anno 2014-15

Introduzione all'Algoritmica per i Licei

3 - Controllo del flusso di esecuzione.

Elio Giovannetti
Dipartimento di Informatica
Università di Torino

versione 1 marzo 2015

Programmazione imperativa: concetti fondamentali.

2)

- sequenza di istruzioni;
- controllo del flusso di esecuzione (istruzioni composte):
  - istruzioni if e if-else;
  - istruzione while;
  - istruzione for

Sequenza di istruzioni.

Un programma è una sequenza di istruzioni, scritte una per riga, che vengono eseguite da Python una dopo l'altra.

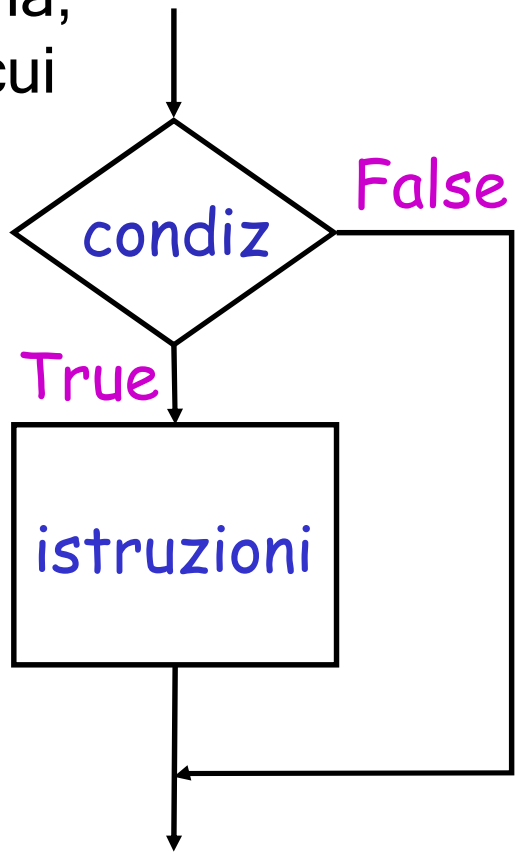
Esempio:

```
nome = input('Come ti chiami?')
print('ciao ' + nome + '!')
m = input('digita un numero: ')
print("il suo quadrato e'", m*m)
```

Controllo del flusso di esecuzione: istruzioni composte.

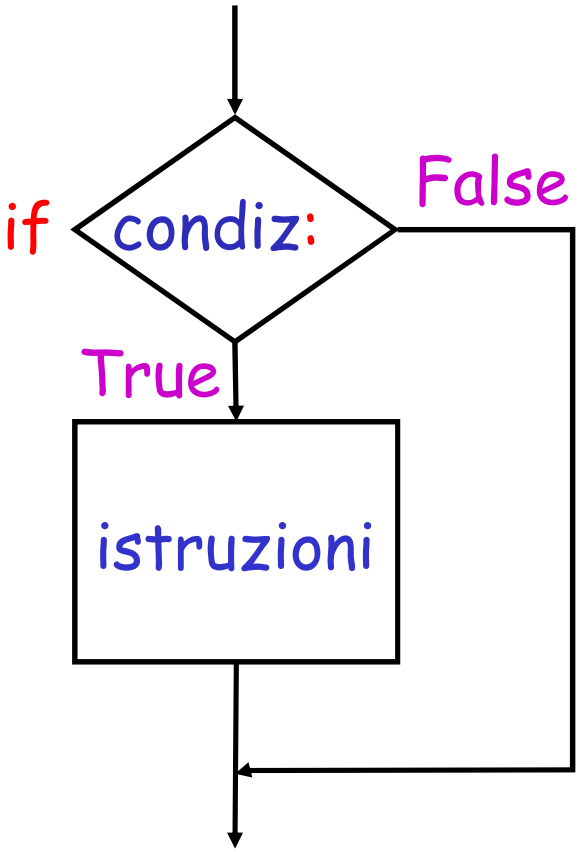
Istruzione if

La condizione deve essere un'espressione booleana, cioè un'espressione il cui valore è True o False.



