

Python

In questa lezione:


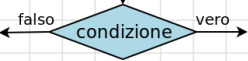

- I diagrammi di flusso
- La programmazione strutturata
- Il blocco di istruzioni
- La selezione
- Il ciclo

I diagrammi di flusso

I diagrammi di flusso

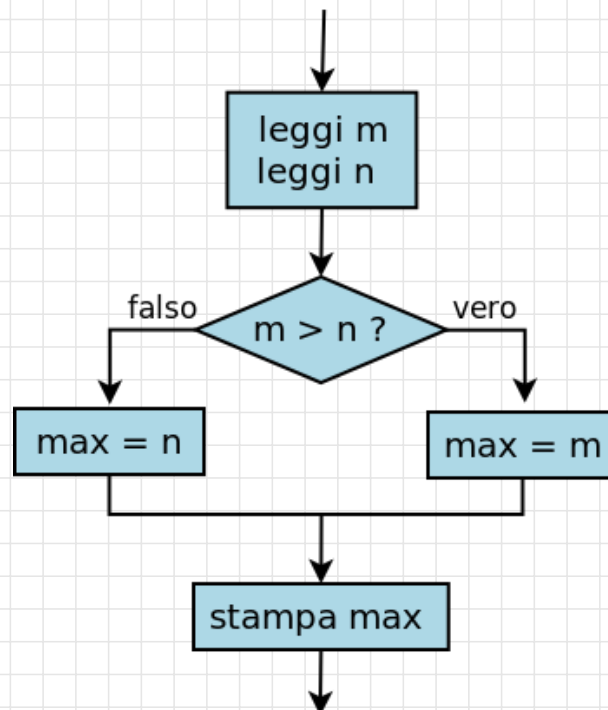
I diagrammi di flusso sono uno strumento grafico per la rappresentazione di algoritmi o programmi e che consentono di evidenziare facilmente i diversi flussi di operazioni da compiere.

Utilizzeremo solamente pochi simboli grafici per rappresentare diagrammi di flusso.

simbolo	nome	funzione
	Blocco	rappresenta una o più operazioni da eseguire in sequenza
	Test	rappresenta una operazione condizionale che determina quale strada seguire
	Linea di flusso	indica l'ordine di simboli da seguire quando collega due simboli. Indica il simbolo iniziale o finale quando non è preceduta o seguita da nessun simbolo.

I diagrammi di flusso - esempio

Esempio di diagramma di flusso per il calcolo del massimo tra due numeri.



La programmazione strutturata (1)

Possiamo immaginare diagrammi di flusso molto complessi in cui è difficile capire la logica che li ha prodotti seguendo le frecce.

Similmente accade nella programmazione, dove l'equivalente della freccia dei diagrammi di flusso sono istruzioni come **GOTO** ("vai a"). Si pensi alla macchina URM e all'istruzione **salto** $J(m, n, q)$.

Un uso non accorto di istruzioni **GOTO** può portare ad un codice poco leggibile, fatto di sequenze di istruzioni e salti ad altre sequenze, il cosiddetto **codice "a spaghetti"**.

La programmazione strutturata (2)

Per controllare il flusso delle istruzioni, ma al contempo evitare la necessità di utilizzare istruzioni GOTO, i linguaggi ad alto livello mettono a disposizione del programmatore delle **strutture di controllo** del flusso di istruzioni da eseguire.

Nel 1966 due ricercatori italiani, Corrado Böhm e Giuseppe Jacopini, dimostrarono che sono sufficienti **tre** sole **strutture di controllo di flusso**.

La programmazione effettuata utilizzando una composizione di queste strutture di controllo prende il nome di **programmazione strutturata**.

Il teorema di Böhm and Jacopini

Il teorema sulla programmazione strutturata di Böhm e Jacopini stabilisce che si può realizzare un qualsiasi programma utilizzando tre strutture di controllo opportunamente combinate:

- il **blocco** (o **sequenza**) che consiste di una sequenza di istruzioni da eseguire una di seguito all'altra
- la **selezione** che consiste nell'eseguire un blocco o un altro blocco di istruzioni in funzione di una condizione che può essere vera o falsa
- il **ciclo** che consiste nel ripetere un blocco di istruzioni mentre una certa condizione è vera.

Il blocco di istruzioni

Una delle strutture fondamentali nei linguaggi di programmazione è il **blocco** o **sequenza** di istruzioni.

Le **istruzioni presenti** nel blocco possono essere **zero, una o più**. Nel caso non siano presenti istruzioni nel blocco si parla di blocco vuoto o sequenza vuota.

I blocchi possono essere **composti** tra loro: un blocco seguito da un altro blocco è ancora un blocco.

