Funzioni

Corso Programmazione Python 2024

Modulo 4

Luca Di Pietro Martinelli

Parte del materiale deriva dai corsi dei proff. Paolo Caressa e Raffaele Nicolussi (Sapienza Università di Roma) e Giorgio Fumera (Università degli Studi di Cagliari)

Riutilizzo del codice

Uno dei concetti principali nella programmazione è il concetto di riutilizzo del codice.

Salvo casi particolari più un programma è complesso e strutturato, più saranno le righe di codice che lo compongono.

Affinché sia possibile tenere questo codice quanto più leggero, uniforme e semplice da mantenere è fondamentale evitare la presenza di porzioni che si ripetono.

Già abbiamo affrontato alcuni casi di riutilizzo del codice, ovvero:

- i cicli `while` e `for`
- alcune funzioni built-in già pronte, che possono essere usate più volte e in parti diverse dei nostri programmi.

In aggiunta alle funzioni integrate offerte da Python abbiamo la possibilità di scrivere le nostre funzioni personalizzate per fargli fare ciò che si desidera.

Chiamata di funzione

La funzione è uno strumento che ci permette di raggruppare un insieme di istruzioni che eseguono un compito specifico. Una funzione può accettare in input 0 o più argomenti (anche definiti parametri) per poi elaborarli e restituire in output un risultato.

Una volta definita una funzione è possibile eseguirla (operazione cui ci si riferisce spesso con la locuzione *chiamata di funzione*), passando argomenti diversi a seconda della situazione. Questo ci permette di rendere il codice più ordinato ed evitare ripetizioni.

Sintassi:

```
nome_funzione(arg1, arg2, ..., argn)
```

- `arg1`, `arg2`, ..., `argn` sono espressioni Python i cui valori costituiranno gli argomenti della funzione
- il numero degli argomenti e il tipo di ciascuno di essi (per es., numeri interi, numeri frazionari, stringhe, valori logici) dipende dalla specifica funzione; se il tipo di un argomento non è tra quelli previsti, si otterrà un messaggio d'errore
- come tutte le espressioni, anche la chiamata di una funzione produce un valore: questo coincide con il valore restituito dalla funzione.

Chiamata di funzione: argomenti

Sintassi:

```
nome_funzione(arg1, arg2, ..., argn)
```

Gli argomenti `arg1, arg2, ..., argn` di una chiamata di funzione sono costituiti da espressioni.

In ciascuno degli argomenti può quindi comparire un'espressione qualsiasi, purché produca un valore di un tipo previsto dalla funzione (in caso contrario si otterrà un messaggio d'errore).

Ne consegue, come caso particolare, che una chiamata di funzione può contenere tra le espressioni dei suoi argomenti altre chiamate di funzioni (chiamate annidate).

Definizione di funzione

Oltre ad usare le funzioni predefinite, è possibile creare nuove funzioni.

La definizione di una nuova funzione avviene attraverso l'uso della keyword `def`.

Sintassi:

```
def nome_funzione (par1, ..., parn):
    corpo_della_funzione
```

- `nome_funzione` è un nome simbolico scelto dal programmatore, con gli stessi vincoli a cui sono soggetti i nomi delle variabili
- `par1, ..., parn` sono nomi (scelti dal programmatore) di variabili, dette parametri della funzione, alle quali l'interprete assegnerà i valori degli argomenti che verranno indicati nella chiamata della funzione
- `corpo_della_funzione` è una sequenza di una o più istruzioni qualsiasi, ciascuna scritta in una riga distinta, con un rientro di almeno un carattere, identico per tutte le istruzioni.

La prima riga della definizione (contenente i nomi della funzione e dei parametri) è detta intestazione della funzione.

Definizione di funzione: return

Per concludere l'esecuzione di una funzione e indicare il valore che la funzione dovrà restituire come risultato della sua chiamata si usa l'istruzione `return`.

Sintassi:

```
return espressione
```

dove `espressione` è un'espressione Python qualsiasi.

L'istruzione `return` può essere usata solo all'interno di una funzione.

Se una funzione non deve restituire alcun valore:

- l'istruzione `return` può essere usata, senza l'indicazione di alcuna espressione, per concludere l'esecuzione della funzione
- se non si usa l'istruzione `return`, l'esecuzione della funzione terminerà dopo l'esecuzione dell'ultima istruzione del corpo.

Definizione e chiamata di funzione

L'esecuzione dell'istruzione `def` non comporta l'esecuzione delle istruzioni della funzione: tali istruzioni verranno eseguite solo attraverso una chiamata della funzione.

L'istruzione `def` dovrà essere eseguita una sola volta, prima di qualsiasi chiamata della funzione. In caso contrario, il nome della funzione non sarà riconosciuto dall'interprete e la chiamata produrrà un messaggio di errore.

L'interprete esegue la chiamata di una funzione nel modo seguente:

- copia il valore di ciascun argomento nel parametro corrispondente (quindi tali variabili possiedono già un valore nel momento in cui inizia l'esecuzione della funzione)
- esegue le istruzioni del corpo della funzione fino all'istruzione `return`, oppure fino all'ultima istruzione del corpo
- se l'eventuale istruzione `return` è seguita da un'espressione restituisce il valore di tale espressione come risultato della chiamata.

Chiamata di funzione in altra funzione

Nelle istruzioni del corpo di una funzione possono comparire chiamate di altre funzioni, sia predefinite che definite dall'utente.

Se si vuole chiamare una funzione predefinita appartenente a una delle librerie Python (come `math` o `random`) sarà necessario inserire prima della chiamata la corrispondente combinazione `from import`, che viene di norma inserita all'inizio del file che contiene il codice.

Esempio:

```
from math import sqrt

def ipotenusa (a, b):
    return sqrt(a ** 2 + b ** 2)
```

Chiamata di funzione all'esterno

Per poter chiamare dall'interno di una funzione un'altra funzione definita dall'utente sono disponibili due alternative:

- la definizione delle due funzioni deve trovarsi nello stesso file
- le due funzioni possono essere definite in file diversi, ma tali file dovranno trovarsi in una stessa cartella e nel file che contiene la chiamata dell'altra funzione si dovrà inserire l'istruzione

```
from nome_file import nome_funzione
```

dove:

- `nome_file` è il nome del file che contiene la definizione dell'altra funzione (senza l'estensione .py)
- `nome_funzione` è il nome della funzione.

Variabili locali

I parametri di una funzione e le eventuali altre variabili alle quali viene assegnato un valore all'interno di essa sono dette locali, in quanto vengono create dall'interprete nel momento in cui la funzione viene eseguita (con una chiamata) e vengono distrutte quando l'esecuzione della funzione termina.

Esempio:

```
n = 4

def stampa_quadrato(x):
    n = x ** 2
    print("Il quadrato di", x, "è", n)

stampa_quadrato(n)
print(n)
```

Console:

```
Il quadrato di 4 è 16
4
```

Variabili globali

Se invece si desidera utilizzare una variabile in tutto il programma, non solo all'interno di una funzione, si deve far uso della cosiddetta variabile globale. Tale variabile deve essere dichiarata al di fuori della funzione che desiderano utilizzarla e all'interno delle funzione deve essere dichiarata con la parole chiave `global`.

Esempio:

```
n = 4

def stampa_quadrato(x):
    global n
    n = x ** 2
    print("Il quadrato di", x, "è", n)

stampa_quadrato(n)
print(n)
```

Console:

```
Il quadrato di 4 è 16
16
```