anno 2014-15

Introduzione all'Algoritmica per i Licei

1 – Primo esempio: il massimo di una sequenza.

Elio Giovannetti
Dipartimento di Informatica
Università di Torino

versione 23 febbraio 2015

Il massimo di una sequenza immessa da tastiera.

Esperimento.

Vi comunico lentamente una sequenza di numeri;
voi, senza usare né carta né matita né altri strumenti,
alla fine dovete dirmi qual è il massimo degli elementi.

...

Come avete fatto per dare la risposta giusta ?

Ad ogni passo:

- avete tenuto in memoria il massimo fino a quel momento;
- avete ricevuto il nuovo numero;
- lo avete confrontato con il massimo:
  - se era maggiore lo avete memorizzato come nuovo massimo dimenticando il precedente;
  - altrimenti avete continuato a memorizzare il massimo, dimenticando il numero ricevuto.

L'algoritmo

ascolta il primo numero e memorizzalo come massimo

ripeti indefinitamente :

    ascolta ciò che vien detto (per brevità, detto)

    se detto è "ho finito": esci dal ciclo

    altrimenti:

        considera il numero espresso da detto;

        se numero > massimo: memorizza numero come massimo

fine del ciclo

scrivi o pronuncia il massimo

Nota: l'algoritmo funziona solo se termino la sequenza di numeri con la frase "ho finito"; se ad un certo punto, invece di dire un numero, dico "Buongiorno", non si sa se ho finito la sequenza o se intendo continuare ...

Il massimo di una sequenza immessa da tastiera

Nel computer servono due celle di memoria: una per tenere il "massimo fino a quel momento", e una per mettervi ogni volta il nuovo numero preso dalla tastiera. Chiamiamole **`max`** e **`num`**.

In Python l'input da tastiera è sempre interpretato come una stringa, cioè una sequenza di caratteri. Se essa rappresenta un numero intero, occorre trasformarla esplicitamente nel numero, per mezzo della funzione **`int`**.

Può quindi essere utile disporre di un'altra cella di memoria, che nelle prossime slides per ragioni di spazio chiameremo semplicemente **`s`**, in cui mettere la stringa presa ogni volta da tastiera, prima di trasformarla in numero.

Inoltre, solo per comodità, come segnale di fine-sequenza scegliamo la stringa vuota (che si ottiene pigiando il tasto INVIO senza aver digitato alcun numero).

Riprendiamo l'algoritmo ...

fai l'input di un numero da tast. e memorizzalo come massimo

ripeti indefinitamente :

    esegui l'input da tastiera una nuova stringa

    se la stringa è la stringa vuota: esci dal ciclo

    altrimenti:

        considera il numero espresso dalla stringa;

        se numero > massimo: memorizza numero come massimo

fine del ciclo

scrivi massimo

... e traduciamolo in Python

```
max = int(input("digita un numero: "))
```

ripeti indefinitamente :

    esegui l'input da tastiera una nuova stringa
    se la stringa è la stringa vuota: esci dal ciclo
    altrimenti:
        considera il numero espresso dalla stringa;
        se numero > massimo: memorizza numero come massimo

fine del ciclo

scrivi massimo

... e traduciamolo in Python

```
max = int(input("digita un numero: "))
while True:
    esegui l'input da tastiera una nuova stringa
    se la stringa è la stringa vuota: esci dal ciclo
    altrimenti:
        considera il numero espresso dalla stringa;
        se numero > massimo: memorizza numero come massimo
fine del ciclo
scrivi massimo
```

... e traduciamolo in Python

```
max = int(input("digita un numero: "))
while True:
    s = input("digita numero, INVIO per terminare: ")
    se la stringa è la stringa vuota: esci dal ciclo
    altrimenti:
        considera il numero espresso dalla stringa;
        se numero > massimo: memorizza numero come massimo
fine del ciclo
scrivi massimo
```

... e traduciamolo in Python

```
max = int(input("digita un numero: "))
while True:
   s = input("digita numero, INVIO per terminare: ")
   if s == "": break
   altrimenti:
      considera il numero espresso dalla stringa;
      se numero > massimo: memorizza numero come massimo
fine del ciclo
scrivi massimo
```

... e traduciamolo in Python

```
max = int(input("digita un numero: "))
while True:
   s = input("digita numero, INVIO per terminare: ")
   if s == "": break
   else:
      num = int(s)
      se numero > massimo: memorizza numero come massimo
fine del ciclo
scrivi massimo
```

