Lezione 2 – I Dizionari

Informatica

21 Aprile 2016

Un esempio di codice inefficiente

Prendiamo come esempio il codice per calcolare la serie di Fibonacci:

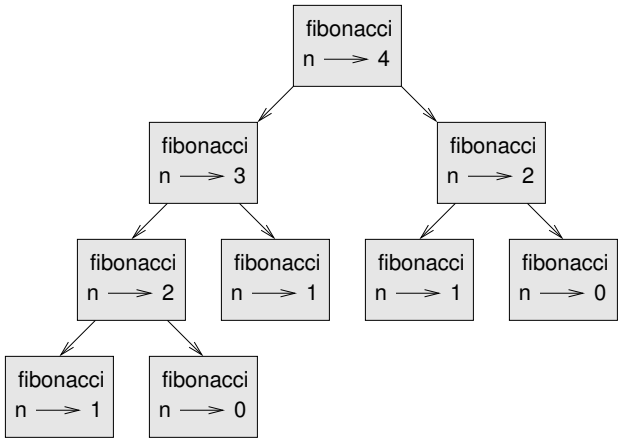
```
def fibonacci(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fibonacci(n-1) + fibonacci(n-2)
```

Questo codice non è molto efficiente: il tempo impiegato per calcolare il risultato cresce molto in fretta

- provate `fibonacci(40)`

Perché `fibonacci` impiega così tanto tempo?

La spiegazione si trova nel *grafo delle chiamate* di `fibonacci`:



La funzione viene eseguita molte volte con lo stesso valore di input!

Usiamo un dizionario per velocizzare `fibonacci`

Per ottenere una soluzione più efficiente possiamo fare come segue:

- teniamo traccia dei valori già calcolati in un dizionario
- i valori memorizzati nel dizionario sono detti *memo*
- ad ogni chiamata, `fibonacci` controlla se l'argomento è presente nel dizionario
- se è già presente, ritorna immediatamente il risultato
- altrimenti, calcola il nuovo valore e aggiunge un memo al dizionario

```
known = {0:0, 1:1}

def fibonacci_memo(n):
    if n in known:
        return known[n]
    res = fibonacci(n-1) + fibonacci(n-2)
    known[n] = res
    return res

print (" --- FIBONACCI MEMO --- ")
f = fibonacci_memo(4)
print(f)
f = fibonacci_memo(10)
print(f)
```

Variabili globali

- In `fibonacci_memo` il dizionario `known` viene creato al di fuori della funzione
  - appartiene al frame speciale chiamato `__main__`
  - le variabili in `__main__` sono dette *globali*
  - sono visibili da qualsiasi funzione
  - permangono tra una chiamata di funzione e la successiva
- le variabili dichiarate all'interno di funzioni sono *locali*
  - vengono distrutte quando la funzione termina

Immutabili globali ed assegnazioni

- Le variabili globali possono essere usate come *flag*:
  - variabili booleane che indicano se una certa condizione è vera
- Per poter riassegnare il valore ad una variabile globale all'interno di una funzione bisogna prima *dichiarare* la variabile globale:

```
def fun():
  global flag
  flag = True
```

- Senza la dichiarazione di `global`, *viene creata una nuova variabile locale con lo stesso nome*

Mutabili globali

- Le variabili mutabili (liste, dizionari) possono essere modificate anche senza dichiarazione:
  - aggiungere, togliere, modificare elementi
- Per essere riassegnate devono invece essere dichiarate `global`:

```
def globale()
  global known
  known = dict()
```

Esempi di uso (corretto e scorretto) di variabili globali:

```
verb = True

def flag():
    if verb:
        print("Flag attivo")

print(" --- verboso --- ")
flag()
verb = False
flag()

been_called = False

def flag2():
    been_called = True

print(" --- flag 2 --- ")
print(been_called)
flag2()
print(been_called)

def flag3():
    global been_called
    been_called = True

print(" --- flag 3 --- ")
been_called = False
print(been_called)
flag3()
print(been_called)
```

Dizionari: operazioni utili

Accesso `d[chiave]` accede/modifica/crea il valore associato a `chiave`

Accesso `d.get(k,default)` restituisce il valore associato alla chiave `k` in `d`, se esiste. Se non esiste restituisce il valore `default`.

Cancellazione `del d[chiave]` elimina la `chiave` ed il valore associato

Chiavi `d.keys()` ritorna la vista dinamica delle chiavi

Valori `d.values()` ritorna la vista dinamica dei valori

Dimensione `len(d)` restituisce il numero di elementi `chiave: valore` presenti nel dizionario

Esercizio 1: Il negozio di libri

Scrivere un programma per gestire il magazzino di un negozio di libri. Il programma deve:

- tenere traccia di quante copie di ogni libro sono presenti in magazzino
- aggiungere una nuova copia di un libro in magazzino
- quando un libro viene venduto, aggiornare il magazzino
- stampare il contenuto del magazzino

```
def aggiungi_libro(lib,mag):
  mag[lib] = mag.get(lib, 0) + 1

def vendi_libro(lib,mag):
  if lib in mag:
    if mag[lib]>1:
      mag[lib]-=1
    elif mag[lib]==1:
        del mag[lib]
    else:
      print("Libro in quantita' non valida")
  else:
    print("Libro non presente in magazzino")

def stampa_mag(mag):
  for lib in mag:
    print(lib,'*'*(mag[lib]),mag[lib])

print(" --- BOOKS ---")
mag=dict()
aggiungi_libro('Siddharta',mag)
aggiungi_libro('Siddharta',mag)
aggiungi_libro('Narciso e Boccadoro',mag)
aggiungi_libro('Io e Dio',mag)
vendi_libro('Io e Dio',mag)
stampa_mag(mag)
```

