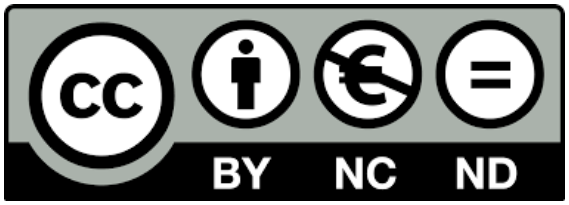
Introduzione alla programmazione

ITS - Umbria

A.S. 2022-23

MATERIA: Fondamenti di informatica

Docente: prof. Paolo Bernardi



Liste: un'introduzione "light"

Python riconosce una certa quantità di tipi di dati composti, usati per raggruppare insieme altri valori. Il più versatile è il tipo lista, che può essere scritto come una sequenza, compresa tra parentesi quadre, di valori (gli elementi della lista) separati da virgole.

Gli elementi della lista non devono essere necessariamente tutti dello stesso tipo.

Liste: un'introduzione "light"

```
Python Console
>>> a = ["ITS", "Matematica", "Informatica", 16, 5, 2019]
>>> a
['ITS', 'Matematica', 'Informatica', 16, 5, 2019]

>>>
```

Attenzione: in questo esempio dobbiamo prestare attenzione ai caratteri "[" e "]". Sono proprio queste ultimi, le **parentesi quadre**, che indicano a Python che vogliamo creare una lista.

Liste: operazioni

Come per le stringhe, è possibile applicare una serie di operazioni sulle liste, come dalla tabella sottoriportata:

Operatore	Descrizione
+	Concatenazione - Aggiunge valori su entrambi i lati dell'operatore
*	Ripetizione: crea nuove liste, concatenando più copie della stessa lista
[]	Slice: fornisce l'elemento dall'indice specificato
[:]	Range Slice - Fornisce gli elementi dell'intervallo specificato
in	Appartenenza - Restituisce vero se un elemento esiste nella lista specificata
not in	Appartenenza - Restituisce vero se un elemento non esiste nella lista specificata

Liste: concatenazione e ripetizione

```
Python Console
>>> a = ["ITS", "Matematica", "Informatica", 16, 5, 2019]
>>> a
['ITS', 'Matematica', 'Informatica', 16, 5, 2019]
>>> b = a + ["fine", 17]
>>> b
['ITS', 'Matematica', 'Informatica', 16, 5, 2019, 'fine', 17]
>>> c = [0.0]
>>> c
[0.0]
>>> c * 10
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

>>>
```

Liste: slicing e operatore appartenenza

Come per gli indici delle stringhe, gli indici delle liste iniziano da 0, e anche le liste possono essere "affettate" (slicing); inoltre, è possibile verificare l'appartenenza o meno di un elemento (operatore in)

```
Python Console
>>> a = ["ITS", "Matematica", "Informatica", 16, 5, 2019]
>>> a[0]
'ITS'
>>> a[3]
16
>>> a[-2]
5
>>> a[1:-1]
['Matematica', 'Informatica', 16, 5]
>>> "ITS" in a
True
>>> 20 in a
False

>>>
```

Liste: slicing e operatore appartenenza

Che cosa succede se specifichiamo però un indice che non esiste? Viene scatenato un errore o, più precisamente, viene sollevata un'eccezione

```
Python Console
>>> a = ["ITS", "Matematica", "Informatica", 16, 5, 2019]
>>> a[10]
Traceback (most recent call last):
  File "<input>", line 1, in <module>
IndexError: list index out of range

>>>
```

Liste: mutabilità delle liste

Al contrario delle stringhe, che sono immutabili, è possibile modificare gli elementi individuali di una lista:

```
Python Console
>>> s = "Ci piace tanto l'ITS!"
>>> s[0] = "V"
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> a = ["ITS", "Matematica", "Informatica", 16, 5, 2019]
>>> a[3] = a[3] + 7
>>> a[2] = "Prosciutto"
>>> a
['ITS', 'Matematica', 'Prosciutto', 23, 5, 2019]

>>>
```

