

# Controllo del flusso: `if, else, for, while`

Rachele Sprugnoli

[rachele.sprugnoli@unipr.it](mailto:rachele.sprugnoli@unipr.it)



**UNIVERSITÀ  
DI PARMA**

# Gestire il flusso di esecuzione

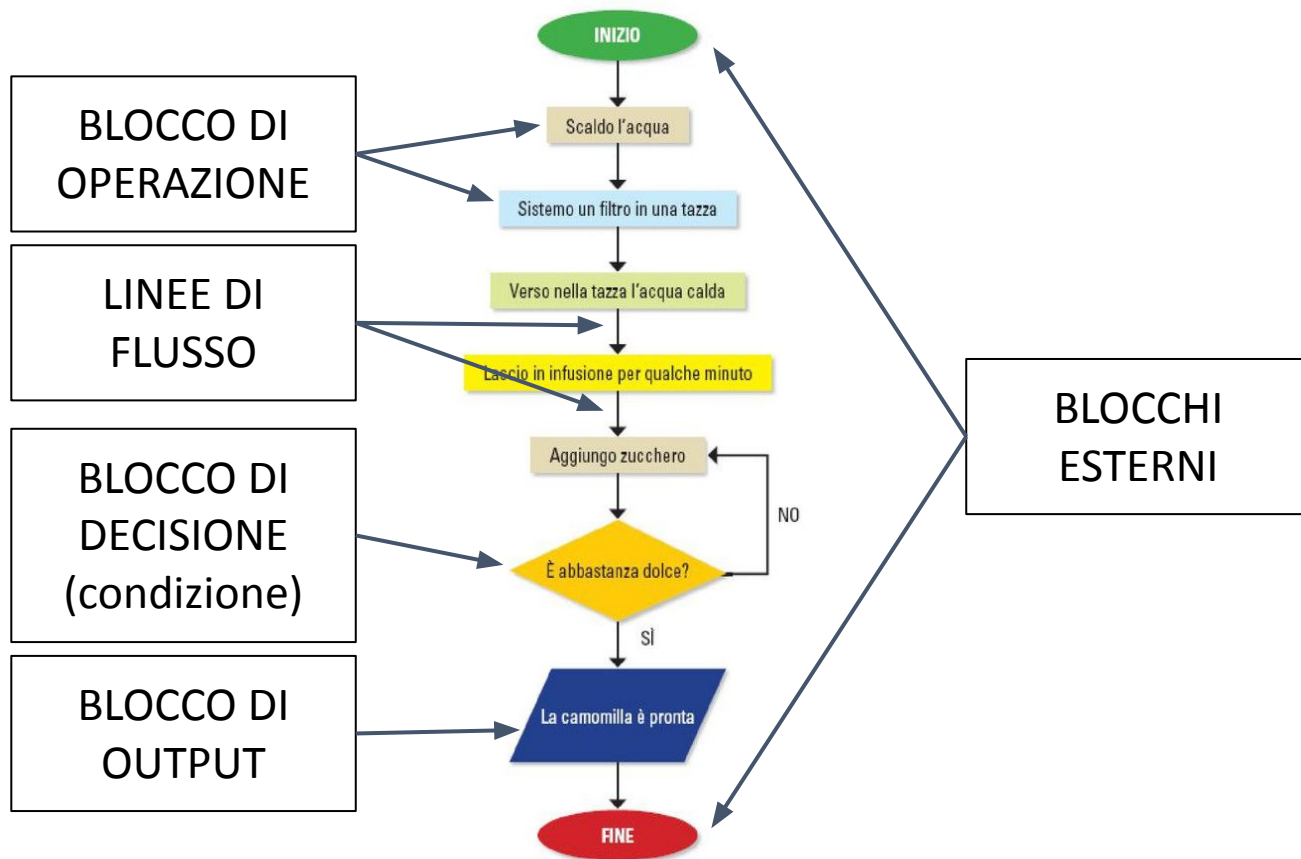
- Flusso di esecuzione: ordine di esecuzione delle varie istruzioni
- Diagrammi di flusso: mappe che guidano alla comprensione logica del programma, specificano che i passi devono essere eseguiti in sequenza, salvo diversa esplicita indicazione

Immagine di Danilo Cimino da:

