Programmazione in Python

Moreno Marzolla
http://www.moreno.marzolla.name/

Python

- Linguaggio di programmazione interpretato
- Inventato negli anni '80 da Guido van Rossum
- Versione attuale del linguaggio: 3.x

Chi usa Python?

- Google
- NASA
- Borsa di New York
- Industrial Light & Magic (quelli che fanno gli effetti speciali di Star Wars)
- Yahoo! (Yahoo mail e gruppi)
- RealNetworks (usato per il testing del sistema)
- RedHat (strumenti per l'installazione di Linux)
- LLNL, Fermilab (controllo di applicazioni scientifiche)
- ...

Python (versione 3)

- Si può utilizzare Python in modalità interattiva o mediante file di programmi
- Modalità interattiva
  - È possibile scrivere espressioni che vengono valutate quando si preme il tasto <Invio>
  - L'interprete mostra subito il risultato delle espressioni
- File di programmi
  - Si scrivono dei file di testo contenenti un programma Python costituito da una sequenza di istruzioni
  - L'interprete esegue l'intero programma, leggendo le istruzioni una dopo l'altra dal file

Valori numerici

- 42 (intero, decimale)
- 0x2A (intero, esadecimale)
- 0.15 (numero reale)
- 1.7e2 (numero reale, notazione scientifica)

Le nostre prime espressioni

- L'espressione più semplice è quella composta da un valore numerico. Python risponde visualizzando il numero che abbiamo inserito

```
>>> 42
42
>>> 13.7
13.7
```

Le nostre prime espressioni

- Possiamo scrivere espressioni più complesse, e Python risponde visualizzandone il valore

```
>>> 2+2
4
>>> 13+7*2
27
>>> 7/2
3.5
>>> 7 // 2          Divisione intera
3
>>> 7 % 2           Resto della Divisione intera
1
```

Stringhe

- Oltre a dati di tipo numerico, possiamo anche usare dati di tipo stringa

```
>>> “pippo”
'pippo'
```

- L'operatore somma (+) concatena due stringhe

```
>>> “pippo” + “pluto”
'pippopluto'
```

- Non è possibile applicare l'operatore somma ad una stringa e ad un numero

```
>>> "pippo"+1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and
'int' objects
```

Stringhe

- Per concatenare una stringa e un numero, occorre trasformare il numero in stringa mediante la funzione str()

```
>>> “pippo”+str(1)
'pippo1'
>>> “pippo”+str(135.2)
'pippo135.2'
```

Operazioni su stringhe

- ...Invece è possibile applicare l'operatore prodotto (*) ad una stringa e ad un numero, e viceversa

```
>>> "pippo"*2
'pippopippo'
>>> 2*"pippo"
'pippopippo'
```

- Possiamo scrivere espressioni più complicate che coinvolgono stringhe

```
>>> (2*"pippo")+"pluto"
'pippopippopluto'
>>> "(" + "pippo"*2 + ")"
'(pippopippo)'
```

Operazioni su stringhe

- È possibile usare la funzione int() per convertire una stringa in un intero, e la funzione float() per convertire una stringa in un numero reale

```
>>> int(“123”)
123
>>> int(“-13”)
-13
>>> int(“13.2”)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '13.2'
>>> float(“13.2”)
13.2
>>> float(“-1”)
-1.0
```

Operatori aritmetici e logici

- Python supporta i valori logici **`True`** e **`False`**
  - Nota: 0, 0.0, “” equivalgono a False
- Operatori aritmetici
  - Le quattro operazioni: a+b, a-b, a*b, a/b
  - Elevamento a potenza: a**b calcola “a elevato alla b”
  - a%b è il resto della divisione intera a/b
- Operatori di confronto
  - a < b, a > b, a <= b, a >= b
  - a == b, a != b
- Operatori booleani
  - a or b (vero se a è vero OPPURE b è vero)
  - a and b (vero se a e b sono ENTRAMBI veri)
  - not a (vero se a è falso)

Variabili

- Una variabile è una etichetta (nome) a cui è assegnato un valore
- Una variabile non può essere utilizzata prima che le venga assegnato un valore
- I nomi delle variabili possono essere composti da lettere, numeri e dal carattere underscore (_)
  - Il primo carattere non può essere un numero
- Esempi di nomi validi
  - `x` `ciao` `x13` `x1_y` `_` `_ciao12`
- Esempi di nomi non validi
  - `1x` `x-y` `$a` `però`

Assegnamenti

- In generale è possibile scrivere cose come:

```
>>> base=12
>>> altezza=3
>>> area=(base*altezza)/2
>>> area
18
```

