

Programmazione in Python

Istruzioni

Dario Pescini - Mirko Cesarini

Università degli Studi di Milano-Bicocca

Dipartimento di Statistica e Metodi Quantitativi

Elementi di un programma

variabili

istruzioni

strutture di controllo

Elementi di un programma

variabili

istruzioni

strutture di controllo

Esecuzione

Esistono diversi tipi di istruzioni o operatori:

- assegnamento
- matematici, incremento/decremento
- confronto, logici
- I/O
- composti o espressioni
- per il controllo del flusso delle informazioni

operatori di assegnamento

Come indicato dal nome, servono ad **assegnare un valore ad una variabile**

```
a = 5
```

Assegnamento: =

- **a** variabile a cui assegnare il valore
- **=** operatore di assegnamento
- **5** valore da assegnare alla variabile

operatori aritmetici

- + somma
- sottrazione
- * prodotto
- ** elevamento a potenza
- / divisione
- // divisione tra interi
- % resto

operatori aritmetici

- + somma
- sottrazione
- * prodotto
- ** elevamento a potenza
- / divisione
- // divisione tra interi
- % resto

divisione /

Ha comportamento diverso se tra interi o tra reali:

```
>>> a = 5
>>> b = a / 2
>>> print(b)
2
```

```
>>> a = 5.0
>>> b = a / 2
>>> print(b)
2.5
```

divisione: Python 2 Vs. Python 3

	Python 2	Python 3
<code>3 / 2</code>	<code>1</code>	<code>1.5</code>
<code>3 // 2</code>	<code>1</code>	<code>1</code>
<code>3 / 2.0</code>	<code>1.5</code>	<code>1.5</code>
<code>3 // 2.0</code>	<code>1.0</code>	<code>1.0</code>

divisione - Esercizio

Scrivere un programma che prenda in input 2 numeri interi, a e b , e restituisca a video

- quoziente a / b
- resto $a \% b$
- prodotto del quoziente per b

divisione - Esercizio

Scrivere un programma che prenda in input 2 numeri interi, a e b, e restituisca a video

- quoziente a / b
- resto $a \% b$
- prodotto del quoziente per b

```
a = 257
b = 11

quoziente = a / b
resto = a % b
prodotto = b * quoziente

print "a = %d b = %d" % (a, b)
print "a / b= %d con resto %d" % (quoziente, resto)
print "b * quoziente = ", prodotto
print "prodotto + resto = ", prodotto + resto
```

divisione - Esercizio

Scrivere un programma che prenda in input 2 numeri reali, a e b , e restituisca a video

- quoziente a / b
- resto $a \% b$
- prodotto del quoziente per b

divisione - Esercizio

Scrivere un programma che prenda in input 2 numeri reali, a e b , e restituisca a video

- quoziente a / b
- resto $a \% b$
- prodotto del quoziente per b

```
a = 257.0
b = 11.0

quoziente = a / b
resto = a % b
prodotto = b * quoziente

print "a = %f b = %f" % (a, b)
print "a / b= %f con resto %f" % (quoziente, resto)
print "b * quoziente = ", prodotto
print "prodotto + resto = ", prodotto + resto
```

operatori +, * e stringhe

concatenazione: +

L'operatore di concatenazione + unisce due stringhe.

`stringaA + stringaB`

```
>>> stringA = 'Hello '  
>>> stringB = 'World !'  
>>> print(stringA + stringB)  
Hello World !
```

ripetizione: *

L'operatore di ripetizione * replica più volte la stringa.

`numero_volte * stringa`

```
>>> print(3 * stringA)  
Hello Hello Hello
```

operatori incremento/decremento

```
a += valore
```

incremento: +=

- `a` variabile da incrementare
- `+=` incrementa di `valore` la variabile `a`

operatori incremento/decremento

```
a += valore
```

incremento: +=

- `a` variabile da incrementare
- `+=` incrementa di `valore` la variabile `a`

```
>>> a = 0
>>> a += 5
>>> print(a)
5
```

operatori incremento/decremento

```
a -= valore
```

decremento: -=

- `a` variabile da decrementare
- `-=` decrementa di `valore` la variabile `a`

operatori incremento/decremento

```
a -= valore
```

decremento: -=

- `a` variabile da decrementare
- `-=` decrementa di `valore` la variabile `a`

```
>>> a = 0
>>> a -= 5
>>> print(a)
-5
```