... e traduciamolo in Python

```
max = int(input("digita un numero: "))
while True:
   s = input("digita numero, INVIO per terminare: ")
   if s == "": break
   else:
      num = int(s)
      if num > max: max = num
```

scrivi massimo

... e traduciamolo in Python

```
max = int(input("digita un numero: "))
while True:
   s = input("digita numero, INVIO per terminare: ")
   if s == "": break
   else:
      num = int(s)
      if num > max: max = num
print(max)
```

Nota: affinché il programma funzioni correttamente occorre che venga immesso almeno un numero.

Analizziamo la prima riga del programma

```
max = int(input("digita un numero: "))
```

1) la funzione predefinita **input** scrive sulla schermo la sequenza di caratteri compresa fra virgolette, poi si mette in attesa che l'utente pigi dei tasti, terminando con il tasto INVIO; a quel punto la funzione dà come risultato al programma la stringa digitata sulla tastiera;

2) il programma prende questa stringa e la passa direttamente alla funzione predefinita **int** che dà come risultato il numero intero rappresentato dalla stringa:

```
int(input("digita un numero: "))
```

3) tale risultato, cioè il numero, viene memorizzato in una cella (una sorta di scatoletta elettronica) di nome **max**.

Come si vede, le istruzioni che compongono la riga vengono eseguite da destra a sinistra, nell'ordine inverso di quello scritto.

Analizziamo le righe successive

```
while True:
```

come vedremo meglio nelle prossime slides, è un comando che fa ripetere indefinitamente la sequenza di istruzioni successive scritte con margine rientrato rispetto all'iniziale di `while`.

Ora analizziamo la sequenza di istruzioni che viene ripetuta.

```
  s = input("digita numero, INVIO per terminare: ")
```

è analoga alla prima riga;

```
  if s == "": break
```

confronta la stringa memorizzata in `s` con la stringa vuota: se sono uguali, esegue l'istruzione `break`, che fa uscire subito dal ciclo di ripetizioni e fa andare alla prima istruzione successiva a quelle del ciclo;

```
  else:
```

altrimenti esegue le righe "rientrate" successive.

Analizziamo le due righe del ramo else

```
num = int(s)
```

come nella prima riga, la funzione `int` prende la stringa contenuta in `s` e dà come risultato il numero corrispondente, che a sua volta viene depositato nella cella `num`.

```
if num > max: max = num
```

questo è il cuore dell'algoritmo: il contenuto della cella `num`, che contiene l'ultimo numero digitato, viene confrontato con il contenuto della cella `max`, che contiene il massimo fino a quel momento, e se il nuovo numero è maggiore di tale massimo, esso viene copiato nella cella `max` cancellando (o, come si dice in gergo informatico, sovrascrivendo) il valore precedente, che non ci serve più e può quindi essere perduto;

se invece il nuovo numero non è maggiore, non si fa niente: il contenuto della cella `max` viene lasciato invariato.

Nota Bene

Anche il contenuto della cella `s` viene ad ogni ciclo sovrascritta con una nuova stringa; quella della volta prima ovviamente non serve più.

Così pure il contenuto della cella `num` viene sovrascritto ogni volta che viene eseguita l'istruzione `num = int(s)`.

Analizziamo la riga finale

```
print(max)
```

tale riga, che è allineata a sinistra senza rientro, non fa parte del ciclo, e viene eseguita una sola volta alla fine, dopo che l'istruzione break ha fatto uscire dal ciclo;

essa si limita a scrivere sullo schermo il contenuto della cella max, che ad ogni ripetizione del ciclo conteneva il massimo fino a quell'istante, e che alla fine conterrà il massimo definitivo.

Una possibile omissione

```
max = int(input("digita un numero: "))
while True:
   s = input("digita numero, INVIO per terminare: ")
   if s == "": break
   else:
      num = int(s)
      if num > max: max = num
print(max)
```

Se la stringa s è la stringa vuota, l'esecuzione della break fa uscire dal ciclo; se non è vuota, anche in assenza di else viene eseguita l'istruzione successiva.

La else si può quindi omettere, anche se metterla non fa male.

