Laboratorio 03

Programmazione - CdS Matematica

Marta Gatto

10 novembre 2015

Dizionari

Un dizionario è

- Contenitore di coppie: $\langle \textit{chiave}, \textit{valore} \rangle$
- **Non** prevede alcun ordinamento

A cosa può servire

- Iterare sulle coppie chiave / valore
- Aggiungere nuove coppie
- Ottenere un valore, data una chiave

Dizionari

Definizioni equivalenti di un dizionario

```
>>> dict1 = dict()
>>> dict1
{}
>>>
```

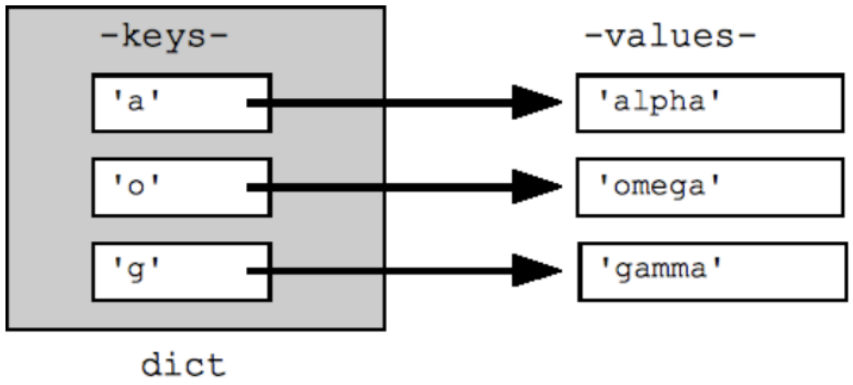
```
>>> dict2 = {}
>>> dict2
{}
>>>
```

Dizionari

Definizione

```
>>> di = {}
>>> di['a'] = 'alpha'
>>> di['g'] = 'gamma'
>>> di'o'] = 'omega'
>>> di
{'a': 'alpha', 'o': 'omega', 'g': 'gamma'}
```

Rappresentazione



Dizionari

Definizione

```
>>> di = {'a':1, 'b':2.3, 'c':'test'}
```

Accesso e modifica di un elemento

```
>>> di['b'] # oggetto indicizzato dalla chiave 'b'
2.3
>>> di['b'] = 'pippo'
>>> di
{'a': 1, 'c': 'test', 'b': 'pippo'}
```

Rimozione di un elemento

```
>>> del di['b']
>>> di
{'a': 1, 'c': 'test'}
```

Dizionari

- Le chiavi e valori possono avere tipo diverso (non omogenee all'interno dello stesso dizionario)
- Le chiavi devono essere **tipi immutabili**

```
>>> {(1,'b'):[1,2]} # chiave e` tupla di immutabili
{(1, 'b'): [1, 2]}
>>> {[1,'b']:[1,2]} # chiave e` lista
[...] TypeError: unhashable type: 'list'
>>> {(1,{2:3}):[1,2]} # chiave e` tupla di mutabili
[...] TypeError: unhashable type: 'dict'
>>> {(1,(2,3)):[1,2]} # chiave e` tupla + tupla
{(1, (2, 3)): [1, 2]}
>>> {(1,(2,[])):[1,2]} # chiave e` tupla + tupla + lista
[...] TypeError: unhashable type: 'list'
```

Dizionari

Alcuni comandi utili:

```
>>> di = {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}

>>> di.keys() # lista di chiavi
['a', 'g', 'o']

>>> di.values() # lista di valori
['alpha', 'gamma', 'omega']

>>> di.items() # lista di tuple con chiave + valore
[('a', 'alpha'), ('g', 'gamma'), ('o', 'omega')]

>>> di.has_key('a') # controllo presenza chiave
True
>>> di.has_key('z')
False

>>> di.clear() # rimozione di tutte le coppie
```

Dizionari – Esercizio

Esercizio

Costruire la propria rubrica telefonica.

Dizionari – Esercizio

Esercizio

Costruire la propria rubrica telefonica.
Anzi, facciamo un "archivio dei contatti": per ciascuno di questi si vuole poter archiviare: *(i)* numero di telefono, *(ii)* e-mail, *(iii)* note.

Dizionari – Esercizio

Esercizio

Costruire la propria rubrica telefonica.
Anzi, facciamo un "archivio dei contatti": per ciascuno di questi si vuole poter archiviare: *(i)* numero di telefono, *(ii)* e-mail, *(iii)* note.

Inserire i contatti almeno di tre persone.