- Questo viene valutato come segue:
  - Prima si valuta l'espressione a destra dell'operatore di assegnamento (il simbolo '=')
  - Il risultato diventa il valore della variabile il cui nome è sinistra del simbolo '='
- Le istruzioni sopra assegnano il valore 18 a **`area`**

Attenzione

- Quanto vale:

```
>>> x = 10
>>> x = x + 1
```

- **x=x+1**, vista come equazione aritmetica, non è risolvibile...
- ...ma questo è Python !
  - Prima si valuta la parte destra (x+1 vale 10+1=11)
  - Poi si assegna tale valore alla parte sinistra
- Dopo l'assegnamento, il nuovo valore di **x** è 11

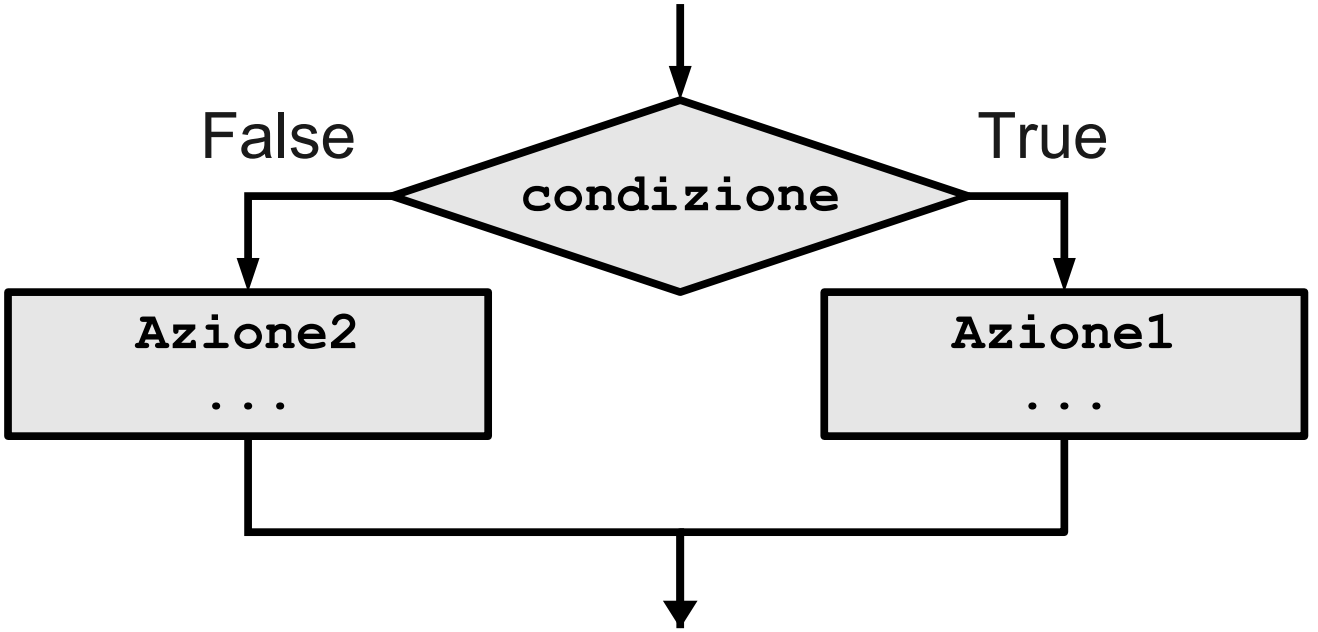
Attenzione

- Quanto vale

```
>>> x = 10
>>> y = 12
>>> z = 2
>>> risultato = x*(2 + y) - 7*z;
```

- Valgono le solite regole di precedenza degli operatori
  - Il prodotto ha precedenza maggiore della somma
  - x*(2+y) – 7*x =
    10*(2+12) - 7*2 =
    10*14 – 7*2 =
    140 - 14 =
    126

Istruzione condizionale “if-else”

