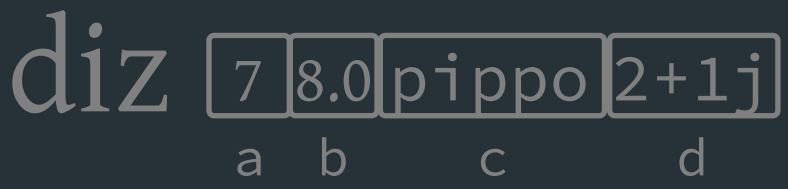
Programmazione in Python

strutture dati: cicli e iteratori

Dario Pescini - Mirko Cesarini

Università degli Studi di Milano-Bicocca

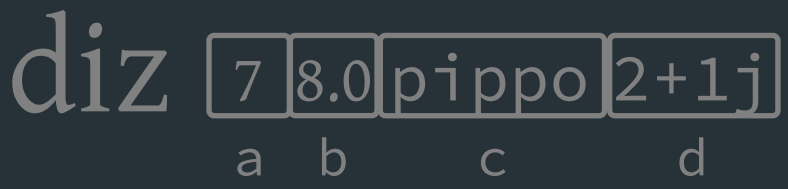
Dizionari e visita dei loro elementi




```
diz = {'a': 7, 'b': 3.0 + 5, 'c': 'pippo', 'd': 2 +1j}

i = 0
while (i < len(diz)):
    print diz[i]
    i = i + 1
```

Dizionari e visita dei loro elementi

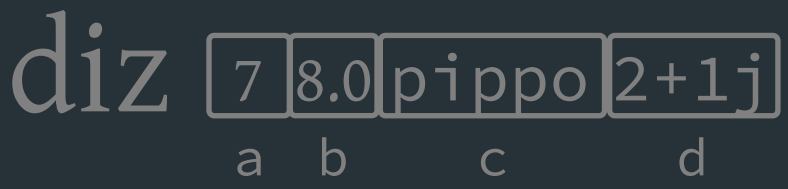



```
diz = {'a': 7, 'b': 3.0 + 5, 'c': 'pippo', 'd': 2 +1j}

i = 0
while (i < len(diz)):
    print diz[i]
    i = i + 1
```

```
dario@vulcano: python DizionarioWhileNotWork.py
Traceback (most recent call last):
  File "DizionarioWhileNotWork.py", line 5, in <module>
    print diz[i]
KeyError: 0
dario@vulcano: _
```

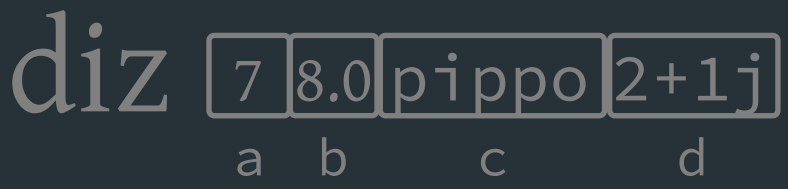
Dizionari e visita dei loro elementi




```
diz = {'a': 7, 'b': 3.0 + 5, 'c': 'pippo', 'd': 2 +1j}

i = 0
listaChiavi = diz.keys()
while (i < len(diz)):
    print diz[listaChiavi[i]]
    i = i + 1
```

Dizionari e visita dei loro elementi

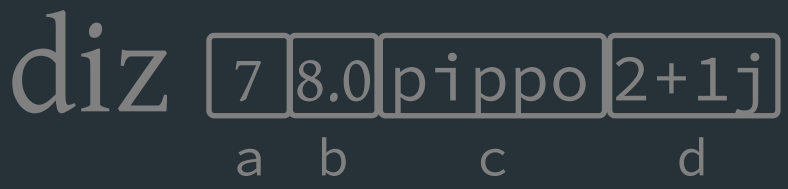



```
diz = {'a': 7, 'b': 3.0 + 5, 'c': 'pippo', 'd': 2 +1j}

i = 0
listaChiavi = diz.keys()
while (i < len(diz)):
    print diz[listaChiavi[i]]
    i = i + 1
```

```
dario@vulcano: python DizionarioWhile.py
7
pippo
8.0
(2+1j)
dario@vulcano:
```

Dizionari e visita dei loro elementi

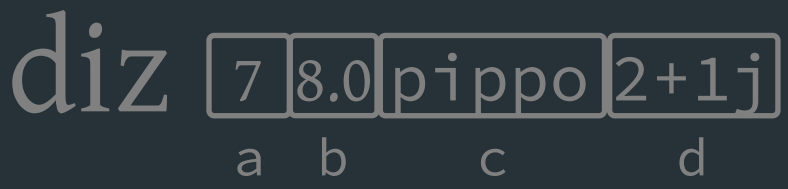



```
diz = {'a': 7, 'b': 3.0 + 5, 'c': 'pippo', 'd': 2 +1j}

i = 0
listaChiavi = diz.keys()
while (i < len(diz)):
    print diz[listaChiavi[i]]
    i = i + 1
```

!!! esiste un modo più semplice !!!