Esercizio 2: Rimuovere le lettere più frequenti

Scrivere una funzione che riceve in input una stringa e rimuove tutte le occorrenze dei caratteri a frequenza più alta.

```
def remove_most_frequent(a):
  b=dict()
  for e in a:
    b[e]=b.get(e,0)+1
  c=0
  for e in b:
    if b[e]>c:
      c=b[e]
  d=""
  for e in a:
    if b[e]<c:
      d=d+e
  return d

print(" --- REMOVE_MOST_FREQUENT --- ")
print(remove_most_frequent("aab"))
print(remove_most_frequent("babbaaccdabaa"))
```

Esercizio 3: L'alfabeto carbonaro

Durante i Moti del 1830-1831, gli aderenti alla Carboneria utilizzavano un *cifrario a sostituzione* per scambiarsi messaggi in codice.

Ogni lettera del messaggio da cifrare veniva sostituita da una lettera diversa, secondo una *tavola di accoppiamento*:

```
A|B|C|D|E|F|G|H|I|L|M|N|O|P|Q|R|S|T|U|V|Z
O|P|G|T|I|V|C|H|E|R|N|M|A|B|Q|L|Z|D|U|F|S
```

1. Utilizzare un dizionario per rappresentare la tavola di accoppiamento (*chiave*)
2. Scrivere una funzione per cifrare un testo
3. Scrivere una funzione per decifrare un testo

```
def cifra(chiave, testo):
    t = testo.lower()
    ris = ""
    for c in t:
        if c in chiave:
            ris = ris + chiave[c]
        else:
            ris = ris + c
    return ris

def inverti(chiave):
    ris = dict()
    for c in chiave:
        val = chiave[c]
        ris[val] = c
    return ris

def decifra(chiave, testo):
    chinv = inverti(chiave)
    return cifra(chinv, testo)

print(" --- CIFRARIO CARBONARO --- ")
chiave = {'a':'o', 'b':'p', 'c':'g', 'd':'t', 'e':'i', 'f':'v', 'g':'c',
          'h':'h', 'i':'e', 'l':'r', 'm':'n', 'n':'m', 'o':'a', 'p':'b',
          'q':'q', 'r':'l', 's':'z', 't':'d', 'u':'u', 'v':'f', 'z':'s'}
chiaro = "Preferisco la macchina alla moto."
print(chiaro)
msg = cifra(chiave, chiaro)
print(msg)
chiaro = decifra(chiave, msg)
print(chiaro)
```

Esercizio 4: Hapax

In linguistica, un *hapax* è una forma linguistica (parola o espressione), che compare una sola volta nell'ambito di un testo.

- scrivere una funzione che ritorni la lista di tutte le parole che compaiono una sola volta in un testo

```
def hapax(s):
    freq = dict()
    parole = s.split()
    for p in parole:
        freq[p] = freq.get(p,0) + 1
    l = []
    for p in freq:
        if freq[p] == 1:
            l.append(p)
    return l

print(" --- HAPAX --- ")
print(hapax("La mattina seguente Don Rodrigo si destò Don Rodrigo"))
```

Esercizio 5: l'elenco telefonico

Le *tuple* possono essere utilizzate nei dizionari sia come *chiavi* che come *valori*:

- esempio: un elenco telefonico è un dizionario che associa tuple `(nome, cognome)` a tuple `(prefisso, numero_tel)`

Scriviamo una serie di funzioni per gestire un elenco telefonico:

1. una funzione per aggiungere un nuovo elemento
2. una funzione per stampare l'elenco telefonico
3. una funzione per cercare un numero telefonico
4. una funzione per cercare il nome associato ad un numero

```
def inserisci(e, cognome, nome, prefisso, tel):
    if (cognome, nome) in e:
        raise ValueError
    else:
        e[cognome, nome] = (prefisso, tel)

def cerca_tel(e, cognome, nome):
    if (cognome, nome) in e:
        return e[cognome, nome]
    else:
        raise ValueError

def cerca_num(e, prefisso, tel):
    for (cognome, nome) in e:
        if e[cognome,nome] == (prefisso, tel):
            return e[cognome, nome]
    raise ValueError

def stampa(e):
    for (cognome, nome) in e:
        (pref, tel) = e[cognome,nome]
        print(cognome, nome, pref, tel)
```

L'elenco telefonico: esempio di esecuzione

```
>>> elenco = dict()
>>> inserisci(elenco, "Rossi", "Mario", 555, 123456)
>>> inserisci(elenco, "Bianchi", "Luigi", 555, 666666)
>>> stampa(elenco)
Rossi Mario 555 123456
Bianchi Luigi 555 666666
>>> (pref,tel) = cerca_tel(elenco, "Rossi", "Mario")
>>> print(pref, tel)
555 123456
>>> (cognome, nome) = cerca_num(elenco, 555, 666666)
>>> print(cognome, nome)
555 666666
```