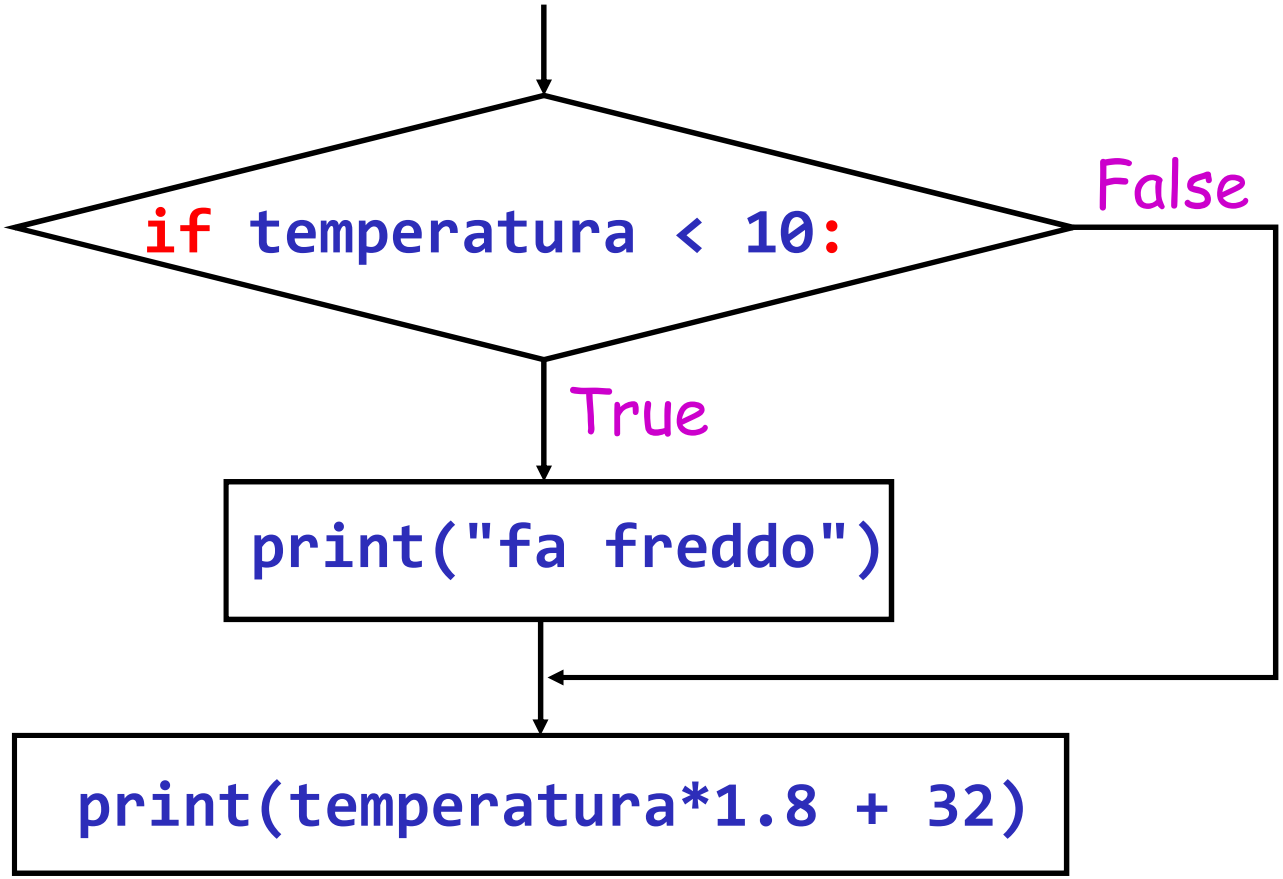
Istruzione if

Sintassi Python



Esempio di istruzione if

```
stringa = input("digita la temperatura: ")
temperatura = int(stringa)
```



