Programmazione in Python

strutture di controllo

Dario Pescini - Mirko Cesarini

Università degli Studi di Milano-Bicocca

Dipartimento di Statistica e Metodi Quantitativi

dario.pescini@unimib.it

Outline

Programmare....

Per evitare disastro bisogna controllare

ordine delle istruzioni

strutture di controllo

Flusso dell' informazione $\equiv$ Strutture di controllo

Sequenza preserva l'ordine delle istruzioni

Selezione sceglie il blocco di istruzioni in base ad una condizione

Iterazione ripete un blocco di istruzioni

Strutture di controllo

Sequenza

Selezione
- if
- if ... else
- if ... elif ... else

Iterazione
- while
- for

Strutture di controllo

Sequenza

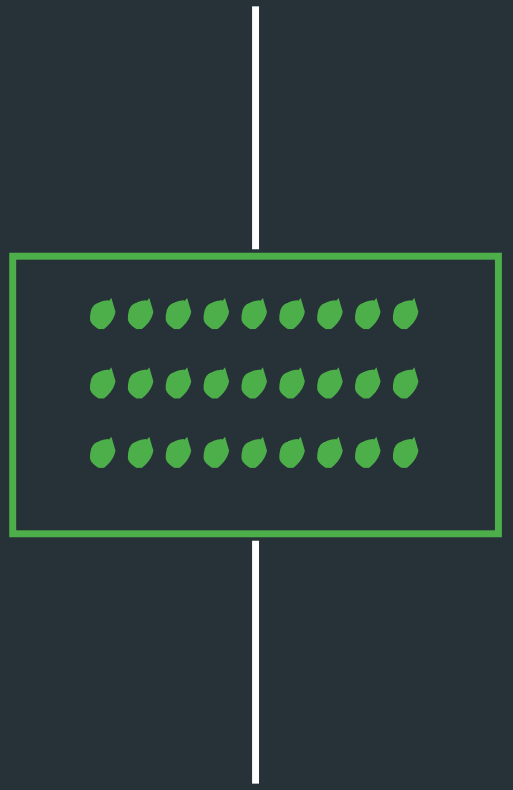
Selezione
- if
- if ... else
- if ... elif ... else

Iterazione
- while
- for

!! in Python i blocchi di istruzioni si isolano tramite indentazione
. . . . (4 spazi) !!

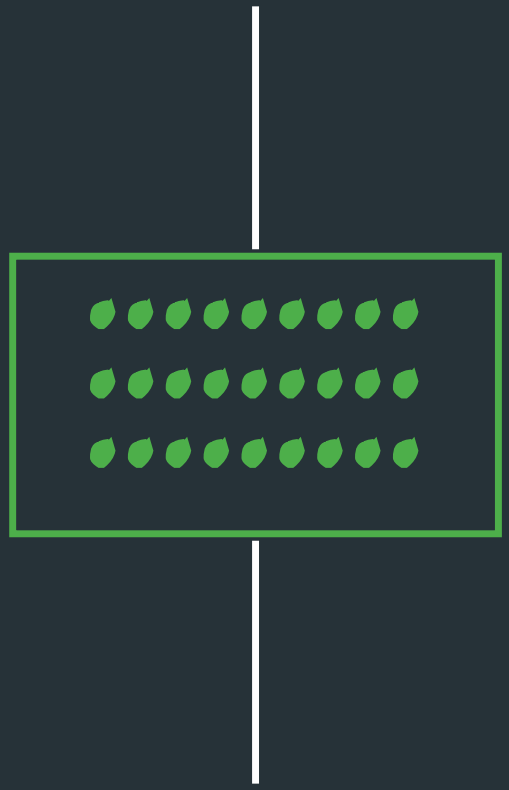
Sequenza

```
istruzione1
istruzione2
   ....
istruzioneN
```



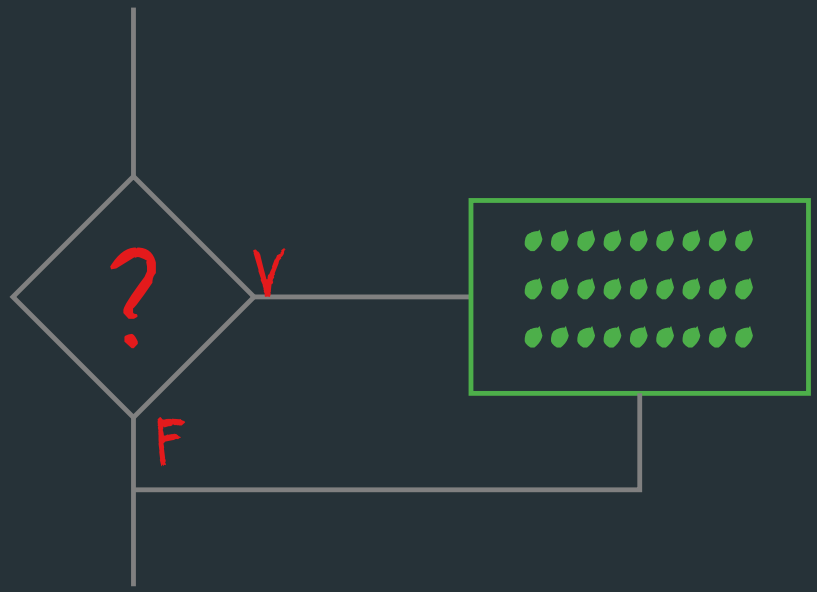
Sequenza

```
a=1
b=3
print "a = ",a, "b = ", b
temp = a
a = b
b = temp
print "a = ",a, "b = ", b
```