Versione senza else

```
max = int(input("digita un numero: "))
while True:
   s = input("digita numero, INVIO per terminare: ")
   if s == "": break
   num = int(s)
   if num > max: max = num
print(max)
```

Nota importante

L'uso della primitiva break per uscire in modo forzato da un ciclo deve essere limitato ai casi, come quello appena visto, in cui esso rende il programma più semplice e più comprensibile.

Ma attenzione: un uso indiscriminato della break conduce a programmi incomprensibili e spesso errati.

Nel seguito del corso, come vedremo, la break non verrà quasi mai usata.

Un problema completamente diverso ... ?

- Vi comunico una sequenza di parole, e alla fine voglio sapere quale di esse è l'ultima in ordine alfabetico – o, come si dice con più precisione in informatica, in ordine lessicografico (da *lessico*, che vuol dire *vocabolario*).
- Ma è lo stesso problema del massimo! Cambia solo il tipo di ordine!
- In Python lo stesso simbolo "<", se usato con le stringhe, ne denota l'ordine lessicografico; si ha quindi, ad esempio:

```
"abate" < "abete" < "cane" < "canestro" < "gatto"
```

Dunque anche il programma Python è lo stesso: l'unica differenza è che ora non devo prima trasformare le stringhe in numeri, ma posso confrontarle direttamente. Più facile!

Esercizi

1. Scrivere e provare il programma Python che riceve in input una sequenza di stringhe terminata dalla stringa vuota, e scrive sullo schermo l'ultima di esse in ordine lessicografico.
2. Scrivere e provare il programma Python che riceve in input una sequenza di stringhe terminata dalla stringa vuota, e scrive sullo schermo la prima di esse in ordine lessicografico.

Un esercizio lievemente diverso.

Esercizio 3. Scrivere un programma Python che riceve in input una sequenza di numeri terminata dalla stringa vuota, e alla fine ne scrive sullo schermo la somma.

Nota.

Mentre è sensato dire che il massimo di una sequenza vuota non esiste, la somma di una sequenza vuota può essere ragionevole considerarla uguale a 0, che è l'elemento neutro dell'addizione.

Assumere che la somma di una sequenza priva di elementi sia 0 permette di scrivere un programma lievemente più semplice di quello del massimo: non c'è bisogno di prendere il primo numero prima di entrare nel ciclo.

Soluzione dell'esercizio 3. La somma degli elementi di una sequenza immessa da tastiera.

- Occorrono anche qui due celle: la somma parziale e il numero letto.
- Corpo del ciclo: ad ogni passo si aggiunge il numero alla somma parziale.
- Inizializzazione: la somma parziale contiene l'elemento neutro della somma, cioè 0.

L'algoritmo, pensando già a Python

```
somma = 0.0
```

ripeti indefinitamente :

    esegui l'input di una nuova stringa

    se la stringa è la stringa vuota: esci dal ciclo

    altrimenti: aggiungi il numero in `somma`

fine del ciclo

scrivi somma

Traduciamolo tutto in Python

```
somma = 0.0
while(True):
    esegui l'input di una nuova stringa
    se la stringa è la stringa vuota: esci dal ciclo
    altrimenti: aggiungi il numero in somma
fine del ciclo
scrivi somma
```

Traduciamolo tutto in Python

```
somma = 0.0
while(True):
    s = input('digita numero, o termina con INVIO: ')
```

se la stringa è la stringa vuota: esci dal ciclo

altrimenti: aggiungi il numero in **somma**

fine del ciclo

scrivi somma

Traduciamolo tutto in Python

```
somma = 0.0
while(True):
   s = input('digita numero, o termina con INVIO: ')
   if s == "": break
```

   altrimenti: aggiungi il numero in **somma**

fine del ciclo

scrivi somma

Traduciamolo tutto in Python

```
somma = 0.0
while(True):
   s = input('digita numero, o termina con INVIO: ')
   if s == "": break
   else: somma = somma + float(s)
```

fine del ciclo

scrivi somma

Traduciamolo tutto in Python

```
somma = 0.0
while(True):
   s = input('digita numero, o termina con INVIO: ')
   if s == "": break
   else: somma = somma + float(s)
print(somma)
```

Nota.

Anche qui la else si può omettere, benché metterla non faccia male.

Versione senza else

```
somma = 0.0
while(True):
   s = input('digita numero, o termina con INVIO: ')
   if s == "": break
   somma = somma + float(s)
print(somma)
```