Lezione 1 – I Dizionari

Informatica

19 Aprile 2016

I dizionari: cosa sono?

Un *dizionario* è un tipo *mutabile* come una lista, ma dalle caratteristiche più generali:

- In una lista, gli indici devono essere numeri interi.
- Gli indici di un dizionario possono essere *di qualsiasi tipo immutabile*.

Si può pensare ad un dizionario come ad una *mappa* tra due insiemi:

- il dominio è un insieme di *chiavi*
- il codominio è un insieme di *valori*
- ad ognuna delle chiavi viene associato un valore
- un'associazione è detta *coppia chiave-valore* o anche *elemento* del dizionario

Esempio: il dizionario Italiano-Inglese

- Costruiamo un dizionario che associ parole in Italiano con la loro traduzione in Inglese.
- In questo dizionario sia le chiavi che i valori sono di tipo stringa.

Cosa faremo:

1. costruiamo un dizionario
2. inseriamo gli elementi nel dizionario
3. stampiamo il dizionario completo
4. stampiamo singoli elementi
5. controlliamo se una *chiave* appartiene al dizionario
6. controlliamo se un *valore* appartiene al dizionario

```
>>> ita2eng = dict()
>>> print(ita2eng)
{}
>>> ita2eng['uno'] = 'one'
>>> print(ita2eng)
{'uno': 'one'}
>>> ita2eng = {'uno': 'one', 'due': 'two', 'tre': 'three'}
>>> print(ita2eng)
{'due': 'two', 'uno': 'one', 'tre': 'three'}
>>> print(ita2eng['due'])
two
>>> print(ita2eng['quattro'])
```

```
KeyError: 'quattro'
>>> 'uno' in ita2eng
True
>>> 'one' in ita2eng
False
>>> valori = ita2eng.values()
>>> 'one' in valori
True
```

Contare le lettere

Esercizio

Scrivere un programma che conti quante volte ogni lettera dell'alfabeto appare in una stringa data come input.

Possiamo risolvere questo problema in diversi modi:

1. Possiamo farlo usando solo variabili semplici
2. Possiamo utilizzare una lista
3. Possiamo utilizzare un dizionario

```
def istogramma(s):
    d = dict()
    for c in s:
        if c not in d:
            d[c] = 1
        else:
            d[c] += 1
    return d
```

Esempio di esecuzione

```
>>> i = istogramma('brontosauro')
>>> print(i)
{'r': 2, 't': 1, 'n': 1, 'u': 1, 'o': 3, 'b': 1, 'a': 1, 's': 1}
```

Miglioriamo l'esempio:

1. Stampiamo gli elementi una lettera per riga
2. Stampiamo in ordine alfabetico
3. Stampiamo l'istogramma a barre:

```
>>> i = istogramma('brontosauro')
>>> stampa_barre(i)
a *
b *
n *
o ***
r **
s *
t *
u *
```

```
def stampa_ist(i):
    for c in i:
        print(c, i[c])

def stampa_ord(i):
    chiavi = list(i.keys())
    chiavi.sort()
```

```
    for c in chiavi:
        print(c, i[c])

def stampa_barre(i):
    chiavi = list(i.keys())
    chiavi.sort()
    for c in chiavi:
        print(c, '*'*i[c])

print(" --- ISTOGRAMMA --- ")
i = istogramma("brontosauro")
stampa_ist(i)
print(" --- STAMPA IN ORDINE --- ")
stampa_ord(i)
print(" --- STAMPA BARRE --- ")
stampa_barre(i)
```

Ricerca nei dizionari

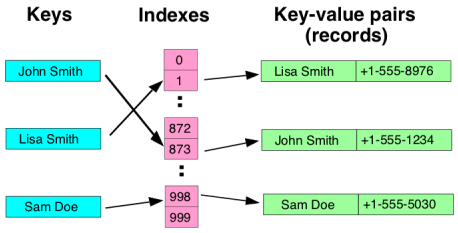
Abbiamo visto che per trovare il valore corrispondente ad una chiave è sufficiente utilizzare le parentesi quadre: `valore = d[chiave]`

- questa operazione si chiama *ricerca* (o *lookup*) in un dizionario

Qual'è la complessità di questa operazione?

- **Liste:** la ricerca impiega un tempo *lineare*
- **Dizionari:** la ricerca impiega tempo (medio) *costante*

Ricerca in tempo costante: tabelle hash



- le chiavi vengono convertite in un intero mediante una *funzione di hash*
- il valore ottenuto (detto *hash*) viene usato come indice di una tabella che contiene le coppie chiave-valore
- perché l'indicizzazione funzioni le chiavi devono essere *immutabili*:
  - stringhe, caratteri, interi, tuple

Ricerca inversa

Ricerca Inversa: dato un valore `v` trovare la chiave `k` corrispondente

Questa operazione pone *due problemi*:

- un valore può essere assegnato a più chiavi diverse
  - restituire solo una chiave? quale?
  - restituire tutte le chiavi?

- Python non mette a disposizione operatori di ricerca inversa:
  - bisogna fare una ricerca nel dizionario

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

print(" --- RICERCA INVERSA --- ")
c = ricerca_inversa(i, 2)
print(c)
c = ricerca_inversa(i, 5)
```

```
def ricerca_tutte(d, v):
    l = []
    for k in d:
        if d[k] == v:
            l.append(k)
    return l

print (" --- RICERCA TUTTE --- ")
chiavi = ricerca_tutte(i, 1)
print(chiavi)
chiavi = ricerca_tutte(i, 5)
print(chiavi)
```

Dizionari e liste
Le *liste* possono essere utilizzate nei dizionari *solo come valori*
Esempio: Invertiamo un dizionario

- Invertire un dizionario significa creare un dizionario che mappa i valori nelle chiavi.
- Poiché un valore può essere associato a più chiavi, il dizionario inverso ha delle *liste* come valori

```
>>> ist = istogramma("pappagallo")
>>> print(ist)
{'g': 1, 'o': 1, 'p': 3, 'l': 2, 'a': 3}
>>> inv = inverti_diz(ist)
>>> print(inv)
{1: ['g', 'o'], 2: ['l'], 3: ['p', 'a']}
```

```
def inverti_diz(d):
    inverso = dict()
    for chiave in d:
        val = d[chiave]
        if val not in inverso:
            inverso[val] = [chiave]
        else:
            inverso[val].append(chiave)
    return inverso
```