Programação I / Introdução à Programação

Capítulo 5, "Data types" (listas em compreensão, dicionários)

João Pedro Pedroso

2024/2025

Última aula:

- listas em compreensão
- dicionários

Nesta aula

- métodos de dicionários
- dicionários em compreensão
- exemplos

Dicionários: operações básicas (1)

```
>>> d = {"one":1, "two":2, "three":3}
```

- número de elementos

```
>>> len(d)
3
```

- acesso a um elemento *(erro se chave não existir)*

```
>>> d["two"]
2
```

- inserção/atualização de um elemento

```
>>> d["four"] = 4
>>> d
{'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

- existência de chave

```
>>> "four" in d
True
>>> "five" in d
False
```

Dicionários: operações básicas (2)

```
>>> d = {'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

- apagar chave

```
>>> del d["two"]
>>> d
{'one': 1, 'three': 3, 'four': 4}
```

- percorrer as chaves:

```
>>> for k in d:
...     print("{:7}{:3}".format(k, d[k]))
...
one      1
three    3
four     4
```

Dicionários: métodos que não os alteram

Alguns métodos pré-definidos:

`d.get(k, x)` valor associado à chave `k`

- `x` se a chave não existir

`d.copy()` retorna cópia superficial de `d`

`d.keys()` sequência com as chaves de `d`

`d.values()` sequência com os valores de `d`

`d.items()` sequência com pares (chave,valor) em `d`

Exemplos

```
>>> d = {"one":1, "two":2, "three":3}
>>> d.get("one", "no value!")
1
>>> d.get("five", "no value!")
'no value!'
>>> list(d.keys())
['one', 'two', 'three']
>>> d = {"one":1, "two":2, "three":3}
>>> list(d.values())
[1, 2, 3]
>>> list(d.items())
[('one', 1), ('two', 2), ('three', 3)]
```

Dicionários: métodos que os alteram

Alguns métodos pré-definidos:

`d.pop(k[, x])` retorna valor associado à chave `k` e apaga-a

- `d[k]` se a chave `k` existir
- `x` se a chave não existir
- se a chave não existir e não se fornecer `x` $\rightarrow$ erro

`d.update(other)` insere todos os elementos de outro dicionário `other` em `d`

`d.clear()` apaga todos os elementos de `d`

Exemplos

```
>>> d = {"one":1, "two":2, "three":3}
>>> d.pop("one")
1
>>> d
{'two': 2, 'three': 3}
>>> d.update({"five":5, "six":6, "inf":"oo"})
>>> d
{'two': 2, 'three': 3, 'five': 5, 'six': 6, 'inf': 'oo'}
>>> d.clear()
>>> d
{}
```

Dicionários em compreensão

- Forma semelhante à das listas em compreensão
- Exemplo:

```
>>> d = {i:(25 + 2*i + 1.5*i**2) for i in [1.23, 2.45, 3.66, 5.12]}
>>> d
{1.23: 29.72935, 2.45: 38.90375, 3.66: 52.4134, 5.12: 74.5616}
```

Dicionários em compreensão

- Forma semelhante à das listas em compreensão
- Exemplo:

```
>>> d = {i:(25 + 2*i + 1.5*i**2) for i in [1.23, 2.45, 3.66, 5.12]}
>>> d
{1.23: 29.72935, 2.45: 38.90375, 3.66: 52.4134, 5.12: 74.5616}
```

```
>>> for k in d:
...     print("{}\t{}".format(k, d[k]))
...
1.23        29.72935
2.45        38.90375
3.66        52.4134
5.12        74.5616
```

Contar ocorrências

- Problema: contar a ocorrência de cada letra do alfabeto num ficheiro de texto
  - *usar um dicionário para associar cada letra à sua contagem*
  - → histograma
- O canto I d'*Os Lusíadas* (Fonte: Projeto Guttenberg `http://www.gutenberg.org/cache/epub/3333/pg3333.txt`)

```
As armas e os barões assinalados,
Que da ocidental praia Lusitana,
Por mares nunca de antes navegados,
Passaram ainda além da Taprobana,
Em perigos e guerras esforçados,
...
```