Dizionari – Esercizio

Esercizio

Costruire la propria rubrica telefonica.
Anzi, facciamo un "archivio dei contatti": per ciascuno di questi si vuole poter archiviare: *(i)* numero di telefono, *(ii)* e-mail, *(iii)* note.

Inserire i contatti almeno di tre persone.

Operazioni da provare:

1. Stampare l'e-mail di un contatto
2. Modificare l'e-mail di un contatto
3. Rimuovere le note di un contatto
4. Stampare la lista delle persone in rubrica
5. Verificare (via codice) se un contatto è in rubrica

Dizionari – Esercizio

Definizione struttura dati

```
>>> archivio = {
...    "Mario Rossi" : {
...       "email" : "mrossi@email.it",
...       "tel" : "0123456789", # come intero era ok?
...       "note" : "studente"
...    }
... }
>>>
```

Dizionari – Esercizio

Definizione struttura dati

```
>>> archivio = {
...    "Mario Rossi" : {
...       "email" : "mrossi@email.it",
...       "tel" : "0123456789", # come intero era ok?
...       "note" : "studente"
...    }
... }
>>>
```

1. Stampare l'e-mail di un contatto

```
>>> archivio['Mario Rossi']['email']
'mrossi@email.it'
```

Dizionari – Esercizio

2. Modificare l'e-mail di un contatto

```
>>> archivio['Mario Rossi']['email'] = 'test@nomail.com'
>>> archivio['Mario Rossi']['email']
'test@nomail.com'
```

3. Rimuovere le note di un contatto

```
>>> del archivio['Mario Rossi']['note']
>>> archivio['Mario Rossi']
{'tel': '0123456789', 'email': 'test@nomail.com'}
```

Dizionari – Esercizio

4. Stampare la lista delle persone in rubrica

```
>>> archivio.keys()
['Mario Rossi', Michele Donini', 'Fabio Aiolli']
```

5. Verificare (via codice) se un contatto è in rubrica

```
>>> archivio.has_key('Fabio Aiolli')
True
>>> archivio.has_key('Bill Gates')
False
```

Insiemi

I `set` rappresentano insiemi di valori immutabili (no ordine, no duplicati).

Esistono due costruttori:

1. Costruttore vuoto
2. Costruttore da lista

```
>>> set()
set([])
>>> set([1,2,3])
set([1, 2, 3])
```

Insiemi

I `set` rappresentano insiemi di valori immutabili (no ordine, no duplicati).

Esistono due costruttori:

1. Costruttore vuoto
2. Costruttore da lista

```
>>> set()
set([])
>>> set([1,2,3])
set([1, 2, 3])
```

```
>>> set("ciao")
set(['i', 'a', 'c', 'o'])
```

Insiemi

```
>>> s = set(range(3))
```

Operazioni su insiemi

```
>>> s.add(2) # succede qualcosa?
>>> s.add(3)
>>> s
set([0, 1, 2, 3])
```

```
>>> s.remove(1)
>>> s
set([0, 2, 3])
>>>
```

```
>>> len(s)
3
```

Insiemi

Operazioni sugli insiemi:

```
>>> disp = set(range(1,10,2))
>>> pari = set(range(2,10,2))

>>> disp | pari # unione
set([1, 2, 3, 4, 5, 6, 7, 8, 9])

>>> mul3 = set([3,6,9]) # multipli di 3
>>> disp & mul3 # intersezione
set([9, 3])

>>> pari - mul3 # complemento (pari ma non multiplo di
    3)
set([8, 2, 4])

>>> disp ^ mul3 # differenza simmetrica (in uno ma non
    nell'altro)
set([7, 5, 6, 1])
```

Insiemi – Esercizio

Esercizio

Contare il numero di **parole** distinte contenute nella frase "`conto su di te per far di conto`" che non siano contenute nella frase "`io per te non conto`".

Insiemi – Esercizio

Esercizio

Contare il numero di **parole** distinte contenute nella frase "`conto su di te per far di conto`" che non siano contenute nella frase "`io per te non conto`".

```
>>> s1 = "conto su di te per far di conto".split()
>>> s2 = "io per te non conto".split()
>>> s1 = set(s1)
>>> s2 = set(s2)
>>> len(s1-s2)
3
```

Tipi iterabili

Operatori di appartenenza (`in` e `not in`)

```
>>> pari = range(2,10,2)
>>> 5 in pari
False
>>> 4 not in pari
False
```

Dimensione (`len`)

```
>>> len([2, 4, 6, 8])
4
>>> len([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
3
```

Tipi iterabili

Operatore di somma per collezioni numeriche (`sum`)

```
>>> sum([1, 2, 3.0])
6.0
```

Somma dei numeri da 0 a 100 (senza Gauss ma con Python)

```
>>> sum(range(101))
5050
```