Esempio di istruzione if

```
stringa = input("digita la temperatura: ")
temperatura = int(stringa)
if temperatura < 10:
  print("fa freddo")
print(temperatura*1.8 + 32)
```

Rientro

Attenzione! una particolarità di Python è che: gli spazi di rientro ("indentazione") hanno significato! Esempio:

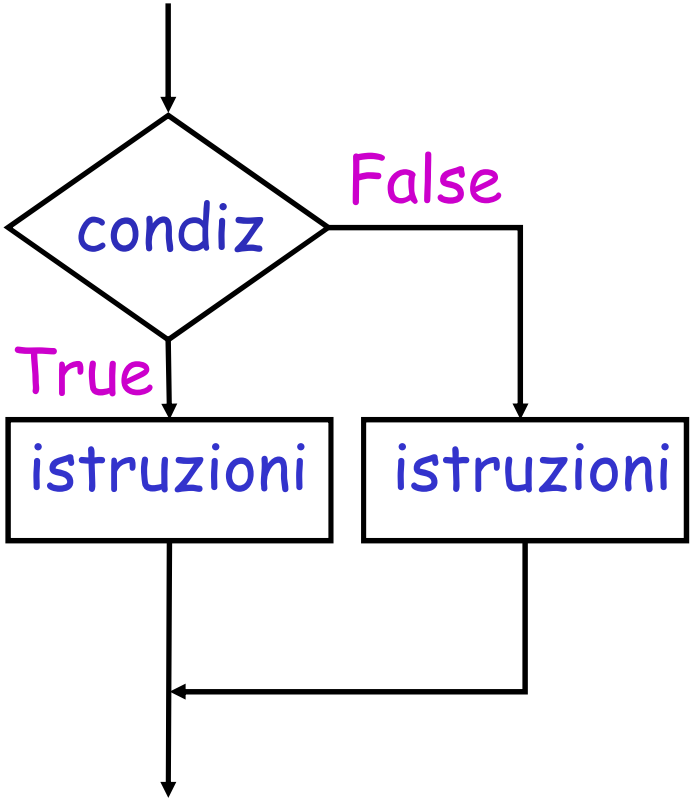
```
if temperatura < 0:
  print("Brrr!")
  print("Mettiti la giacca imbottita!")
print(temperatura*1.8 + 32, "Fahrenheit")
```

è diverso da

```
if temperatura < 0:
  print("Brrr!")
  print("Mettiti la giacca imbottita!")
  print(temperatura*1.8 + 32, "Fahrenheit")
```

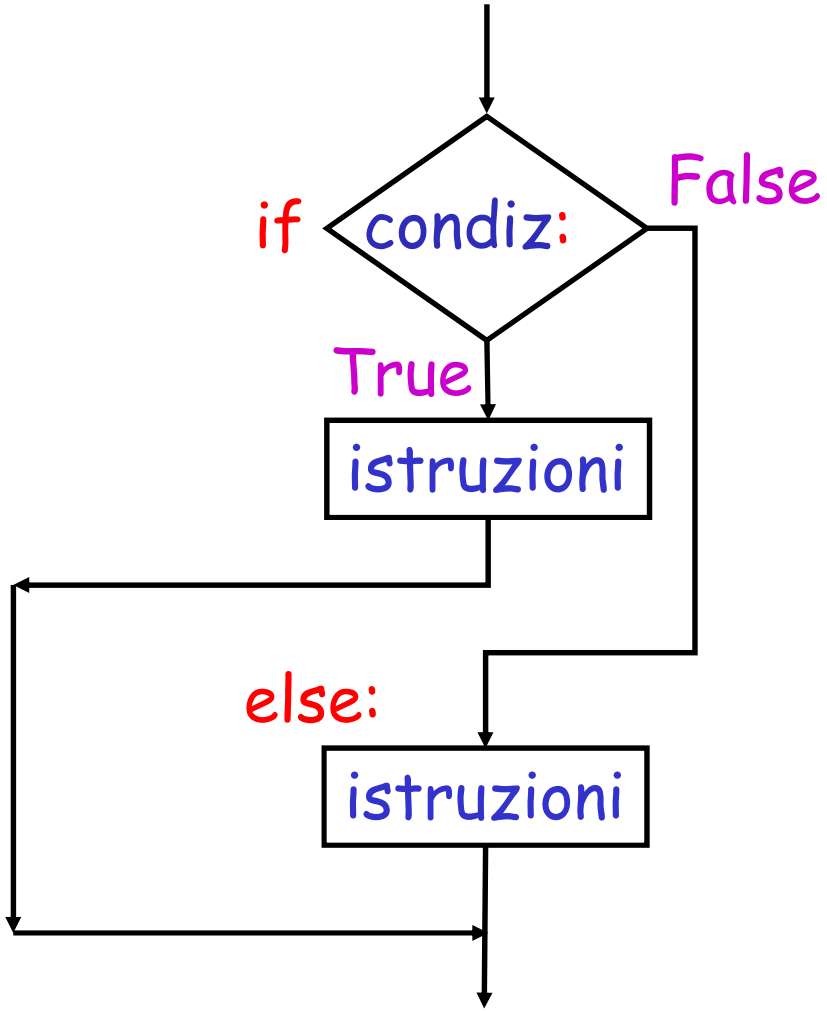
Il 1° programma scrive la temperatura in Fahrenheit sempre, il 2° la scrive solo se è minore di zero.