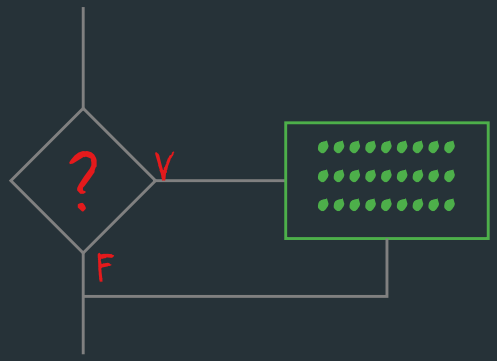
Strutture controllo - Selezione: IF

```
if condizione :
....lista comandi
```



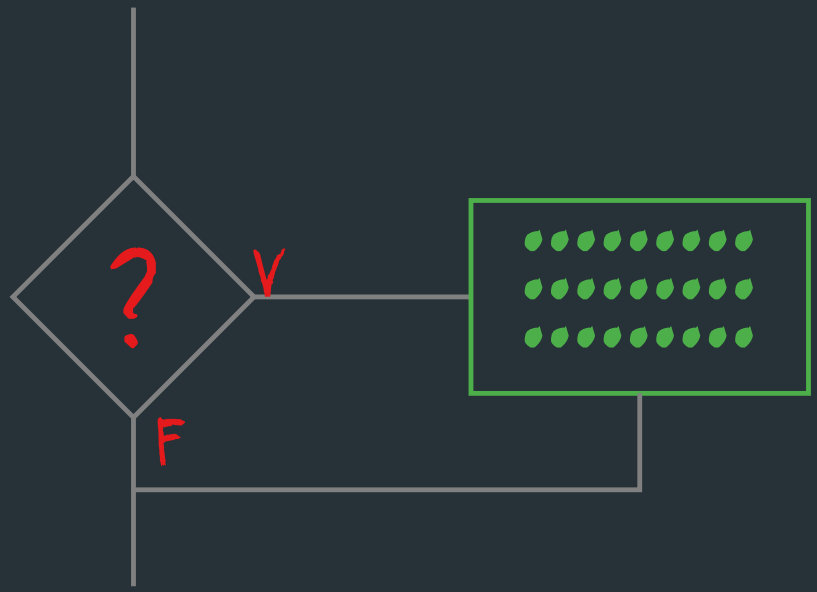
Strutture controllo - Selezione: IF

```
a = input ("Inserire un numero intero a:\n")

if a < 0 :
    a = -a

print "Il valore assoluto di a è ", a
```



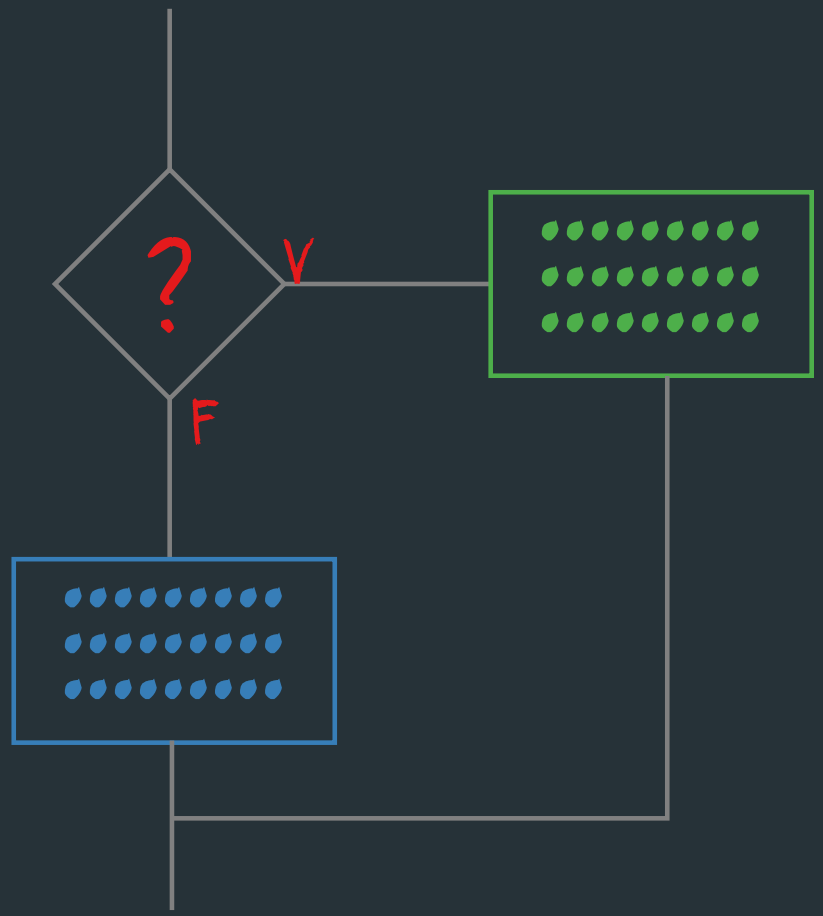
Strutture controllo - Selezione: IF

```
dario@vulcano: python if.py
Inserire un numero intero a:
5
Il valore assoluto di a e'  5
dario@vulcano: python if.py
Inserire un numero intero a:
-5
Il valore assoluto di a e'  5
```



