Programmazione in Python

strutture dati: liste

Dario Pescini - Mirko Cesarini

Università degli Studi di Milano-Bicocca

Dipartimento di Statistica e Metodi Quantitativi

nome.cognome@unimib.it

Strutture dati complesse: liste

lista

possiede un nome `l` ed aggrega più oggetti organizzati sequenzialmente:

```
l = a 'pippo' 6 d … z
```

Strutture dati complesse: liste

lista

possiede un nome `l` ed aggrega più oggetti organizzati sequenzialmente:

`l = l[0] l[1] l[2] l[3] … l[n-1]`

Strutture dati complesse: liste

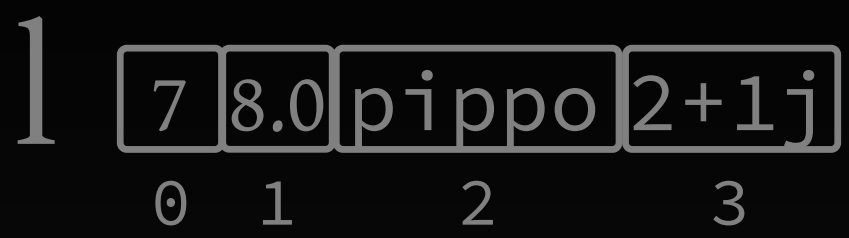
La lista è una struttura dati complessa di tipo sequenza, dinamica ed eterogenea.

```
l = [7, 3.0 + 5, 'pippo', 2 +1j]
```

dichiarazione lista: `[]`

- `l` nome della lista
- `[]` delimitatori della lista
- `7, 3.0 + 5, 'pippo', 2 +1j` elementi della lista
- `,` separatore degli elementi

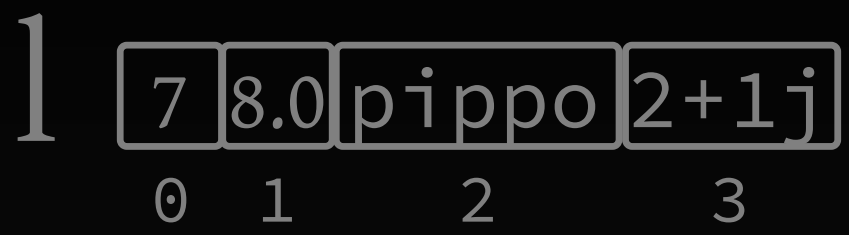
Strutture dati complesse: liste



sequenza:

```
>>> l = [7, 8.0, 'pippo', 2+1j]
>>> print l
[7, 8.0, 'pippo', (2+1j)]
>>> print l[3]
(2+1j)
>>> _
```

Strutture dati complesse: liste



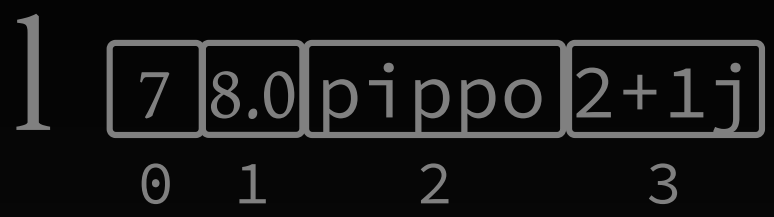
sequenza:

```
>>> l = [7, 8.0, 'pippo', 2+1j]
>>> print l
[7, 8.0, 'pippo', (2+1j)]
>>> print l[3]
(2+1j)
>>> _
```

indici negativi:

```
>>> l[-2]
'pippo'
>>> _
```

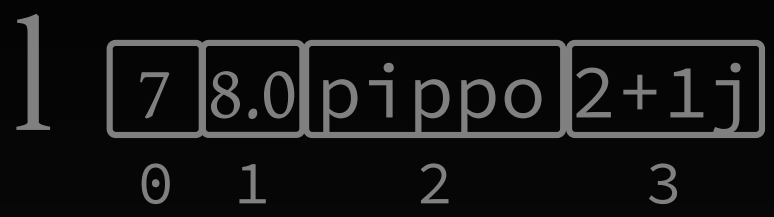
Strutture dati complesse: liste



lunghezza:

```
>>> len(l)
4
>>> _
```

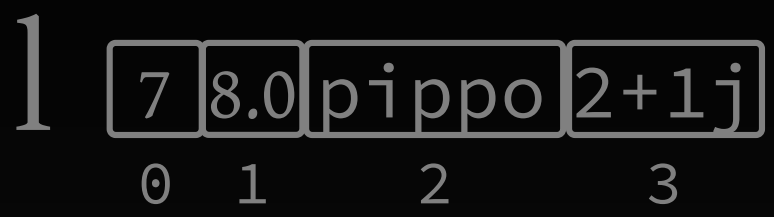
Strutture dati complesse: liste



!! mutabile: !!

```
>>> l[2] = 'nuova'
>>> print l
[7, 8.0, 'nuova', (2+1j)]
>>> _
```

Strutture dati complesse: liste



Slicing:

```
>>> l[:2]
[7, 8.0]
>>> l[2:]
['pippo', (2+1j)]
>>> l[1:3]
[8.0, 'pippo']
>>> _
```