Istruzione if-else



Istruzione if-else

Sintassi Python



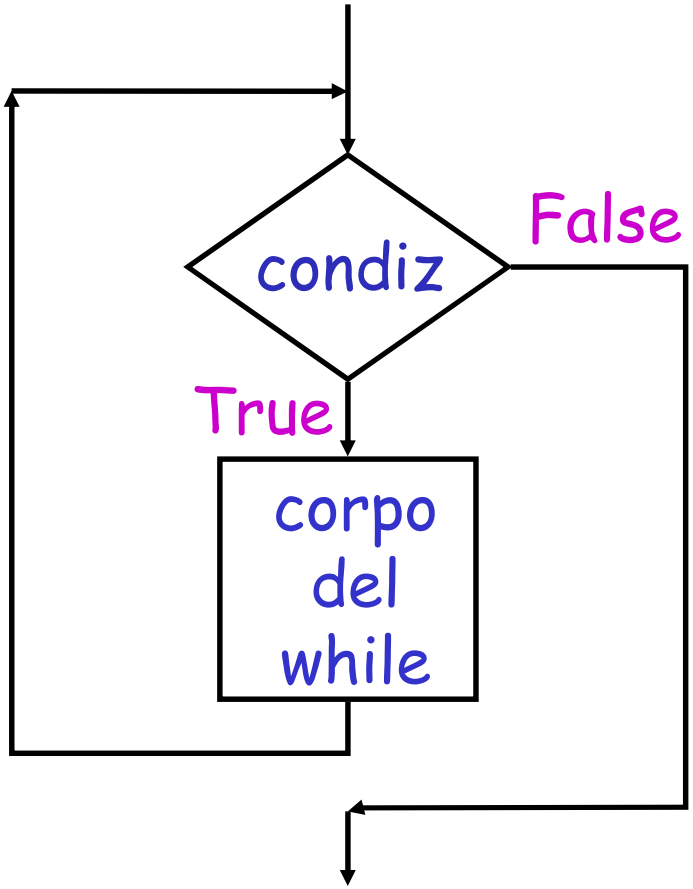
Esempio

```
stringa = input("digita la temperatura: ")
temperatura = int(stringa)
if temperatura < 10:
  print("fa freddo")
else:
  print("non fa freddo")
print("ho finito")
```

Iterazione: l'istruzione while.

