Esercitazione di Programmazione

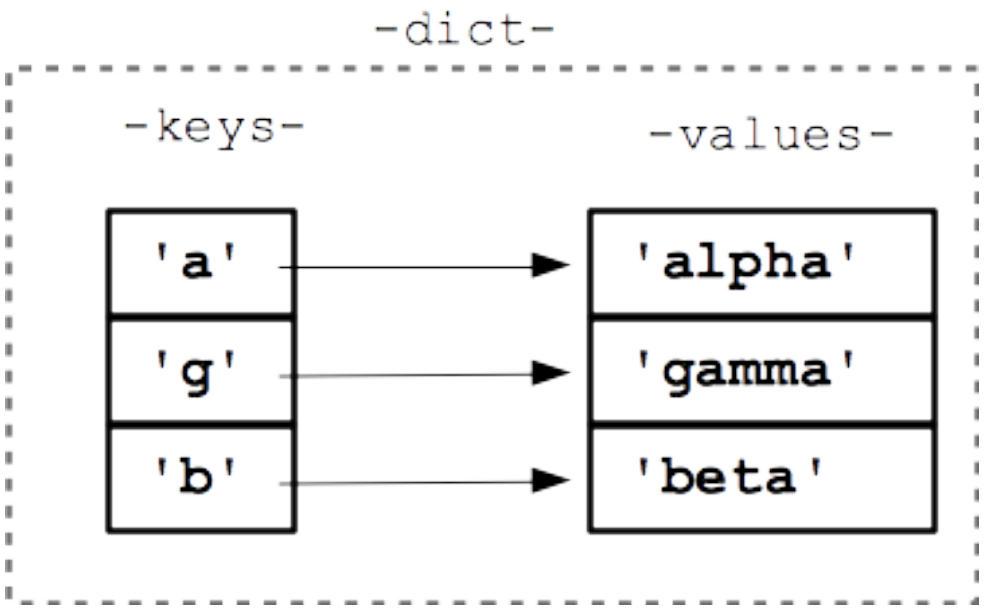
Michele Beretta

michele.beretta@unibg.it

Gruppo A	Gruppo B/C	Argomenti
12/03	13/03	Variabili, tipi base, operazioni e funzioni standard di I/O
17/03	20/03	Uso di funzioni e metodi, uso stringhe, conversioni di tipo ed espressioni booleane
16/04	17/04	Costrutto condizionale: `if-elif-else`, strutture iterative `while` e `for`
23/04	24/04	Esercizi su `if-elif-else`, uso dei cicli `while` e `for` (`break`, `continue` ed `else`)
07/05	08/05	Esercizi sui cicli `while` e `for`
14/05	15/05	Liste (introduzione ed esercizi)
21/05	22/05	Moduli e funzioni (introduzione ed esercizi)
27/05	29/05	File, complementi sulle stringhe (introduzione ed esercizi)
04/06	05/06	Dizionari ed esercizi di ripasso

Uso di base dei dizionari

Un dizionario è un'associazione fra **chiavi** e **valori**.



Vedere il file `introdict.py`.

Esempio uso dei dizionari

Scrivere un programma che acquisisca da tastiera un numero positivo `N`, successivamente acquisire i dati anagrafici di `N` persone: nome, cognome, età, città di residenza. I dati anagrafici di ciascun individuo dovranno essere strutturati in una variabile di tipo dizionario (una variabile distinta per ogni persona) con chiavi: `'nome'`, `'cognome'`, `'eta'`, `'citta di residenza'`.

I dizionari dovranno essere strutturati come elementi di una lista, che sarà costruita iterativamente per concatenazione a partire da una lista vuota.

Il programma dovrà visualizzare a video l'età media delle persone inserite e il nome della città che compare più spesso nella lista.

Strutturare il codice prevedendo le seguenti tre funzioni:

```
# Ritorna un dizionario con le informazioni di una persona
acquisisci_persona()

# Ritorna l'età media in una lista di dizionari
calcola_eta_media(listaDiDizionari)

# Ritorna la stringa con il nome della città
# di residenza ripetuta più spesso nella lista passata come parametro
calcola_citta_piu_comune(listaDiDizionari)
```

Esercizio 1

Scrivere un programma che acquisisca le coordinate di `N > 1` punti nel piano cartesiano (`N` deve essere acquisito precedentemente).