Histograma de ocorrências:

```
def histograma(txt):
    count = {}
    for t in txt:
        c = t.lower()
        if c>='a' and c<='z':
            count[c] = 1+count.get(c,0)
    return count
```

- Resultado no segmento anterior:

```
>>> histograma(txt)
{'a': 30, 's': 17, 'r': 11, 'm': 5, 'e': 13, 'o': 9, 'b': 2, 'i': 6,
'n': 9, 'l': 4, 'd': 8, 'q': 1, 'u': 4, 'c': 2, 't': 4, 'p': 5,
'v': 1, 'g': 3, 'f': 1}
```

- Visualização: imprimir uma tabela de ocorrências de letras

```
hist = histograma(txt)
for c in hist:
    print(c, hist[c])
a 30
s 17
r 11
m 5
```

Observações

- Analisando a totalidade do Canto 1:

```
{'a': 3022, 's': 1829, 'r': 1581, 'm': 1035, 'e': 3068, 'o': 2623,
 'b': 249, 'i': 1156, 'n': 1295, 'l': 573, 'd': 1220, 'q': 391,
 'u': 1033, 'c': 659, 't': 1231, 'p': 532, 'v': 425, 'g': 379,
 'f': 240, 'h': 239, 'j': 103, 'z': 68, 'x': 29, 'y': 1}
```

- A letra `'e'` é a que ocorre mais vezes (3068)
- A letra `'y'` ocorre apenas 1 vez
- Contudo: os resultados são incorretos porque não contabilizamos letras acentuadas!
  - $\rightarrow$ converter letras acentuadas em não acentuadas

Normalização

- Vimos numa das aulas passadas: módulo `unidecode`

```
>>> txt = """As armas e os barões assinalados,
... Que da ocidental praia Lusitana,
... Por mares nunca de antes navegados,
... Passaram ainda além da Taprobana,
... Em perigos e guerras esforçados,
... """
>>> from unidecode import unidecode
>>> print(unidecode(txt))
As armas e os baroes assinalados,
Que da ocidental praia Lusitana,
Por mares nunca de antes navegados,
Passaram ainda alem da Taprobana,
Em perigos e guerras esforcados,
```

Histograma de ocorrência: versão *unicode*

```
from unidecode import unidecode

def histograma(txt):
    count = {}
    for t in unidecode(txt):
        c = t.lower()
        if c>='a' and c<='z':
            count[c] = 1+count.get(c,0)
    return count
```

```
>>> histograma(canto1)
{'a': 3303, 's': 1829, 'r': 1581, 'm': 1035, 'e': 3184, 'o': 2708,
 'b': 249, 'i': 1212, 'n': 1295, 'l': 573, 'd': 1220, 'q': 391,
 'u': 1044, 'c': 740, 't': 1231, 'p': 532, 'v': 425, 'g': 379,
 'f': 240, 'h': 239, 'j': 103, 'z': 68, 'x': 29, 'y': 1}
```

Cópia (superficial) de dicionários

- Através do método copy:

```
>>> d = {i:(25 + 2*i + 1.5*i**2) for i in [1.23, 2.45, 3.66, 5.12]}
>>> e = d.copy()
>>> e
{1.23: 29.72935, 2.45: 38.90375, 3.66: 52.4134, 5.12: 74.5616}
```

- Com

```
f = dict(d)
```

Copia superficial (*shallow copy*)

- Copia as *referências* de objectos contidos em listas/dicionários
- Não copia esses objetos...