Tipi iterabili

Relazioni d'ordine (`min` e `max`)

```
>>> a = range(100)
>>> min(a), max(a)
(0, 99)
>>> b = ["ZZZ", "aaa"]
>>> min(b), max(b)
('ZZZ', 'aaa')
```

Operazioni di ordinamento (`sorted`)

```
>>> b = set(["ZZZ", "aaa", "AAA", "zzz"])
>>> sorted(b)
['AAA', 'ZZZ', 'aaa', 'zzz']
>>> sorted(b, reverse=True)
['zzz', 'aaa', 'ZZZ', 'AAA']
```

Tipi iterabili

Operatore di aggregazione (`zip`)

```
>>> nomi = ["Fabio", "Mirko", "Michele", "Marta"]
>>> cognomi = ["Aiolli", "Polato", "Donini", "Gatto"]
>>> zip(nomi, cognomi)
[('Fabio', 'Aiolli'), ('Mirko', 'Polato'), ('Michele', '
    Donini'), ('Marta', 'Gatto')]
```

```
>>> nomi = set(["Fabio", "Mirko", "Michele", "Marta"])
>>> cognomi = set(["Aiolli", "Polato", "Donini", "Gatto
    "])
>>> zip(nomi, cognomi)
[('Marta', 'Donini'), ('Fabio', 'Aiolli'), ('Mirko', '
    Gatto'), ('Michele', 'Polato')]
```

Tipi iterabili

Operatore di aggregazione (`zip`) (cont.)

```
>>> a = {"a" : (1, 2), "b" : 4, "c" : "prova"}
>>> b = [1, 2]
>>> zip(a, b)
[('a', 1), ('c', 2)]
```

Descrittori di lista

Data una collezione iterabile, si può costruire agilmente una lista analizzando gli elementi della collezione.

Operatore descrittore di lista

```
[f(x) for x in l] = [f(l[0]), ..., f(l[N])]
```

Descrittori di lista

Data una collezione iterabile, si può costruire agilmente una lista analizzando gli elementi della collezione.

Operatore descrittore di lista

`[f(x) for x in l] = [f(l[0]), ..., f(l[N])]`

Calcolare il quadrato di una lista di numeri

```
>>> l = [2, 3, 5, 7, 11, 13]
>>> [i**2 for i in l]
[4, 9, 25, 49, 121, 169]
```

Descrittori di lista

Descrittori di lista condizionali

`[f(x) for x in l if g(x)]`

Quadrati solo per i numeri maggiori o uguali a 3

```
>>> l = [1, 2, 3, 4]
>>> [i**2 for i in l if i >= 3]
[9, 16]
```

Descrittori di lista

Descrittori con più iteratori

```
[f(x1,x2) for x1 in l1 for x2 in l2 if g(x1,x2)]
```

Tutte le combinazioni di valori distinti da due iterabili.

```
>>> l1 = [-1, 0, 1]
>>> l2 = [-1, 0, 1]
>>> [(x,y) for x in l1 for y in l2 if x != y]
[(-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0)]
```

Descrittori di lista

Descrittori con più iteratori

```
[f(x1,x2) for x1 in l1 for x2 in l2 if g(x1,x2)]
```

Tutte le combinazioni di valori distinti da due iterabili.

```
>>> l1 = [-1, 0, 1]
>>> l2 = [-1, 0, 1]
>>> [(x,y) for x in l1 for y in l2 if x != y]
[(-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0)]
```

Idea generale

$\forall x \in \mathrm{IT}_x \quad \forall y \in \mathrm{IT}_y(x),$ se vale $g(x, y),$ allora $f(x, y)$

(La valutazione procede **da sinistra a destra**)

Descrittori di lista

È possibile annidare descrittori di lista.

Ricostruire una matrice tramite descrittori di lista

```
>>> matrix = [[1, 2], [3, 4]]

>>> [[item for item in row] for row in matrix]
[[1, 2], [3, 4]]
```

Descrittori di lista

È possibile annidare descrittori di lista.

Ricostruire una matrice tramite descrittori di lista

```
>>> matrix = [[1, 2], [3, 4]]

>>> [[item for item in row] for row in matrix]
[[1, 2], [3, 4]]
```

Costruire una lista con tutti gli elementi di una matrice

```
>>> [item for item in row2 for row2 in matrix]
```

Descrittori di lista

È possibile annidare descrittori di lista.

