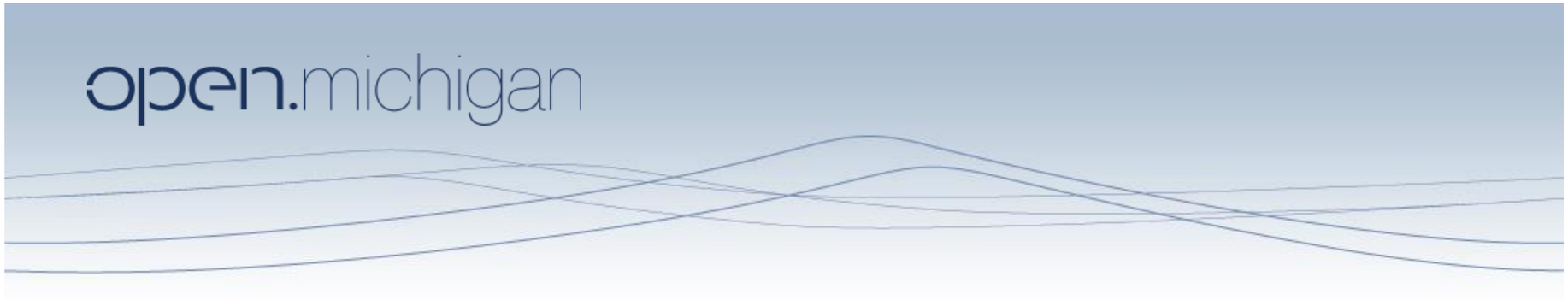
Le Liste in Python

Capitolo 8

Python for Informatics: Exploring Information
www.pythonlearn.com



Una Lista è come una Collezione

- Una collezione che permette di mettere più valori in una singola “variabile”

- Una collezione è utile perché si possono inserire diversi valori tutti in un unico contenitore.

```
amici = [ 'Joseph', 'Glenn', 'Sally' ]

indossano = [ 'calze', 'camica', 'profumo' ]
```

Cosa non è una “Collezione”

- La maggior parte delle variabili ha un unico valore all'interno – quando si assegna un nuovo valore alla variabile – il vecchio valore viene sovrascritto e sostituito dal nuovo

```
$ python
Python 2.5.2 (r252:60911, Feb 22 2008, 07:57:53)
[GCC 4.0.1 (Apple Computer, Inc. build 5363)] on darwin
>>> x = 2
>>> x = 4
>>> print x
4
```

Liste di Constanti

- Le Liste di constanti sono indicate dalle parentesi quadre e gli elementi all'interno della lista sono separati da virgole.

- Una lista di elementi può essere un qualsiasi oggetto gestito da Python – anche un'altra lista

- Una lista può essere vuota

```
>>> print [1, 24, 76]
[1, 24, 76]
>>> print ['rosso', 'giallo', 'blu']
['rosso', 'giallo', 'blu']
>>> print ['rosso', 24, 98.6]
['rosso', 24, 98.59999999999994]
>>> print [ 1, [5, 6], 7]
[1, [5, 6], 7]
>>> print []
[]
```

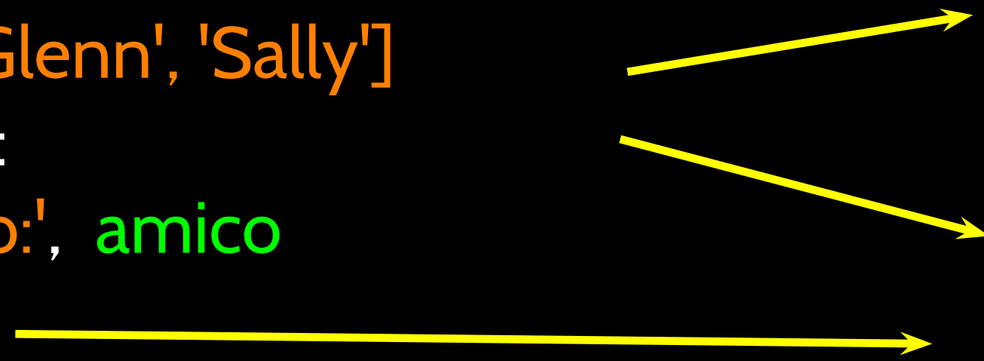
Si sono già usate le liste!

```
for i in [5, 4, 3, 2, 1] :
    print i
print 'Partenza!'
```

```
5
4
3
2
1
Partenza!
```

