Dizionari e Tabelle di dati

Informatica@SEFA 2018/2019 - Laboratorio 6

Massimo Lauria <massimo.lauria@uniroma1.it>
http://massimolauria.net/courses/infosefa2018/

Lunedì, 26 Novembre 2018

Dizionari

Dizionari

Il dizionario è una struttura dati che

- contiene coppie **chiave-valore**
- data una **chiave**, forniscce il corrispondente **valore**

Caratteristiche:

- dalla **chiave** si accede al **valore**, non viceversa
- non ci sono due coppie con la stessa **chiave**

Implementazione di un dizionario

Si potrebbe realizzare un dizionario come una sequenza di coppie, ordinate rispetto alla chiave.

```
[('Ferrari','rosso'), ('Jaguar','verde'), ('Panda','bianco')] 1
```

- ricerca di un elemento $O(\log(n))$
- inserimento/cancellazione di un elemento $O(n)$

Ma esiste un modo per cui

- ricerca di un elemento $O(1)$
- inserimento/cancellazione di un elemento $O(1)$
- (necessità di ristrutturare di tanto in tanto)

Dizionari in Python

Python fornisce dizionari efficienti. La sintassi è

```
{ key1 : val1 , key2 : val2 , key3 : val3, key4 : val4 }
```

- delimitato da parentesi graffe
- ogni coppia chiave-valore è separata i due punti
- le coppie sono separate da virgole

```
{'Ferrari':'rossa', 'Jaguar':'verde', 'Panda':'bianco'}
```

Esempio

```
# dizionario con chiavi-valori                                    1
rubrica = { 'Sergio': '123456', 'Bruno': '654321' }               2
print(rubrica)                                                    3
print(type(rubrica))                                              4
                                                                  5
# accesso ad una chiave                                           6
print( rubrica['Sergio'] )                                        7
```

```
{'Sergio': '123456', 'Bruno': '654321'}
<class 'dict'>
123456
```

Accesso agli elementi del dizionario

Sintassi simile a quella degli indici di lista

- ma invece di posizioni, si usano le chiavi

```
dizionario = { 'maria': 84834, (23,"verde"): [1,2,3],          1
               100 : 'Ferrari' }                               2
                                                               3
print( dizionario[ 'maria'       ])                            4
print( dizionario[ (23,'verde')  ])                            5
print( dizionario[ 50 + 50  ])                                 6
```

```
84834
[1, 2, 3]
Ferrari
```

Chiavi ammissibili

Liste non sono ammissibili

- Numeri
- Stringhe
- Tuple di espressioni ammissibili

```
d1 = { (12,("casa",'tetto'),12.4) : "dati importantissimi" }  1
                                                              2
d2 = { (12,("casa",'tetto'),[1,2,3]) : "dati inutili" }       3
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    d2 = { (12,("casa",'tetto'),[1,2,3]) : "dati inutili" }
TypeError: unhashable type: 'list'
```

Chiave assente

Se si tenta di accedere al valore di una chiave che non è nel dizionario si ottiene l'errore `KeyError`

```
d = { 'gatto' : 'miao', 'cane': 'bau'}                    1
print(d['gatto'])                                         2
print(d['topo'])                                          3
```

```
miao
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    print(d['topo'])
KeyError: 'topo'
```

Operazioni su un dizionario

```
voti = {}          # dizionario vuoto                          1
print('#chiavi',len(voti), voti)                             2
                                                             3
voti['Rossi'] = 23      # 'Rossi' è una nuova chiave         4
print('#chiavi',len(voti), voti)                             5
                                                             6
voti['Bianchi'] = 19    # 'Bianchi' è una nuova chiave       7
print('#chiavi',len(voti), voti)                             8
                                                             9
voti['Bianchi'] = 26    # 'Bianchi' esiste già               10
print('#chiavi',len(voti), voti)                             11
                                                             12
voti.pop('Rossi')       # eliminiamo il valore per 'Rossi'   13
print('#chiavi',len(voti), voti)                             14
```

```
#chiavi 0 {}
#chiavi 1 {'Rossi': 23}
#chiavi 2 {'Rossi': 23, 'Bianchi': 19}
#chiavi 2 {'Rossi': 23, 'Bianchi': 26}
#chiavi 1 {'Bianchi': 26}
```

Test di appartenenza

```
voti = {'Rossi': 23, 'Bianchi': 26}

print('Rossi' in voti)

print('Bianchi' not in voti)

if 'Bianchi' in voti:
    voti.pop('Bianchi')

print(voti)
print('Bianchi' in voti)
```

```
True
False
{'Rossi': 23}
False
```

Domanda: che succede se cancellate una chiave che non è nel dizionario?