operatori di confronto

> maggiore
>= maggiore o uguale
< minore
<= minore o uguale
== uguaglianza
!= disuguaglianza

is identità
is not non identità

Restituiscono **True** **False** a seconda che il confronto sia vero falso.

```
>>> a = 5  
>>> b = 10  
>>> a > b  
False
```

operatori di confronto

É possibile immagazzinare l'informazione {Vero, Falso} in una variabile di tipo "Booleano" $\in \{\text{True}, \text{False}\}$

```
>>> a = True
>>> type(a)
<type 'bool'>
>>> _
```

```
>>> a = True
>>> b = False
>>> c = a == b
```

operatori di appartenenza

Verificano se un elemento appartiene ad un gruppo:

`in` appartiene

`not in` non appartiene

Restituiscono `True` `False` a seconda che il test sia vero falso.

```
>>> s = 'appartenenza'
>>> print ('z' in s)
True
>>> print ('b' in s)
False
```

operatori logici

Ricevono in **ingresso** variabili di tipo “Booleano” e restituiscono **un valore** ancora Booleano:

$$F : \{\text{True}, \text{False}\}^n \rightarrow \{\text{True}, \text{False}\}$$

operatori logici

Ricevono in **ingresso** variabili di tipo "Booleano" e restituiscono **un valore** ancora Booleano:

not: {True, False} \rightarrow {True, False}

not

A	not(A)
V	F
F	V

value = **not** A

operatori logici

Ricevono in **ingresso** di variabili di tipo "Booleano" e restituiscono **un valore** ancora Booleano:

and: $\{\text{True}, \text{False}\}^2 \rightarrow \{\text{True}, \text{False}\}$

and

	V	F
V	V	F
F	F	F

value = A **and** B

operatori logici

Ricevono in **ingresso** di variabili di tipo "Booleano" e restituiscono **un valore** ancora Booleano:

or: $\{\text{True}, \text{False}\}^2 \rightarrow \{\text{True}, \text{False}\}$

or

	V	F
V	V	V
F	V	F

value = A **or** B

operatori I/O: print

```
print(una_Stringa)
```

print - scrittura

- `print()` è l'istruzione per stampare una stringa a video
- `una_Stringa` è la stringa che verrà visualizzata. si può sostituire la variabile con una stringa racchiusa tra singoli `' '` o doppi `" "` apici.

```
print ("del testo %d altro testo %d" % (var1, var2) )
```

print - scrittura

- `"del testo %d altro testo %d"` è la stringa che verrà visualizzata dove `%d` è l'identificatore di un parametro.
- `% (var1, var2)` è l'elenco dei valori che saranno assegnati ai parametri

operatori I/O - formattazione

Per inserire nelle stringhe dei valori associati a variabili si utilizzano degli identificatori di formattazione:

%d	sostituisci con un intero
%f	sostituisci con un float
%r	sostituisci con contenuto raw della variabile
%s	sostituisci con una stringa
\t	sostituisci con una tabulazione
\n	sostituisci con un a capo

operatori I/O: print

```
>>> a = 10

>>> print("si possono" + "concatenare le stringhe e mescolare
valori numerici", a)
si possonoconcatenare le stringhe e mescolare valori numerici
10

>>> print("pippo %d " % 10)
pippo 10

>>> print("pippo %d e %f" % (10, 10.0))
pippo 10 e 10.000000

>>> print("pippo %d e %r" % (10, 10.0))
pippo 10 e 10.0

>>> print("pippo %d e %s" % (10, 10.0))
pippo 10 e 10.0
```

operatori I/O: input

```
una_variabile = input("inserisci un valore")
```

input - lettura

- `input()` è l'istruzione per leggere un valore da terminale
- `"inserisci un valore"` è la stringa che viene mostrata all'utente
- `una_variabile` è la variabile a cui assegnare il **valore**

```
(var1, var2) = input("inserisci due valori")
```

input - lettura

- `"inserisci due valori"` è la stringa che viene mostrata all'utente
- `(var1, var2)` sono le due variabili a cui assegnare i valori numerici inseriti e separati da `,`

operatori I/O: raw_input

```
una_variabile = raw_input("inserisci del testo")
```

raw_input - lettura

- `raw_input()` è l'istruzione per leggere un valore da terminale
- `"inserisci del testo"` è la stringa che viene mostrata all'utente
- `una_variabile` è la variabile a cui assegnare la **stringa** inserita dall'utente