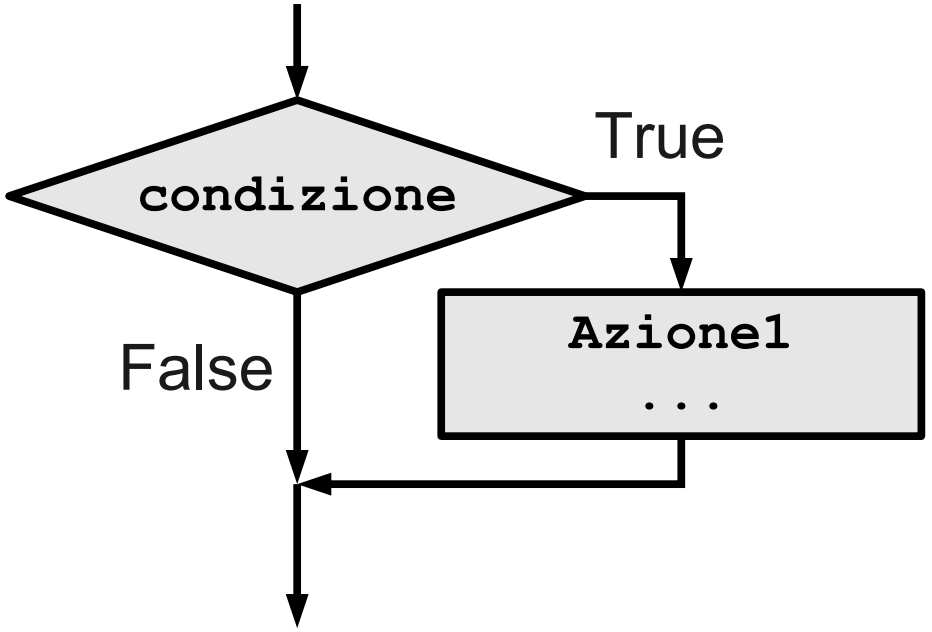


Python

```
if condizione:
    Azione1
    ...
else:
    Azione2
    ...
```

```
x=10
y=5
if x >= y:
   print(x, “maggiore o uguale a“, y)
else:
   print(x, “minore di“, y)
```

Istruzione condizionale “if-else”

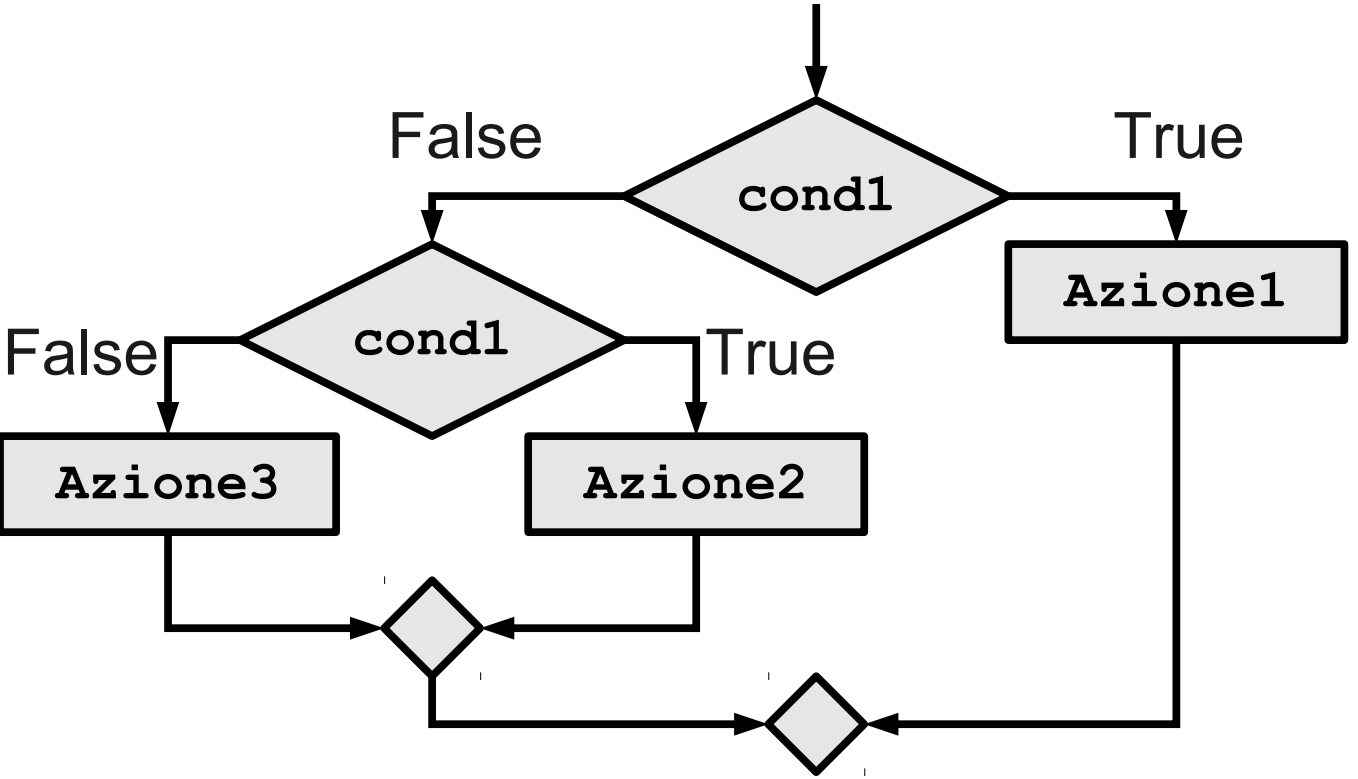


Python

```
if condizione:
    Azione1
    ...
```

```
if 0:
    print(“aaa”)
print(“bbb”)
```