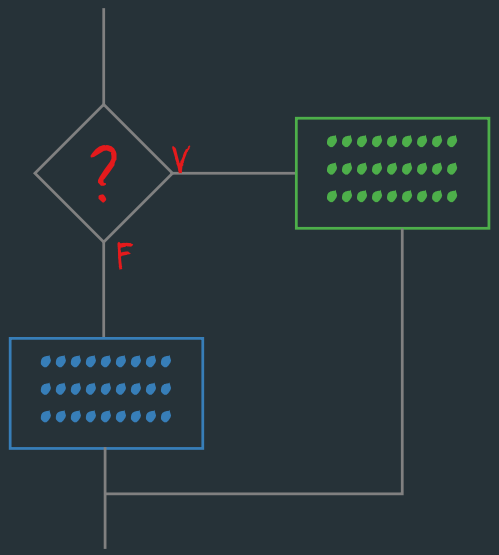
Strutture controllo - Selezione: IF - ELSE

```
if condizione :
    lista comandi
else :
    lista comandi
```



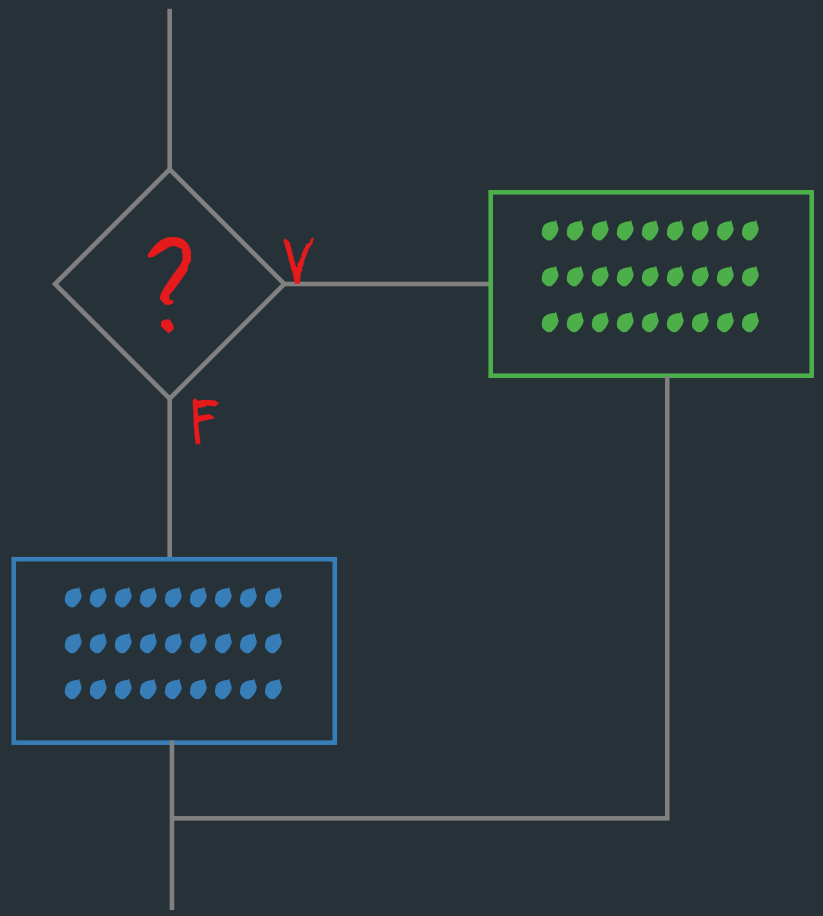
Strutture controllo - Selezione: IF - ELSE

```
print "Inserire due numeri interi:\n"
a = input("primo numero ")
b = input("secondo numero ")

if a > b :
    print "Il maggiore dei due è a!\n"

else :
        "Il maggiore dei due è b!\n"
```



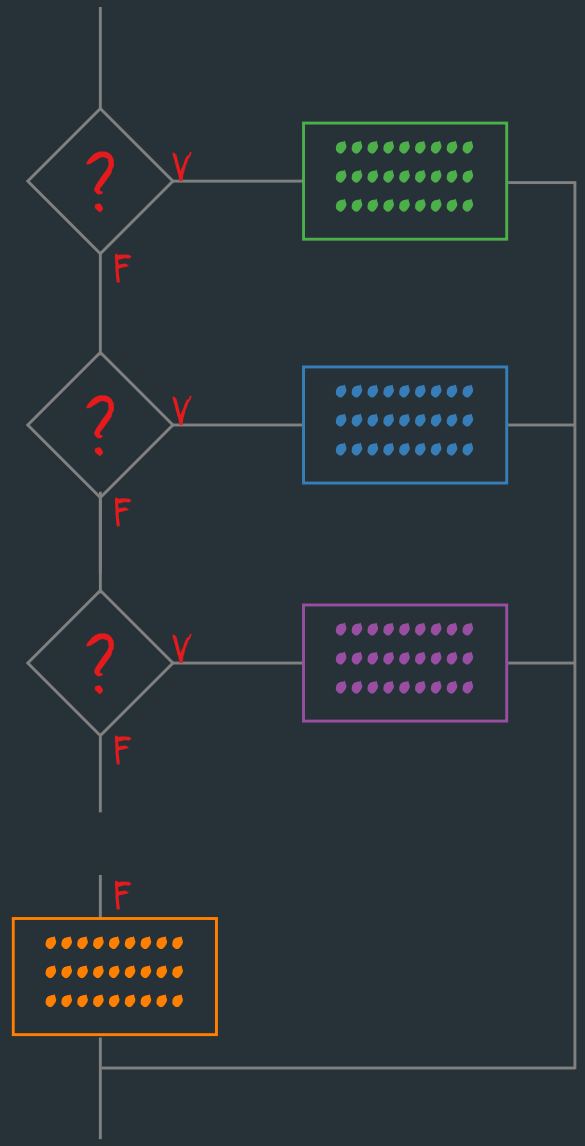
Strutture controllo - Selezione: IF - ELSE

```
dario@vulcano: python ifElse.py
Inserire due numeri interi:

primo numero 3
secondo numero 5
Il maggiore dei due e' b!

dario@vulcano: python ifElse.py
Inserire due numeri interi:

primo numero 5
secondo numero 3
Il maggiore dei due e' a!
```

