Indentazione

Informatica@DSS 2025/2026

Massimo Lauria <massimo.lauria@uniroma1.it>
https://massimolauria.net/informatica2025/

Indentazione

L'inserimento di spazio vuoto all'inizio della riga, per

- identificare blocchi logici di codice
- rendere il codice più leggibile

Esempio di indentazione annidata

```
def scontato(prezzo,sconto):                                    1
    if sconto < 0 or sconto > 100:                              2
        print("Errore nell'input")                              3
        print("Lo sconto deve essere tra 0 e 100")              4
        return                                                  5
                                                                6
    percentuale = 100 - sconto                                  7
    prezzo_scontato = prezzo * percentuale / 100                8
    return prezzo_scontato                                      9
                                                                10
print(scontato(1000,20))                                        11
print(scontato(500,15))                                         12
```

In Python l'indentazione è importante

Le istruzioni nello stesso blocco devono essere **allineate**

```
print("Prima riga")                                    1
  print("seconda riga")                                2
```

```
: Traceback (most recent call last):
:   File "<stdin>", line 1, in <module>
:   File "/tmp/babel-YszcCQ/python-VErkEL", line 2
:     print("seconda riga")
:     ^
: IndentationError: unexpected indent
```

```
print("Prima riga")                                    1
print("seconda riga")                                  2
```

```
Prima riga
seconda riga
```

Tab vs Spazi

Il carattere “tabulazione” (TAB) indica ”aggiungi un livello di indentazione”. Sfortunatamente

- è visivamente uguale a una sequenza di spazi
- la sequenza ha lunghezza differente (2,3,4,8… spazi) a seconda della visualizzazione.

```
<spazio><spazio><spazio><spazio>istruzione1
<tabulazione>istruzione2
```

In editor o terminali diversi si ottiene:

```
    istruzione1
    istruzione2
```

```
      istruzione1
    istruzione2
```

```
        istruzione1
            istruzione2
```

Tab vs Spazi in python3

In python3

- è vietato mischiare Tab e Spazi nell'identazione
- si consiglia di usare solo spazi (tipicamente 4 per livello)
- è possibile impostare l'editor così che inserisca 4 spazi ogni volta che si preme TAB.

Quanto indentare

Io suggerisco 4 spazi.

- la lunghezza dell'indentazione è facoltativa
- non compromettete la leggibilità

```
x = 12                                                          1
print("Primo livello di indentazione, 0 spazi")                2
if x > 0:                                                       3
  print("Secondo livello di indentazione, 2 spazi")             4
  if x<100:                                                     5
   print("Terzo livello di indentazione, 1 spazio")             6
else:                                                           7
     print("Secondo livello di indentazione, 5 spazi")          8
```

De-indentare

Ridure l'indentazione comunica al Python che la nuova istruzione fa parte di un blocco di codice più esterno, al quale questa **deve** essere allineata.

```
x = 10

def gruppo_istruzioni():
    print("Tizio")
    print('Caio')
    print("Sempronio")

  gruppo_istruzioni()
```

```
: Traceback (most recent call last):
:   File "<stdin>", line 1, in <module>
:   File "/var/folders/kf/p7km5ptj1p52hvz5nl6j9gr80000gn/T/babel-oj12Dc/python
:     gruppo_istruzioni()
:                       ^
: IndentationError: unindent does not match any outer indentation level
```

Commenti e righe vuote

Ignorati da python. L'indentazione è sempre considerata rispetto alla precedente riga contentente vero codice.

```
x = 10

def gruppo_istruzioni():
    print("Tizio")
 # commento mal indentato. Brutto ma corretto
    print('Caio')
    print("Sempronio")

        # altro commento mal indentato
gruppo_istruzioni()
```

```
Tizio
Caio
Sempronio
```