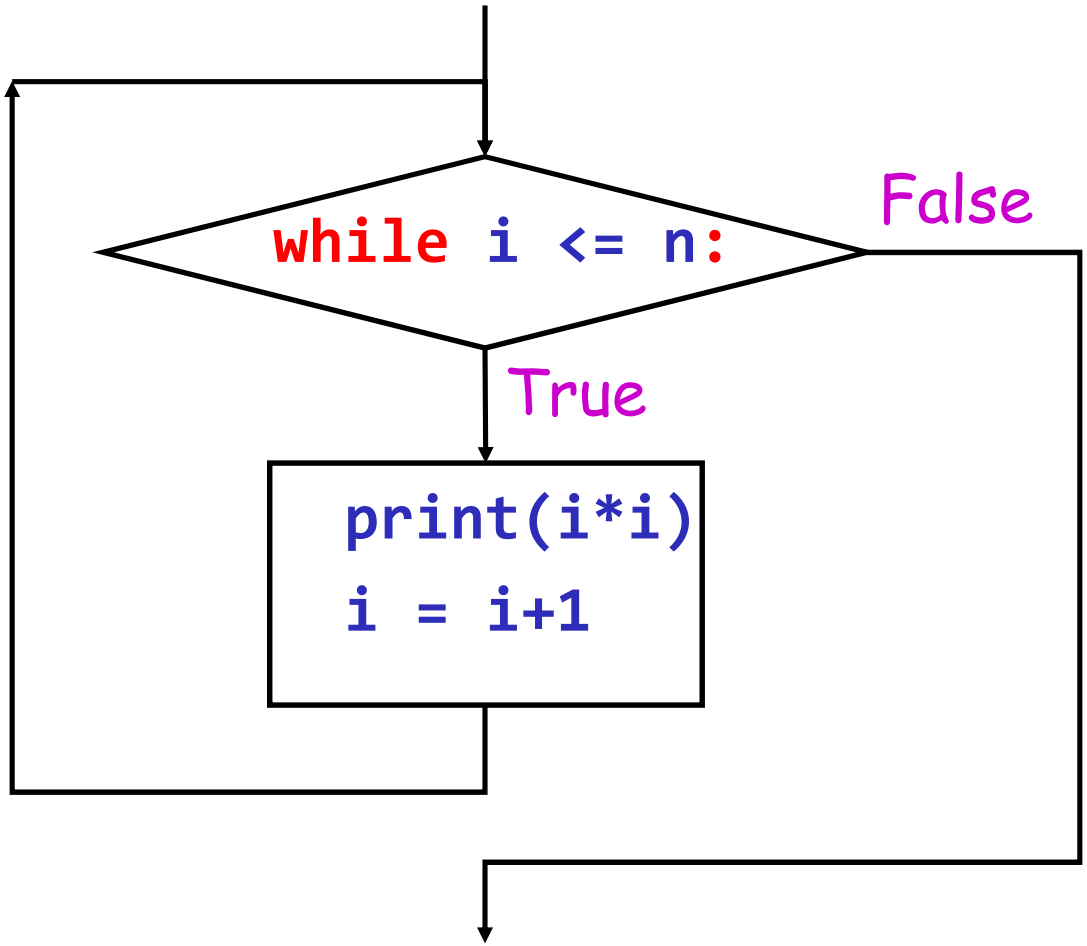
Sintassi Python

```
while condizione:
    corpo-del-while
```



Esempio: scrivere i quadrati degli interi da 1 a n.

```
n = int(input("digita un numero: "))
i = 1
```



Esempio: scrivere i quadrati degli interi da 1 a n.

```
n = int(input("digita un numero: "))
i = 1
while i <= n:
  print(i*i)
  i = i+1
print("ho finito")
```

L'istruzione for … in

Sintassi Python

```
for identificatore in espressioneLista:
    corpo del for
```

L'istruzione **for...in** esegue le istruzioni del corpo su tutti gli elementi della lista uno dopo l'altro, iniziando dal primo.

Esempio

```
nomi = ['Ada', 'Alan', 'John']
for elemento in nomi:
  print('ciao', elemento)
```

scrive sullo schermo

```
ciao Ada
ciao Alan
ciao John
```

Nota Bene: il **for** si crea internamente una copia di ciascun elemento e agisce su tale copia.

Uso combinato di for e range.