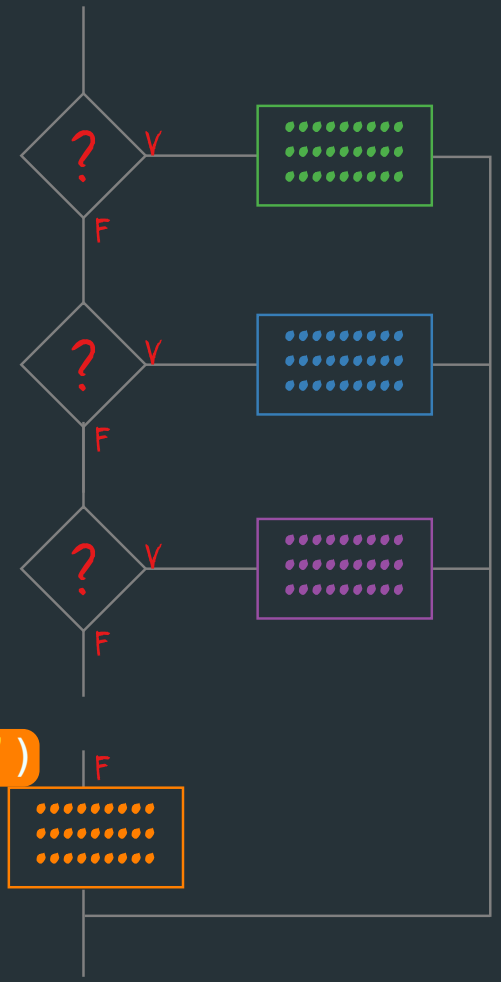


Strutture controllo - Selezione concatenata: IF-ELIF-ELSE

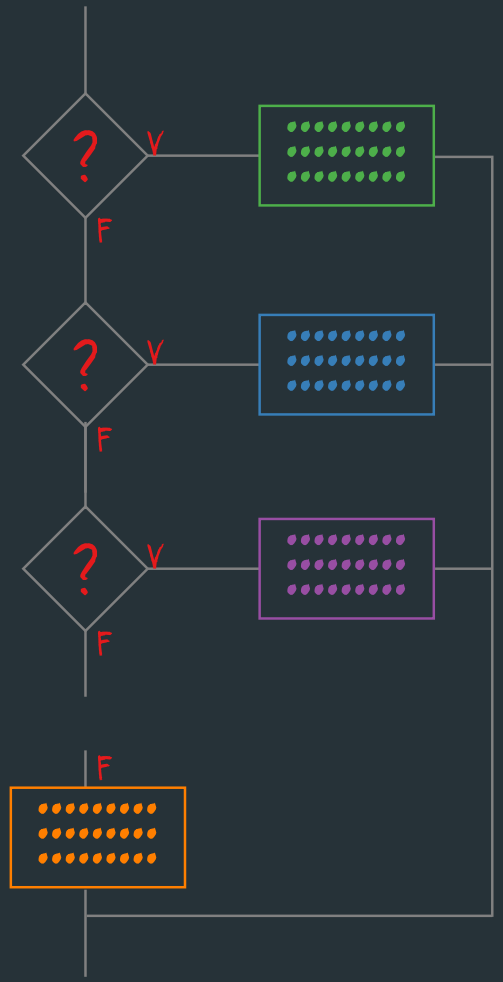
```
if confronto1 :
    lista comandi
elif confronto2 :
    lista comandi
...
else :
    lista comandi
```



```
selezione = raw_input("Scegliere una delle
    seguenti opzioni: a,b,c\n")

if selezione == 'a':
    print "hai scelto l'opzione a\n"
elif selezione == 'b':
        "hai scelto l'opzione b\n")
elif selezione == 'c':
    print "hai scelto l'opzione c\n")

else:

    print "hai scelto un'opzione non valida...\n")
```



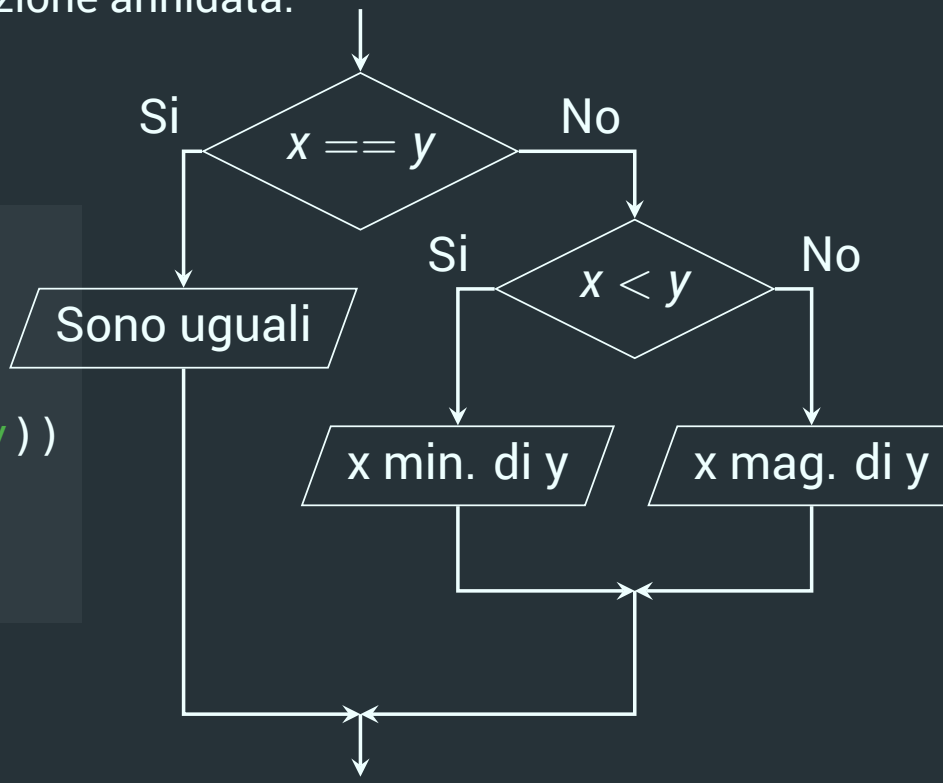
```
dario@vulcano: python Elif.py
Scegliere una delle seguenti opzioni: a,b,c
'd'
hai scelto un'opzione non valida...

dario@vulcano: python Elif.py
Scegliere una delle seguenti opzioni: a,b,c
'b'
hai scelto l'opzione b
```