<https://www.cosedicomputer.com/cos-e-un-algoritmo/>



# Diagramma di flusso: esempio



# Istruzione `if`

- Consente di eseguire in modo condizionale blocchi di istruzioni
- Viene usata per determinare l'esecuzione del codice in base alla valutazione di un'espressione booleana (`True` o `False`)
  - Se l'espressione `if` è `True`, viene eseguito il codice indentato dopo l'istruzione
  - Se invece è `False`, allora il codice indentato dopo l'`if` viene saltato e il programma esegue la riga di codice successiva che è allo stesso livello di indentazione dell'`if`

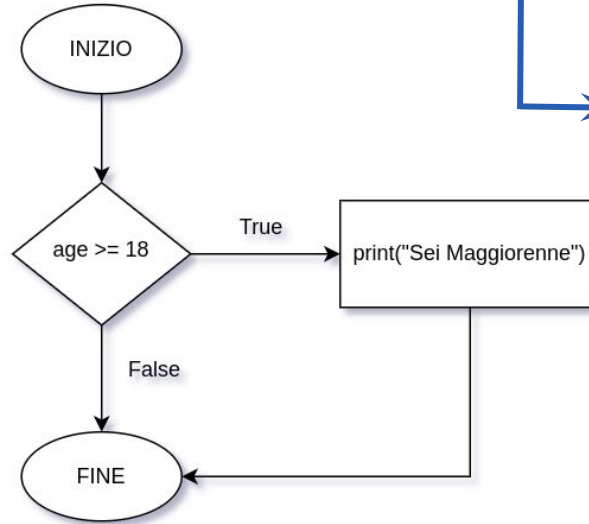
# Istruzione if

```
age = 20
```

```
if age >= 18:  
    #codice  
#codice dopo l'if
```

```
age = 10
```

```
if age >= 0:  
    #codice  
#codice dopo l'if
```



Immagini diagrammi tratti da:

<https://www.programmareinpython.it/video-corso-python-base/controllo-di-flusso-if-elif-ed-else/>

# Istruzione `else`

- Se la condizione di controllo nell'istruzione `if` restituisce `False`, è possibile aggiungere una clausola `else` (“altrimenti”) facoltativa
- L'istruzione `else` viene eseguita prima che il programma finisca se la condizione nell'istruzione `if` restituisce `False`
- Il blocco `else` è sempre l'ultimo a essere analizzato ed è eseguito solo se tutte le condizioni precedenti a `else` sono risultate `False`
  - La parola chiave `else` deve essere seguita dai due punti
  - Il codice indentato dell'istruzione `else` deve essere allo stesso livello dell'`if` iniziale

# Istruzione else

```
age = 20
```

```
if age >= 18:  
    #codice_1
```

```
else:
```

```
    #codice_2
```

```
#codice dopo l'if
```

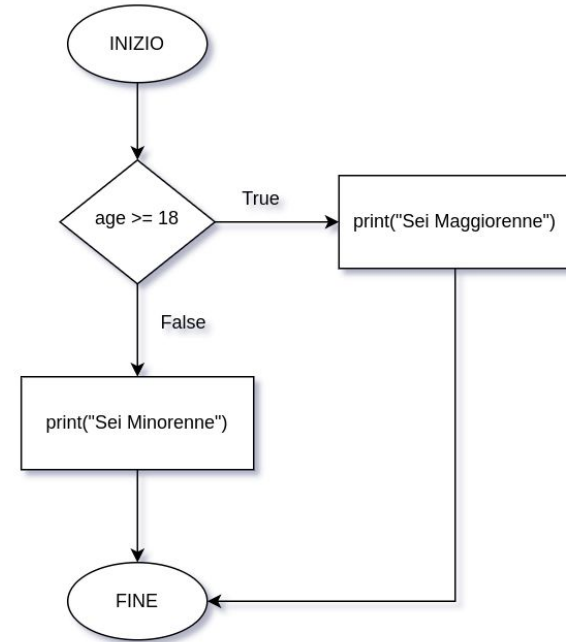
```
age = 10
```

```
if age >= 18:  
    #codice_1
```

```
else:
```

```
    #codice_2
```

```
#codice dopo l'if
```



Immagini diagrammi tratti da:

<https://www.programmareinpython.it/video-corso-python-base/controllo-di-flusso-if-elif-ed-else/>

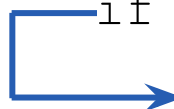
# Istruzione `elif`

- L'enunciato `if...else` si usa per eseguire un blocco di codice tra due alternative; se sono possibili più di due opzioni si usa l'istruzione `elif`
- Permette di aggiungere tutti i blocchi di controllo di cui abbiamo bisogno
- Fornisce un'altra espressione (condizione) da verificare, proprio come accade per l'`if` iniziale
  - L'istruzione `elif` viene eseguita SOLO SE la sua espressione di controllo restituisce `True` e l'`if` o eventuali `elif` precedenti hanno restituito `False`




