# Dicionários

Programação I

2016.2017

Teresa Gonçalves tcg@uevora.pt

Departamento de Informática, ECT-UÉ

# Como programar?



#### Perceber o problema

Dados e Resultados

#### Pensar numa solução

Dividir o problema em problemas mais simples

Resolver os problemas mais simples

Resolver o problema mais complexo

#### Implementar a solução

Utilizar funções para estruturar a resolução dos sub-problemas

#### Testar a solução

Fazer vários testes

Escolher valores que induzam comportamentos diferentes do programa



## Sumário

Dicionário

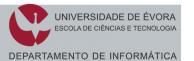
**Propriedades** 

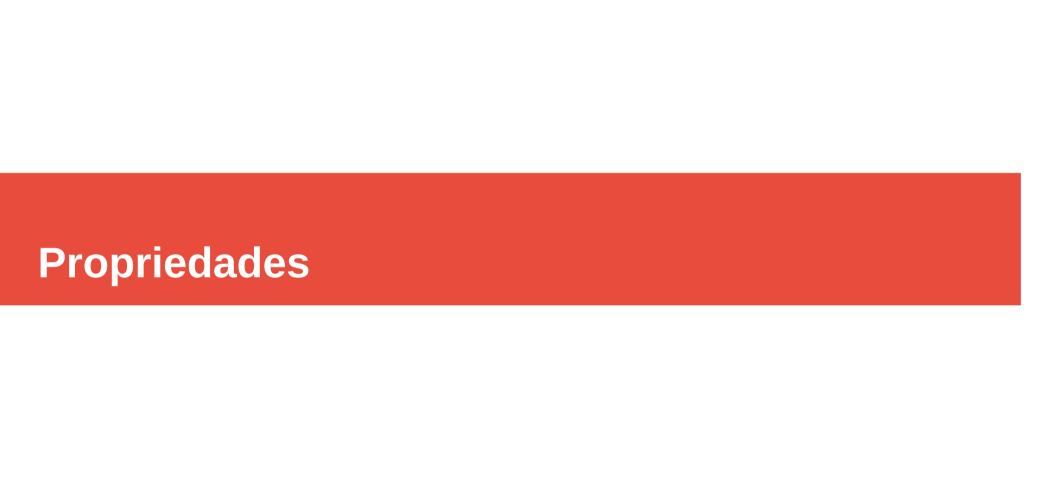
Métodos

Iteração

**Aplicações** 

**Outras operações** 





## Dicionário

#### Refere um conjunto de itens

Conjunto não ordenado de associações chave:valor

Cada par chave:valor designa-se por entrada (no dicionário)

Definição entre {}

#### **Nota**

Não é sequencial!

Estrutura mais complexa que as listas

#### **Exemplo**

```
>>> dicionario={'two':2, 'one':1}
>>> type(dicionario)
<class 'dict'>
```



## **Propriedades**

#### **Associativo**

Mantém correspondência entre valores

#### Mutável

É possível alterar o conteúdo de um dicionário

#### Indexável

A indexação é feita através de chaves

#### Unicidade da chave

Há uma entrada **única** para cada chave

## Características

## Conjunto não ordenado de associações chave:valor Chave

tipo imutável

Valores numéricos, strings

Tuplos se não incluirem itens mutáveis

#### **Valor**

tipo qualquer

# Criação de dicionários

#### Atribuição

```
>>> d1 = {}
>>> type(d1)
<class 'dict'>
```

## Função dict()

```
>>> d2 = 2011,
>>> type(d2)
<class 'dict'>
>>> d1 == d2
True
```

# Função dict()

#### Aceita uma sequência de valores com 2 elementos

```
1° – chave
2° – valor
```

#### **Exemplo**

```
>>> dict([['one', 1],('two', 2)])
{'one':1, 'two':2}
```



# Manipulação

#### **Adicionar**

```
d[chave]=valor
>>> d=d{'one':1],'two':2)
>>> d['three']=3
{'one':1,'two':2,'three':3}
```

#### Remover

```
del d[chave]
>>>del d('two'); d
{'one':1,'three':3}
```



# Manipulação

#### Aceder

```
d[chave]
Dá erro se chave não existir
>>> d={'one':1,'two':2,'three':3}
>>> d['two']
2
>>> d(3)
Traceback (most recent call last):...
KeyError: 3
```

## **Outras funcionalidades**

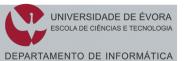
## len()

```
tamanho do dicionário
>>> d={1:'um', 2:'dois'}
>>> len(d)
2
```

#### chave in d

Verifica se existe a entrada chave no dicionario d

```
>>> 3 in d
False
>>> 'mickey' not in {'tweety':'passaro',
'willy':'baleia', 'donald':'pato'}
True
```





# copy() e clear()

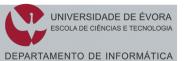
## d.copy()

Devolve um novo dicionário que é cópia de d

```
>>> d={1:'um', 2:'dois'}
>>> d1=d.copy()
>>> del d[1]
>>> len(d)
2
>>> len(d1)
1
```

