dizionari

June 5, 2023

0.1 Dizionari

Un dizionario è un tipo di dato che permette di memorizzare delle coppie (chiave, valore), così che data una potenziale chiave si possa determinare efficententemente il valore corrispondente.

Un esempio banale è una rubrica telefonica:

Nome	Telefono
Mario	11111
Luisa	55555
Fabrizio	33333

Il tipo di dato `dict` permette di memorizzare queste informazioni e permette di rispondere facilmente a domande come: - `Giulio` è in rubrica? - Qual è il numero di `Luisa`?

Vediamo subito un esempio:

[2]:
```
rubrica = { "Mario" : "11111", "Fabrizio": "33333", "Luisa" : "55555" }
```

[3]:
```
print(rubrica)
```

```
{'Mario': '11111', 'Fabrizio': '33333', 'Luisa': '55555'}
```

[4]:
```
print("La rubrica ha", len(rubrica), "nomi." )
```

```
La rubrica ha 3 nomi.
```

[5]:
```
print( rubrica["Luisa"] )
print( rubrica["Mario"] )
```

```
55555
11111
```

Precisiamo la sintassi. Alla riga 1 dell'esempio precedente assegnamo alla variabile `rubrica` un valore di tipo `dict` (il tipo di dati python che realizza il dizionario).

- Un dizionario vuoto si specifica come `{ }`,
- altrimenti come un elenco `{ chiave : valore , chiave: valore , ... }` di coppie `chiave : valore` separate da virgole e contenuto tra graffe.

0.1.1 Verificare se una chiave è nel dizionario: operatore `in`

La caratteristica principale dei dizionari è che la ricerca di una chiave è **estremamente veloce**. Se ho una lista `L` di 10.000.000 di elementi e voglio sapere se contiene un certo valore `x` posso usare l'espressione booleana `x in L`. Per determinare il valore di `x in L` Python scorre tutta la lista fino a che non trova `x`, eseguendo anche fino a 10.000.000 passi.

Invece l'operazione `in` su un dizionario è estremamente **più efficiente**. Nei nostri piccoli esempi questa differenza non si noterà, tuttavia la funzionalità di associare una chiave ad un valore è sufficiente per farci utilizzare questo tipo di dato.

Vediamo un esempio di dizionario con dati più variegati.

[6]:
```
D = {  14 : "quattordici" , 7.3 : "sette virgola tre", "Harry" : "Potter",␣
 ↪"Frodo" : "Baggins"  }
```

[8]:
```
print ( 7.3 in D  )
print ( "Asterix" in D)
print ( 'Frodo' in D)
print ('Potter' in D)  # la ricerca è effettuata sulle chiavi
```

```
True
False
True
False
```

0.1.2 Accedere al valore di una chiave (esistente)

In una sequenza si accede all'elemento i-esimo con la sintassi `seq[i]`. I dizionari **non hanno una natura sequenziale** l'idea che un valore o una chiave siano in una certa posizione di una sequenza è concettualmente sbagliato.

Per accedere alla chiave `key` di un dizionario `D` si una una sintassi simile `D[key]` con l'enorme differenza che `key` può essere quasi qualunque dato, e non solo un numero.

[9]:
```
D = {  7+7 : "quattordici" , 7.3 : "sette " + "virgola tre", "Harry" :␣
 ↪"Potter", "Frodo" : "Baggins"  }
```

[10]:
```
print ( D[14]  )
print ( D[7.3]  )
print ( D['Frodo'])
print ( D['Har' + 'ry'])
```

```
quattordici
sette virgola tre
Baggins
Potter
```

Se la chiave non è presente nel dizionario, un tentativo di accesso maldestro genererà un `KeyError`.

[11]:
```
Capitali = { "Italia" : "Roma" , "Francia" : "Parigi" }
```

[12]:
```
print ( Capitali["Italia"]  )
print ( Capitali["Francia"]  )
print ( Capitali["Jugoslavia"])
```

```
Roma
Parigi
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [12], in <cell line: 3>()
      1 print ( Capitali["Italia"]  )
      2 print ( Capitali["Francia"]  )
----> 3 print ( Capitali["Jugoslavia"])

KeyError: 'Jugoslavia'
```

0.1.3 Modificare il dizionario

Così come le liste, i dizionari possono essere modificati.

- Memorizzare o sovrascrivere il valore associato ad una chiave: `D[key]=val`
- Eliminare una coppia (chiave,valore) : `D.pop(key)`

[17]:
```
D = { 'A' : 'a' , 'B': 'b'}
```

[18]:
```
D['C'] = 'c'
print(D)
```

```
{'A': 'a', 'B': 'b', 'C': 'c'}
```

[19]:
```
D['A'] = 1000
print(D)
```

```
{'A': 1000, 'B': 'b', 'C': 'c'}
```

[20]:
```
D.pop('B')
print(D)
```

```
{'A': 1000, 'C': 'c'}
```

0.1.4 Ciclare sul dizionario

Il dizionario **non è una sequenza**. Le chiavi non hanno un ordine specifico, e l'eventuale ordine può essere modificato senza preavviso. Quindi nel dizionario l'ordine delle chiavi è **irrilevante e mutevole**.

[21]:
```
D = {  7+7 : "quattordici" , 7.3 : "sette " + "virgola tre",
       "Harry" : "Potter", "Frodo" : "Baggins"  }
```

[22]:
```
for chiave in D:
    print( chiave, "-->", D[chiave] )
```

```
14 --> quattordici
7.3 --> sette virgola tre
Harry --> Potter
Frodo --> Baggins
```

0.1.5 Esempi/Esercizi

Scrivere delle funzioni che realizzino i seguenti compiti.

- Data una lista di coppie `[ (key,val), (key,val), ...]` costruite il dizionario corrispondente.

[23]:
```
def costruiredict(coppie):
    D = { }
    for coppia in coppie:
        chiave = coppia[0]
        valore = coppia[1]
        D[chiave] = valore
    return D
```

[24]:
```
L1 = [ ('A','a') , ('B','b'), ('C','c') ]
D1 = costruiredict( L1 )
print(D1)
```

```
{'A': 'a', 'B': 'b', 'C': 'c'}
```

[25]:
```
L2 = [ ('uno','one') , ('casa','house'), ('automobile','car') ]
D2 = costruiredict( L2 )
print(D2)
```

```
{'uno': 'one', 'casa': 'house', 'automobile': 'car'}
```

- Dato un dizionario, costruite un altro dizionario invertendo le corrispondenze `(key,val)` e transformandole in `(val,key)`. Ovvero scambi le chiavi con i valori e viceversa.

[30]:
```
def revert(dizionario):
    Res = {}
    for key in dizionario:
        val = dizionario[key]
        Res[val] = key
    return Res
```

[31]:
```
D1 = {'A': 'a', 'B': 'b', 'C': 'c'}
print( revert(D1) )
```

```
{'a': 'A', 'b': 'B', 'c': 'C'}
```

[32]:
```
D2 = {'uno': 'one', 'casa': 'house', 'automobile': 'car'}
print( revert(D2) )
```

```
{'one': 'uno', 'house': 'casa', 'car': 'automobile'}
```

0.1.6 Riassumendo

Abbiamo visto: - il tipo di dato `dict` che realizza il **dizionario**; - operazioni di **ricerca**, **accesso**; - operazioni di **aggiunta** e **cancellazione**; - come ciclare sulle chiavi del dizionario.