# Istruzione elif

age = 20




```
if age > 18:
    #codice_1
elif age = 18:
    #codice_2
else:
    #codice_3
#codice dopo l'if
```

age = 18



```
if age > 18:
    #codice_1
elif age = 18:
    #codice_2
else:
    #codice_3
#codice dopo l'if
```

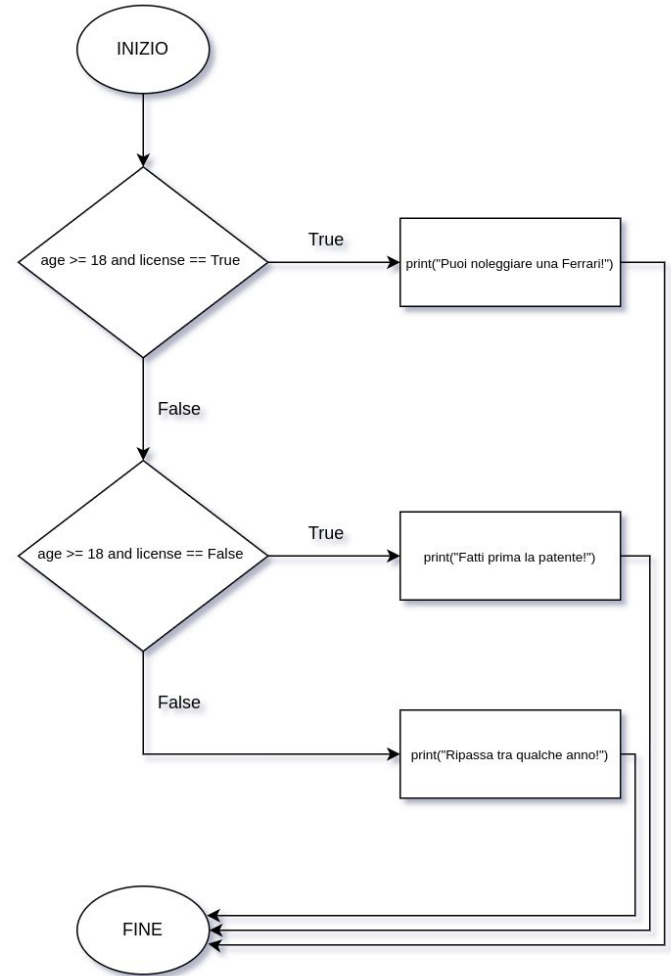
age = 15



```
if age > 18:
    #codice_1
elif age = 18:
    #codice_2
else:
    #codice_3
#codice dopo l'if
```

# Istruzione elif

Immagini diagrammi tratti da:  
<https://www.programmareinpython.it/video-corso-python-base/controllo-di-flusso-if-elif-ed-else/>



# Cicli

- Loop: modo per ripetere più volte un insieme di linee di codice
- 2 tipi:
  - `for`: può essere usato per iterare su una sequenza, come una lista, un dizionario o una stringa
  - `while`: continua a essere eseguito fino a quando una certa condizione è soddisfatta

