Python for Technologies #2

Controllo di flusso e logica booleana

Alvaro Gaiotti — alvaro.gaiotti@randstad.it

Indice

Il controllo di flusso

Sinora abbiamo visto esempi di programmi lineari, ma spesso abbiamo la necessità di svolgere un'operazione od un'altra nel caso si verifichi o meno una determinata condizione: in questi casi il flusso del programma non è più lineare, ma inizia a prendere dei bivi o delle diramazioni.

Python, come molti altri linguaggi, mette a disposizione uno strumento apposito per situazioni di questo tipo: l'istruzione `if...elif...else`.

Esempio

Aiutiamo un utente a sapere se un numero è divisibile per 13. Dopo aver raccolto il numero richiesto (l'input `x`), ci troviamo di fronte a due casi:

1. Divisibile per 13 $\Longrightarrow$ `print(x, " è divisibile per 13")`
1. Non divisibile per 13 $\Longrightarrow$ `print(x, " non è divisibile per 13")`.

Possiamo usare la funzione `input(testo_da_stampare)` per ricevere il numero che l'utente vuole controllare.

Suggerimento

`input()` riceve in input un testo da stampare e restituisce in output quanto inserito dall'utente sulla riga di comando:
`numero_da_controllare = input("Inserisci un numero")`

Per la divisibilità, la scorsa lezione abbiamo visto un operatore che potrebbe aiutarci in questa situazione: lo ricordate?

Si tratta dell'operatore modulo (`%`): infatti `(13 * n) % 13 = 0`; quindi se `numero_da_controllare % 13` restituisce `0`, allora è divisibile per 13!

Attenzione!

Python è sensibile all'indentazione, ovvero al numero di spazi presenti prima dell'inizio di una riga: risulta quindi necessario, al seguito dell'istruzione `if`, inserire 4 spazi prima del testo delle righe appartenenti al *corpo* da eseguire se la condizione si verifica.

Se non sono presenti, l'interprete Python ritorna un `SyntaxError`.

Si ritorna alla normale indentazione al di fuori delle condizioni `if...elif...else`.

```python
"""Il numero è divisibile per 13?"""
# La funzione int(some) converte `some` in un `int`
n_da_controllare = int(input("Inserisci il numero: "))
if n_da_controllare % 13 == 0:  # Indentare!
    print(n_da_controllare, " è divisibile per 13")
    # Non indentare: non è il corpo dell'if!
else:                           # Indentare!
    print(n_da_controllare, " non è divisibile per 13")
    # Non indentare: non è il corpo dell'else!
print("Remember to rate my usefulness from 10 to 10!")
```

La logica booleanea

L'istruzione `if` valuta un espressione che può essere vera o falsa. Python, per rappresentare questi stati, utilizza il tipo `bool`, che può assumere i valori `True` e `False`.

Per ragionare su questi valori utilizziamo la logica booleana e i suoi operatori:

- `and`
- `or`
- `not`

La logica booleanea

Questa è una *tabella di verità* utile per combinare le condizioni in modo da ottenere il risultato di interesse:

A	B	A and B	A or B	not A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Gli operatori di comparazione

Spesso utilizziamo, nelle espressioni valutate dall'istruzione `if`, delle comparazioni tra valori o variabili. Per fare queste comparazioni utilizziamo gli operatori di comparazione, che restituiscono un `bool`:

`==`	uguale
`!=`	non uguale
`>`	maggiore
`>=`	maggiore uguale
`<`	minore
`<=`	minore o uguale

Esercizio — Saluta solo se...

Proviamo a scrivere uno script che saluti un utente solo se, dopo richiesta, l'utente conferma l'intenzione, fornendo la risposta `[s]ì`.

Suggerimento

Potete utilizzare gli operatori di comparazione anche sull'input dell'utente:

```
istruzione = input("Saluto o no? s=sì") # istruzione == 's'
```

Esercizio — Il più grande di tre numeri

Proviamo a scrivere uno script che, tra tre numeri forniti da un utente, stampi il più grande di questi tre.

Suggerimento

Potete pensare a due versioni: una che utilizza l'istruzione `if` e una che utilizza la funzione *built-in* `max()`

Esercizio — Tre lati di un triangolo

Scriviamo uno script che, dati tre lati di un potenziale triangolo, ci dica se è possibile costruirlo. Di seguito le formule di validità:

$$a > 0 \text{ and } a < b + c$$
$$b > 0 \text{ and } b < a + c$$
$$c > 0 \text{ and } c < a + b$$

Ricordate che anche i `bool` possono essere salvati nelle variabili!

```
lato1_ok = a > 0 and a < b + c
```

Cicli

Spesso abbiamo la necessità di effettuare le stesse operazioni più e più volte in maniera ripetitiva: per gestire questo tipo di situazioni, i linguaggi di programmazione hanno delle istruzioni per i *cicli*.

Python, in particolare, ha due di queste istruzioni:

- `while`
- `for`

Il ciclo `while`

Il ciclo while controlla se una condizione si verifica e, in caso positivo, esegue le istruzioni nel proprio *corpo*, in caso negativo esce dal ciclo e fa proseguire il programma:

```python
x = 0
while x <= 10:
    print(x)
    x += 1
print("Finito!")
```

Il ciclo `for`

Il ciclo `for` esegue il proprio corpo per un numero di volte pari agli elementi in una serie.

Codice equivalente a quello visto per il ciclo `while`:

```python
for x in range(11):
    print(x)
    x += 1
print("Finito!")
```

La funzione `range()`

La funzione `range()` viene utilizzata molto spesso per i cicli `for` in Python, ma non solo. Questa funzione produce una serie di numeri e ha la seguente sintassi:

```python
range(start = 0, stop, step = 1)
```

con il numero dopo “=” ad indicare un valore di default se non fornito. Possiamo anche contare all’indietro!

```python
range(10, 0, -1)
```

Esercizio — ...Lancio!

Scrivete uno script che esegue un conto alla rovescia e che produca un output simile a quanto segue:

```
3
2
1
Lancio!
```

Esercizio — Divisibili per 3 e 5

Scrivete uno script che chiede all'utente un numero `n` e stampa tutti i numeri tra `0` ed `n` divisibili per 3 e 5.

Esercizio — Quanti incontri in Serie A?

La Serie A è composta da 20 squadre. Nel corso di una stagione, ciascuna incontra ogni altra squadra due volte.

Scrivete uno script che calcoli il numero di partite che vengono svolte in una stagione.

Esercizio — Somma di quadrati

Scrivete un programma che stampi i quadrati dei numeri da 1 a `n`, con `n` inserito dall'utente e la loro somma.

Suggerimento

Usate un ciclo all'interno di un altro ciclo