Il blocco in Python

Generalmente nei linguaggi di programmazione un blocco è individuato da **delimitatori**, cioè parole chiave come **begin** e **end** (in Pascal) oppure parentesi come **{** e **}** (in C, C++ e Java).

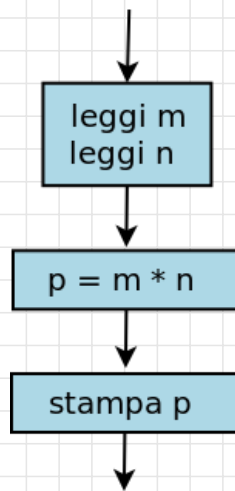
In Python **non esistono delimitatori**. Un blocco è individuato da una sequenza di istruzioni i cui enunciati sono posti alla stessa distanza dall'inizio della linea (**indentazione**).

Normalmente si scrive un'istruzione per linea. Se su una linea sono presenti più istruzioni queste sono separate da un **“;”**.

Con questo modo di rappresentare un blocco in Python non sarebbe possibile indicare il blocco vuoto. A questo fine esiste l'istruzione **pass**, che specifica di non far nulla.

Esempio di blocco in Python

Un programma Python costituito da un solo blocco per calcolare il prodotto di due numeri.



```

"""
calcolo del prodotto di due numeri
"""
n = int(input("Primo numero: "))
m = int(input("Secondo numero: "))

p = m * n

print("Prodotto: ", p)
  
```

Notare come tutte le istruzioni Python sono tutte allineate al primo carattere di ogni linea.

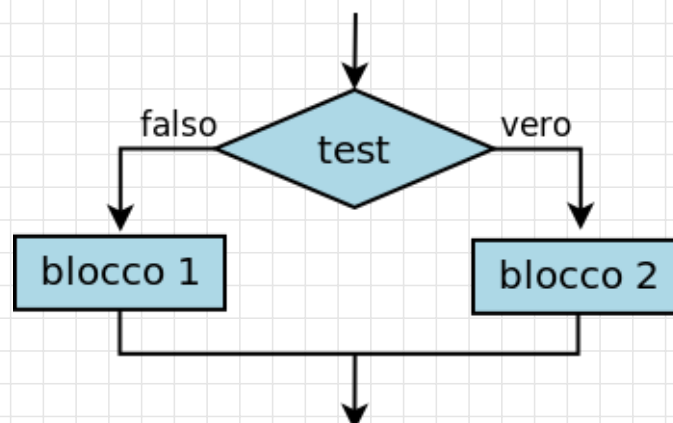
La selezione

La selezione

La seconda struttura fondamentale è la [selezione](#).

La selezione si usa quando deve essere eseguito un blocco di istruzioni o un altro, a seconda della valutazione di un test.

Corrisponde al seguente diagramma di flusso:



Il test nella selezione

La selezione prevede la valutazione di un **test**. Vediamo come realizzare un test. Il test è una espressione da valutare il cui risultato è **vero** o **falso**.

Operatori sui tipi già visti (interi, reali e stringhe) che restituiscono un valore vero o falso sono gli **operatori di confronto**.

Operatori di confronto in Python sono:

Operatore	Significato
<code>==</code>	uguale a
<code>!=</code>	diverso da
<code><</code>	minore di
<code>></code>	maggiore di
<code><=</code>	minore o uguale a
<code>>=</code>	maggiore o uguale a

Questi operatori restituiscono un valore nell'insieme **{False, True}**.

Il tipo booleano

Molti linguaggi, tra cui il Python, definiscono il tipo di dato **booleano** (da George Boole) costituito dall'insieme **{False, True}** con gli **operatori logici** di **congiunzione (AND)**, **disgiunzione (OR)** e **negazione (NOT)**.

Gli operatori AND e OR sono operatori binari, il NOT è unario, e operano secondo le seguenti tabelle:

a	b	a AND b
True	True	True
True	False	False
False	True	False
False	False	False

a	b	a OR b
True	True	True
True	False	True
False	True	True
False	False	False

a	NOT a
True	False
False	True

Il tipo booleano

In Python il tipo booleano è il `boolean` e gli operatori logici sono `and`, `or`, `not`.

