Fundamentos de Programação
@ LEIC/LETI

Semana 8

Dicionários

O tipo dicionário. Exemplos. Frequência de letras num texto. Dicionários de dicionários.

Alberto Abad, Tagus Park, IST, 2018

O tipo Dicionário em Python

- Os dicionários, também conhecidos como mapas (*mapping*), são entidades formadas por um conjunto de pares *chave/valor*, em que cada chave é associada a um valor.
  - Outros nomes: Associative arrays (Perl/PHP), HashMaps (Java).
- Em Python existe o tipo `dict`, um tipo **mutável** que representa um dicionário. Em BNF:

```
<dicionário> ::= {} | {<pares>}
<pares> ::= <par> | <par>, <pares>
<par> ::= <chave> : <valor>
<chave> ::= <expressão>
<valor> ::= <expressão> | <tuplo> | <lista> | <dicionário>
```

- Em Python os dicionários são como listas em que:
  - Os elementos também são referenciados por indexação, mas são acedidos por chave
  - As chaves tem de ser de um tipo imutável (e são únicas)
  - Ao contrário das listas, os elementos de um dicionário não estão ordenados

Dicionários em Python - Exemplos

```
>>> vazio = {} # definir dicionarios vazios
>>> outrovazio = dict()
>>> notasFP = {'ist40000':14, 'ist40001':9,  'ist40002':17} #init dicionarios
>>> notasFP['ist40000']  #indexar elementos
>>> notasFP['ist40008']  #e se não existem!?
>>> notasFP['ist40001'] += 1 #alterar elementos

>>> notasFP['ist40001'] = [9, 10] # o tipo dos valores heterogéneos
>>> notasFP['ist40001'][0]
>>> notasFP['ist40007'] = 12 #inserir novos elemetos
>>> notasFP[('ist40008', 'Primeiro exame')] = 13  # tipo chaves hegerogéneo
>>> outrasnotas = dict((('ist40000', 14),('ist40001',9),('ist40002',17)))
```

In [ ]:
```
# notasFP = {'ist40000':14, 'ist40001':9,  'ist40002':17}
# notasFP['ist40004'] = [7, 13]
outrasnotas = dict((('ist40000', 14),('ist40001',9),('ist40002',17)))
outrasnotas
```

Operações básicas com dicionários

Operação	*Tipo dos argumentos*	*Valor*
`del(`d`[`e`])`	Elemento de dicionário	Remove do dicionário d o elemento com índice e.
c `in` d	Chave e dicionário	`True` se a chave c pertence ao dicionário d; `False` em caso contrário.
c `not in` d	Chave e dicionário	A negação do resultado da operação c `in` d.
`len(`d`)`	Dicionário	O número de elementos do dicionário d.

Tabela 7.1: Operações sobre dicionários em Python.

Operações básicas com dicionários - Exemplos

```
>>> notasFP = {'ist40000':14, 'ist40001':9,  'ist40002':17}
>>> 'ist40000' in notasFP # True or False?
>>> 14 in notasFP         # True or False?
>>>  del notasFP['ist40000']  #apagar elementos
>>> 'ist40000' in notasFP
>>> 'ist40000' not in notasFP
>>> len(notasFP)
```

In [ ]:
```
notasFP = {'ist40000':14, 'ist40001':9,  'ist40002':17}
del notasFP['ist40000']
notasFP
'ist40000' in notasFP
len(notasFP)
```

Outras operações e métodos

- Iterar um dicionário (sobre as chaves):

```
for key in notasFP:
  print(key, " --> ", notasFP[key])
```

