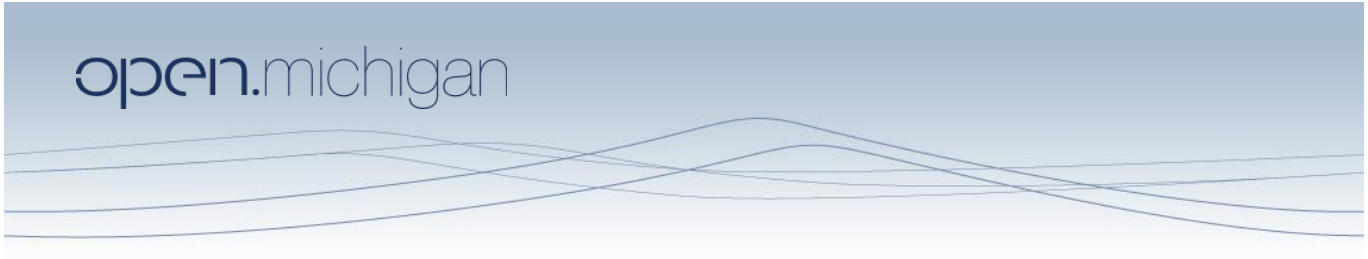
Stringhe

Capitolo 6

Python for Informatics: Exploring Information
www.pythonlearn.com



Tipo Dati Stringa

- Una stringa è una sequenza di caratteri
- Una stringa è identificata da singoli o doppi apici 'Ciao' oppure “Ciao”
- Tra due stringhe, il segno + “concatena”
- Una stringa può contenere anche numeri
- I numeri in formato stringa possono essere convertiti usando int()

```
>>> str1 = "Ciao”
>>> str2 = 'a tutti'
>>> bob = str1 + str2
>>> print bob
Hellothere
>>> str3 = '123'
>>> str3 = str3 + 1
Traceback (most recent call last):
File "<stdin>", line 1, in
<module>TypeError: cannot
concatenate 'str' and 'int' objects
>>> x = int(str3) + 1
>>> print x
124
>>>
```

Lettura e Conversione

- E' preferibile acquisire i dati in entrata usando le stringhe per poi convertirle secondo il tipo richiesto
- Questo ci dà la possibilità di avere un migliore controllo sulle criticità e/o su input errati dell'utente
- Se si vuole acquisire un tipo numerico usando raw_input() occorre convertire la stringa

```
>>> name = raw_input('Enter:')
Enter:Chuck
>>> print name
Chuck
>>> apple = raw_input('Enter:')
Enter:100
>>> x = apple – 10
Traceback (most recent call last):  File
''<stdin>'', line 1, in
<module>TypeError: unsupported
operand type(s) for -: 'str' and 'int'
>>> x = int(apple) – 10
>>> print x
90
```

Guardando dentro le stringhe

- Si può ottenere un qualsiasi carattere all'interno della stringa usando un indice fra parentesi quadre
- Il valore dell'indice deve essere un numero intero e l'indice del primo elemento è sempre 0
- Il valore dell'indice può essere il risultato di una espressione

b	a	n	a	n	a
0	1	2	3	4	5

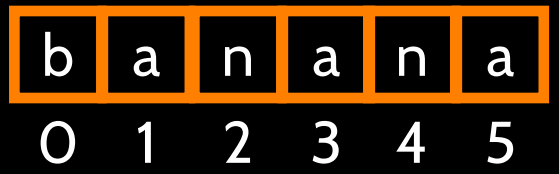
```
>>> fruit = 'banana'
>>> letter = fruit[1]
>>> print letter
a
>>> n = 3
>>> w = fruit[n - 1]
>>> print w
n
```

Un carattere di troppo

- Si avrà un ”python error” se si cerca di forzare l'indice oltre la fine della stringa
- Quindi fate attenzione quando utilizzate un indice e lo fate scorrere all'interno della stringa

```
>>> zot = 'abc'
>>> print zot[5]
Traceback (most recent call last):
File "<stdin>", line 1, in
<module>IndexError: string index
out of range
>>>
```