Operatore	Descrizione	Sintassi	Esempi
<code>and</code>	coniunzione	<code>a and b</code>	<code>True and False → False</code>
<code>or</code>	disgiunzione	<code>a or b</code>	<code>True or False → True</code>
<code>not</code>	negazione	<code>not a</code>	<code>not True → False</code>

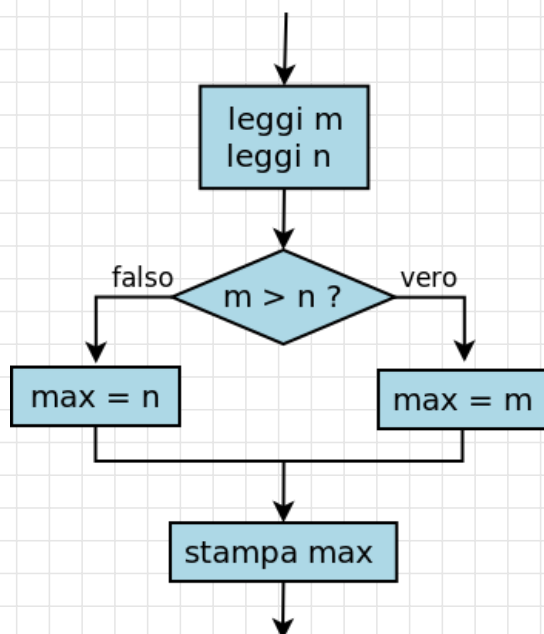
Le `variabili booleane` hanno uno dei due valori possibili. Con esse si possono definire `espressioni booleane`. Es.:

```
a = False
b = ( 10 < 3 )      # b vale False
c = not a or b      # c riceve il risultato di una espressione booleana
print(c) #stampa True
```

La selezione in Python

if-else

In Python, la selezione di un blocco o di un altro blocco di istruzioni si realizza mediante il costrutto `if - else`.



```
"""
calcolo del massimo tra due numeri
"""

m = int(input("Primo numero: "))
n = int(input("Secondo numero: "))

if m > n :
    max = m
else :
    max = n

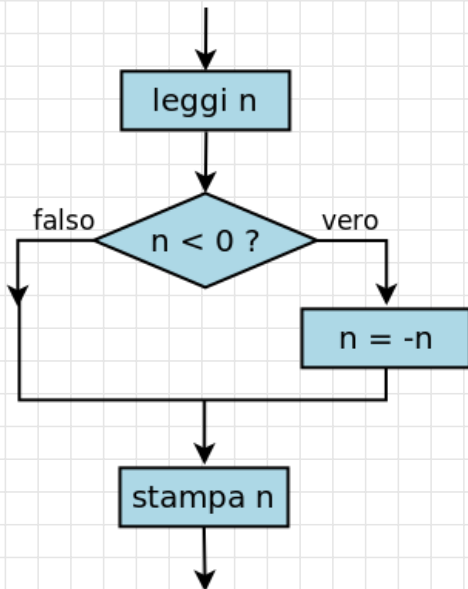
print("Valore massimo: ", max)
```

Notare come i blocchi corrispondenti ai due rami del flusso hanno una

La selezione in Python

if

Nel caso un blocco sia vuoto si può utilizzare l'istruzione `pass`, ma anche omettere il ramo `else`.



```

"""
valore assoluto di un numero
"""

n = int(input("Un numero: "))

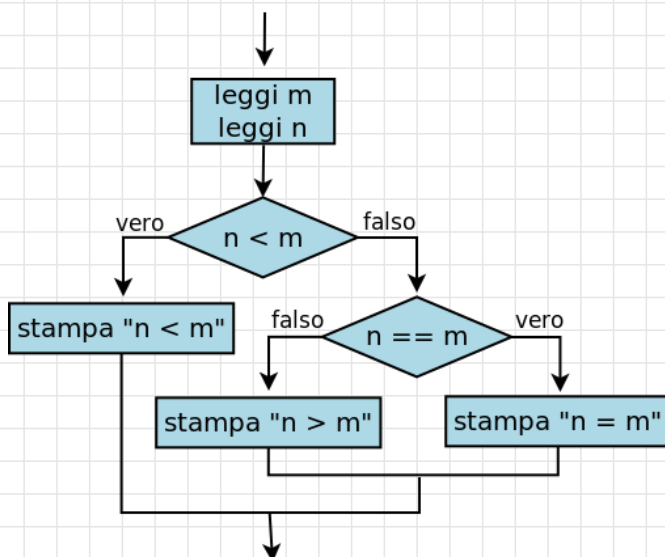
if n < 0:
    n = -n #cambio del segno

print("Valore assoluto: ", n)
  
```

La selezione in Python

if-elif-else

Si presenta spesso il caso di selezioni annidate. In questo caso può essere utilizzato il costrutto `if-elif-else`.



```

"""
confronto tra numeri
"""

m = int(input("Primo numero: "))
n = int(input("Secondo numero: "))

if n < m :
    print( n, " è minore di ", m )
elif n == m :
    print( "I numeri sono uguali" )
else:
    print( n,"è maggiore di ", m )
  