Condizioni annidate

Una struttura condizionale può essere inserita nel corpo di un'altra struttura condizionale. Si parla in questo caso di condizione annidata:

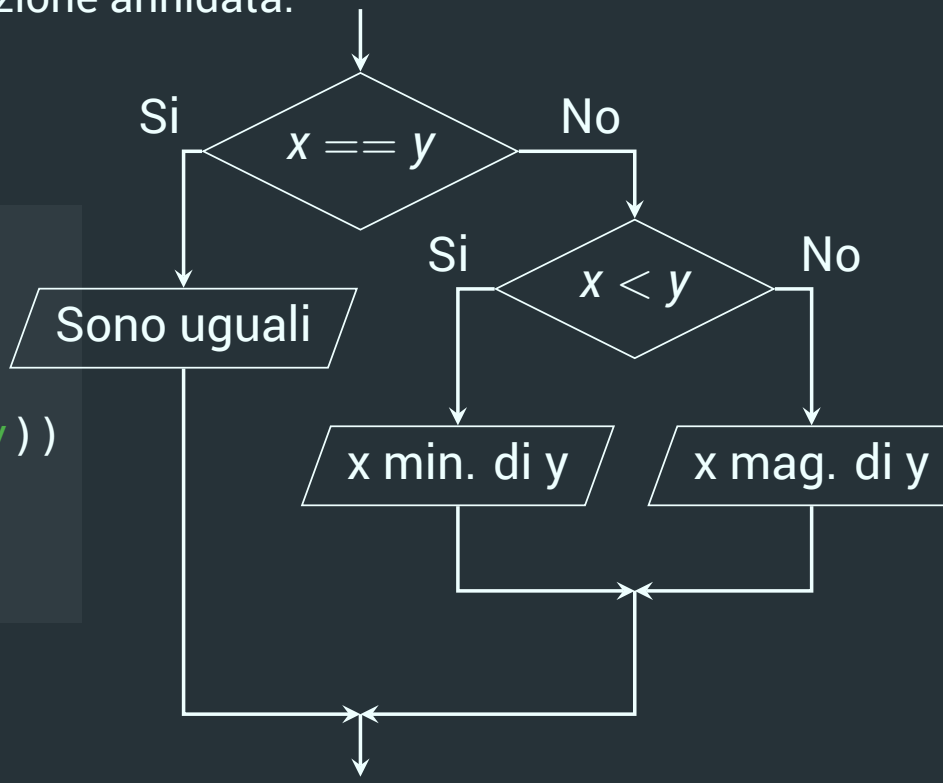
```
if x == y:
   print("sono uguali")
else:
   if x < y:
      print (str(x)+"e' minore di"+str(y))
   else:
      print (str(x)+"e' maggiore
    di"+str(y))
```



Condizioni annidate

Una struttura condizionale può essere inserita nel corpo di un'altra struttura condizionale. Si parla in questo caso di condizione annidata:

```
if x == y:
   print("sono uguali")
else:
   if x < y:
      print (str(x)+"e' minore di"+str(y))
   else:
      print (str(x)+"e' maggiore
    di"+str(y))
```

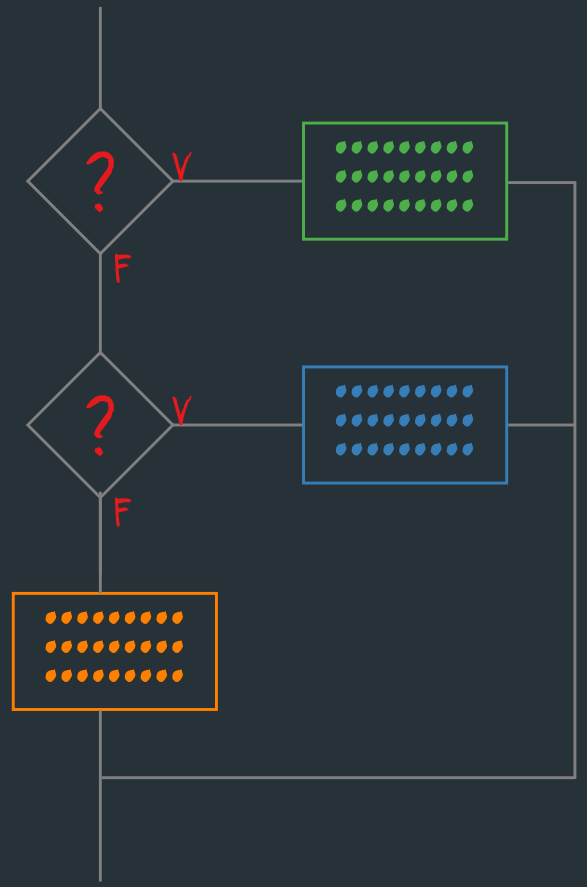


Python usa l'indentazione per decidere a quale ramificazione appartiene un'istruzione

Strutture controllo - Selezione concatenata: IF-ELIF-ELSE

```
if confronto1 :
    lista comandi
elif confronto2 :
    lista comandi
else :
    lista comandi
```

```
if confronto1 :
    lista comandi
else :
    if confronto2 :
        lista comandi
    else :
        lista comandi
```



Esercizio

Scrivere un programma che, dati tre parametri a, b, c, determini le due radici x_1 e x_2 dell'equazione $ax^2 + bx + c = 0$.