Liste: mutabilità delle liste

È anche possibile assegnare valori alle "fette", e questo può pure modificare le dimensioni della lista:

```
Python Console
>>> a = ["ITS", "Matematica", "Informatica", 16, 5, 2019]
>>> a[0:3] = [1, 2, 3]    #  Rimpiazza alcuni elementi
>>> a
[1, 2, 3, 16, 5, 2019]
>>> a[0:3] = []           #  Rimuove alcuni elementi
>>> a
[16, 5, 2019]
>>> a[1:1] = ["x", "y"]   #  Inserisce alcuni elementi
>>> a
[16, 'x', 'y', 5, 2019]

>>>
```

Liste: funzione len

Analogamente alle stringhe, è possibile ottenere il numero degli elementi contenuti in una lista attraverso la funzione len(lista)

```
Python Console
>>> a = ["ITS", "Matematica", "Informatica", 16, 5, 2019]
>>> len(a)
6
>>> a[0:3] = []            #  Rimuove alcuni elementi
>>> a
[16, 5, 2019]
>>> len(a)
3
>>> a[1:1] = ["x", "y"]    #  Inserisce alcuni elementi
>>> a
[16, 'x', 'y', 5, 2019]
>>> len(a)
5

>>>
```

L'istruzione for: introduzione

L'istruzione **for** di Python differisce un po' da quella a cui si è abituati in C o Pascal. Piuttosto che iterare sempre su una progressione aritmetica (come in **Pascal**), o dare all'utente la possibilità di definire sia il passo iterativo che la condizione di arresto (come in **C**), in Python l'istruzione **for** compie un'iterazione sugli elementi di una qualsiasi sequenza (p.e. una lista o una stringa), nell'ordine in cui appaiono nella sequenza.

L'istruzione for: sintassi

Sintassi

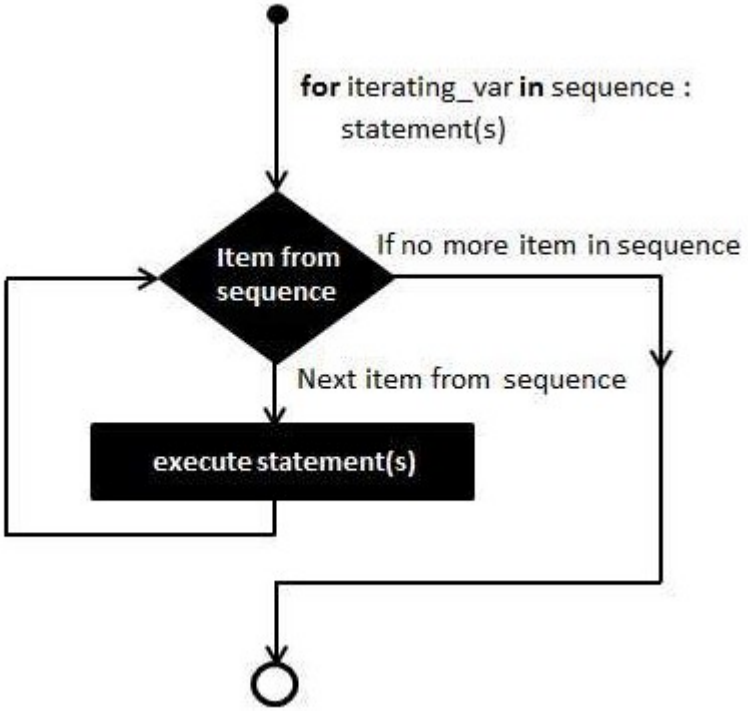
```
>>> for iterating_var in sequence:
...   statements(s)
```

Il primo elemento valutato è sequence. Quindi, il primo elemento della sequenza viene assegnato alla variabile iterating_var. Successivamente, viene eseguito il blocco di istruzioni.

Ogni elemento della sequenza sequence è assegnato a iterating_var e il blocco delle istruzioni viene eseguito fino a quando non viene esaurita l'intera sequenza.

L'istruzione for: diagramma di flusso

Diagramma di Flusso



L'istruzione for: esempi

```
Python Console
>>> a = "Giovedì"
>>> for s in a:
...     print("Elemento:", s)
...
Elemento: G
Elemento: i
Elemento: o
Elemento: v
Elemento: e
Elemento: d
Elemento: ì

>>>
```

L'istruzione for itera su tutti gli elementi (caratteri) della stringa "Giovedì", stampandoli in sequenza uno dopo l'altro

L'istruzione for: esempi

Altro esempio, con una lista di stringhe e formattazioni diverse

```
Python Console
>>> for parola in ["Oggi", "è", "giovedì"]:
...     print(parola)
...
Oggi
è
giovedì

>>>
```

```
Python Console
>>> for parola in ["Oggi", "è", "giovedì"]:
...     print(parola, end=" ")
...
Oggi è giovedì
>>>
```

La funzione range: sintassi

Se è necessario iterare su una successione di numeri, viene in aiuto la funzione built-in range(), che genera un iteratore.