```


La selezione in Python

if-elif-else

Il numero di annidamenti può essere arbitrario, così come i rami `elif`.

```
""" lancia un dado e ne stampa il valore """  
  
import random  
faccia = random.randint(1, 6) #lancio del dado  
  
if faccia == 1 :  
    print("Uno")  
  
elif faccia == 2 :  
    print("Due")  
  
elif faccia == 3 :  
    print("Tre")  
  
elif faccia == 4 :  
    print("Quattro")  
  
elif faccia == 5 :  
    print("Cinque")  
  
else:  
    print("Sei")
```

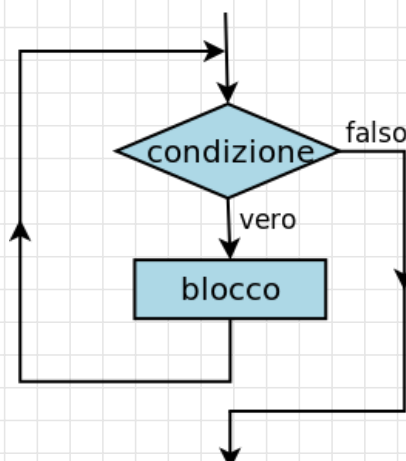
Il ciclo

Il ciclo

La terza struttura fondamentale è il `ciclo`.

Il ciclo si usa quando deve essere ripetutamente eseguito un blocco di istruzioni mentre è valida una condizione.

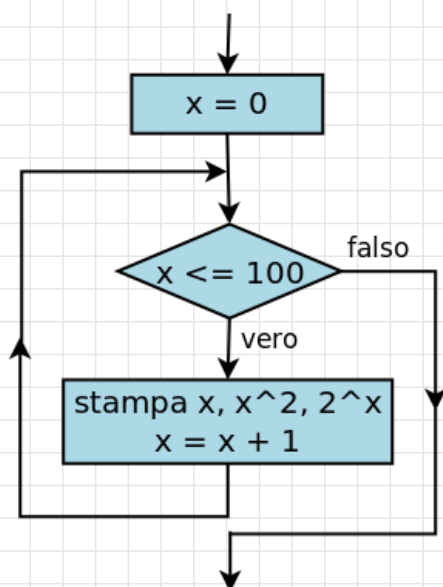
Corrisponde al seguente diagramma di flusso:



Il ciclo in Python

while

In Python, il ciclo si realizza mediante il costrutto `while`.



```

"""
stampa i valori di x, x^2 e 2^x
per i valori di x da 0 a 100
"""

x = 0
while x <= 100:
    print( x, x*x, 2**x)
    x = x + 1
  
```

Il ciclo in Python

for-in

Spesso è necessario fare delle azioni per tutti i valori di un insieme o di una sequenza di elementi. In questi casi è conveniente utilizzare il costrutto `for-in`, la cui sintassi è:

```

for <variabile> in <insieme di valori> :
    <blocco>
  
```

Il blocco viene ripetuto un numero di volte pari alla cardinalità dell'insieme dei valori. Ogni volta, la variabile assume un valore dell'insieme. Per esempio, una stringa può essere vista come una sequenza di valori, i suoi caratteri.

```

"""
stampa i caratteri di una stringa e il loro codice unicode
"""

for c in "Salve!": # c assume i valori 'S','a','l','v','e','!'
    print (c, ord(c)) # per ognuno stampa il suo codice unicode
  
```

Il ciclo in Python

for-in

Spesso bisogna iterare su un intervallo di valori interi. Questo intervallo può essere fornito dall'istruzione `range`. L'istruzione `range(a,b)` restituisce un intervallo di valori tra `a` e `b` escluso.

Esempi:

```
for n in range(0,10):    # n assume i valori 0,1,2,3,4,5,6,7,8,9
    print (n)
```

```
for x in range(0,101):   # x assume i valori tra 0 e 100
    print( x, x*x, 2**x)
```

```
""" calcola la somma dei primi n interi """
n = int(input("Un intero: "))
s = 0      # in ogni momento s contiene le somme parziali
for x in range(1,n+1):
    s = s + x

print(s)   # stampa la somma dei primi n interi
```

Esercizi

- Realizzare un programma che, chieste le coordinate dei centri e i raggi di due cerchi, controlli se i due cerchi hanno punti in comune.
- Realizzare un programma che scriva le cifre binarie di un intero dato in ingresso.
- Controllare se un numero intero dato in ingresso contenga la cifra sette.
- Scrivere un programma che simuli mille lanci di un dado e che al termine stampi, per ogni faccia del dado, quante volte è uscita.
- Scrivere un programma che simuli mille lanci di due dadi e che al termine stampi il numero delle volte che è uscita una coppia di valori uguali.
- (Turtle graphic) Scrivere un programma che richieda un intero `n` e disegni un poligono regolare di `n` lati.