Iterazione sulle chiavi

```
rubrica = {'Mario': '112233', 'Bruno': '654321', 'Sergio': '
    123456'}

for chiave in rubrica:
    print(chiave, rubrica[chiave])
```

```
Mario 112233
Bruno 654321
Sergio 123456
```

```
rubrica = {'Mario': '112233', 'Bruno': '654321', 'Sergio': '
    123456'}

for chiave in rubrica.keys():
    print(chiave, rubrica[chiave])
```

```
Mario 112233
Bruno 654321
Sergio 123456
```

Altre iterazioni

```
rubrica = {'Mario': '112233', 'Bruno': '654321', 'Sergio': '
    123456'}

for valore in rubrica.values():
    print(valore)
```

```
112233
654321
123456
```

```
rubrica = {'Mario': '112233', 'Bruno': '654321', 'Sergio': '
    123456'}

for chiave,valore in rubrica.items():
    print(chiave,valore)
```

```
Mario 112233
Bruno 654321
Sergio 123456
```

I dizionari non hanno un ordine specifico

- cambia durante l'esecuzione del programma
- **insiemi** di coppie chiave-valore

```
reddito1 = { 'Marta': 200000, 'Luisa': 250000 }
reddito2 = { 'Luisa': 250000, 'Marta': 200000 }

print(reddito1 == reddito2)

for k in reddito1:
    print('reddito1',k)

for k in reddito2:
    print('reddito2',k)
```

```
True
reddito1 Marta
reddito1 Luisa
reddito2 Luisa
reddito2 Marta
```

Esercizio: manipolazione di tabelle

Struttura di una tabella

codice	esercizio	percentuale voto
A1001	calcolo dello sconto	10
A1002	formattazione strighe	20
A1003	ghms2	20
A1004	operazioni su tabelle	15
A1005	sequenze crescenti	35

Una tabella di N colonne può essere realizzata attraverso un dizionario on N chiavi (e.g. `codice`, `esercizio`, `percentuale voto`). Dove ad ogni chiave è associata una lista di valori.

Nota: le liste devono avere la stessa lunghezza.

Esempio

```
tabella={}                                                          1
tabella['codice']=["A1001"   ,"A1002", "A1003" ,"A1004", "          2
    A1005"]
tabella['esercizio']=[                                              3
    "calcolo dello sconto",                                         4
    "formattazione strighe",                                        5
    "ghms2",                                                        6
    "operazioni su tabelle",                                        7
    "sequenze crescenti"]                                           8
tabella['percentuale voto']=[10,20,20,15,35]                        9
print(tabella)                                                      10
```

```
{'codice': ['A1001', 'A1002', 'A1003', 'A1004', 'A1005'],
 'esercizio': ['calcolo dello sconto', 'formattazione strighe', 'ghms2',
               'operazioni su tabelle', 'sequenze crescenti'],
 'percentuale voto': [10, 20, 20, 15, 35]}
```

Esercizio 10: tabella ben formata

La funzione `benformata` deve restituire **True** quando simultaneamente (a) l'input è tipo `dict`; (b) tutti i valori sono di tipo `list`; (c) queste liste hanno la stessa lunghezza. Restituisce **False** altrimenti.

```
from lab06 import benformata                                    1
                                                                2
print(  benformata({ 'A' : [1,2,3], 'B': [4,5,6]}) )            3
                                                                4
print(  benformata({ 'A' : [1,2,3], 'B': [4,5]})   )            5
                                                                6
print(  benformata('dizionario')      )                         7
                                                                8
print(  benformata({ 'A' : 28, 'B': [4,5,6]})      )            9
```

```
True
False
False
False
```

Esercizio 11: proiezione

Una proiezione di una tabella è una selezione delle sue colonne.

codice	esercizio	percentuale voto
A1001	calcolo dello sconto	10
A1002	formattazione strighe	20
A1003	ghms2	20
A1004	operazioni su tabelle	15
A1005	sequenze crescenti	35

Due esempi di proiezione

codice	percentuale voto
A1001	10
A1002	20
A1003	20
A1004	15
A1005	35

esercizio	percentuale voto
calcolo dello sconto	10
formattazione strighe	20
ghms2	20
operazioni su tabelle	15
sequenze crescenti	35

Esercizio 11: proiezione (2)

Scrivere la funzione `proiezione`, tale che

```
proiezione(tabella,colonne)
```

restituisca una nuova tabella (codificata come dizionario di liste), contentente le colonne scelte. Oltretutto

- `colonne` è una sequenza di chiavi
- solleva `TypeError` se la tabella non è ben formata
- solleva `ValueError` se una delle colonne richieste non è nella tabella.

Esercizio 11: proiezione (esempio)

```
from lab06 import proiezione

tabella={}
tabella['codice']=["A1001"   ,"A1002", "A1003" ,"A1004", "
    A1005"]
tabella['esercizio']=[
    "calcolo dello sconto",
    "formattazione strighe",
    "ghms2",
    "operazioni su tabelle",
    "sequenze crescenti"]
tabella['percentuale voto']=[10,20,20,15,35]

print(proiezione(tabella,['codice','percentuale voto']))
print('')
print(proiezione(tabella,['percentuale voto']))
```

```
{'codice': ['A1001', 'A1002', 'A1003', 'A1004', 'A1005'],
 'percentuale voto': [10, 20, 20, 15, 35]}

{'percentuale voto': [10, 20, 20, 15, 35]}
```

Esercitazione

1. scrivere **un** programma python contenente
   - le funzioni che risolvono gli esercizi
   - nient'altro
   - il file deve chiamarsi `lab06.py`
2. scrivete le vostre funzioni nel file `lab06.py`
3. scaricate il file test `test_lab06.py`
4. eseguite, nella cartella che contiene entrambi,

```
$ python3 test_lab06.py
```

5. migliorate fino a che non ottenete una cosa **COME**

```
....................
----------------------------------------------------------
Ran 20 tests in 0.005s

OK
```