Si ricorda che $x_{1,2} = -b \pm \frac{\sqrt{b^2 - 4ac}}{2a}$

```
a = input("Immetti il coefficiente a ")
b = input("Immetti il coefficiente b ")
c = input("Immetti il coefficiente c ")
print ("Data l'equazione algebrica " + str(a) + "*X^2+" +
    str(b) + "*X+" + str(c) + "=0 ")
delta = b*b - 4*a*c
rad_delta=delta**0.5
x1 = -(b - rad_delta)/(2*a)
x2 = -(b + rad_delta)/(2*a)
print ("Le soluzioni sono: " + str(x1) + " e " + str(x2))
```

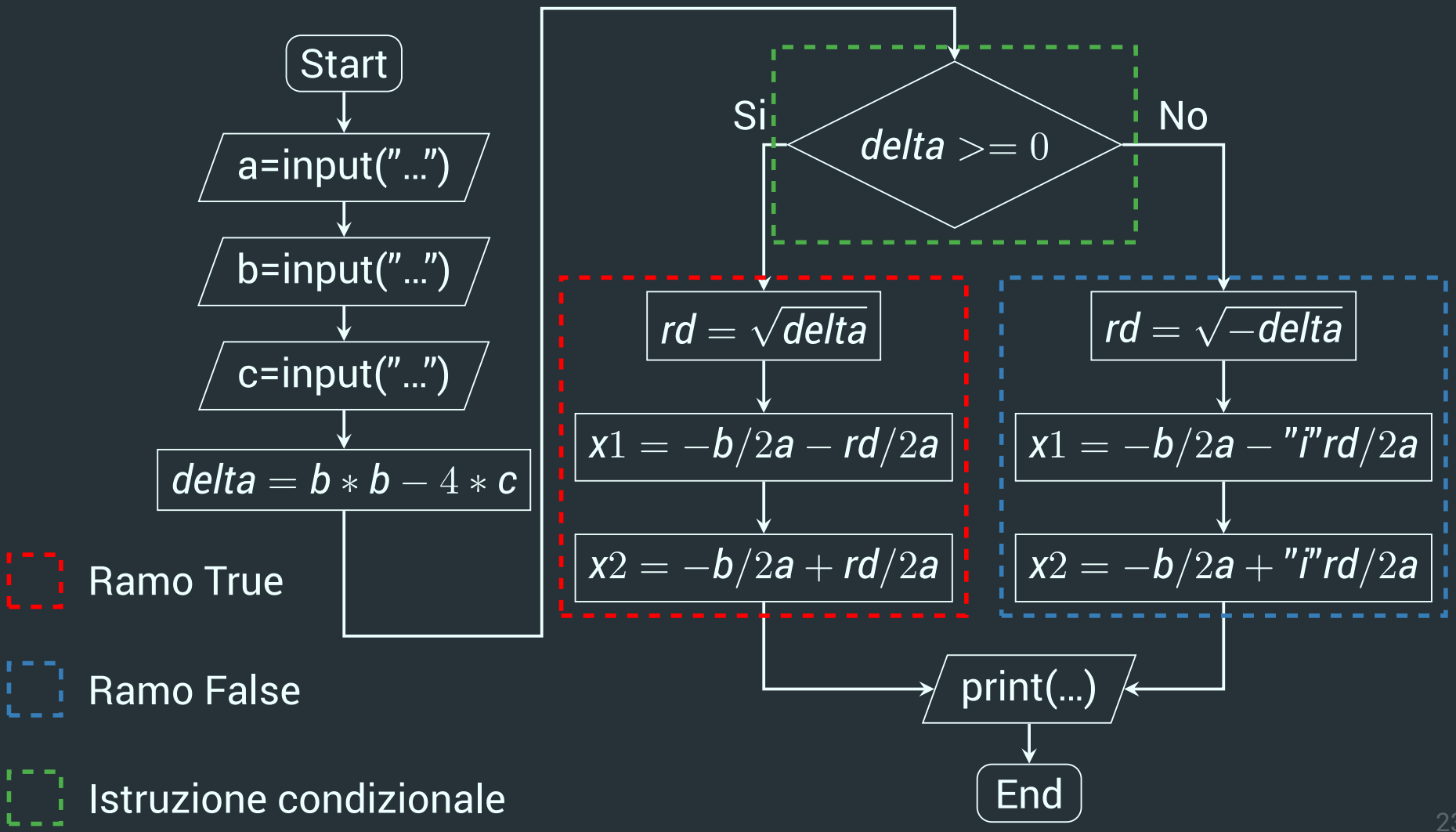
...eseguiamo il programma...

```
dario@vulcano: ipython radiciWrong.py
Immetti il coefficiente a 4
Immetti il coefficiente b 4
Immetti il coefficiente c 4
Data l'equazione algebrica 4*X^2+4*X+4=0
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/Users/dario/ownCloud/Works/Didattica/Lezioni/Informatica/Esercizi/radiciWrong.py in <module>()
      5         "*X^2+" + str(b) + "*X+" + str(c) + "=0 ")
      6 delta = b * b - 4 * a * c
----> 7 rad_delta = delta**0.5
      8 x1 = -(b - rad_delta) / (2 * a)
      9 x2 = -(b + rad_delta) / (2 * a)

ValueError: negative number cannot be raised to a fractional power
```

...debuggiamo...