Iteratori - ciclo for




```
diz = {'a': 7, 'b': 3.0 + 5, 'c': 'pippo', 'd': 2 +1j}

for parola in diz:
    print diz[parola]
```

```
dario@vulcano: python DizionarioFor.py
7
pippo
8.0
(2+1j)
dario@vulcano: _
```

Strutture Controllo - Iterazione: FOR

```
for elemento in iteratore :
    lista comandi
```

Iteratori - ciclo for

```
diz = {'uno': 'one', 'due': 'two', 'tre': 'three', 'quattro':
    'four', 'cinque': 'five'}

for chiave in diz:
    print diz[chiave]
```

```
dariopescini@gorgona: python DizionarioIterarore.py
four
five
three
two
one
```

Iteratori - ciclo for

```
diz = {'uno': 'one', 'due': 'two', 'tre': 'three', 'quattro':
    'four', 'cinque': 'five'}

for chiave in diz:
    print diz[chiave]
```

```
dariopescini@gorgona: python DizionarioIterarore.py
four
five
three
two
one
```

Cosa succede?

Iteratori - ciclo for

```
diz = {'uno': 'one', 'due': 'two', 'tre': 'three', 'quattro':
    'four', 'cinque': 'five'}

for chiave in diz:
    print diz[chiave]
```

```
dariopescini@gorgona: python DizionarioIterarore.py
four
five
three
two
one
```

Cosa succede?

viene creata una lista contente le chiavi ≡ insieme iteratore

```
>>> diz.keys()
['quattro', 'cinque', 'tre', 'due', 'uno']
>>>
```

Iteratori - ciclo for

```
diz = {'uno': 'one', 'due': 'two', 'tre': 'three', 'quattro':
    'four', 'cinque': 'five'}

for chiave in diz:
    print diz[chiave]
```

```
dariopescini@gorgona: python DizionarioIterarore.py
four
five
three
two
one
```

Cosa succede?

ad ogni iterazione viene estratto un elemento diverso dall'insieme

```
iteratore =  quattro      - valore =  four
iteratore =  cinque       - valore =  five
iteratore =  tre          - valore =  three
iteratore =  due          - valore =  two
iteratore =  uno          - valore =  one
```

Iteratori - ciclo for

```
diz = {'uno': 'one', 'due': 'two', 'tre': 'three', 'quattro':
    'four', 'cinque': 'five'}

for chiave in diz:
    print diz[chiave]
```

```
dariopescini@gorgona: python DizionarioIterarore.py
four
five
three
two
one
```

Cosa succede?

L'insieme iteratore non ha un ordinamento

Iteratori - stringhe

```
stringa = 'unaStringa'

for elemento in stringa:
    print elemento
```

```
u
n
a
S
t
r
i
n
g
a
```

Iteratori - liste

```
lista = ['uno', 'due', 'tre', 'quattro', 'cinque']

for elemento in lista:
    print elemento
```

```
uno
due
tre
quattro
cinque
```

Iteratori - tuple

```
tupla = (1, 2, 'tre', 'quattro', 5)

for elemento in tupla:
    print elemento
```

```
1
2
tre
quattro
5
```

Iteratori - ciclo for

```
diz = {'uno': 'one', 'due': 'two', 'tre': 'three', 'quattro':
    'four', 'cinque': 'five'}

for chiave in diz:
    print chiave
```

```
uno
due
tre
quattro
cinque
```

Iteratori - range

È possibile costruire un iteratore per eseguire un numero prefissato di iterazioni tramite il comando `range()`.

```
iterabile = range(0, 100, 3)
```

`range(start, stop, step)` costruisce iterabile di interi

- `range()` è l'istruzione che restituisce una lista di interi iterabile
- `start` è il valore iniziale
- `stop` è il valore massimo necessario
- `step` è l'incremento con cui vengono costruiti gli elementi

Iteratori - range

```
lista = ['uno', 'due', 'tre', 'quattro', 'cinque']
iterabile = range(len(lista))
print iterabile

for indice in iterabile:
    print indice, lista[indice]
```

Iteratori - range

```
lista = ['uno', 'due', 'tre', 'quattro', 'cinque']
iterabile = range(len(lista))
print iterabile

for indice in iterabile:
    print indice, lista[indice]
```

```
In [14]: lista = ['uno', 'due', 'tre', 'quattro', 'cinque']
    ...: iterabile = range(len(lista))
    ...: print iterabile
    ...:
    ...: for indice in iterabile:
    ...:     print indice, lista[indice]
    ...:
[0, 1, 2, 3, 4]
0 uno
1 due
2 tre
3 quattro
4 cinque
```

Propietà principali strutture dati

	aggregazione	plasticità	omogeneità	indice	iterabile
scalare	semplice	statica	omogenea		
stringa	complessa	statica	omogenea	intero	si
tupla	complessa	statica	eterogenea	intero	si
lista	complessa	dinamica	eterogenea	intero	si
dizionario	complessa	dinamica	eterogenea	chiave	si - lista