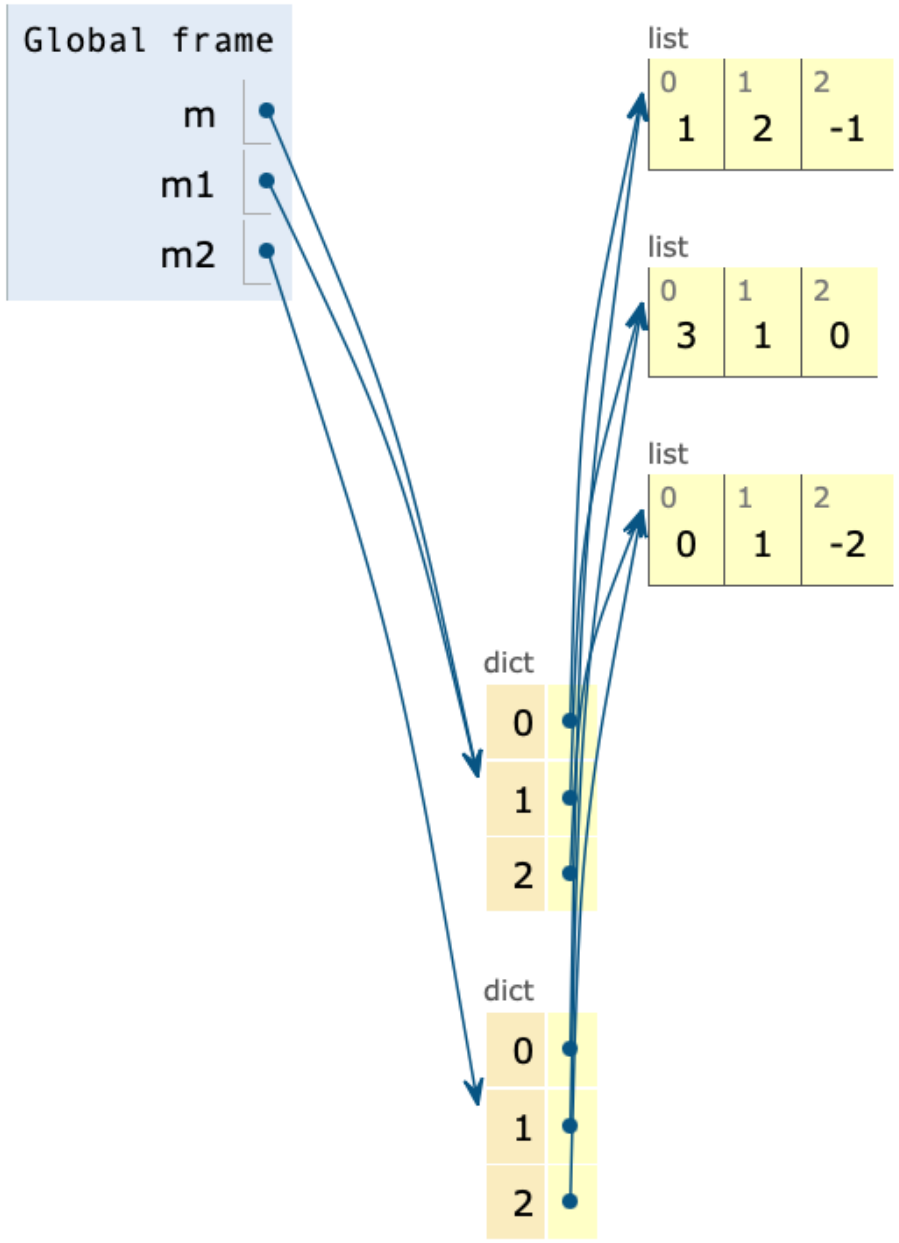
```
>>> m = {0:[1,2,-1],
...      1:[3,1,0],
...      2:[0,1,-2]}
>>> m1 = m     # nova referência para 'm'
>>> m2 = m.copy()    # cópia superficial de 'm'
>>> m2[1][0] = 999999
>>> m2
{0: [1, 2, -1], 1: [999999, 1, 0], 2: [0, 1, -2]}
>>> m
{0: [1, 2, -1], 1: [999999, 1, 0], 2: [0, 1, -2]}
```

Módulo copy

copy	copia o primeiro nível de um objecto (cópia superficial)
deepcopy	copia todos os níveis de um objecto (cópia profunda)