Le stringhe hanno una lunghezza



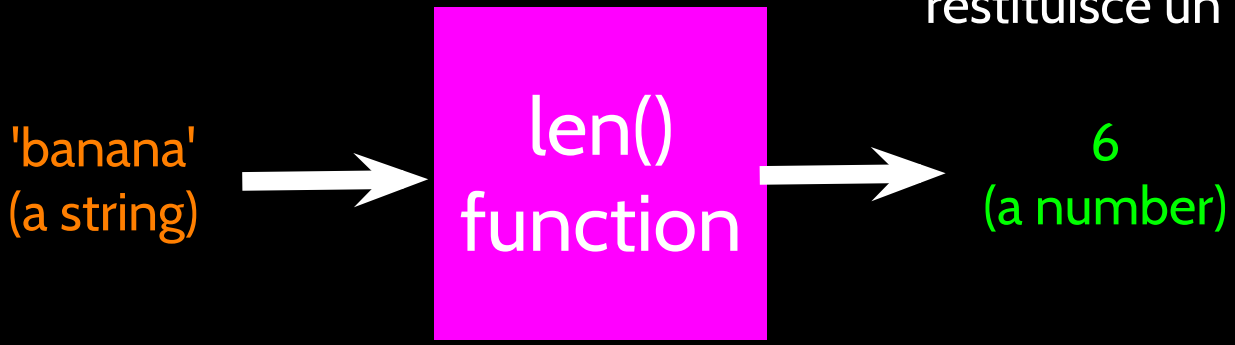
- len() è una funzione interna che ci restituisce la lunghezza della stringa

```
>>> fruit = 'banana'
>>> print len(fruit)
6
```

Funzione len

```
>>> fruit = 'banana'
>>> x = len(fruit)
>>> print x
6
```

Una funzione è un tipo di codice già presente che possiamo usare. Una funzione prende in entrata un input e restituisce un output

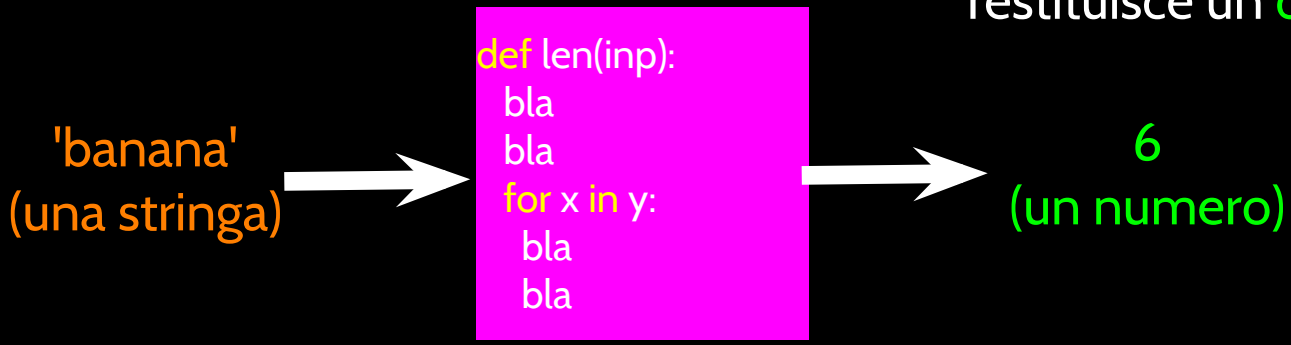


Guido ha scritto questo codice

Funzione len

```
>>> fruit = 'banana'
>>> x = len(fruit)
>>> print x
6
```

Una funzione è un tipo di codice già presente che possiamo usare. Una funzione prende in entrata un input e restituisce un output



Scorrere una Stringa

- Utilizzando un ciclo while e variabile iterativa, e la funzione len, si può costruire un ciclo per visualizzare individualmente ogni singolo carattere presente nella stringa

```
fruit = 'banana'
index = 0
while index < len(fruit) :
  letter = fruit[index]
   print index, letter
   index = index + 1
```

```
0 b
1 a
2 n
3 a
4 n
5 a
```