Strutture dati complesse: liste


```
lista = [7, 3.0 + 5, 'pippo', 2 +1j]

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

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

Strutture dati complesse: liste

É possibile estendere una tupla tramite concatenazione:

```
>>> l = [7, 8.0, 'pippo', 2+1j]
>>> id(l)
4455295960
>>> l = l + [12345,]
>>> print l
[7, 8.0, 'pippo', (2+1j), 12345]
```

Metodi per le liste

Alcuni metodi che potete utilizzare per le liste:

- `append(x)` aggiunge l'elemento x in coda alla lista
- `extend(lista2)` aggiunge la lista2 in coda alla lista
- `del lista[i]` elimina l'elemento i-esimo dalla lista
- `sort()` ordina la lista
- ...

- `https://docs.python.org/2.7/library/stdtypes.html#mutable-sequence-types`

Metodi per le liste

`append(x)` aggiunge l'elemento x in coda alla lista

```
CARDINALITA = 5
serie = []

i = 0
while i < CARDINALITA:
    numero = input('inserisci un numero ')
    serie.append(numero)
    print serie
    i += 1
```

```
dario@vulcano: ipython inserimentoValoriLista.py
isnserisci un numero -1
[-1]
isnserisci un numero 4
[-1, 4]
isnserisci un numero 12361
[-1, 4, 12361]
isnserisci un numero -32130
[-1, 4, 12361, -32130]
isnserisci un numero 1
[-1, 4, 12361, -32130, 1]
dario@vulcano: _
```

Metodi per le liste

`extend(lista2)` aggiunge lista2 in coda alla lista

```
CARDINALITA = 5
serie = []
serie2 = ['terzultimo', 'penultimo', 'ultimo']

i = 0
while i < CARDINALITA:
    numero = input('isnserisci un numero ')
    serie.append(numero)
    print serie
    i += 1
serie.extend(serie2)
print serie
```

```
dario@vulcano: ipython inserimentoValoriListaExtend.py
isnserisci un numero 1
[1]
isnserisci un numero 2
[1, 2]
isnserisci un numero 3
[1, 2, 3]
isnserisci un numero 4
[1, 2, 3, 4]
isnserisci un numero 5
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 'terzultimo', 'penultimo', 'ultimo']
dario@vulcano: _
```

Metodi per le liste

`del lista[i]` elimina l'elemento i-esimo dalla lista

```
lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
del lista[5:9]
print lista
```

```
In [5]: lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

In [6]: del lista[5:9]

In [7]: print lista
[1, 2, 3, 4, 5, 10, 11, 12]

In [8]: _
```

Metodi per le liste

`sort()` ordina la lista

```
CARDINALITA = 5
serie = []

i = 0
while i < CARDINALITA:
    numero = input('inserisci un numero ')
    serie.append(numero)
    print serie
    i += 1

serie.sort()
print serie
```

```
dario@vulcano: ipython inserimentoValoriListaSort.py
isnserisci un numero 7381
[7381]
isnserisci un numero 32
[7381, 32]
isnserisci un numero 3
[7381, 32, 3]
isnserisci un numero 431
[7381, 32, 3, 431]
isnserisci un numero 6
[7381, 32, 3, 431, 6]
[3, 6, 32, 431, 7381]
```

Copia di liste

Supponete di dover copiare una lista per crearne un'altra identica da manipolare.

```
listaA = [1, 2, 3, 4]
```

Verrebbe istintivo usare l'operatore di assegnamento =

```
listaB = listaA
```

Stampando la lista sembra che tutto sia avvenuto regolarmente:

```
>>> print listaB
[1, 2, 3, 4]
```

lavoriamo con la listaB:

```
>>>listaB.extend([5, 6, 7, 8])
```

listaB è stata correttamente estesa:

```
print listaB
[1, 2, 3, 4, 5, 6, 7, 8]
```

ma è stata modificata anche listaA:

```
print listaA
[1, 2, 3, 4, 5, 6, 7, 8]
```

Copia di liste

Cos’è successo?

Copia di liste

Cos'è successo?

```
In [7]: id(listaA)
Out[7]: 4496640120

In [8]: id(listaB)
Out[8]: 4496640120
```

Copia di liste

Riproviamo sfruttando il comando `list()`.

```
listaA = [1, 2, 3, 4]
listaB = list(listaA)
```

Stampando la lista sembra che tutto sia avvenuto regolarmente:

```
>>> print listaB
[1, 2, 3, 4]
```

lavoriamo con la listaB:

```
>>>listaB.extend([5, 6, 7, 8])
```

listaB è stata correttamente estesa:

```
print listaB
[1, 2, 3, 4, 5, 6, 7, 8]
```

è stata modificata anche listaA?

```
print listaA
[1, 2, 3, 4]
```

Copia di liste

Cos'è successo?

Copia di liste

Cos'è successo?

```
In [17]: id(listaA)
Out[17]: 4495603688

In [18]: id(listaB)
Out[18]: 4495953864
```

Abbiamo lavorato con due oggetti distinti