- Il problema nasce in questo gruppo di istruzioni:
```
delta = b*b - 4*a*c
rad_delta = delta**0.5
```
- quando delta assume un valore negativo
  - Python non esegue la radice quadrata di un numero negativo (si potrebbero utilizzare le librerie per manipolare i numeri complessi ..., ma per ora non consideriamo questa opzione)
- Cosa si può fare?
  - Il programma dovrebbe comportarsi in modo diverso a seconda del valore della variabile delta

...miglioriamo l'algoritmo...



...implementiamo nuovamente...

```
a = input("Immetti il coefficiente a ")
b = input("Immetti il coefficiente b ")
c = input("Immetti il coefficiente c ")
print ("Data l'equazione algebrica " + str(a) +
       "*X^2+" + str(b) + "*X+" + str(c) + "=0 ")
delta = b * b - 4 * a * c
if delta >= 0:
    rad_delta = delta**0.5
    x1 = -(b - rad_delta) / (2 * a)
    x2 = -(b + rad_delta) / (2 * a)
    print ("Le soluzioni sono: " + str(x1) + " e " + str(x2))
else:
    rad_delta = (-delta)**0.5
    reale = -b / (2 * a)
    imm = rad_delta / (2 * a)
    print ("Le soluzioni sono: " + str(reale) + " + i " +
    str(imm) +
           " e " + str(reale) + " - i " + str(imm))
```

...eseguiamo il programma...

```
dario@vulcano: ipython radici.py
Immetti il coefficiente a 4
Immetti il coefficiente b 4
Immetti il coefficiente c 4
Data l'equazione algebrica 4*X^2+4*X+4=0
Le soluzioni sono: -1 + i 0.866025403784 e -1 - i 0.866025403784
dario@vulcano: _
```

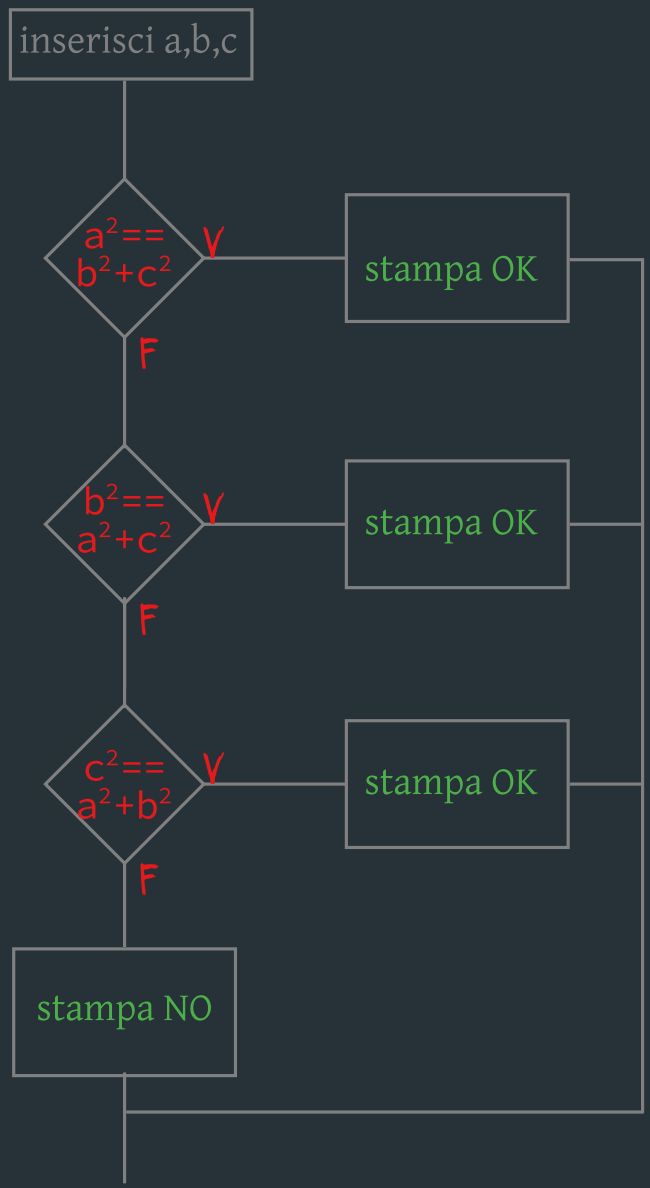
Esercizio

Scrivere un programma che determini se tre numeri, inseriti dall'utente, siano i tre lati di un triangolo rettangolo.

$$a^2 = b^2 + c^2$$

algoritmo

b


inserisci a,b,c
$a^2 == b^2+c^2$
V
stampa OK
F
$b^2 == a^2+c^2$
V
stampa OK
F
$c^2 == a^2+b^2$
V
stampa OK
F
stampa NO

implementazione

```
a = input("inserisci il valore a ")
b = input("inserisci il valore b ")
c = input("inserisci il valore c ")

if a**2 == b**2 + c**2:
    print "triangolo rettangolo\n"
else:
    if b**2 == a**2 + c**2:
        print "triangolo rettangolo\n"
    else:
        if c**2 == a**2 + b**2:
            print "triangolo rettangolo\n"
        else:
            print "non è un triangolo rettangolo\n"
```