Liste e cicli finiti – migliori amici

```
amici = ['Joseph', 'Glenn', 'Sally']
for amico in amici :
    print 'Buon Anno:',  amico
print 'Finito!'
```



```
Buon Anno: Joseph
Buona Anno: Glenn
Buon Anno: Sally
```

Leggere dentro le Liste

- Proprio come le stringhe, si può ottenere un singolo elemento di una lista usando un indice inserito all'interno delle parentesi quadre

Joseph	Glenn	Sally
0	1	2

```
>>> amici = [ 'Joseph', 'Glenn', 'Sally' ]
>>> print amici[1]
Glenn
>>>
```

Le liste sono Modificabili

- Le stringhe sono "immutabili" - *non si può* cambiare il contenuto di una stringa – bisogna creare una nuova stringa per poter fare qualsiasi cambiamento

- Le liste sono "modificabili" si può we *can* cambiare un elemento di una lista usando l'operatore indice

```
>>> fruit = 'Banana'
>>> fruit[0] = 'b'
Traceback
TypeError: 'str' object does not support item assignment
>>> x = fruit.lower()
>>> print x
banana
>>> lotto = [2, 14, 26, 41, 63]
>>> print lotto[2, 14, 26, 41, 63]
>>> lotto[2] = 28
>>> print lotto
[2, 14, 28, 41, 63]
```

Quanto è Lunga una Lista?

La funzione len() prende una lista come parametro e restituisce il numero di *elementi* nella lista

- In realtà len() indica il numero di elementi di *qualsiasi* collezione o sequenza (es. come una stringa...)

```
>>> saluto = 'Ciao Bob'
>>> print len(saluto)
9
>>> x = [ 1, 2, 'joe', 99]
>>> print len(x)
4
>>>
```

Usare la funzione range

La funzione range restituisce una lista di numeri che variano da zero a uno meno lo stesso parametro

- Si può costruire un ciclo indicizzato usando for ed un iteratore intero

```
>>> print range(4)
[0, 1, 2, 3]
>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print len(friends)
3
>>> print range(len(friends))
[0, 1, 2]
>>>
```

A tale of two loops...

```
friends = ['Joseph', 'Glenn', 'Sally']

for amico in amici :
    print 'Buon anno:',  amico

for i in range(len(amici)) :
    amico = amici[i]
    print 'Buon anno:', amico
```

```
>>> amici = ['Joseph', 'Glenn', 'Sally']
>>> print len(amici)
3
>>> print range(len(amici))
[0, 1, 2]
>>>
```

```
Buon anno: Joseph
Buon anno: Glenn
Buon anno: Sally
```

Concatenazione di liste con l'uso di +

- Si può creare una nuova lista unendo due liste già esistenti

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print c
[1, 2, 3, 4, 5, 6]
>>> print a
[1, 2, 3]
```

Le liste possono essere suddivise utilizzando :

```
>>> t = [9, 41, 12, 3, 74, 15]
>>> t[1:3]
[41,12]
>>> t[:4]
[9, 41, 12, 3]
>>> t[3:]
[3, 74, 15]
>>> t[:]
[9, 41, 12, 3, 74, 15]
```

Ricordarsi: *Proprio come nelle stringhe*, il secondo numero è "fino a, ma non incluso"

I Metodi delle Liste

```
>>> x = list()
>>> type(x)<type 'list'>
>>> dir(x)['append', 'count', 'extend', 'index', 'insert', 'pop',
'remove', 'reverse', 'sort']
>>>
```

http://docs.python.org/tutorial/datastructures.html

Costruire una lista da zero

Si può creare una lista vuota e dopo aggiungere gli elementi usando il metodo append

- La lista mantiene l'ordine e i nuovi elementi sono aggiunti alla fine della lista

```
>>> cose = list()
>>> cose.append('libro')
>>> cose.append(99)
>>> print stuff
['libro', 99]
>>> stuff.append('biscotto')
>>> print stuff
['libro', 99, 'biscotto']
```

C'è qualcosa in una Lista?

- Python fornisce due operatori che permettono di controllare se un elemento è all'interno di una lista
- Sono operatori logici che restituiscono True (vero) o False (falso)
- Non modificano la lista

```
>>> qualcosa = [1, 9, 21, 10, 16]
>>> 9 in qualcosa
True
>>> 15 in qualcosa
False
>>> 20 not in qualcosa
True
>>>
```

Una Lista è Ordinata in Sequenza

- Una lista può contenere diversi elementi e li mantiene nell'ordine dato fintanto non si faccia qualcos per cambiarlo

- Una lista può essere ordinata (es. Il suo ordine cambia)

- Il metodo sort (diversamente dalle stringhe) significa ''ordina te stessa''

