Laboratorio di Informatica

Corso di Laurea Magistrale in Ingegneria per l'Ambiente e il Territorio

A.A. 2021/2022

Docente: Lorenzo Putzu

Lezione 02

Il linguaggio Python

Istruzioni condizionali

- In tutti i linguaggi di alto livello sono disponibili per questo scopo:
  - le espressioni condizionali, che consentono di esprimere condizioni che possono essere **vere** o **false** (per es., *a* ”= 0)
  - l’istruzione condizionale, che consente di eseguire due diverse sequenze d’istruzioni alternative tra loro, in base al valore di un’espressione condizionale

Espressioni condizionali

- Il tipo più semplice di espressione condizionale consiste nel confronto tra due espressioni.
  - **Sintassi:** espressione1 operatore espressione2
  - espressione1 e espressione2 sono due espressioni Python **qualsiasi**
  - operatore è un simbolo che indica il confronto da eseguire tra le due espressioni

simbolo	significato
==	"uguale a" (notare il **doppio** simbolo =, per evitare ambiguità con l'istruzione di assegnamento)
!=	"diverso da"
<	"minore di"
<=	"minore o uguale a"
>	"maggiore di"
>=	"maggiore o uguale a"

Espressioni condizionali

- Il risultato di un espressione condizionale assume il valore logico
  - True (vero)
  - False (falso)
- Il loro valore può essere assegnato a una variabile o stampato nella *shell*; per esempio, assumendo che alla variabile x sia già stato assegnato un valore numerico:
  - risultato = (x > 0)
  - print(x > 0)

Espressioni condizionali composte

- Le espressioni condizionali viste finora si dicono semplici, poiché consistono in un singolo confronto tra due valori.
- Possiamo costruire espressioni condizionali composte, ottenute combinando espressioni condizionali **qualsiasi** (anche a loro volta composte) per mezzo di operatori logici :
  - **Sintassi**:
  - espr-cond1 and espr-cond2
  - espr-cond1 or espr-cond2
  - not espr-cond
- E' possibile usare le parentesi tonde per definire l'ordine degli operatori logici

Istruzioni condizionali

- Le espressioni condizionali sono uno dei componenti delle **istruzioni** condizionali, che consentono di esprimere in un programma la scelta tra **due** diverse sequenze di istruzioni in base al verificarsi o meno di una certa **condizione** durante l'**esecuzione** dello stesso programma.
- In linguaggio naturale, un'istruzione condizionale esprime la seguente richiesta all'esecutore di un algoritmo:
  - **se** una data **condizione** è vera,
  - **allora** esegui una certa sequenza d'istruzioni,
  - **altrimenti** esegui un'altra sequenza d'istruzioni

Istruzioni condizionali

- Sintassi

  if espr-cond :

  sequenza di istruzioni 1

  else :

  sequenza di istruzioni 2

- espr-cond è un'espressione condizionale **qualsiasi**
- sequenza di istruzioni 1 e sequenza di istruzioni 2 sono due sequenze di una o più istruzioni **qualsiasi** e devono essere correttamente **indentate (rientri)**

Istruzioni condizionali

- Esempio
- Assumiamo di aver acqusito un input da tastiera e che vanga memorizzato nella variabile x

```
if x > 0 :
  print("La condizione è vera.")
else:
  print("La condizione è falsa.")
```

- ATTENZIONE alla sintassi e all'indentazione

Istruzioni condizionali

- Il ramo **else** è opzionale perciò non deve essere indicata un'istruzione da eseguire se la **condizione** è falsa
- A volte invece potrebbe essere necessario verificare più **condizioni** perciò si possono aggiungere un numero di rami **elif** pari alle condizioni da verificare
  - Sintassi completa
  - `if...`
  - `if...else...`
  - `if...elif...`
  - `if...elif...else...`

Istruzioni condizionali nidificate

- Un'istruzione condizionale può contenere al suo interno istruzioni qualsiasi, e quindi anche altre istruzioni condizionali. Si parla in questo caso di istruzioni *nidificate* (o *annidate*).
- L'uso di istruzioni condizionali nidificate consente di esprimere la scelta tra **più di due** sequenze di istruzioni alternative.
- Un'istruzione condizionale nidificata all'interno di un'altra si scrive con la stessa sintassi mostrata in precedenza, questo implica che:
  - le parole-chiave if e (se presente) else devono essere allineate tra loro, ma con un **rientro** rispetto a quelle dell'istruzione condizionale che le contiene
  - le sequenze di istruzioni che seguono if e else devono essere scritte con un ulteriore rientro