Istruzione condizionale “if-else”



Python

```
if cond1:
    Azione1
    ...
elif cond2:
    Azione2
    ...
else:
    Azione3
    ...
```

- Nota: ci possono essere tanti “elif” quanti si vuole
- “Azione1”, “Azione2” ecc. possono essere composte da più righe di codice, purché siano tutte indentate allo stesso livello

Esempio

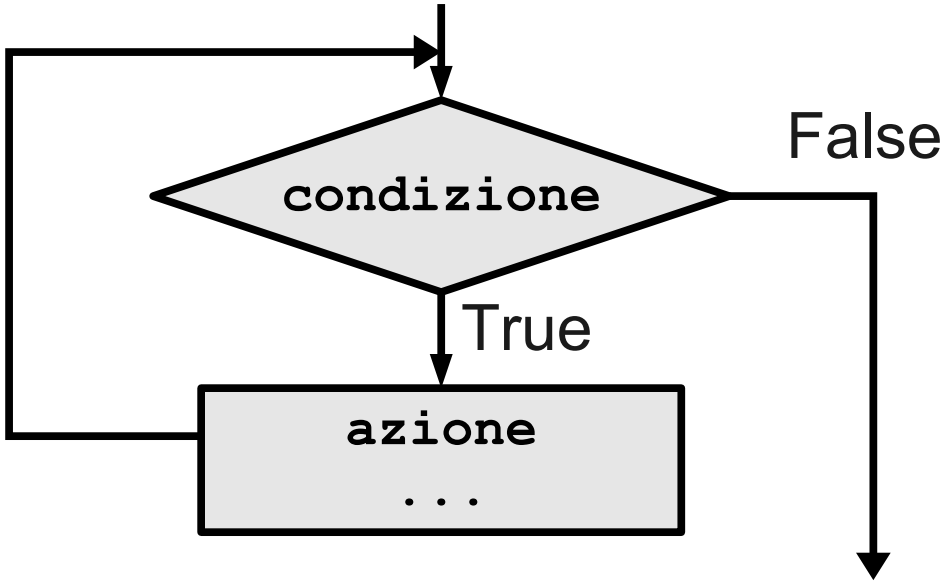
```
x = int(input("Scrivi un numero "))
if x>0:
   print(x, "e' positivo")
elif x == 0:
   print(x, "e' zero")
else:
   print(x, "e' negativo")
```

Esercizio

- Leggere da tastiera tre numeri interi da assegnare ad altrettante variabili diverse
- Scrivere a video il valore più grande tra quelli inseriti
- Esempio:

```
Digita un numero: 3
Digita un numero: 2
Digita un numero: 7
Il valore massimo e' 7
```

Ciclo “while”



Python

```
while condizione:
    azione
    ...
```

```
x=1
while x<4:
    print(x**2)
    x=x+1
print(“Fine”)
```

L'output generato

```
1
4
9
Fine
```

Esercizio

- Chiedere all'utente un intero N>0
- Chiedere all'utente N numero interi
- Stampare la somma di tutti gli N valori inseriti dall'utente

```
Digita N 3
Digita un valore 1
Digita un valore 31
Digita un valore -2
La somma dei valori inseriti e' 30
```

Possibile soluzione

```
N=int(input("Digita N "))
somma=0
while N>0:
  a=int(input("Digita un valore "))
  somma = somma + a
  N = N - 1
print("La somma dei valori inseriti e'", somma)
```

Errore da evitare

- Se la condizione di un ciclo while rimane sempre vera, il ciclo non termina mai! Quindi è importante assicurarsi che “prima o poi” la condizione diventi falsa
- Esempio

```
x=1
while x == 1:
   print(“ciao”)
```

questo programma continua a stampare “ciao” finché non lo si interrompe con Ctrl+C

Esercizi

- Scrivere un programma Python che calcola il prodotto di due interi positivi *x* e *y* utilizzando la somma
  - Cioè, x*y deve essere calcolato come x+x+... + x (y volte)
- Scrivere un programma Python che chiede in input un intero positivo N e stampa a video i numeri pari compresi tra 1 e N (estremi inclusi)
  - Es: se N=13, stampa i numeri 2, 4, 6, 8, 10 e 12
  - Es: se N=1 non stampa nulla
- Modificare l'esercizio precedente per stampare tutti i numeri dispari compresi tra 1 e N (estremi inclusi)