esecuzione

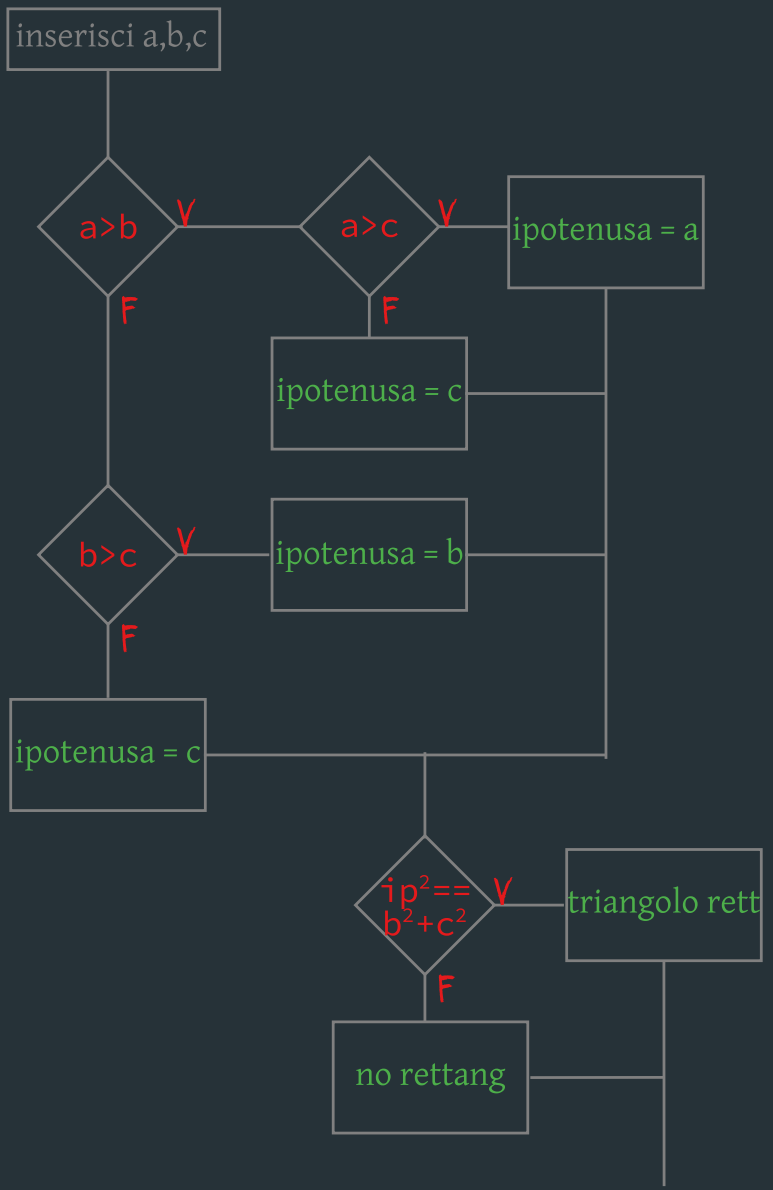
```
dario@vulcano: ipython latiTriangoloFB.py
inserisci il valore a 3
inserisci il valore b 4
inserisci il valore c 5
triangolo rettangolo

dario@vulcano: ipython latiTriangoloFB.py
inserisci il valore a 4
inserisci il valore b 3
inserisci il valore c 5
triangolo rettangolo

dario@vulcano: ipython latiTriangoloFB.py
inserisci il valore a 5
inserisci il valore b 4
inserisci il valore c 3
triangolo rettangolo

dario@vulcano:
dario@vulcano: ipython latiTriangoloFB.py
inserisci il valore a 4
inserisci il valore b 4
inserisci il valore c 4
non è un triangolo rettangolo
```

variante

```
a = input("inserisci il valore a ")
b = input("inserisci il valore b ")
c = input("inserisci il valore c ")

if a**2 == b**2 + c**2:
    print "triangolo rettangolo\n"
elif b**2 == a**2 + c**2:
    print "triangolo rettangolo\n"
elif c**2 == a**2 + b**2:
    print "triangolo rettangolo\n"
else:
    print "non è un triangolo rettangolo\n"
```

variante 2

b


inserisci a,b,c
a>b
V
F
a>c
V
F
ipotenusa = a
ipotenusa = c
b>c
V
F
ipotenusa = b
ipotenusa = c
ip²==
b²+c²
V
F
triangolo rett
no rettang

```
a = input("inserisci il valore a ")
b = input("inserisci il valore b ")
c = input("inserisci il valore c ")

if a > b:
    cat1 = b
    if a > c:
        cat2 = c
        ip = a
        print "a e' l'ipotenusa? "
    else:
        cat2 = a
        ip = c
        print "c e' l'ipotenusa? "
else:
    cat1 = a
    if b > c:
        cat2 = c
        ip = b
        print "b e' l'ipotenusa? "
    else:
        cat2 = b
        ip = c
        print "c e' l'ipotenusa? "
if ip**2 == cat1**2 + cat2**2:
    print "si ed è un triangolo rettangolo"
else:
    print "non è un triangolo rettangolo"
```

L'istruzione pass

- In qualche occasione può essere utile avere un blocco vuoto. In questo caso può essere usata l'istruzione pass, che è solo un segnaposto.

```
if x>0:
   pass
else:
   print('Ciao')
```

- Utile durante lo sviluppo, in attesa che il codice mancante venga scritto
  - Nell'esempio, se non ci fosse il pass, l'interprete si rifiuterebbe di eseguire il programma:
    `syntax error - expected an indented block`

- Non è buono stile scrivere una condizione come la seguente (il pass è tollerato solo se usato temporaneamente)

```
if x>=0:
   pass
else:
   print('Negativo')
```

- Sarebbe meglio riscrivere l'istruzione di cui sopra in questo modo

```
if x<0: # equivalente a not(x>=0)
   print('Negativo')
```