Scorrere una Stringa

- Utilizzare un ciclo for per scorrere la stringa è più elegante
- La variabile iterativa è completamente gestista dal ciclo for

```
fruit = 'banana'
for letter in fruit :
   print letter
```

```
b
a
n
a
n
a
```

Cicli per leggere le stringhe

Utilizzare un ciclo for per scorrere la stringa è più elegante

La variabile iterativa è completamente gestita dal ciclo for

```
fruit = 'banana'
for letter in fruit :
  print letter
```

```
index = 0
while index < len(fruit) :
   letter = fruit[index]
   print letter
   index = index + 1
```

```
b
a
n
a
n
a
```

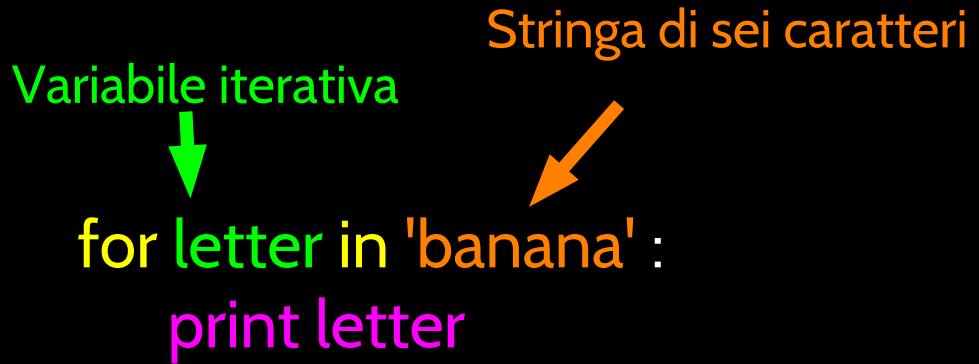
Cicli e conteggi

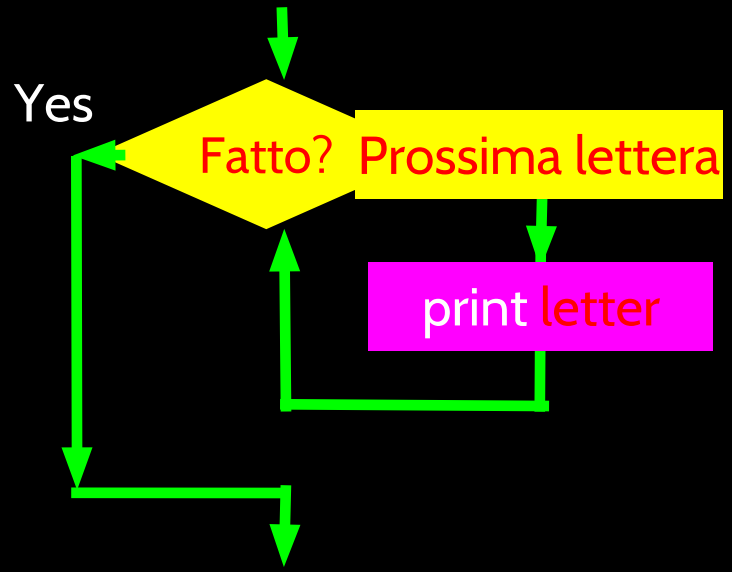
- Questo è un semplice ciclo che legge carattere per carattere una stringa e conta il numero di volte che durante la scansione trova la lettera 'a'.

```
word = 'banana'
count = 0
for letter in word :
    if letter == 'a' :
        count = count + 1
print count
```