Sintassi

range([start,] stop[, step])

- *start : numero iniziale della sequenza.*
- *stop : genera numeri fino a, ma non incluso, questo numero.*
- *step : differenza tra ogni numero nella sequenza.*

La funzione range: esempi

Se è necessario generare una sequenza, è necessario far precedere la funzione range dalla funzione list(), per convertire l'iteratore in una sequenza (lista, nel nostro caso).

```
Python Console
>>> range(10)                # Genera un iteratore
range(0, 10)
>>> list(range(10))          # Genera una lista
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>>
```

La funzione range: esempi

L'estremo destro passato alla funzione non fa mai parte della lista generata; list(range(10)) genera una lista di 10 valori, esattamente gli indici leciti per gli elementi di una sequenza di lunghezza 10. È possibile far partire l'intervallo da un altro numero (start), o specificare un incremento diverso (step).

```
>>> range(10)                # Genera un iteratore
range(0, 10)
>>> list(range(10))          # Genera una lista
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(3, 10))       # Genera una lista partendo da 3
[3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 10, 2))    # Genera una lista partendo da 1 con passo 2
[1, 3, 5, 7, 9]
>>> list(range(10, 0, -1))   # Genera una lista ordinata in senso decrescente
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> list(range(9, -1, -1))   # Genera una lista ordinata in senso decrescente
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

>>>
```

La funzione range: esempi

Per effettuare un'iterazione sugli indici di una sequenza, si usino in combinazione range() e len() come segue:

```
Python Console
>>> a = ['Ma', 'quanto', 'ci', 'piace', "l'ITS!"]
>>> for i in range(len(a)):
...     print(i, ":", a[i])
...
0 : Ma
1 : quanto
2 : ci
3 : piace
4 : l'ITS!

>>>
```

L'istruzione while: sintassi

Sintassi

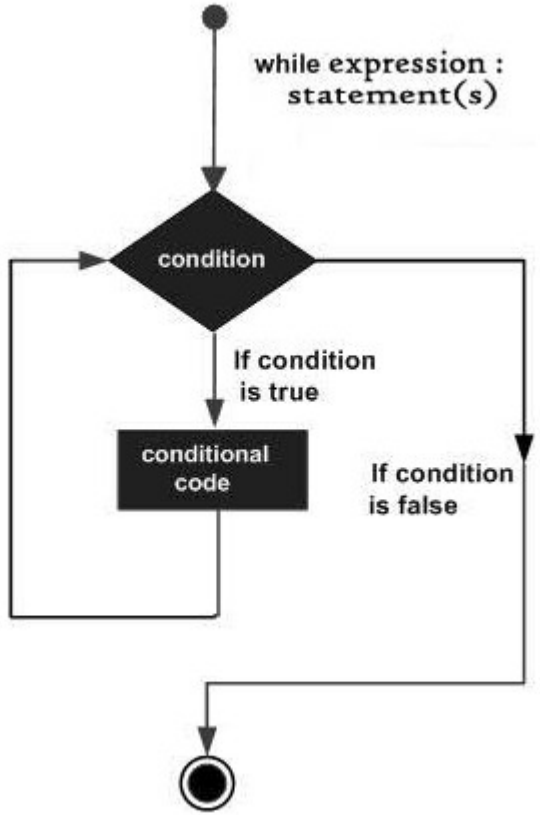
```
>>> while expression:
...   statements(s)
```

Una dichiarazione di ciclo **while** nel linguaggio di programmazione Python esegue ripetutamente un blocco di istruzioni fintanto che una determinata condizione è vera.

Quando la condizione diventa falsa, il controllo del programma passa alla linea immediatamente successiva al ciclo.