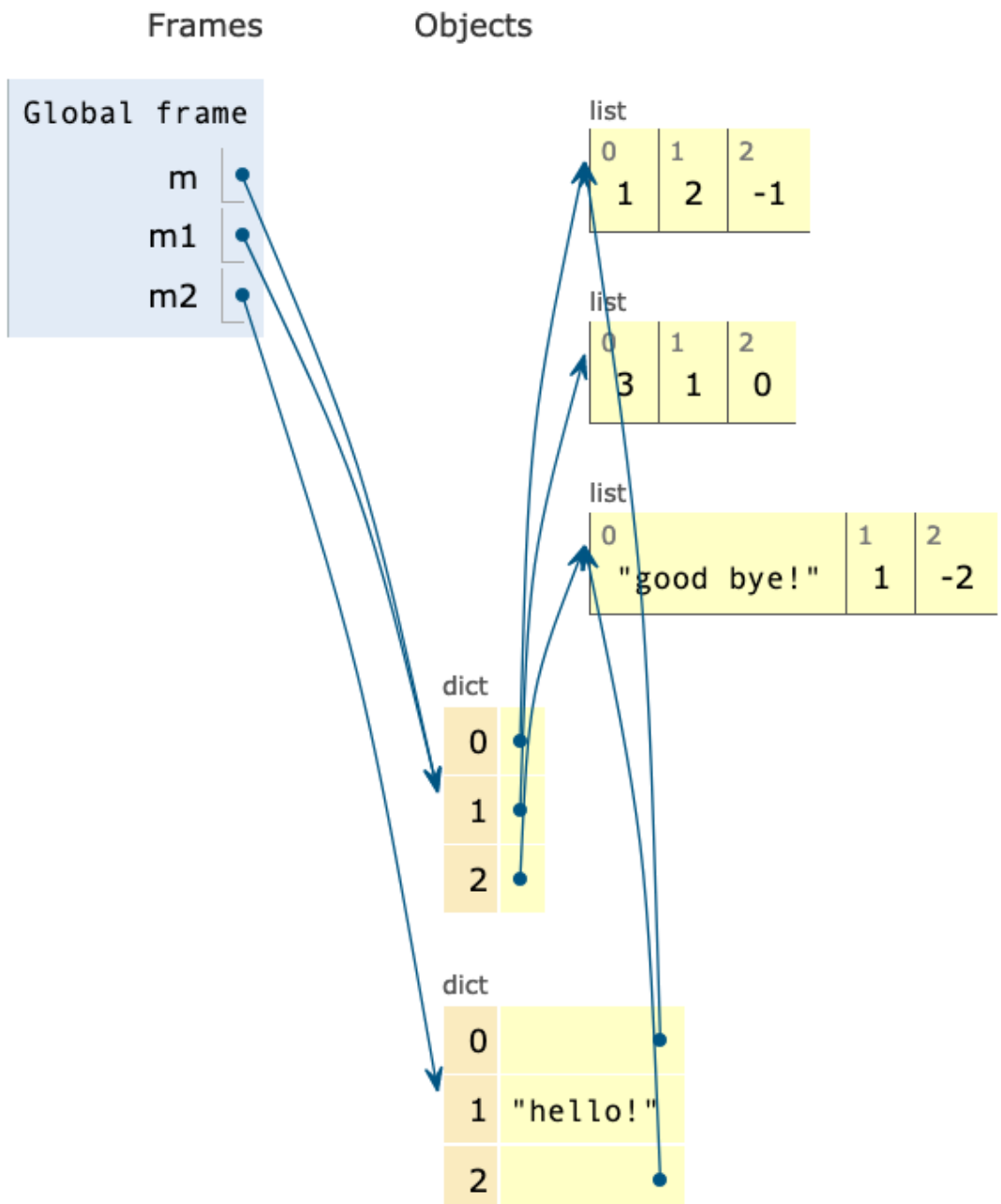
Cópia superficial

```
m = {0:[1,2,-1],
     1:[3,1,0],
     2:[0,1,-2]}

m1 = m

m2 = m.copy()
```