L'uso combinato di un ciclo for e di un range permette di scrivere programmi eleganti e compatti. Il range, infatti, genera automaticamente una sequenza di interi.

Ad esempio per ripetere 4 volte un'istruzione o una sequenza basta scrivere `for i in range(4):`.

Così, per disegnare con la tartaruga un quadrato di 50 pixel di lato, vedremo che si scriverà:

```
for i in range(4):  # ripeti 4 volte
  forward(50)       # vai avanti di 50 pixel
  right(90)         # ruota su te stesso verso destra di 90°
```

Nota che la variabile i assume successivamente i valori 0,1,2,3 che in questo caso non vengono usati.

Scrivere i quadrati degli interi da 1 a n: prog. rivisitato

Avevamo scritto il programma:

```
n = int(input("digita un numero: "))
i = 1
while i <= n:
  print(i*i)
  i = i+1
print("ho finito")
```

Si scrive in modo più compatto ed elegante con un for:

```
n = int(input("digita un numero: "))
for i in range(1, n+1): print(i*i)
print("ho finito")
```

Attenzione! Ricorda che nel range il valore finale è escluso!

Altri cicli su liste.

Vogliamo trasformare le temperature della nostra sequenza da gradi centigradi a gradi Fahrenheit.

In Python la funzione `len(list)` dà come risultato il numero di elementi della lista `list` ("len" è l'abbreviazione della parola inglese "length", "lunghezza"). Allora:

```
n = len(temperature)
for i in range(n):
  temperature[i] = temperature[i]*1.8 + 32
```

`print(temperature)` scrive sullo schermo:

```
[73.4, 65.12, 62.6, 66.2, 76.28, 60.8]
```

Liste di stringhe

```
nomi = ['al Khwarismi','Euclide','Don Knuth','007']
```

Esempio. Facciamo diventare maiuscole tutte le lettere:

```
for i in range(len(nomi)):
  nomi[i] = nomi[i].upper()
print(nomi)
```

scrive sullo schermo:

```
['AL KHWARISMI','EUCLIDE', 'DON KNUTH', '007']
```

Il massimo di una lista.

Supponiamo di aver memorizzato in una lista di nome **spese** una sequenza di numeri che rappresentano delle spese, e di voler ora trovare la spesa massima.

Si inizia assumendo come massimo il primo elemento. Poi si percorre la lista dal secondo elemento in avanti.

```
max = spese[0]
for i in range(1, len(spese)):
  if spese[i] > max: max = spese[i]
print(max)
```

Usando la notazione di Python per le sottoliste, il programma si scrive in modo più compatto ed elegante senza usare l'indice **i**:

```
max = spese[0]
for elem in spese[1:]: if elem > max: max = elem
print(max)
```

Somma degli elementi di una lista

Supponiamo di avere memorizzato in una lista `spese` una sequenza di numeri (interi o con la virgola). Per calcolare la somma delle spese basterà fare:

```
somma = 0
for i in range(len(spese)):
   somma = somma + spese[i]
print somma
```

oppure, in forma più compatta ed elegante senza usare indici:

```
somma = 0
for spesa in spese:
   somma = somma + spesa
print somma
```

Ancora più concisamente.

L'istruzione `somma = somma + spesa`

si può scrivere `somma += spesa`

Ad esempio:

```
somma = 0
for spesa in spese: somma += spesa
print somma
```

Esercizi

Esercizio 1 (molto facile).

Data una lista di parole, trovarne una di lunghezza massima.

Nota: la funzione len, applicata a una stringa, dà come risultato la lunghezza della stringa. Esempio:

len("abate") → 5 len("ciao a tutti") → 12 ecc.

Esercizio 2 (molto facile)
Data una lista di parole, trovare la prima in ordine alfabetico.

Ricorda: nei linguaggi di programmazione moderni l'ordine alfabetico fra stringhe è denotato dallo stesso simbolo <.

Esercizio 3 (elementare)

Data una lista di numeri, calcolarne il prodotto.

Esercizio 4: Data una lista di numeri di almeno due elementi, trovare i due numeri più grandi (non necessariamente distinti).

