Esercizio

Produrre uno script che (senza utilizzare le funzioni min/max/sorted/ecc.):

I. determini il minimo tra due numeri
II. determini il minimo della lista [1,5,2,-4,1,55,3,-1]
III. dato un numero n inizializzato a piacere, determini se n e' minore di ogni elemento di una lista di 5 elementi inseriti dall'utente
IV. dato un numero n inizializzato a piacere, determini se n e' minore di ogni elemento di una lista arbitraria inserita dall'utente (si usi `raw_input` all'interno di un ciclo `while`, uscendo dal ciclo all'inserimento del carattere q)
V. determini il minore in una lista di numeri nel range (1,10) inseriti dall'utente
VI. determini il minore in una lista di numeri arbitrari inseriti dall'utente

In [1]:
```
# I
a,b = 5,7
minimo = a
if minimo > b:
    minimo = b
print "il minimo tra %d e %d e' %d"%(a,b,minimo)
```

In [2]:
```
# II
l =  [1,5,2,-4,1,55,3,-1]
minimo = l[0]
for num in l:
    if minimo > num:
        minimo = num
print "il minimo in %s e' %d"%(l,minimo)
```

In [6]:
```
# III
n = 7
l = []
for i in range(5):
    element = raw_input("inserisci un numero: ")
    l.append(element)

n_is_min = True
for num in l:
    if n > num:
        n_is_min = False
        break
print "n "+(not n_is_min)*"non "+"e' il minimo"
```

In [ ]:
```
#IV
n = 7
l = []
ch = raw_input("inserisci un numero (o 'q' per uscire): ")
```

```
while ch != 'q':
    el = int(ch)
    l.append(el)
    ch = raw_input("inserisci un numero (o 'q' per uscire): ")

# si procede come al punto III
n_is_min = True
for num in l:
    if n > num:
        n_is_min = False
        break
print "n = %d "%n + (not n_is_min)*"non " + "e' il minimo"
```

In [ ]:
```
#V
# l'inserimento della lista e' analogo al punto IV
l = []
ch = raw_input("inserisci un numero (o 'q' per uscire): ")
while ch != 'q':
    el = int(ch)
    l.append(el)
    ch = raw_input("inserisci un numero (o 'q' per uscire): ")

minimo = 10 # questo valore verra' certamente sostituito,
            # visto che tutti gli elementi di l sono < 10
# si procede come al punto II
for num in l:
    if minimo > num:
        minimo = num
if len(l) == 0: # l'utente ha inserito subito 'q'
    print "la lista non contiene elementi"
else:
    print "il minimo in %s e' %d"%(l,minimo)
```

In [ ]:
```
#VI
# il problema e' simile al precedente. Dobbiamo solo cambiare
# l'inizializzazione della variabile "minimo"
l = []
ch = raw_input("inserisci un numero (o 'q' per uscire): ")
while ch != 'q':
    el = int(ch)
    l.append(el)
    ch = raw_input("inserisci un numero (o 'q' per uscire): ")

if len(l) > 0: # se la lista non e' vuota
    minimo = l[0] # inizializzo al primo elemento di l

for num in l:
    if minimo > num:
        minimo = num
if len(l) == 0: # l'utente ha inserito subito 'q'
    print "la lista non contiene elementi"
else:
```

```
    print "il minimo in %s e' %d"%(l,minimo)
```

In [7]:
```
# extra. Esempio di funzione che determina l'elemento minimo di una lista
def find_min(l):
    if len(l) == 0: # se la lista e' vuota
        return None
    minimo = l[0] # inizializzo al primo elemento di l
    for num in l:
        if minimo > num:
            minimo = num
    return minimo
```

In [13]:
```
print find_min([]) # -> None
print find_min([1,5,2,44,2,-11,3,-5]) # -> -11
```

Esercizio

1) Script: "cartasassoforbice1.py"

Creare uno script che inizializza una stringa g1 alla stringa "carta" ed una stringa g2 a scelta vostra tra carta, sasso, forbice. Stampare il risultato del confronto secondo le regole del gioco carta sasso forbice:

- carta vince contro sasso
- sasso vince contro forbice
- forbice vince contro carta

Lo script deve funzionare per ogni possibile scelta valida di g2. Esempi:

se g2=carta -> stampa "il gioco e' terminato in pareggio"

se g2=sasso -> stampa "il giocatore 2 ha perso"

se g2=forbice -> stampa "il giocatore 2 ha vinto"

In [1]:
```
g1="carta"
g2="sasso"

print "Scelta giocatore 1: %s" % g1;
print "Scelta giocatore 2: %s" % g2;

#dato che g1==carta, ci sono tre casi possibili
```

2) Script: "cartasassoforbice2.py"

Estendere lo script in modo che il valore della variabile g2 sia immesso da tastiera dall'utente. Controllare che l'input sia una delle tre stringhe ammissibili. Sono valide anche stringhe con lettere maiuscole, ad es. "Carta", "CArTa" ecc..Se l'input non e' ammissibile lo script non deve stampare l'esito del gioco.

Esempi:

input -> output

Carta -> "il gioco e' terminato in pareggio"

CArTa -> "il gioco e' terminato in pareggio"

SASSO -> "il giocatore 2 ha perso"

sassom -> "la scelta non e' ammissibile"

3) Script: cartasassoforbice3.py

Modificare lo script precedente in modo che l'input del giocatore 2 sia determinato casualmente

4) Script: "cartasassoforbice4.py"

Estendere gli script precedenti in modo che l'input del giocatore uno sia immesso da tastiera dall'utente e l'input del giocatore 2 sia scelto casualmente. Notate che adesso si complica la verifica dell'esito della partita

Suggerimento: dati i possibili valori delle variabili, ci sono 9 configurazioni possibili

carta - carta -> il gioco e' terminato in pareggio

carta - sasso -> giocatore 1 vince

carta - forbice -> giocatore 2 vince

sasso - carta -> giocatore 2 vince

sasso - sasso -> il gioco e' terminato in pareggio

sasso - forbice -> giocatore 1 vince

forbice - carta -> giocatore 1 vince

forbice - sasso -> giocatore 2 vince

forbice - forbice -> il gioco e' terminato in pareggio

ma SOLAMENTE TRE output diversi. Cercate di raggruppare le condizioni in modo da sfruttare questa osservazione

5) Script: "cartasassoforbice5.py"

Estendere lo script precedente in modo che se l'input immesso dall'utente non e' corretto, l'utente deve rieffettuare la scelta finche' questa non e' corretta.

6) Script: "cartasassoforbice6.py"

A partire dallo script precedente, si estenda il gioco a numeroround round (con numeroround variabile inizializzata all'interno dello script a 3) stampando per ogni round l'esito. Al termine si stampi il numero di vittorie del giocatore uno, il numero di pareggi ed il numero di volte in cui ha perso.

7) Script: "cartasassoforbice7.py"
A partire dallo script precedente, si apporti la seguente modifica: se l'input dell'utente non e' corretto si passi direttamente al round seguente utilizzando il comando continue, invece di richiedere all'utente di reinserire una stringa finche' questa non e' corretta.

8) Script: "cartasassoforbice8.py"

A partire dallo script precedente, si estenda il gioco in modo che, inizializzata all'interno dello script una variabile x, 0<=x<=1, il giocatore 2 (la cui scelta e' ancora casuale) vinca con probabilita' almeno x. Utilizzate il comando random.random() se avete bisogno di generare un numero casuale tra 0 e 1