Esercizio

1. Un anno è bisestile se è multiplo di 100 e divisibile per 400, oppure se **non** è multiplo di 100 ma è divisibile per 4. Questa regola vale **solo** per gli anni successivi al 1582. Scrivere un programma che acquisisca un anno, verifichi se è successivo al 1582, e in questo caso stabilisca se sia bisestile o meno

Istruzione iterativa

- In molti algoritmi è necessario **ripetere** una stessa sequenza di operazioni per una o più volte.
- In tutti i linguaggi di alto livello è presente a questo scopo l'istruzione iterativa, che consente di esprimere la seguente richiesta all'esecutore di un algoritmo:
  - **finché** una data **condizione** è vera,
  - **esegui** una certa sequenza di istruzioni

Istruzione iterativa

- Sintassi

  while espr-cond :

  sequenza di istruzioni

- espr-cond è un'espressione condizionale **qualsiasi**
- sequenza di istruzioni consiste in una o più istruzioni **qualsiasi**
- Un'istruzione iterativa viene eseguita **ripetendo** ciclicamente (iterativamente) i seguenti passi:
  - 1. viene valutata espr-cond
  - 2. se espr-cond è vera, si esegue la sequenza di istruzioni, e si **ritorna** al punto 1; se invece espr-cond è falsa l'esecuzione dell'istruzione iterativa termina

Istruzione iterativa

- Esempio
- Assumiamo di aver acqusito un input da tastiera e che vanga memorizzato nella variabile x

```
while x < 2 :
    print("Python")
    x = 3
```

- ATTENZIONE alla sintassi e all'indentazione

Istruzione iterativa

- Nella maggior parte dei casi un'istruzione iterativa ha lo scopo di ripetere una certa sequenza di istruzioni per un numero di volte noto nel momento in cui si **scrive** il programma

- A questo scopo si usa comunemente una alla quale **prima** dell'istruzione iterativa si assegna un valore scelto dal programmatore (di norma, 1); tale valore viene poi **incrementato** di **una** unità in ogni ripetizione dell'istruzione iterativa;
  - l'espressione condizionale verificherà quindi che il valore della stessa variabile sia minore o uguale al numero desiderato di ripetizioni.

- Poiché le variabili usate in questo modo hanno la funzione di tenere traccia del (contare il) numero di iterazioni, vengono comunemente indicate con il termine "**contatori**".

Istruzione iterativa

- Esempio
- Assumiamo di aver acqusito un input da tastiera e che vanga memorizzato nella variabile x

```
x = eval(input("Inserire un numero: "))
k = 1
while k <= x :
    print("Buongiorno")
    k = k + 1
```

- ATTENZIONE alla sintassi e all'indentazione

Assegnamenti composti

- Gli assegnamenti composti rendono più compatto il codice nella scrittura delle operazioni aritmetiche nel caso in cui una variabile venga utilizzata nella stessa e per salvarne poi il contenuto.
  - += > incrementa la variabile che lo precede aggiungendo il valore susseguente;
  - -= > decrementa la variabile che lo precede sottraendo il valore susseguente;
  - *= > moltiplica la variabile che lo precede per il valore susseguente;
  - /= > divide la variabile che lo precede per il valore susseguente;
  - //= > divide la variabile che lo precede per il valore susseguente e restituisce il quoziente intero;
  - %= > calcola il modulo tra la variabile che lo precede ed il valore susseguente;

Assegnamenti composti

- i = i + 1; → i += 1;
- a = a * 3; → a *= 3;
- b = b / 2; → b /= 2;
- y = y % 5; → y %= 5;

- Precedenza più bassa rispetto agli operatori aritmetici.

```
i *= j + k;    →   i = i * ( j + k );
```

Assegnamenti composti

- L’esempio di prima può quindi essere riscritto come segue

```
x = eval(input("Inserire un numero: "))
k = 1
while k <= x :
  print("Buongiorno")
  k += 1
```

Istruzione iterativa nidificata

- Come l'istruzione condizionale anche l'istruzione iterativa può contenere al suo interno istruzioni qualsiasi, quindi anche altre istruzioni iterative e anche in questo caso si parla di istruzioni *nidificate* (o *annidate*).

- L'uso di istruzioni iterative nidificate può essere molto utile quando si lavora su delle tabelle o matrici che quindi presentano delle righe e colonne

- **ATTENZIONE** all'efficienza: troppe istruzioni annidate possono influire pesantemente sui tempi di esecuzione del programma

Esercizi

2. Scrivere un programma che acquisisca una sequenza di numeri di lunghezza data e ne calcoli la somma

3. Scrivere un programma che acquisica una sequenza di numeri fino all'inserimento del valore 0, e ne calcoli la somma

4. Modificare il programma precedente in modo che calcoli anche il numero di valori positivi inseriti