```
>>>amici = [ 'Joseph', 'Glenn', 'Sally' ]
>>> amici.sort()
>>> print amici
['Glenn', 'Joseph', 'Sally']
>>> print amici[1]
Joseph>>>
```

Funzioni presenti e Liste

- Ci sono diverse funzioni già presenti in Python che utilizzano le liste come parametri

- Ricordate I cicli già scritti? Le funzioni sono molto più sempici

```
>>> nums = [3, 41, 12, 9, 74, 15]
>>> print len(nums)
6
>>> print max(nums)
74>>> print min(nums)
3
>>> print sum(nums)
154
>>> print sum(nums)/len(nums)
25
```

http://docs.python.org/lib/built-in-funcs.html

```
total = 0
count = 0
while True :
    inp = raw_input('Enter a number: ')
    if inp == 'done' : break
    value = float(inp)
    total = total + value
    count = count + 1

average = total / count
print 'Average:', average
```

```
Enter a number: 3
Enter a number: 9
Enter a number: 5
Enter a number: done
Average: 5.66666666667
```

```
numlist = list()
while True :
    inp = raw_input('Enter a number: ')
    if inp == 'done' : break
    value = float(inp)
    numlist.append(value)

average = sum(numlist) / len(numlist)
print 'Average:', average
```

I migliori amici: Stringhe e Liste

```
>>> abc = 'Con tre parole'
>>> cose = abc.split()
>>> print cose
['Con', 'tre', 'parole']
>>> print len(cose)
3
>>> print cose[0]
Con
```

```
>>> print cose
['Con', 'tre', 'parole']
>>> for p in cose :
...         print p
...
Con
Tre
Parole
>>>
```

Split divide una stringa in singole parti creando una lista di stringhe. Se si vedono come parole si può accedere ad una particolare parola o creare un ciclo per scansionare tutte le parole.

```
>>> linea = 'Tanti                    spazi vuoti'
>>> ecc = linea.split()
>>> print ecc['Tanti', 'spazi', 'vuoti'']
>>>
>>> linea = 'primo;secondo;terzo'
>>> cosa = linea.split()
>>> print cosa['primo;secondo;terzo']
>>> print len(cosa)
1
>>> cosa = linea.split(';')
>>> print cosa['primo', 'secondo', 'terzo']
>>> print len(cosa)
3
>>>
```

Quando non si specifica un separatore, gli spazi vuoti sono considerati come un "unico" separatore.

Si può specificare il carattere che è il separatore da usare nella funzione split.

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if not line.startswith('From ') : continue
    words = line.split()
    print words[2]
```

```
Sat
Fri
Fri
Fri
  ...
```

```
>>> line = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
>>> words = line.split()
>>> print words
['From', 'stephen.marquard@uct.ac.za', 'Sat', 'Jan', '5', '09:14:16', '2008']
>>>
```

Il metodo della doppia divisione

- Qualche volta si ha il bisogno di dividere una stringa per poi dividere di nuovo la sottostringa ottenuta dalla prima divisione

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
words = line.split()
email = words[1]
```

Il metodo della doppia divisione

- Qualche volta si ha il bisogno di dividere una stringa per poi dividere di nuovo la sottostringa ottenuta dalla prima divisione

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
words = line.split()
email = words[1]
pieces = email.split('@')
```

stephen.marquard@uct.ac.za

['stephen.marquard', 'uct.ac.za']

Il metodo della doppia divisione

- Qualche volta si ha il bisogno di dividere una stringa per poi dividere di nuovo la sottostringa ottenuta dalla prima divisione

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
words = line.split()
email = words[1]
pieces = email.split('@')
print pieces[1]
```

stephen.marquard@uct.ac.za

['stephen.marquard', 'uct.ac.za']

Il metodo della doppia divisione

- Qualche volta si ha il bisogno di dividere una stringa per poi dividere di nuovo la sottostringa ottenuta dalla prima divisione

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
words = line.split()
email = words[1]
pieces = email.split('@')
print pieces[1]
```

stephen.marquard@uct.ac.za

['stephen.marquard', 'uct.ac.za']

Riepilogo Liste

- Concetto di collezione
- Liste e cicli finiti
- Indicizzazione e ricerca
- Modificare le liste
- Funzioni: len, min, max e sum
- Partizionare le liste
- Liste metodi: append, remove
- Ordinare le liste
- Dividere le liste in stringhe di parole
- Usare il metodo split per analizzare le stringhe