## d.clear()

Remove todos os itens do dicionário d



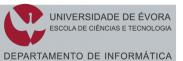
# get()

## d.get(chave,default)

Devolve o valor associado à chave no dicionário d

```
Se a chave não existir, devolve
```

```
default - valor por omissão
  None - se default não existir
>>> d={'tweety':'passaro', 'willy':'baleia', 'donald':'pato'}
>>> d.get('donald')
'pato'
>>> d.get('mickey')
```



# keys(), values() e items()

#### d.keys()

```
Devolve uma sequência iterável das chaves em d
>>> d={'tweety':'passaro', 'willy':'baleia', 'donald':'pato'}
>>> d.keys()
['tweety','willy','donald']
```

#### d.values()

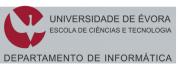
Devolve uma sequência iterável dos valores em d

```
>>> d.values()
['passaro','baleia','pato']
```

#### d.items()

Devolve uma sequência iterável dos tuplos em d

```
>>> d.items()
[('tweety':'passaro'),('willy':'baleia'),('donald':'pato')]
```



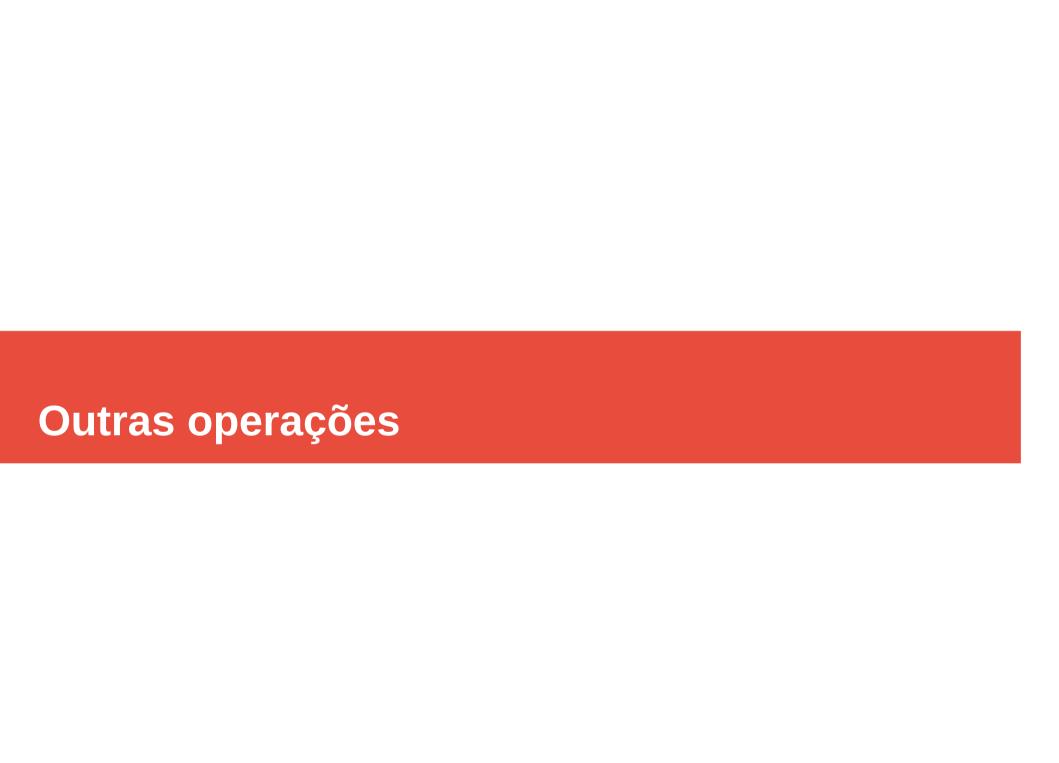
# Iteração

# Iteração sobre dicionário

```
for chave in d:
    print(d[chave])

for chave in d.keys():
    print(d.get(chave))

for chave, valor in d.items():
    print("%s : %s" % (chave, valor))
```