L'istruzione while: diagramma di flusso

Diagramma di Flusso



L'istruzione while: esempi

```
Python Console
>>> i = 0
>>> while i < 10:
...     print(i)
...     i = i + 1
...
0
1
2
3
4
5
6
7
8
9
>>>
```

L'istruzione while viene eseguita fintantoché la condizione i < 10 resta vera. La variabile i viene stampata e incrementata

L'istruzione while: cicli infiniti

Un loop diventa loop infinito se una condizione non diventa FALSE.

È necessario essere cauti quando si utilizzano cicli while a causa della possibilità che questa condizione non si risolva mai in un valore FALSE.

Ciò si traduce in un ciclo che non finisce mai. Tale ciclo è chiamato ciclo infinito (infinite loop).

Un ciclo infinito potrebbe essere utile nella programmazione client / server in cui il server deve essere eseguito continuamente in modo che i programmi client possano comunicare con esso come e quando richiesto.

L'istruzione while: cicli infiniti (esempio)

```
1   var = 1
2   while var == 1:      # ATTENZIONE: loop infinito
3       num = int(input("Inserisci un numero : "))
4       print("Hai immesso:", num)
5
6   print("Ciclo finito!")
7
```

L'esempio precedente esegue un ciclo infinito e diventa necessario usare CTRL + C (CTRL + D) per uscire dal programma.

L'istruzione while: cicli infiniti (esempio)

```
Inserisci un numero : 10
Hai immesso: 10
Inserisci un numero : 2
Hai immesso: 2
Inserisci un numero : -5
Hai immesso: -5
Inserisci un numero : 8
Hai immesso: 8
Inserisci un numero : ^D
Traceback (most recent call last):
  File "/run/media/marcos/PB/Simonetti/Marco/Studio/Informatica/Programmazione/Python/ITS/Lezione_01/prova01.py", line 3, in <module>
    num = int(input("Inserisci un numero : "))
EOFError: EOF when reading a line

Process finished with exit code 1
```

Le istruzioni break e continue; else nei cicli

L'istruzione **break**, come in C, esce immediatamente dal ciclo **for** o **while** più interno che la racchiude.

L'istruzione **continue**, anch'essa presa a prestito dal C, prosegue con l'iterazione seguente del ciclo.

Le istruzioni di ciclo possono avere una clausola **else** che viene eseguita quando il ciclo termina per esaurimento della lista (con **for**) o quando la condizione diviene falsa (con **while**), ma non quando il ciclo è terminato da un'istruzione break.

Di seguito si veda un'applicazione di quanto detto mediante il ciclo seguente, che ricerca numeri primi.

L'istruzioni break e continue: diagrammi

Diagramma di Flusso: BREAK

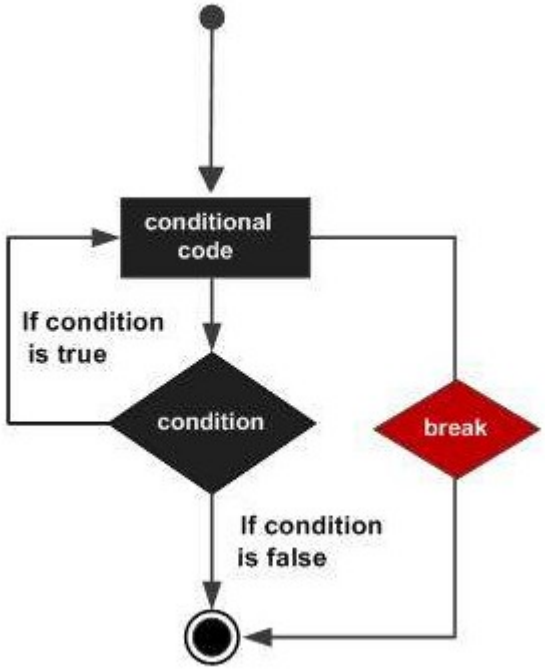
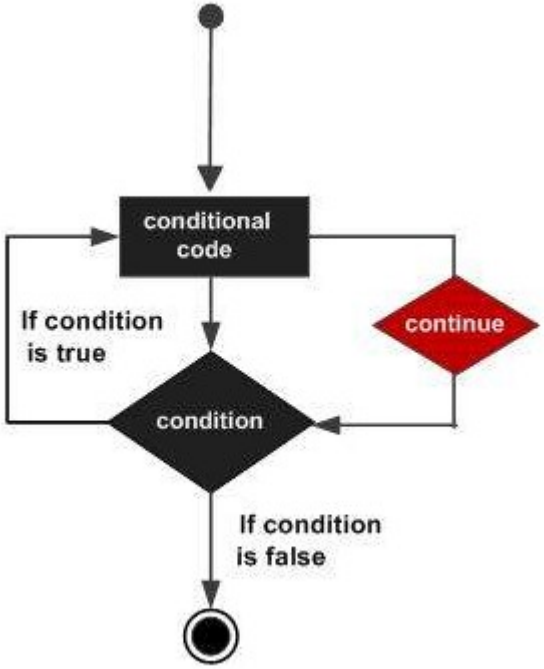