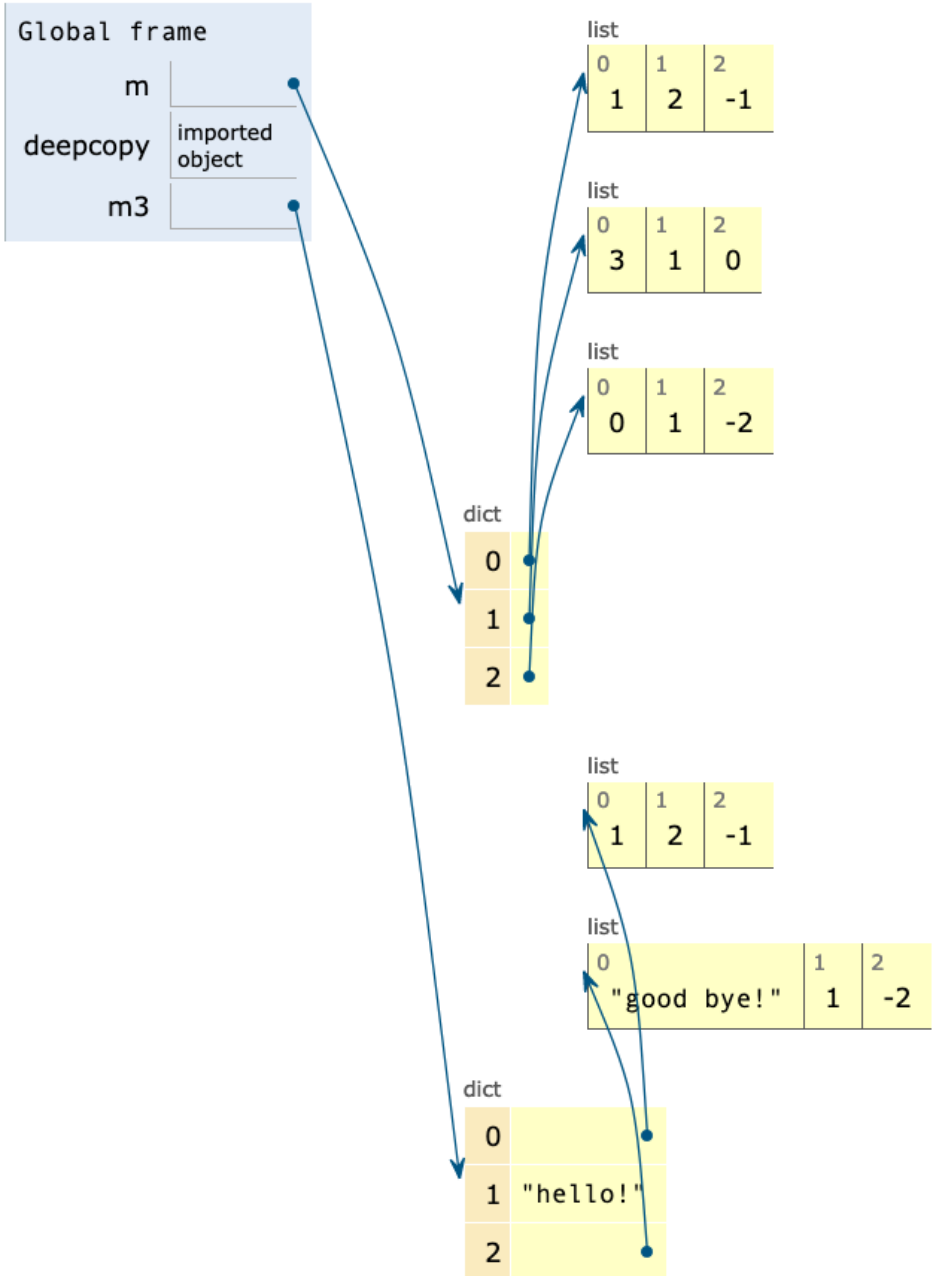
Cópia superficial

```
m = {0:[1,2,-1],
     1:[3,1,0],
     2:[0,1,-2]}

m1 = m

m2 = m.copy()
m2[1] = "hello!"
m2[2][0] = "good bye!"
```



Cópia profunda

```
m = {0:[1,2,-1],
     1:[3,1,0],
     2:[0,1,-2]}

from copy import deepcopy
m3 = deepcopy(m)
m3[1] = "hello!"
m3[2][0] = "good bye!"
```



Noções estudadas

- funções e métodos de dicionários
- dicionários em compreensão
- cópias de objetos

Próxima aula

- Tipos de dados do Python: exemplos, exercícios