Cosa vuol dire in

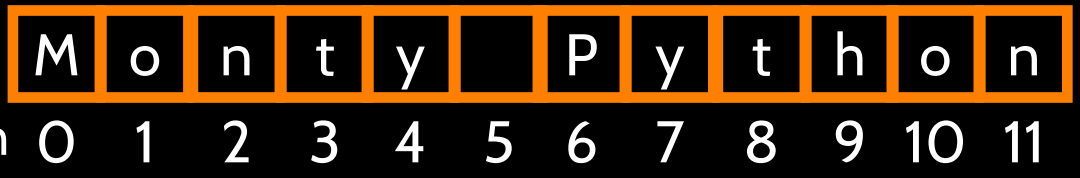
- La variabile iterativa “itera” lungo la sequenza (insieme ordinato)
- Il blocco (corpo) di codice è eseguito ogni volta per ogni valore in sequenza
- La variabile iterativa si sposta attraverso tutti valori in sequenza






```
for letter in 'banana' :
    print letter
```

La variabile iterativa “itera” lungo la sequenza.
Il blocco di codice è eseguito ogni volta per ogni valore in sequenza



Sezione di Stringa

- Si può anche prendere in considerazine una sezione continua di una stringa usando i due punti

- Il secondo indice si posiziona subito dopo la fine della sezione - “fino a, ma non incluso”

- Se il secondo indice è più grande della lunghezza della stringa, arriva solo fino alla

```
>>> s = 'Monty Python'
>>> print s[0:4]
Mont
>>> print s[6:7]
P
>>> print s[6:20]
Python
```

- Se si tralascia il primo o l'ultimo indice della sezione, si prendono rispettivamente come riferimento l'inizio e la fine della stringa

```
>>> s = 'Monty Python'
>>> print s[:2]
Mo
>>> print s[8:]
Thon
>>> print s[:]
Monty Python
```

Sezione di Stringa

Concatenazione di stringhe

- Quando l'operatore + è usato con le stringhe, vuol dire "concatenazione"

```
>>> a = 'Hello'
>>> b = a + 'a tutti'
>>> print b
Ciaoa tutti
>>> c = a + '  ' + 'a tutti'
>>> print c
Ciao a tutti
>>>
```

Usare in come operatore

- La parola riservata in può essere usata per controllare se una stringa è contenuta ''in'' un'altra stringa

- L'epressione in è un'epressione logica e restituisce True o False e può essere usata come condizione all'interno di un if

```
>>> fruit = 'banana'
>>> 'n' in fruit
True
>>> 'm' in fruit
False
>>> 'nan' in fruit
True
>>> if 'a' in fruit :
...     print 'Found it!'
...
Found it!
>>>
```

Confronto tra stringhe

```
if word == 'gente':
    print  'Tutto ok, gente.'

if word < 'gente':
    print 'La tua parola,' + word + ', viene prima di gente.’
elif word > 'gente':
    print 'La tua parola,' + word + ', viene dopo gente.’
else:
    print 'Tutto ok, gente.'
```

String Library

- Python ha diverse function (funzioni) che operano sulle stringhe che si trovano in string library

- Queste function sono *già presenti* per ogni variabile di tipo stringa – si richiamano attacando alla stringa un punto seguito dal nome della funzione

- Queste function non modificano la stringa di partenza ma restituiscono una nuova seconda stringa che

```
>>> greet = 'Hello Bob'>>> zap = greet.lower()>>> print zaphello bob
>>> print greet
Hello Bob>>> print 'Hi There'.lower()
hi there
>>>
```

```
>>> stuff = 'Hello world'
>>> type(stuff)<type 'str'>
>>> dir(stuff)
['capitalize', 'center', 'count', 'decode', 'encode', 'endswith',
'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit',
'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip',
'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
```

http://docs.python.org/lib/string-methods.html

`str.`**`replace`**(*old*, *new*[, *count*])

Return a copy of the string with all occurrences of substring *old* replaced by *new*. If the optional argument *count* is given, only the first *count* occurrences are replaced.

`str.`**`rfind`**(*sub*[, *start*[, *end*]])

Return the highest index in the string where substring *sub* is found, such that *sub* is contained within s[start,end]. Optional arguments *start* and *end* are interpreted as in slice notation. Return `-1` on failure.