Le coordinate di ciascun punto saranno memorizzate in un dizionario avente chiavi `'x'` e `'y'` e i dizionari saranno a loro volta memorizzati all'interno di una lista.

Il programma deve calcolare e visualizzare a video la somma delle distanze di ciascun punto dal successivo e sommare a tale quantità anche la distanza dell'ultimo punto dal primo. Strutturare il programma mediante l'uso di funzioni.

Esercizio 2

Si scriva un sottoprogramma che:

- Acquisisca una stringa `s` dall'utente, con la funzione `lettere`; tale funzione deve creare una variabile di tipo dizionario, contenente come chiavi le lettere di `s` e come valori le occorrenze di tali lettere in `s` e restituirla al chiamante
- Visualizzare a monitor il dizionario creato
- E successivamente per ogni coppia chiave-valore visualizzare la chiave seguita dal valore tra parentesi tonde

Ad esempio, acquisendo la stringa `'casa'` dev'essere visualizzato

```
{'c': 1, 'a': 2, 's': 1}
c ( 1 )
a ( 2 )
s ( 1 )
```

Esercizio 3.1

Scrivere un programma che permetta di riempire una lista di studenti.

Ogni studente è un dizionario che contiene nome, cognome, matricola e una lista di esami superati.

Il programma deve fornire un menu da cui l'utente può scegliere se

- Inserire un altro studente
- Stampare tutti gli studenti inseriti
- Terminare l'esecuzione

Strutturare il programma mediante l'uso di funzioni.

Esercizio 3.2

Si completi l'esercizio 3.1 aggiungendo le seguenti funzionalità, implementate tramite un menù:

- Ogni studente anziché mantenere una lista di esami superati mantiene un dizionario di esami superati (chiavi) e voto per quell'esame (valore). Ad esempio: `{'analisi': 30, 'fisica': 2}`.
- Si deve offrire all'utente anche la possibilità di aggiornare la lista esami/voti per gli studenti con la funzione `aggiorna_voto_esame`, solo se l'esame è già presente tra quelli sostenuti dallo studente e se il nuovo voto è ≥ 18 e ≤ 30.
- Si deve poter stampare la media dei voti di uno studente con la funzione `media_voti`.

Esercizio 1 di ripasso (per casa)

Definire una funzione che riceva come argomenti due liste di numeri interi, ognuna delle quali si assume essere composta da numeri tutti differenti, e restituisca la stringa

- `'identiche'` se le liste sono identiche
- `'uguali ma in ordine differente'` se contengono gli stessi elementi ma in ordine differente
- `'diverse'` in tutti gli altri casi

Se le due liste sono diverse il programma dovrà creare una nuova lista contenete i valori di entrambe senza ripetizioni e in ordine crescente e dovrà memorizzare quest'ultima lista in un file `elementi_ordinati.txt` contenete un valore per ogni rigo.

Suggerimenti:

- per verificare se due liste sono `'identiche'` si può utilizzare l'operatore `==`
- due liste di lunghezza diversa non possono contenere esattamente gli stessi elementi, neanche in ordine differente
- per verificare se un elemento di una lista sia presente in un'altra (in posizione qualsiasi) si può utilizzare l'operatore `in`
- se due liste hanno la stessa lunghezza e *tutti* gli elementi di una di esse si trovano nell'altra si può concludere le liste siano `'uguali ma in ordine differente'`; se anche *uno solo* degli elementi di una non si trovasse nell'altra si può invece concludere che le liste siano `'diverse'`

Esercizio 2 di ripasso (per casa)

Si scriva un programma dove l'utente inserisce una lista di interi (l'inserimento termina quando l'utente inserisce `0`), e da questa vengono create due nuove liste: una contenente gli interi pari e una contenente i dispari.

- Stampare la lunghezza delle due liste e stampare qual è la lista con più elementi (`'pari'` o `'dispari'`)
- Sommare tutti gli interi delle due liste, stampare le somme e stampare qual è la lista con la somma maggiore
- Stampare infine le liste ordinate come due colonne di un file di testo chiamato `liste_ordinate.txt`

Strutturare il programma in sottoprogrammi.