Diagramma di Flusso: CONTINUE



L'istruzioni break e continue: esempio

prova01.py

```
1    for n in range(2, 10):
2        for x in range(2, n):
3            if n % x == 0:
4                print(n, "è uguale a", x, "*", n // x)
5                break
6        else:    # Il ciclo scorre la sequenza senza trovare il fattore
7            print(n, "è un numero primo")
8
```

```
2 è un numero primo
3 è un numero primo
4 è uguale a 2 * 2
5 è un numero primo
6 è uguale a 2 * 3
7 è un numero primo
8 è uguale a 2 * 4
9 è uguale a 3 * 3

Process finished with exit code 0
```

Tipi booleani

Anche Python, come ogni linguaggio che si rispetti, ha i suoi valori **booleani vero e falso**, che corrispondo agli oggetti costanti **True** e **False**.

```
Python Console
>>> numeri_pari = [2, 4, 6, 8]
>>> x = 1 in numeri_pari
>>> print(x)
False
>>> y = 6 in numeri_pari
>>> print(y)
True

>>>
```

Tipi booleani

Se vogliamo inizializzare un valore booleano dobbiamo ricordarci di non usare gli apici come per le stringhe

```
Python Console
>>> var1 = True       # così va bene
>>> var2 = "True"     # così non va bene: è una stringa

>>>
```

Tipi booleani: operatori di confronto

Tabella riepilogativa degli operatori che restituiscono un tipo booleano

Operatore	Descrizione	Esempio
==	*Se i valori di due operandi sono uguali, la condizione diventa vera.*	*(a == b)*
! =	*Se i valori di due operandi non sono uguali, la condizione diventa vera.*	*(a != b)*
<>	*Se i valori di due operandi non sono uguali, la condizione diventa vera.*	*(a <> b)*
>	*Se il valore dell'operando sinistro è maggiore del valore dell'operando destro, allora la condizione diventa vera.*	*(a > b)*
<	*Se il valore dell'operando sinistro è inferiore al valore dell'operando destro, allora la condizione diventa vera.*	*(a < b)*
> =	*Se il valore dell'operando di sinistra è maggiore o uguale al valore dell'operando di destra, la condizione diventa vera.*	*(a >= b)*
<=	*Se il valore dell'operando di sinistra è minore o uguale al valore dell'operando di destra, la condizione diventa vera.*	*(a <= b)*

Tipi booleani: operatori logici

Esistono i seguenti operatori logici supportati dal linguaggio Python

Operatore	Descrizione	Esempio
and	*Se entrambi gli operandi sono veri, la condizione diventa vera.*	*a and b*
or	*Se uno qualsiasi dei due operandi è diverso da zero, la condizione diventa vera.*	*a or b*
not	*Utilizzato per invertire lo stato logico del suo operando.*	*not a*

Essi possono anche essere combinati con gli operatori di confronto

Tipi booleani: tabelle di verità

not

A	$\overline{A}$
F	**V**
V	**F**

and

A	B	$A \wedge B$
F	F	**F**
F	V	**F**
V	F	**F**
V	V	**V**

or

A	B	$A \vee B$
F	F	**F**
F	V	**V**
V	F	**V**
V	V	**V**

Tipi booleani: esempio

```
Python Console
>>> a, b = 3, 5
>>> a < b
True
>>> (a < b) and (b > 0)
True
>>> (a > b) or (b < 0)
False

>>>
```