`str.`**`rindex`**(*sub*[, *start*[, *end*]])

Like `rfind()` but raises `ValueError` when the substring *sub* is not found.

`str.`**`rjust`**(*width*[, *fillchar*])

Return the string right justified in a string of length *width*. Padding is done using the specified *fillchar* (default is a space). The original string is returned if *width* is less than `len(s)`.

http://docs.python.org/lib/string-methods.html

String Library

str.capitalize()
str.center(larghezza[, fillchar])
str.endswith(suffisso[, inizio[, fine]])
str.find(sub[, inizio[, fine]])
str.lstrip([chars])
str.replace(vecchio, nuovo[, indice])
str.lower()
str.rstrip([chars])
str.strip([chars])
str.upper()

http://docs.python.org/lib/string-methods.html

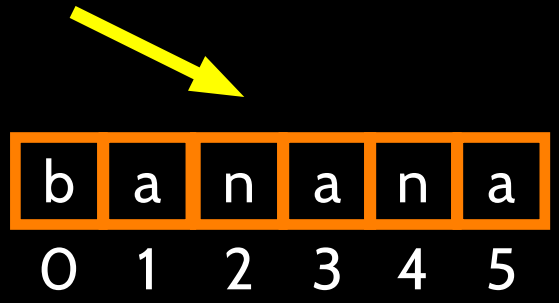
Ricerca di una stringa

- Si usa la (function) funzione find() per cercare una sottostringa all'interno di una stringa

- find() trova la prima occorenza della sottostringa

- Se la sottostringa non si trova, find() restituisce -1

- Ricordarsi che il numero di posizione delle stringhe




```
>>> fruit = 'banana'
>>> pos = fruit.find('na')
>>> print pos
2
>>> aa = fruit.find('z')
>>> print aa
-1
```

Cambiare tutti I caratteri IN MAIUSCOLO

- Si può fare una copia di una stringa con tutti I caratteri minuscoli oppure maiuscoli

- Speso quando si utilizza per una ricerca di una sottostringa find() - conviene prima convertire tutti I caratteri della stringa in minuscolo così da non doversi preoccupare anche per I caratteri maiuscoli perché in python un carattere in minuscolo è diverso dallo stesso in maiuscolo

```
>>> saluto = 'Ciao Bob'
>>> nnn = saluto.upper()
>>> print nnn
CIAO BOB
>>> www = saluto.lower()
>>> print www
ciao bob
>>>
```

Trova e sostiuisci

- La funzione replace() agisce come l'operazione “trova e sostiuisci” in un word processor
- Sostituisce tutte le le occorrenze della vecchia stringa con la nuova stringa

```
>>> greet = 'Hello Bob'
>>> nstr = greet.replace('Bob','Jane')
>>> print nstr
Hello Jane
>>> nstr = greet.replace('o','X')
>>> print nstrHellX BXb
>>>
```

Rimozione degli spazi

- Qualche volta si ha bisogno di rimuovere (strip) gli spazi presenti all'inizio e/o fine di una stringa

- lstrip() e rstrip()
  rispettivamente solo per l'estremo sinsitro (left) o l'estremo destro (right) della stringa

- strip() rimuove gli spazi sia alla testa che alla coda della

```
>>> greet = '   Hello Bob  '
>>> greet.lstrip()
'Hello Bob  '
>>> greet.rstrip()
'   Hello Bob'
>>> greet.strip()
'Hello Bob'
>>>
```

Prefissi

```
>>> line = 'Ti auguro una buona giornata'
>>> line.startswith('Ti')
True
>>> line.startswith('t')
False
```

21 31

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

```
>>> data = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
>>> atpos = data.find('@')
>>> print atpos
21
>>> sppos = data.find(' ',atpos)
>>> print sppos
31
>>> host = data[atpos+1 : sppos]
>>> print host
uct.ac.za
```

Parsing and Extracting

Summary

- Il tipo stringa
- Leggere/Convertire
- Indicizzazione delle stringhe []
- Sezioni di stringhe [2:4]
- I cicli for e while per leggere una stringa