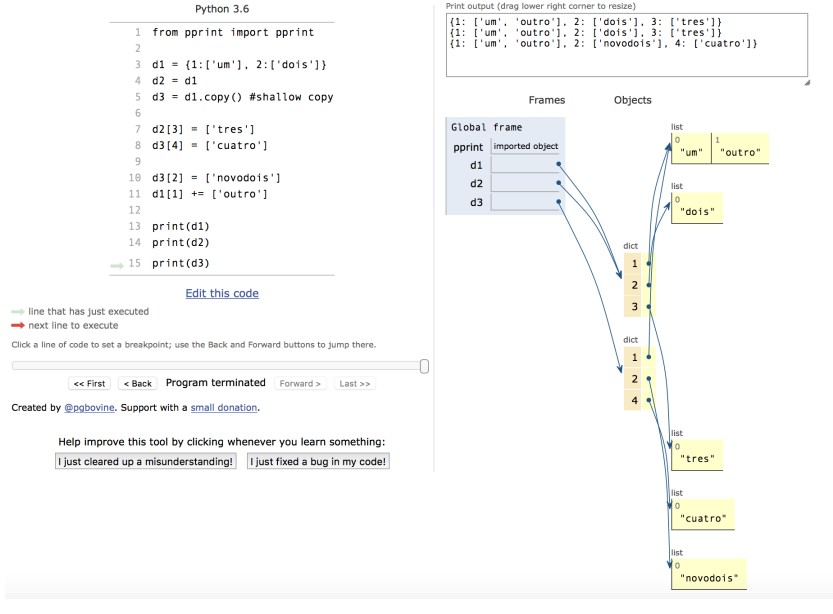
- `notasFP.keys()`, `notasFP.values()` e `notasFP.items()`
- Outros: `notasFP.clear()`, `notasFP.get('ist1000')`, `dict.fromkeys(('ist2000', 'ist3000'), 0)`
- Ver mais info no `help(dict)` ou em https://jeffknupp.com/blog/2015/08/30/python-dictionaries/ (https://jeffknupp.com/blog/2015/08/30/python-dictionaries/)

In [ ]:
```
notasFP.get('ist4', 0)
```

Dicionários

Sobre a mutabilidade dos dicionários

http://pythontutor.com/visualize.html (http://pythontutor.com/visualize.html)



Exemplo 1: Reverse Lookup

- Encontrar a chave que corresponde a um valor:
  - Pode exisitir mais de uma e então voltamos uma qualquer
  - Pode não existir `LookupError`

In [ ]:
```
def reverse_lookup(d, value):
    lista = []
    for e in d:
        if d[e] == value:
            lista.append(e)
    return lista
d = {'1':'um', '2':'dois', '3':'tres', '4':'dois'}
reverse_lookup(d,'cinco')
```

Exemplo 2: Contagem de símbolos/letras

- Programa que conta o número de ocorrências de cada símbolo em uma sequência de caracteres.
- Alterar para não diferenciar minúsculas e maiúsculas.
- Alterar para ignorar pontuação e espaços em branco.
- Mostrar o resultado, em que ordem aparece?

In [ ]:

```
import string
def symbolstable(s):
    table = {}

    toignore = string.punctuation + string.whitespace

    s = s.lower()

    for c in s:
        if c not in toignore:
            table[c] = 1 if c not in table else table[c] + 1

    return table


res = symbolstable("Sed ut perspiciatis unde omnis iste natus error sit voluptatem
 accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inv
entore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim
 ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia conseq
uuntur magni dolores eos qui ratione voluptatem sequi nesciunt.")
for k in res:
    print(k, '-->', res[k])
```

Exemplo 2 cont.: Contagem de símbolos/letras

- Mostrar/printar o resultado, em que ordem aparecem?
- Como mostrar em ordem?
- Como contar o total de símbolos?
- Como alterar (ou gerar um novo dicionário) com as frequências?
- Como mostrar em ordem de frequência?

In [ ]:

```
from functools import reduce
res = symbolstable("Sed ut perspiciatis unde omnis iste natus error sit voluptatem
 accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inv
entore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim
 ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia conseq
uuntur magni dolores eos qui ratione voluptatem sequi nesciunt.")

# How can we print the results in alphabetical order?
for k in sorted(res, reverse=True):
    print(k, '-->', res[k])

# Count the total number of symbols
# soma = 0
# for v in res.values():
#     soma += v
soma = reduce(lambda x, y: x+y, res.values())



# Create a new dict (or modify the original one) to store the frequencies
# newres = {}
# for k in res:
#    newres[k] = res[k]/soma
newres = dict([(k,v/soma) for k, v in res.items()])


# help(sorted)
# How can we print the results sorted by  frequency?
for k, v in sorted(newres.items(), key=lambda x: -x[1]):
    print(k, '-->', v)

# How can we print the results sorted by fequency, with a minumum value?
for k, v in filter(lambda x: x[1] > 0.05, sorted(newres.items(), key=lambda x: -x[
1])):
    print(k, '-->', v)
```