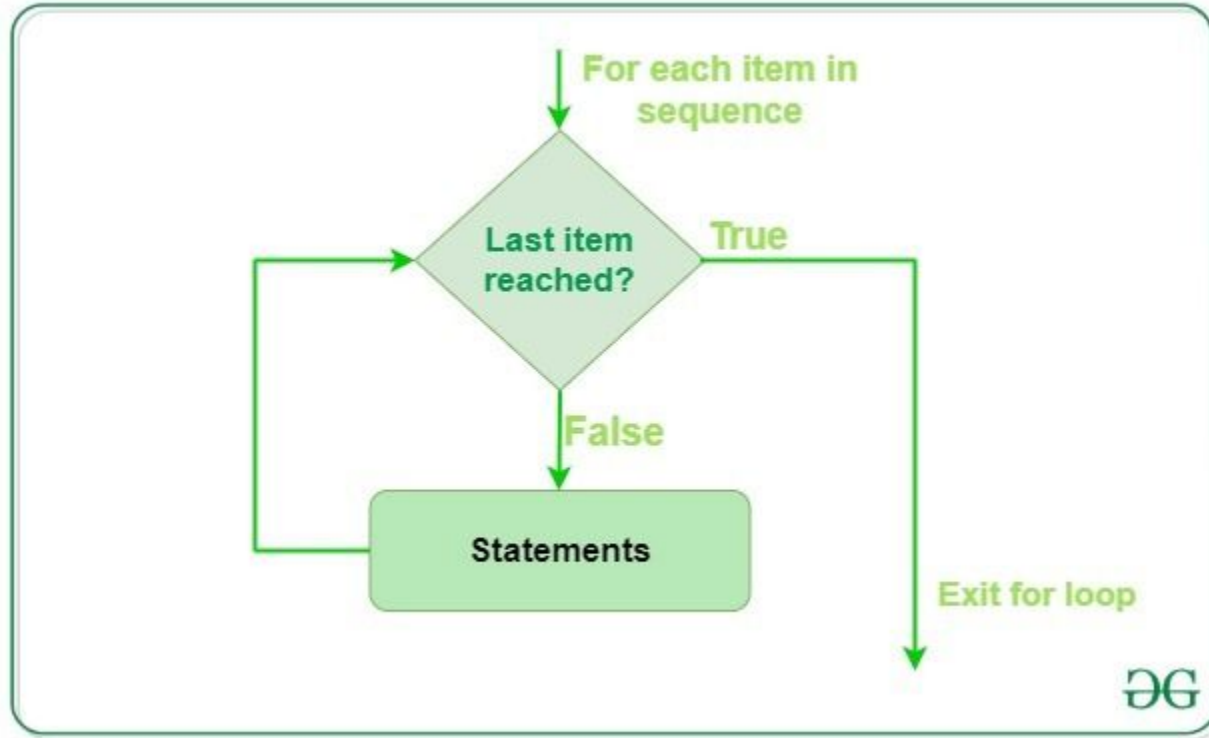
# Ciclo `for`

- Itera una lista di elementi ed esegue una serie di azioni su ogni elemento
- La sintassi di un ciclo `for` consiste nell'assegnare un valore temporaneo a una variabile a ogni iterazione successiva: il ciclo `for` si occupa sia della creazione che dell'assegnazione del valore alla variabile ad ogni passo

**`for`** elemento **`in`** sequenza:

    fare\_qualcosa(elemento) → attenzione all'indentazione

# Ciclo for



Da: <https://www.geeksforgeeks.org/python-for-loops/>

# Ciclo `for` con la funzione `range()`

- La funzione `range()` genera una sequenza di numeri interi che controlla le iterazioni del ciclo
  - L'intervallo di `range()` corrisponde al numero di iterazioni che verranno eseguite, ovvero al numero di volte che il ciclo `for` verrà processato
  - Il ciclo termina alla fine dell'intervallo di `range()`
  - L'intervallo definito nel `range` non include il numero passato in sé perché inizia a contare a partire da 0: ad es., `range(11)` va da 0 a 10

# Ciclo for con la funzione range ()

`for` numero `in` `range`(11)

parola chiave

nome variabile  
(scelto da voi)

parola chiave

chiamata alla  
funzione range()

`range`(start, stop, step)

punto iniziale  
dell'intervallo

punto finale  
dell'intervallo

passo di  
avanzamento

# Ciclo for con la funzione range()

- Esempi:
  - `for i in range(6)` → 0,1,2,3,4,5
  - `for i in range(10,16)` → 10,11,12,13,14,15
  - `for i in range(0,9,2)` → 0,2,4,6,8
  - `for i in range(5,0,-1)` → 5,4,3,2,1