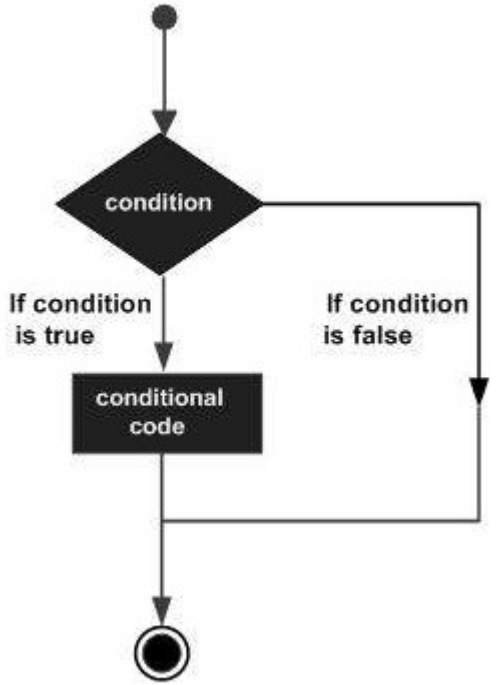
Processo decisionale: if, elif e else

Il processo decisionale è un'anticipazione delle condizioni che si verificano durante l'esecuzione del programma e la specifica delle azioni intraprese in base alle condizioni.

Le strutture decisionali valutano più espressioni che producono VERO (True) o FALSO (False) come risultato. È necessario determinare quale azione intraprendere e quali istruzioni eseguire se il risultato è VERO oppure FALSO.

Processo decisionale: diagramma di flusso

Di seguito è riportata la forma generale di una tipica struttura decisionale presente nella maggior parte dei linguaggi di programmazione.



Istruzione if: sintassi

È simile a quello di altri linguaggi. L'istruzione if contiene un'espressione logica che consente di confrontare i dati e di prendere una decisione in base al risultato del confronto.

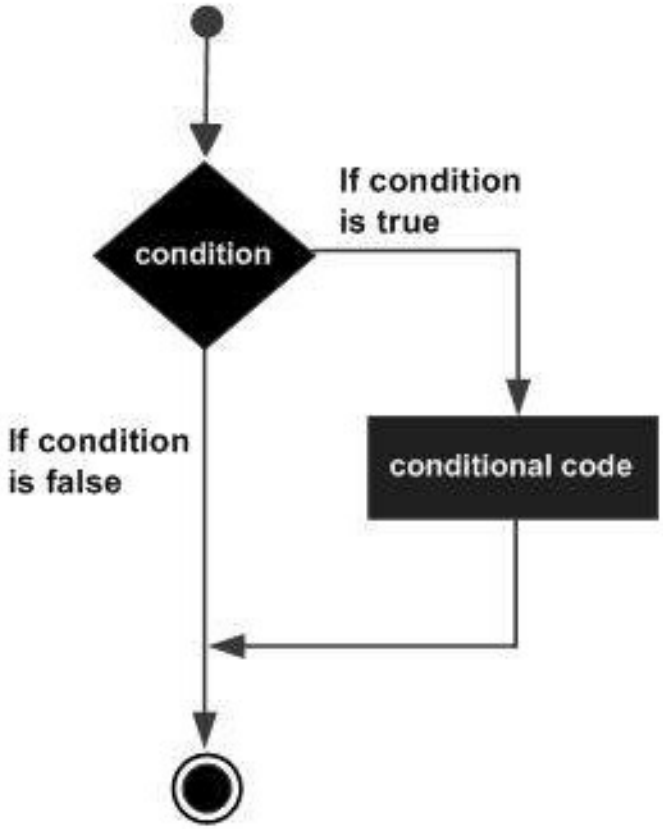
Sintassi

```
>>> if expression:
...   statements(s)
```

Istruzione if: diagramma di flusso

Se l'espressione booleana restituisce TRUE, viene eseguito il blocco di istruzioni all'interno dell'istruzione if.

Se l'espressione booleana viene valutata su FALSE, viene eseguito il primo set di codice dopo la fine dell'istruzione if.