Exemplo 3: Contagem de palavras

In [198]:

```
import string

def wordstable(s):
    table = {}

    toignore = string.punctuation + string.whitespace

    s = s.lower()

    # if the string does not end with a punctuation/whitespace, I add one
    # this is done to guarantee that I don't miss the last word
    if s[-1] not in toignore:
        s += '.'

    start = 0 #store the position where the current word starts
    for i in range(len(s)):
        if s[i] in toignore: ## search for possible word ending
            if i > start: # if it is only one caracter, it must be a punctuation/w
hitespace symbol
                          # and needs to be escaped
                          # otherwise, it is a valid word to add to our table
                table[s[start:i]] = 1 if s[start:i] not in table else table[s[star
t:i]] + 1
            start = i + 1 # reset the start of next word


    return table


res = wordstable("Sed ut perspiciatis unde omnis iste natus error sit voluptatem a
ccusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inven
tore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ip
sam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequun
tur magni dolores eos qui ratione voluptatem sequi nesciunt.")
print(res)
```

```
{'sed': 2, 'ut': 1, 'perspiciatis': 1, 'unde': 1, 'omnis': 1, 'iste': 1, 'natu
```

Exemplo 4: Dicionário de dicionários

- Representar fichas académicas de alunos duma universidade: número do aluno (index), nome (primeiro nome e apelido) e disciplinas que frequentou, contendo ano(s) letivo(s) e classificação.

In [199]:
```
from pprint import pprint
alunos = {1000:
          {'nome': {'primeiro nome': 'John', 'apelido': 'Boy'},
          'disc': {'FP': {'2017-2018': 8, '2018-2019': 13}, 'IEI':{'2017-2018': 17
}}}}
alunos[1001] = {'nome': {'primeiro nome': 'Sarah', 'apelido': 'Girl'},
          'disc': {'FP': {'2017-2018': 18}, 'IEI':{'2017-2018': 15}}}
alunos[1000]['disc']['FP']
```

Out[199]:
```
{'2017-2018': 8, '2018-2019': 13}
```

- Função que recebe estrutura como a anterior e uma pauta e insere:
  - Ex: `pauta = ('PO', '2018-2019', ((1000,'RE'), (1001,15)))`

In [202]:

```
def insere_notas(alunos, pauta):
    disc = pauta[0]
    ano = pauta[1]

    for aluno, nota in pauta[2]:
        if aluno not in alunos:
            raise LookupError("o aluno não existe")
        if disc not in alunos[aluno]['disc']: # primeira inscricao
            alunos[aluno]['disc'][disc] = {ano : nota}
        else:
            alunos[aluno]['disc'][disc][ano] = nota

pauta = ('PO', '2018-2019', ((1000,'RE'),(1001,15)))
insere_notas(alunos, pauta)
pprint(alunos)
```

```
{1000: {'disc': {'FP': {'2017-2018': 8, '2018-2019': 13, '2019-2020': 'RE'},
                 'IEI': {'2017-2018': 17},
                 'PO': {'2018-2019': 'RE'}},
        'nome': {'apelido': 'Boy', 'primeiro nome': 'John'}},
 1001: {'disc': {'FP': {'2017-2018': 18, '2019-2020': 15},
                 'IEI': {'2017-2018': 15},
                 'PO': {'2018-2019': 15}},
        'nome': {'apelido': 'Girl', 'primeiro nome': 'Sarah'}}}
```

Exemplo 5: Números de Fibonacci com memória

In [206]:
```
def fib(n):
    if n == 0 or n ==1:
        return n
    else:
        return fib(n-1) + fib(n-2)

def fib_mem(n):
    memo = {0:0, 1:1}
    def fib_aux(n):
        if n in memo:
            return memo[n]
        else:
            res = fib_aux(n-1) + fib_aux(n-2)
            memo[n] = res
            return res

    return fib_aux(n)

print(fib(20))
print(fib_mem(20))
```

```
6765
6765
```

In [205]:
```
%timeit fib(20)
%timeit fib_mem(20)
```

```
2.47 ms ± 74.6 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
6.44 µs ± 19.8 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```