Concatenazione di liste, moltiplicazione di lista.

La valutazione dell'espressione

["paperino", "zio Paperone"] + ["qui", "quo", "qua"]

produce la lista:

["paperino", "zio Paperone", "qui", "quo", "qua"]

Quindi attenzione: la somma di liste non è commutativa !

$$lista_1 + lista_2 \neq lista_2 + lista_1$$

La valutazione dell'espressione

["qui", "quo", "qua"] *3

produce la lista:

["qui", "quo", "qua", "qui", "quo", "qua", "qui", "quo", "qua"]

Per i più curiosi: quanto fa ["qui", "quo", "qua"] *0 ?
E ["qui", "quo", "qua"] * -2 ?

Esercizi

1. Scrivi un programma che, date due liste numeriche numeri1 e numeri2 uguale lunghezza, costruisce e scrive sullo schermo una terza lista in cui ciascun elemento è la somma dei corrispondenti elementi delle due liste.

2. Scrivi un programma che, date due liste di stringhe `nomi` e `cognomi` uguale lunghezza, costruisce una terza lista in cui ciascun elemento è la stringa composta da nome e cognome corrispondenti, separati da uno spazio. Esempio:

```
nomi = ["Ada", "Alan", "Tony", "Li"]
cognomi = ["Byron", "Turing", "Hoare", "Gong"]
```

risult.: `["Ada Byron", "Alan Turing", "Tony Hoare", "Li Gong"]`

3. Perfeziona i programmi precedenti in modo che se le liste non sono di uguale lunghezza non faccia nulla ma scriva un opportuno messaggio sullo schermo.

Esercizi

4. Modifica i programmi precedenti in modo che gli "elementi finali in più" della lista più lunga vengano inseriti in fondo nel risultato.

```
nNomi = len(nomi)
nCognomi = len(cognomi)
risult = []
i = 0
while ... :
  risult.append( ... )
  ...
if ... :
  risult.extend(...)
else: ...
```

List comprehension (comprensione di lista)

Nel linguaggio matematico, per indicare un insieme di elementi che soddisfano a una certa condizione P, si scrive:

$$\{x : P(x)\}$$

che si legge: l'insieme di tutti gli x tali che P(x).

Si può anche scrivere, ad es.:

$$\{x^2 : x \text{ è un intero positivo} < 10\}$$

che si legge: l'insieme degli x^2 tali che x è un intero positivo < 10
cioè l'insieme {1, 4, 9, 16, 25, 36, 49, 64, 81}

In Python e in qualcuno dei linguaggi più moderni esiste un costrutto analogo per creare delle liste, la list comprehension:

```
quadrati = [x**2 for x in range(1,10)]
```

La list comprehension è un costrutto sintattico molto potente che permette di costruire delle liste in modo molto conciso.

List comprehension: altri esempi.

Data una lista

```
temperature = [5,-2,11,-6, 0, 8, 23,-4, 17,-2,-1]
```

possiamo ottenere la lista delle temperature negative semplicemente scrivendo:

```
gelate = [x for x in temperature if x < 0]
```

si ottiene che `gelate` è la lista `[-2,-6,-4,-2,-1]`

Una comprensione più complessa: le terne pitagoriche.

```
terneP = [(a,b,c) for a in range(1,20)
                  for b in range(a,20)
                  for c in range(b,20)
                  if a**2 + b**2 == c**2]
```

`terneP` è la lista: `[(3, 4, 5), (5, 12, 13), (6, 8, 10), (8, 15, 17), (9, 12, 15)]`

Esercizi

1. Scrivere una list comprehension che, utilizzando gli operatori logici, fornisca la lista dei valori compresi fra -3 e 10 nella lista temperature dell'esempio precedente.

2. Scrivere una list comprehension che fornisca la lista di tutte le potenze di 2 con esponente da 0 a 10 inclusi (cioè 1, 2, 4, 8, 16, …, 1024), ricordando che x^n in Python si scrive `x**n`.