Istruzione if: esempio

```
Python Console
>>> a, b = 3, 5
>>> if a < b:
...         print("a < b")
...
a < b
>>> if a > b:
...         print("a < b")
...
...

>>>
```

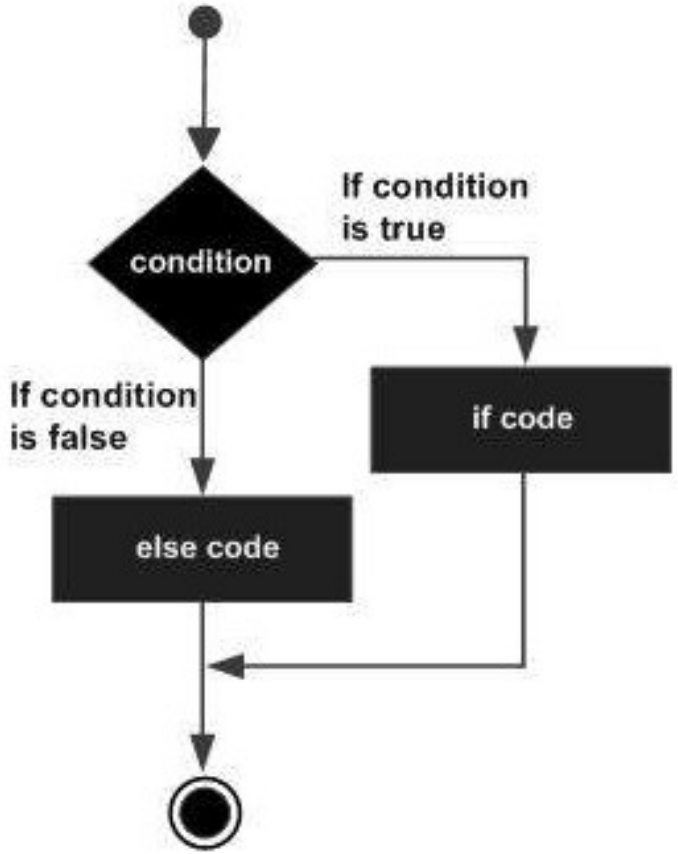
Istruzioni if else: sintassi

Un'altra istruzione può essere combinata con un'istruzione if . L'istruzione ELSE contiene il blocco di codice che viene eseguito se l'espressione condizionale nell'istruzione if restituisce un valore pari a 0 o un valore FALSE. La dichiarazione else è una dichiarazione opzionale; se esiste, essa è unica e chiude il blocco condizionale.

Sintassi

```
>>> if expression:
...   true_statements(s)
... else:
...   false_statements(s)
```

Istruzioni if else: diagramma di flusso



Istruzioni if else: esempio

```
Python Console
>>> a, b = 3, 5
>>> if a < b:
...         print("a < b")
... else:
...         print("a > b")
...
a < b

>>>
```

```
Python Console
>>> a, b = 3, 2
>>> if a < b:
...         print("a < b")
... else:
...         print("a > b")
...
a > b

>>>
```

Istruzioni if elif else: sintassi

L'istruzione elif consente di controllare una sequenza di espressioni booleane ed eseguire un blocco di codice non appena una delle condizioni viene valutata VERA.

Sintassi

```
>>> if expression_1:
...   statements_1(s)
... elif expression_2:
...   statements_2(s)
... elif expression_3:
...   statements_3(s)
... elif expression_4:
...   statements_4(s)
... else:
...   statements_5(s)
```

Istruzioni if elif else: utilizzo

Simile all'istruzione **else**, la dichiarazione **elif** è facoltativa.

Tuttavia, a differenza di **else**, per la quale può esserci al massimo una dichiarazione, possono essere presenti un numero arbitrario di istruzioni **elif** che seguono un **if**.

Python non possiede istruzioni **switch** o **case** come in altri linguaggi, ma possono essere utilizzati costrutti equivalenti usando **if** con **elif**

Istruzioni if elif else: esempio

prova01.py | prova02.py

```
1   var = 100
2   if var == 200:
3       print("1 - Got a true expression value")
4   elif var == 150:
5       print("2 - Got a true expression value")
6   elif var == 100:
7       print("3 - Got a true expression value")
8   else:
9       print("4 - Got a false expression value")
10  print(var)
11  print("Good bye!")
12
```

Istruzioni if elif else: esempio

Output ottenuto

```
3 - Got a true expression value
100
Good bye!

Process finished with exit code 0
```

ESERCIZI: comprensione di un programma

prova01.py | prova02.py

```
while True:
    s = input("Inserisci un numero intero positivo: ")
    if s.isdecimal():
        break
    print("Errore di inserimento.")
    print("-" * 35)
n = int(s)
a = 1
for i in range(n + 1):
    s = "{:>4}".format(a)
    print(s)
    a = a * 2

```

Il linguaggio Python (ver. 3)

PROBLEMI

14 Calcola la somma di tre valori forniti in input.

15 Calcola l'area di un triangolo di cui sono forniti in input la base e l'altezza.

16 Ricevute in ingresso le lunghezze dei tre lati di un triangolo, determina se si tratta di un triangolo equilatero, isoscele o scaleno.