N.B. Per contare a ritroso si usa un incremento negativo



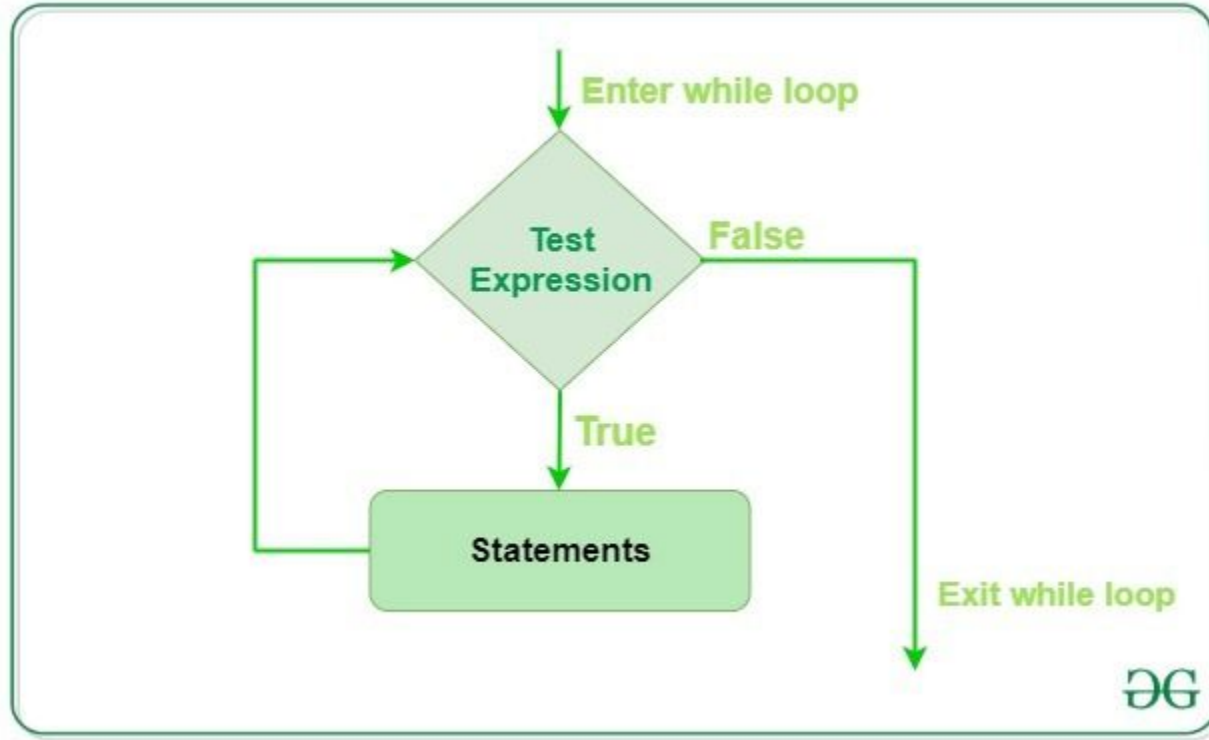
# Ciclo `while`

- Traducibile con “finché”: esegue ripetutamente un blocco di codice finché una condizione è `True`
  - La condizione viene controllata prima che il blocco di codice venga eseguito: se la condizione non è soddisfatta inizialmente, allora il blocco di codice non viene eseguito

**while** condizione:

    fare\_qualcosa() → attenzione all'indentazione

# Ciclo while



Da: <https://www.geeksforgeeks.org/python-while-loop/>

# Differenze

**for**

```
for numero in range(11):  
    print(numero)
```

- Itera su una sequenza
- Si usa quando si ha una sequenza già definita

**while**

```
counter = 0  
while counter <= 10:  
    print(counter)  
    counter += 1
```

- Esegue ripetutamente un blocco di istruzioni finché una condizione è vera
- Si usa quando non si conosce in anticipo il numero di iterazioni

# Cicli infiniti

- Infinite loop: un ciclo che non termina mai per un errore di battitura nella dichiarazione condizionale all'interno del loop o a causa di una logica errata
  - Negli esempi che seguono la condizione non diventa mai falsa per cui il ciclo non termina

```
while 15 == 15:  
    print("aiuto")
```

```
i = 1  
while i < 6:  
    print (i)
```

# Un po' di pratica



- Lezione7.ipynb

<https://colab.research.google.com/drive/1iAi80XZ1dfwyORRqsmDiPIKVYJ3tzQa2?usp=sharing>



## Esercizio 1

- Stampare tutti i numeri da 10 a 20 (compresi):
  - usando il ciclo `for`
  - usando il ciclo `while`

## Esercizio 2

- Stampare i numeri pari a ritroso da 10 a 2:
  - 10, 8, 6, 4, 2, 0

## Esercizio 3



- Definite una lista che ha come elementi le parole del primo verso della Commedia e scrivete uno script che controlla se nella lista è presente la parola “vita”
  - Stampate un output nel caso sia presente e uno nel caso non sia presente
  - Se la parola è presente l’output deve indicare la posizione nella lista

# Soluzioni esercizi



- [https://colab.research.google.com/drive/1v0gh9M\\_K\\_z8ugaE498qP\\_VNUb6rRCHUYt?usp=sharing](https://colab.research.google.com/drive/1v0gh9M_K_z8ugaE498qP_VNUb6rRCHUYt?usp=sharing)