e conta o número de vezes que cada letra ocorre na string.
Soluções?
Dependem da "estrutura de dados"utilizada:

Dicionários como um conjunto de contadores

Considere que pretendemos fazer uma função que recebe uma string e conta o número de vezes que cada letra ocorre na string.
Soluções?
Dependem da "estrutura de dados"utilizada:

1. Utilizando 26 variáveis, uma por cada letra do alfabeto.

Dicionários como um conjunto de contadores

Considere que pretendemos fazer uma função que recebe uma string e conta o número de vezes que cada letra ocorre na string.
Soluções?
Dependem da "estrutura de dados"utilizada:

1. Utilizando 26 variáveis, uma por cada letra do alfabeto.
2. Utilizando listas. Ver a função `ord`

Dicionários como um conjunto de contadores

Considere que pretendemos fazer uma função que recebe uma string e conta o número de vezes que cada letra ocorre na string.
Soluções?
Dependem da "estrutura de dados"utilizada:

1. Utilizando 26 variáveis, uma por cada letra do alfabeto.
2. Utilizando listas. Ver a função `ord`
3. Utilizando dicionários.

Histograma

Exemplo

```
def histogram(s):
    d = dict()
    for c in s:
        if c in d:
            d[c] += 1
        else:
            d[c] = 1
    return d
```

Histograma

Exemplo

```
def histogram(s):
    d = dict()
    for c in s:
        if c in d:
            d[c] += 1
        else:
            d[c] = 1
    return d
```

Exemplo (Utilização)

```
>>> histogram('banana')
```

Histograma

Exemplo

```
def histogram(s):
    d = dict()
    for c in s:
        if c in d:
            d[c] += 1
        else:
            d[c] = 1
    return d
```

Exemplo (Utilização)

```
>>> histogram('banana')
    {'a':3, 'b':1, 'n':2}
```

Exemplo (Escrever conteúdo do dicionário)

```
def print_hist(h):
    for c in h:
        print(c, h[c])
```

Ciclos e dicionários

Exemplo (Escrever conteúdo do dicionário)

```
def print_hist(h):
    for c in h:
        print(c, h[c])
```

Exemplo (Utilização)

```
>>> print_hist(histogram('banana'))
```

Exemplo (Escrever conteúdo do dicionário)

```
def print_hist(h):
    for c in h:
        print(c, h[c])
```

Exemplo (Utilização)

```
>>> print_hist(histogram('banana'))
a 3
n 2
b 1
```

Pesquisa inversa

- *lookup*: dada uma chave e um dicionário, podemos facilmente obter o valor correspondente
- Mas como podemos proceder se tivermos o dicionário e o valor e pretendermos obter a chave?

Pesquisa inversa

- *lookup*: dada uma chave e um dicionário, podemos facilmente obter o valor correspondente
- Mas como podemos proceder se tivermos o dicionário e o valor e pretendermos obter a chave?
  - Podemos ter mais que uma chave para tal valor!
  - Não existe nenhum método específico para tal. Por isso, temos de fazer uma pesquisa inversa

Exemplo (Reverse Lookup)

```
def reverse_lookup(d, v):
    for c in d:
        if d[c] == v:
            return c
    raise ValueError
```

Utilizando reverse lookup

Exemplo (Utilização)

```
>>> h = histogram('parrot')
>>> k = reverse_lookup(h, 2)
>>> k
    r
```

Utilizando reverse lookup

Exemplo (Utilização)

```
>>> h = histogram('parrot')
>>> k = reverse_lookup(h, 2)
>>> k
    r
>>> k = reverse_lookup(h, 3)
    Traceback (most recent call last):
    File «stdin>", line 1, in ?
    File «stdin>", line 5, in reverse_lookup
    ValueError
```

Raise

- A instrução `raise` tem um argumento opcional que pode ser utilizado para detalhar a mensagem de erro.
- Por exemplo, podemos fazer:

```
>>> raise ValueError('value does not appear
in the dictionary')
Traceback (most recent call last):
File "<stdin>", line 1, in ?
ValueError: value does not appear in the dictionary
```

Exemplo (Inversa de um dicionário)

```
def invert_dict(d):
    inv = dict()
    for key in d:
        val = d[key]
        if val in inv:
            inv[val].append(key)
        else:
            inv[val] = [key]
    return inv
```

Exemplo

```
>>> h = histogram('parrot')
>>> inv_h = invert_dict(h)
>>> inv_h
```

Exemplo (Inversa de um dicionário)

```
def invert_dict(d):
    inv = dict()
    for key in d:
        val = d[key]
        if val in inv:
            inv[val].append(key)
        else:
            inv[val] = [key]
    return inv
```

Exemplo

```
>>> h = histogram('parrot')
>>> inv_h = invert_dict(h)
>>> inv_h
    {1: ['a', 'p', 't', 'o'], 2: ['r']}
```

Keys must be hashable

Exemplo

```
>>> t = [1, 2, 3]
>>> d = dict()
>>> d[t] = 'oops'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

Keys must be hashable

Exemplo

```
>>> t = [1, 2, 3]
>>> d = dict()
>>> d[t] = 'oops'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