17 Dato in ingresso il raggio, calcola la circonferenza e l'area del cerchio.

18 Calcola il volume di un parallelepipedo di cui sono forniti in input altezza, larghezza e profondità.

19 Ricevuti in ingresso due valori, fanne la sottrazione se il primo è maggiore del secondo o fanne la somma se il primo è minore del secondo. Se sono uguali, segnalalo a video.

20 Determina se un voto ricevuto in ingresso è sufficiente o insufficiente.

TUTOR
Diagramma di flusso e codifica in Scratch - C - Pascal - Python - Java

21 Calcola la media di cinque voti ricevuti in ingresso.

22 Ricevuti in ingresso cinque voti, conta quanti sono sufficienti, quanti insufficienti e visualizza se sono di più i voti positivi o quelli negativi.

23 Determina il voto più basso tra cinque voti ricevuti in input.

TUTOR
Diagramma di flusso e codifica in Scratch - C - Pascal - Python - Java

24 Determina il voto più alto tra sei voti ricevuti in input.

25 Visualizza in ordine crescente e decrescente tre valori ricevuti in input.

26 Visualizza in ordine crescente quattro valori ricevuti in input.

27 Ricevuti in ingresso i nomi di due squadre di calcio e il risultato della partita, visualizza la squadra vincente o, in caso di pareggio, entrambi i nomi delle squadre.

TUTOR
Diagramma di flusso e codifica in Scratch - C - Pascal - Python - Java

28 Ricevuti in ingresso il nome di una squadra di calcio, il numero di partite vinte e il numero di partite pareggiate, calcola i punti in classifica.

29 Realizza una semplice calcolatrice che, ricevuti in input due operandi e un operatore aritmetico, fornisca in uscita il risultato dell'operazione richiesta.

30 Dati in ingresso il costo di quattro libri, calcola il costo medio per libro.

31 Dati in ingresso il costo di cinque libri, conta quanti di questi hanno un prezzo superiore a 15 €.

32 Un libro deve essere restituito in biblioteca dopo 15 giorni di prestito altrimenti si è multati di 0,80 € al giorno di ritardo. Ricevuto in ingresso il numero di giorni di un prestito, visualizza se il socio deve essere multato per il ritardo e a quanto ammonta la multa da pagare.

TUTOR
Diagramma di flusso e codifica in Scratch - C - Pascal - Python - Java

33 Calcola e applica lo sconto al prezzo fornito in ingresso di un hoverboard a seconda delle seguenti fasce di prezzo:

- se prezzo < 100 € → sconto 5%
- se prezzo ≥ 100 € e < 200 € → sconto 10%
- se prezzo ≥ 200 € → sconto 15%

34 Ricevuti in ingresso il numero di persone adulte e il numero di bambini che vanno al cinema, calcola il costo totale del biglietto sapendo che gli adulti pagano 7 € mentre i bambini pagano 4 €.

35 Ricevuti in ingresso il costo del biglietto per visitare un museo e il numero di alunni di una classe in gita scolastica, calcola la spesa totale.

Ora a voi...

Bibliografia

- Marco Beri, Programmare in Python, Apogeo
- Kenneth A. Lambert, Programmazione in Python, Apogeo
- Python Software Foundation, https://docs.python.org/3/
- Python - sito ufficiale della comunità italiana, www.python.it/doc/
- W3C programming language tutorials, www.w3schools.com/python/
- "The Hitchhiker's Guide to Python!", Kenneth Reitz, Tanya Schlusser, 2018,
  - https://media.readthedocs.org/pdf/python-guide/latest/python-guide.pdf
- www.tutorialspoint.com
  - www.tutorialspoint.com/python3/index.htm

Bibliografia

Software (free license)

- Interprete Python3: www.python.org/downloads/
- IDE: PyCharm (community): www.jetbrains.com/pycharm/download/

Immagini (free license)

- Fonte: www.duckduckgo.com