

Programmazione in Python

strutture dati: stringhe

Dario Pescini - Mirko Cesarini

Università degli Studi di Milano-Bicocca

Dipartimento di Statistica e Metodi Quantitativi

Strutture dati complesse: stringhe

stringa



possiede un nome **A** ed aggrega più stringhe di un elemento (caratteri) organizzate sequenzialmente:

A = a b c d ... z



Strutture dati complesse: stringhe

stringa



possiede un nome **A** ed aggrega più stringhe di un elemento (caratteri) organizzate sequenzialmente:

$A = A[0] \ A[1] \ A[2] \ A[3] \ \dots \ A[n-1]$



Strutture dati complesse: stringhe

La stringa è una struttura dati **complessa** di tipo **sequenza**, **statica** ed **omogenea**.

```
a = 'corso di Informatica'
```

```
a = "corso di Informatica"
```

dichiarazione stringa: ' ' o ""

- **a** nome della stringa
- ' ' o "" delimitatori della stringa
- **corso di Informatica** contenuto della stringa

Strutture dati complesse: stringhe

a c o r s o d i I n f o r m a t i c a

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

sequenza:

```
>>> a[4]
'o'
>>> print a[4]
o
```

Strutture dati complesse: stringhe

a c o r s o d i I n f o r m a t i c a

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

sequenza:

```
>>> a[4]
'o'
>>> print a[4]
o
```

indici negativi:

```
>>> a[-1]
'a'
>>> _
```

Strutture dati complesse: stringhe

a c o r s o d i I n f o r m a t i c a

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

lunghezza:

```
>>> len(a)
20
>>> _
```

Strutture dati complesse: stringhe

a c o r s o d i I n f o r m a t i c a
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

immutabile:

```
>>> a[4] = 'i'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> _
```


Strutture dati complesse: stringhe

a c o r s o d i I n f o r m a t i c a

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Slicing:

```
>>> a[:5]
'corso'
>>> a[9:]
'Informatica'
>>> a[6:8]
'di'
>>> a[-6:-3]
'mat'
>>> a[-3:1]
''
>>>
```

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

a c o r s o d i I n f o r m a t i c a

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Strutture dati complesse: stringhe

a c o r s o d i I n f o r m a t i c a

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

```
stringa = 'corso di Informatica'
```

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

```
dario@vulcano: python stringaWhile.py
```

```
c
o
r
s
o

d
i

I
n
f
o
r
m
a
t
i
c
a
```

Strutture dati complesse: stringhe

a

c	o	r	s	o		d	i		I	n	f	o	r	m	a	t	i	c	a
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

É possibile estendere le stringhe tramite **concatenazione** + :

```
>>> a = 'corso di Informatica'
>>> a = a + ' ^^'
>>> print a
corso di Informatica ^^
>>> _
```

Strutture dati complesse: stringhe

a c o r s o d i I n f o r m a t i c a

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

É possibile **replicare** le stringhe tramite l'operatore ***** :

```
>>> a = '--^^'
>>> print a * 4
>>> --^^--^^--^^--^^
_
```

Metodi per le stringhe

Alcuni metodi che potete utilizzare per le stringhe:

- `upper()` restituisce la stringa in MAIUSCOLO
- `find(stringToFind)` restituisce l'indice del primo carattere della prima occorrenza della stringa cercata o -1 se non viene trovata
- `count(subStr)` restituisce il numero delle occorrenze di `subStr` nella stringa
- `split([sep])` restituisce la lista delle parole divise da `sep` all'interno della stringa
- `strip([chars])` restituisce una copia della stringa con i caratteri in `chars` rimossi dall'inizio e dalla fine
- ...
- <https://docs.python.org/2.7/library/stdtypes.html#string-methods>

Metodi: esempi

```
s = ' una stringa particolare '  
t = '__| qualche carattere |__'
```

- `s.strip()` → `'una stringa particolare'`
`t.strip()` → `'__| qualche carattere |__'`

Metodi: esempi

```
s = ' una stringa particolare '  
t = '__| qualche carattere |__'
```

- `s.strip()` → 'una stringa particolare'
`t.strip()` → '__| qualche carattere |__'
- `s.strip(' ')` → ' una stringa particolare'
`t.strip('|')` → '| qualche carattere |'

Metodi: esempi

```
s = ' una stringa particolare '  
t = '__| qualche carattere |__'
```

- `s.strip()` → 'una stringa particolare'
`t.strip()` → '__| qualche carattere |__'
- `s.strip('_')` → ' una stringa particolare'
`t.strip('_')` → '| qualche carattere |'
- `s.strip('|')` → ' una stringa particolare'
`t.strip('|')` → '__| qualche carattere |__'

Metodi: esempi

```
s = ' una stringa particolare '  
t = '__| qualche carattere |__'
```

- `s.strip()` → `'una stringa particolare'`
`t.strip()` → `'__| qualche carattere |__'`
- `s.strip('_')` → `' una stringa particolare'`
`t.strip('_')` → `'| qualche carattere |'`
- `s.strip('|')` → `' una stringa particolare'`
`t.strip('|')` → `'__| qualche carattere |__'`
- `s.find('a')` → `3`
`s.find('x')` → `-1`

Metodi: esempi

```
s = ' una stringa particolare '  
t = '__| qualche carattere |__'
```

- `s.strip()` → 'una stringa particolare'
`t.strip()` → '__| qualche carattere |__'
- `s.strip('_')` → ' una stringa particolare'
`t.strip('_')` → '| qualche carattere |'
- `s.strip('|')` → ' una stringa particolare'
`t.strip('|')` → '__| qualche carattere |__'
- `s.find('a')` → 3
`s.find('x')` → -1
- `s.split()` → ['una', 'stringa', 'particolare']
`s.split('a')` → [' un', ' string', ' p', 'rticol', 're ']