Ricostruire una matrice tramite descrittori di lista

```
>>> matrix = [[1, 2], [3, 4]]

>>> [[item for item in row] for row in matrix]
[[1, 2], [3, 4]]
```

Costruire una lista con tutti gli elementi di una matrice

```
>>> [item for item in row2 for row2 in matrix]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'row2' is not defined

>>> [item for row3 in matrix for item in row3]
```

Descrittori di lista

È possibile annidare descrittori di lista.

Ricostruire una matrice tramite descrittori di lista

```
>>> matrix = [[1, 2], [3, 4]]

>>> [[item for item in row] for row in matrix]
[[1, 2], [3, 4]]
```

Costruire una lista con tutti gli elementi di una matrice

```
>>> [item for item in row2 for row2 in matrix]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'row2' is not defined

>>> [item for row3 in matrix for item in row3]
[1, 2, 3, 4]
```

Descrittori di lista – Esercizio

Esercizio

Data la stringa "`conto su di te per far di conto`":

1. Costruire una lista i cui elementi rappresentino il *numero di lettere di ogni parola* della stringa
2. Data la lista `l1 =['a','b','c']`, costruire una lista della stessa lunghezza in cui ogni elemento indichi (con `True`/`False`) se ciascuna lettera della lista `l1` compare o meno nella stringa di partenza

Descrittori di lista – Esercizio

Costruire una lista i cui elementi rappresentino il *numero di lettere di ogni parola* della stringa

```
>>> s = "conto su di te per far di conto"
>>> [len(w) for w in s.split()]
[5, 2, 2, 2, 3, 3, 2, 5]
```

Descrittori di lista – Esercizio

Costruire una lista i cui elementi rappresentino il *numero di lettere di ogni parola* della stringa

```
>>> s = "conto su di te per far di conto"
>>> [len(w) for w in s.split()]
[5, 2, 2, 2, 3, 3, 2, 5]
```

Costruire una lista i cui elementi indichino (con `True`/`False`) se le lettere `['a','b','c']` compaiono o meno

```
>>> l = ['a','b','c']
>>> [v in s for v in l]
[True, False, True]
```

Descrittori di lista – Esercizio

Esercizio

Data la lista di coordinate $(1, 5), (5, 2), (3, 9), (1, -3)$:

1. Per ogni punto, calcolare la somma delle coordinate
2. Calcolare il quadrato della distanza dei punti della lista dal punto `P(1,1)`
3. Calcolare le coordinate dei punti in un sistema di riferimento con origine `O(3,-1)`

Descrittori di lista – Esercizio

Per ogni punto, calcolare la somma delle coordinate

```
>>> l = [(1,5),(5,2),(3,9),(1,-3)]
>>> [sum(coord) for coord in l]
[6, 7, 12, -2]
```

Descrittori di lista – Esercizio

Per ogni punto, calcolare la somma delle coordinate

```
>>> l = [(1,5),(5,2),(3,9),(1,-3)]
>>> [sum(coord) for coord in l]
[6, 7, 12, -2]
```

Calcolare il quadrato della distanza dei punti `l[i]` dal punto `P(1,1)`

```
>>> P = (1,1)
>>> [(x - P[0])**2 + (y - P[1])**2 for x,y in l]
[16, 17, 68, 16]
```

Descrittori di lista – Esercizio

Calcolare le coordinate dei punti in un sistema di riferimento con origine `O(3,-1)`

```
>>> O = (3, -1)
>>> [(x - O[0], y - O[1]) for x,y in l]
[(-2, 6), (2, 3), (0, 10), (-2, -2)]
```

Descrittori di lista – Cifrario di Cesare

Cifrario di Cesare per singolo carattere:

```
>>> k = 3
>>> chiaro = "a"
>>> chr(ord("a") + ((ord(chiaro) - ord("a") + k) % 26))
'd'
```

Esercizio

Estendere l'esercizio al trattamento di stringhe.
Note: gli spazi vanno preservati come tali; per comodità, convertire il messaggio in minuscolo.

Descrittori di lista – Cifrario di Cesare

```
>>> k = 3
>>> t = "Questo e un esempio di testo di prova"
>>> t = t.lower()
>>> code = [[chr(ord("a") + ((ord(c) - ord("a") + k) %
    26)) for c in parola] for parola in t.split()]
>>> code
[['t', 'x', 'h', 'v', 'w', 'r'], ['h'], ['x', 'q'], ['h
    ', 'v', 'h', 'p', 's', 'l', 'r'], ['g', 'l'], ['w', '
    h', 'v', 'w', 'r'], ['g', 'l'], ['s', 'u', 'r', 'y',
    'd']]

>>> code = ["".join(c) for c in code]
>>> code = " ".join(code)
>>> code
'txhvwr h xq hvhpslr gl whvwr gl suryd'
```