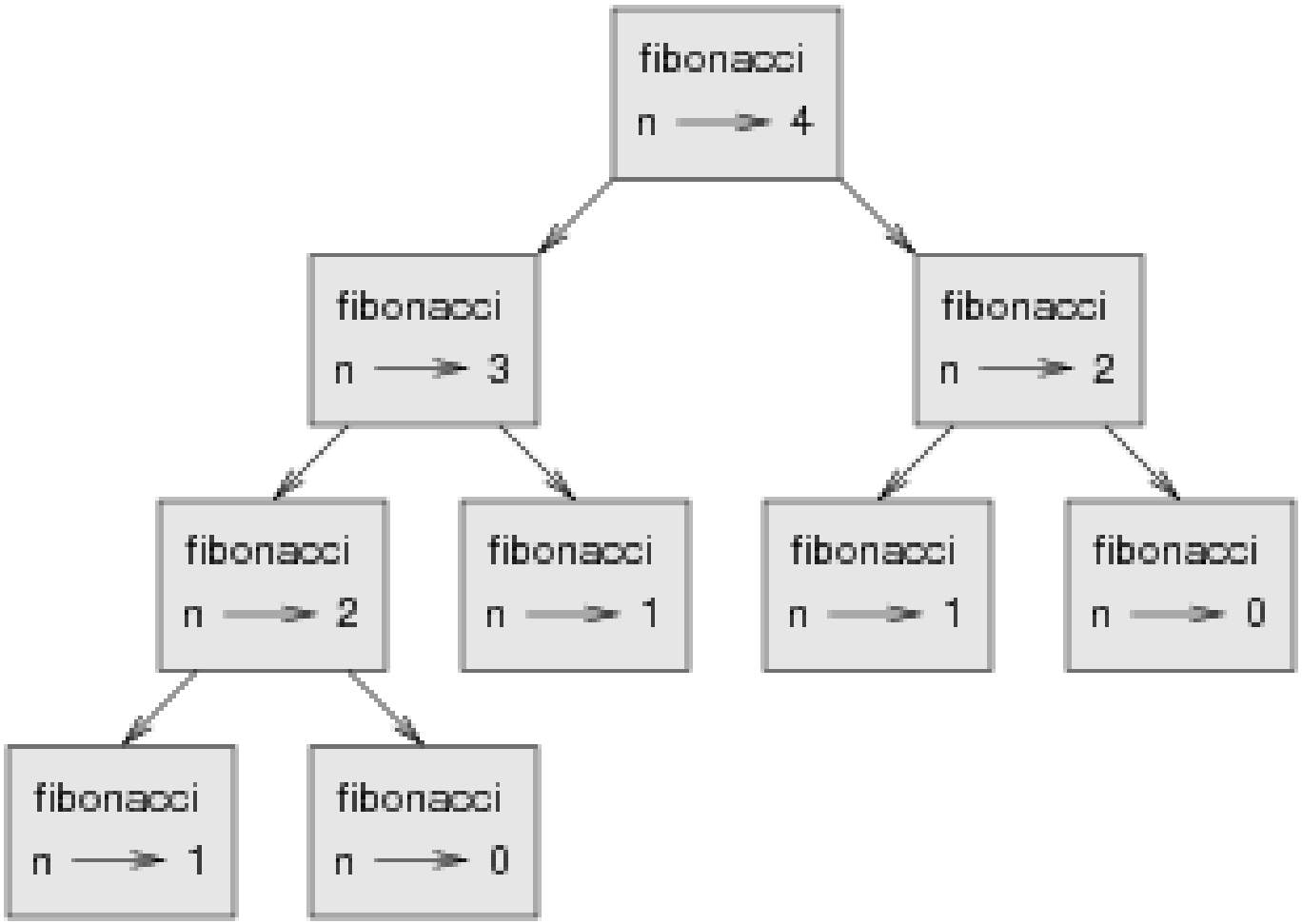
Definição (Hash)

Uma função de *hash* é uma função que recebe um valor (de qualquer tipo) e devolve um inteiro.

Exemplo (função `fibonnaci`)

```
def fibonnaci(n):
    if n == 0:
        return 0
    else:
        if n == 1:
            return 1
        else:
            return fibonnaci(n-1)+fibonnaci(n-2)
```

Grafo de chamadas de `fibonnaci(4)`



- Existe muita redundância
- Como resolver?

Fibonacci com memoization

Exemplo (Fibonacci com memoization)

```
known = {0:1, 1:1}

def fibonacci_memo(n):
    if n in known:
        return known[n]
    fib_n = fibonacci_memo(n-1) + \
            fibonacci_memo(n-2)
    known[n] = fib_n
```

Utilizando dicionários para representar grafos

- Resumidamente um grafo é como um conjunto de pontos (*nós*) ligados por segmentos de recta (*arestas*).
- Em grafos dirigidos, as arestas são direccionadas e "setas".
- Existem imensos algoritmo de grafos para:
  - descobrir o caminho entre dois nós
  - encontrar ciclos num grafo
  - encontrar um caminho que passe por todos os nós

Exemplo de um grafo utilizando dicionários

Exemplo

```
grafo = {'A': ['B', 'C'],
         'B': ['C', 'D'],
         'C': ['D'],
         'D': ['C'],
         'E': ['F'],
         'F': ['C']}
```

Pesquisando caminhos

Exemplo

```
def find_path(graph, start, end, path=[]):
    path = path + [start]
    if start == end:
        return path
    if not (start in graph):
        return None
    for node in graph[start]:
        if node not in path:
            newpath = find_path(graph, node, end, path)
            if newpath:
                return newpath
    return None
```

Utilizando a pesquisa

Exemplo

```
>>> print (find_path(grafo, 'A', 'D'))
```

Utilizando a pesquisa

Exemplo

```
>>> print (find_path(grafo, 'A', 'D'))
    ['A', 'B', 'C', 'D']
```

Utilizando a pesquisa

Exemplo

```
>>> print (find_path(grafo, 'A', 'D'))
    ['A', 'B', 'C', 'D']
```