UNIVERSITÀ di VERONA

Linguaggio di Programmazione Python

12/04/2022

Giulio Mazzi

giulio.mazzi@univr.it

Stringhe in python

Abbiamo già visto molte variabili di tipo stringa, contengono al loro interno del testo, e sono immutabili.

Possiamo pensare alle stringhe come *liste di caratteri*

```
nome = 'Giulio'
for char in nome:
    print(char, end=',')
```

```
$ python3 test.py
G,i,u,l,i,o,
```

Possiamo usare molte delle operazioni per le liste anche sulle stringhe (e.g., slicing, `in`, `len()`, `find(x)`, `[x]`,…)

Le stringhe hanno molti metodi personalizzati: isalpha(), isdigit(), islower(), isupper(), isspace(),…

Trovate qui una lista completa:

https://docs.python.org/3/library/stdtypes.html#string-methods

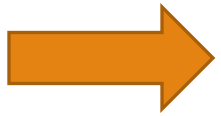
Stringhe - join & split

Due metodi molto usati, quando si lavora con le stringhe, sono join e split.

- Con `' '.join(…)` posso unire più elementi in un unica stringa. Il contenuto della stringa su cui chiamo il metodo è il separatore (e.g., `','.join(…)` per stampare parole separate da virgola
- Con `split()` posso spezzare una stringa complessa In una lista di stringhe. Devo specificare il valore che fa da separatore

```
l = ['nome', 'cognome', 'matricola']
print(','.join(l))

u = 'Mario,Rossi,VR123456'
print(u.split(','))
```



```
nome,cognome,matricola
['Mario', 'Rossi', 'VR123456']
```

Stringhe - metodo format

Possiamo costruire stringhe complesse che incorporano dati e valori usando `format()`.

Il metodo prende una stringa con dei "buchi" (indicati con un {}) e li sostituisce con dei valori. I valori vengono presi in ordine, oppure posso specificare la loro posizione (con {0}, {1}, …) oppure usare keyword arguments ('{x}'.format(x=…))

```
x = 2
y = 4
print('la somma di {} e {} fa {}'.format(x, y, x+y))
print('la somma di {1} e {0} fa {2}'.format(y, x, x+y))
print('la somma di {x} e {y} fa {res}'.format(x=x, y=y, res=x+y))
```

Opzioni Format

Posso anche specificare delle opzioni all'interno delle parentesi graffe per stampare il valore secondo determinati criteri.

```
data = [1.3225342, 32.453002, 122]
# stampa solo le prime due cifre
print('{:.2f}'.format(data[0]))
# occupa 10 caratteri, allinea a destra
print('{:>10}'.format(data[1]))
# stampa i valori della lista
print('|{:>8.2f}|{:>8.2f}|{:>8.2f}|'.format(*data))
```

```
1.32
 32.453002
|    1.32|   32.45|  122.00|
```

Trovate la lista completa qui:

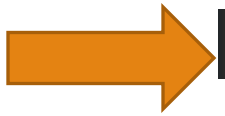
https://docs.python.org/3/library/string.html#formatstrings

Formatted string literals

Python 3.6 ha aggiunto un nuovo modo di gestire la formattazione, i formatted string literals (o f-string). Questi snelliscono ed estendono i metodo forniti da `.format()`.

Identifico le stringhe con una f davanti al corpo, posso mettere il nome di variabili (ma anche vere espressioni!) dentro il {}.

```
x = 2
y = 4
print(f'la somma di {x} e {y} fa {x+y}')
```

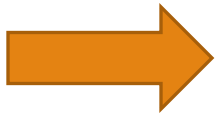


```
la somma di 2 e 4 fa 6
```

Posso usare le stesse opzioni di format (e.g., {:.2f},…).

Un estensione molto è l'opzione =, stampa nome e valore di una variabile (utile per fare introspezione!)

```
print(f'{x=}, {y=}')
```



```
x=2, y=4
```

Formattazione in stile c

Esiste un metodo più vecchio di formattare stringhe, basato su c.

Meno usato degli altri metodi, ma ancora presente, conoscerlo è utile. È più semplice e limitato, ma anche leggermente più veloce in casi semplici.

Devo specificare il tipo dei valori (d per decimali, f per float, s per stringhe...) e collegare i valori con % e una tupla

```
x = 2
y = 4
print('la somma di %d e %d fa %d' % (x, y, x+y))
```

Librerie utili

Spesso dobbiamo gestire stringhe scritte in formati speciali (json, csv, xml,…).

Python offre una ricca lista di librerie built-in per gestire questi formati.

- Import json per gestire json
- Import csv per gestire comma separated values
- Import xml (in particolare ElementTree) per gestire file xml
- …

- Un altra libreria molto utile è pickle, che serve a serializzare codice python