# Pesquisa inversa

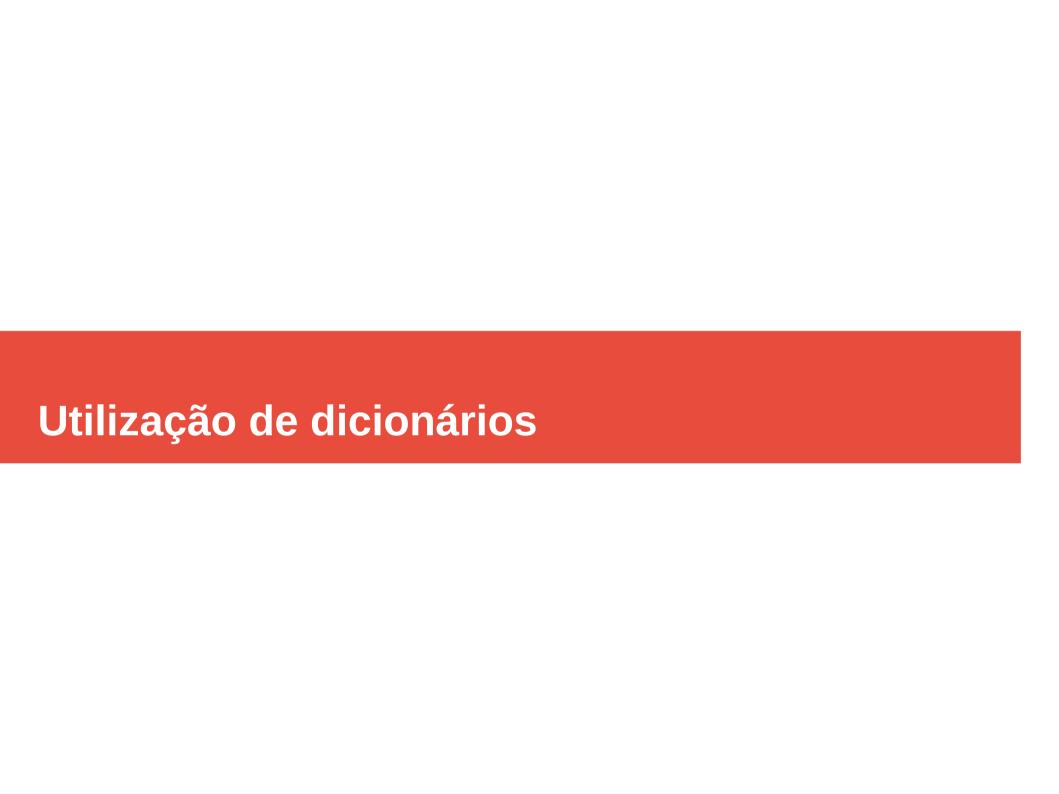
#### Dado um valor, encontrar a chave associada a esse valor

```
def obter_chave(valor,dict):
    for ch in dict:
        if dict[ch]==valor:
            return c
    return None
```

## Inverter dicionário

#### O dicionário garante a unicidade das chaves, não dos valores!

```
def inverte_dict(dict):
    d={}
    for chave,valor in dict.items():
        if val not in dict:
            d[valor]=[chave]
        else:
            d[valor].append(chave)
    return d
>>> jog={'figo':'portugal','ronaldo':'portugal','zidane':'franca','guardiola':'espanha'}
>>> print( inverte_dic(jog)
{'franca': ['zidane'],'portugal': ['figo', 'ronaldo'], 'espanha': ['guardiola']}
```



## Contar a ocorrência de cada palavra num texto

## Manter lista de palavras e nº ocorrências respetivas

```
freq_pals={}
```

## Saber o nº de ocorrências de uma palavra

```
freq_pals.get(pal,0)
```

#### Processar uma palavra do texto

```
if not pal in freq_pals:
    freq_pals[pal]=1 # adicionar palavra, freq=1
else:
    freq_pals[pal]=freq_pals[pal]+1
```



# Desenhar um histograma da ocorrência de letras numa string

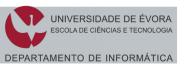
```
def contagem(str):
                                  def barra(n):
  letras={}
                                    s=' '
  for c in str:
                                    for i in range(n):
    if c in letras:
                                       s=s+'='
      letras[c]=letras[c]+1
                                    return s
    else:
      letras[c]=1
                              def histograma(str):
  return letras
                                c=contagem(str)
                                for c,n in d.items():
                                  h=barra(n)
                                  print("%c:%s"%(c,h))
```

# Desenhar um histograma da ocorrência de letras numa string

```
>>> print(contagem('hipopotamo'))
{'a':1,'i':1,'h':1,'m':1,'o':3,'p':2,'t':1}
>>> histograma('hipopotamo')
a:=
i:=
h:=
m:=
o:===
p:==
t:=
```

#### Nota

Os elementos do dicionário não aparecem ordenados!



# Agência de aluguer de automóveis

## Dicionário para automóveis

```
automoveis={'11-10-AB':['ford',5,'gasolina,
'12-10-AB':['renault'35,'diesel'],
'13-10-AB':['mercedes',5,'gasolina]}
```

#### Dicionário para clientes

```
clientes={'ana':[12345678,'Evora',24],
'joao':[23456789,'Lisboa',21],
'bruno':[234567890,'Abrantes',30]}
```

#### Dicionário para alugueres

```
alugueres={'11-10-AB':['ana','2016-10-10'], '11-10-AB':['bruno','2016-11-01']}
```



## **Funcionalidades**

## O carro '11-10-AB' está alugado?

Existe uma entrada em alugueres com chave '11-10-AB'?

#### Qual a idade do cliente Bruno?

Consultar no dicionário clientes o índice 2 da lista associada à chave 'Bruno'

#### **Quantos carros existem com 5 portas?**

Contabilizar no dicionário automoveis o nº de entradas cujo índice 2 da lista de valores é igual a 5



# Outras utilizações

## Problemas que necessitem de estruturas associativas

Lista telefónica

Índice remissivo

**Stocks** 

Produtos e respetivas unidades

Produtos e respetivo preço

