Operatore modulo

Mirko Cesarini - Dario Pescini
nome.cognome@unimib.it

Università di Milano Bicocca

Operatore modulo

- Operatore modulo, rappresentato dal simbolo %
- Secondo voi, che cosa fa l'operatore modulo?

```
1 print(5 % 3)
```

```
2
```

```
2 a=31
3 b=7
4 c = a % b
5 print(c)
```

```
3
```

- Risposta: l'operatore modulo calcola il resto della divisione intera tra due numeri
- Si può applicare anche a valori di tipo float?

```
6 print(5 % 3.2)
```

- Che cosa viene visualizzato?

```
1.8
```

Esempio d'uso di %

- Secondo voi che cosa fa l'algoritmo qua sotto?

```
a=int(input('Inserisci un numero: '))
if a%2 == 0:
    print('Pari')
else:
    print('Dispari')
print('Fine programma')
```

- Determina se un numero è pari o dispari

Polimorfismo operatore %

- Abbiamo già visto l'operatore % usato in un contesto completamente diverso

```
1 a=10
2 b=5
3 print("Dei %d passeg. saliti, %d sono scesi" % (a,b))
```

```
Dei 10 passeg. saliti, 5 sono scesi
```

- Qua sopra, % è applicato a due operandi: una stringa e (a,b) . . . vedremo più avanti che cos'è una struttura dati con valori racchiusi tra parentesi
- Nel caso di operandi solo numerici, il comportamento è:

```
1 print(7%4)
```

```
3
```

- Polimorfismo: a seconda dei contesti, un operatore (nel nostro caso %) ha dei comportamenti diversi

Commenti

Mirko Cesarini - Dario Pescini
nome.cognome@unimib.it

Università di Milano Bicocca

Commenti

- In uno script python

```
1 lato=5
2 raggio=10
3 PI=3.14 #valore pi greco
4 # qua sotto c'e' un errore. Quale?
5 print("area circonferenza: %f " % (lato**2)) #** elevam. al quadrato
```

- Tutto ciò che si trova a destra del simbolo # verrà ignorato dall'interprete Python
- Utile per
  - inserire promemoria, commenti, ...
  - comunicare all'interprete di (temporaneamente) non interpretare una o più righe ...senza doverle cancellare

Commenti 2

- In uno script python

```
1 lato=5
2 raggio=10
3 PI=3.14 #valore pi greca
4 # Versione corretta
5 print("area circonferenza: %f " % (PI*raggio**2)) #** elevam. al quadrato
```

- Tutto ciò che si trova a destra del simbolo # verrà ignorato dall'interprete Python
- Utile per
  - inserire promemoria, commenti, ...
  - comunicare all'interprete di (temporaneamente) non interpretare una o più righe ...senza doverle cancellare

Commenti: escamotage frequenti

- Trucco per trovare un errore in una riga
  - Commentate la riga che contiene l'errore
  - Provate a riscrivere il comando nella riga sotto, un passo alla volta ...
- Per evitare temporaneamente che l'interprete processi alcune righe
  - Dovete scrivere un # all'inizio di ogni riga che volete escludere dall'interpretazione
  - L'effetto del # termina con la fine della riga
- Altri linguaggi di programmazione mettono a disposizione costrutti per commentare più linee in un colpo solo. Python offre non uno strumento ma un escamotage ...

Stringhe su più righe

- Il comando

```
a="""
oggi abbiamo
      quasi
finito
"""
# Dentro la stringa precedente non devono
# essere rispettate le regole di indentazione
```

- E' equivalente al comando

```
a="oggi abbiamo\n      quasi\nfinito\n"
```

- Le 3 virgolette permettono di iniziare e terminare stringhe che si estendono su più righe
  - Nota bene: le 3 virgolette vanno scritte tra loro attaccate
  - Al posto delle 3 virgolette si possono usare i 3 apici ' ' ' (scritti sempre attaccati)

Commenti: escamotage frequenti 2

- Per commentare su più righe ...
- ... potete trasformare le righe da commentare in una stringa che si estende su più righe, es.

```
a="""
istruzione 1
...
istruzione n
"""
```

- ... Non occorre assegnare la stringa ad una variabile, es.

```
"""
istruzione 1
...
istruzione n
"""
```