Dizionari

Informatica@DSS 2024/2025

Massimo Lauria <massimo.lauria@uniroma1.it>
https://massimolauria.net/informatica2024/

Dizionari

Il dizionario è una struttura dati che

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

Caratteristiche:

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

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],
               100 : 'Ferrari' }

print( dizionario[ 'maria'       ])
print( dizionario[ (23,'verde')  ])
print( dizionario[ 50 + 50  ])
```

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

```
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'}
print(d['gatto'])
print(d['topo'])
```

```
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
```

Differenza di efficienza nella ricerca

Esempio:

- prepariamo 1000000 numeri casuali tra 0 e 100 milioni;
- li carichiamo (a) come chiavi in un dizionario, (b) in una lista non ordinata e (c) in una lista ordinata;
- effettuiamo 1000 ricerche dello zero sulle tre strutture dati.

```
Preparazione dei dati ...
 fatto in 2.134997014 secondi!
Inizio 1000 ricerche nel dizionario ...
 fatto in 0.00010509499999988847 secondi!
Inizio 1000 ricerche sequenziali nella lista ...
 fatto in 13.978430125000001 secondi!
Inizio 1000 ricerche binarie nella lista ordinata ...
 fatto in 0.0044725649999682 secondi!
```

Frequenza delle parole

```
from pprint import pprint as print                                    1
                                                                      2
testo="""la rana in spagna gracida                                    3
         in campagna quanto è felice la rana in spagna"""             4
frequenze = {}                                                        5
                                                                      6
for parola in testo.split():                                          7
    if parola in frequenze:                                           8
        frequenze[parola] +=1                                         9
    else:                                                             10
        frequenze[parola] = 1                                         11
print(frequenze)                                                      12
```

```
{'campagna': 1,
 'felice': 1,
 'gracida': 1,
 'in': 3,
 'la': 2,
 'quanto': 1,
 'rana': 2,
 'spagna': 2,
 'è': 1}
```