Laboratorio 0

Elementi di Informatica e Programmazione

Esercizio 0-1

Progettare il programma `treePrinter.py` che visualizzi sullo schermo un albero di Natale, fatto esattamente in questo modo:

```
    *
   ***
  *****
 *******
*********
    *
    *
```

Soluzione

Possiamo sfruttare le operazioni sulle stringhe per concatenare gli elementi che compongono l'albero di Natale. Dopo aver studiato i cicli, potete anche risolvere l'esercizio usando un ciclo.

```
print(" "*4 + "*")
print(" "*3 + "*"*3)
print(" "*2 + "*"*5)
print(" "*1 + "*"*7)
print("*"*9)
print(" "*4 + "*")
print(" "*4 + "*")
```

```
  *
 ***
```

```
  *****
 *******
*********
    *
    *
```

Esercizio 0-2

Progettare il programmaprintDigits.py che chieda all'utente di fornire un numero intero positivo avente al massimo cinque cifre e visualizzi le singole cifre del numero, una per riga, ordinatamente Ad esempio, se l'utente scrive 14356, il programma deve visualizzare:

```
Le cifre sono:
1
4
3
5
6
```

Se, invece, l'utente scrive 763, il programma deve visualizzare:

```
Le cifre sono:
0
0
7
6
3
```

Suggerimento: ricordare le proprietà della divisione intera: quoziente e resto.

MOLTO IMPORTANTE: (come sempre) individuare un algoritmo che risolva il problema e verificarlo con cura PRIMA di iniziare a scrivere il programma.

Verificare il corretto funzionamento del programma in casi diversi (in particolare, con numeri aventi meno di cinque cifre).

Come si comporta il programma nel caso in cui il dato d'ingresso non rispetti e specifiche, cioè, ad esempio, abbia più di 5 cifre? Fare qualche esperimento e cercare di capire perché il programma si comporta così.

Soluzione

Dividere per `1000` consente di "selezionare" le migliaia del numero. Calcolare il resto della divisione per 10 del risutato consente di selezionare la cifra meno significativa del risutato dell'operazione precedente, perchè siamo in base 10.

```
number = int(input("Inserire un numero intero positivo con al massimo 5 cifre: "))
print("Le cifre sono")
print( number // 10000 )
print( ( number // 1000 ) % 10 )
print( ( number // 100 ) % 10 )
print( ( number // 10 ) % 10 )
print( number % 10 )
```

Esercizio 0-3

Scrivere il programma printEasterDateForYear.py che calcoli la data della domenica di Pasqua dell'anno specificato dall'utente. La domenica di Pasqua è la prima domenica dopo la prima luna piena di primavera e la sua data può essere calcolata con questo algoritmo, individuato da Carl Friedrich Gauss nel 1800.

1. Chiedi all'utente l'anno Yes, un numero intero non negativo.
2. Dividi y per 19, ottenendo il resto a. Ignora il quoziente.
3. Dividi y per 100, ottenendo quoziente b e resto c.
4. Dividi b per 4, ottenendo quoziente d e resto e.
5. Dividi (8b+13) per 25, ottenendo il quoziente g. Ignora il resto.
6. Dividi (19a+b-d-g+15) per 30, ottenendo il resto h. Ignora il quoziente.
7. Dividi c per 4, ottenendo quoziente j e resto k.
8. Dividi (a+11h) per 319, ottenendo il quoziente m. Ignora il resto.
9. Dividi (2e+2j-k-h+m+32) per 7, ottenendo il resto r. Ignora il quoziente.
10. Dividi (h-m+r+90) per 25, ottenendo il quoziente n. Ignora il resto.
11. Dividi (h-m+r+n+19) per 32, ottenendo il resto p. Ignora il quoziente.
12. Pasqua è il giorno p del mese n dell'anno y.

Verificare che nel 2001 il giorno di Pasqua sia stato il 15 aprile, mentre nel 2030 sarà il 21 aprile.

Soluzione

Risolvere l'esercizio richiede di codificare attentamente i passaggi riportati sopra.

```
y = 2030
a = y % 19
b = y // 100
c = y % 100
d = b // 4
e = b % 4
g = (8*b + 13) // 25
h = (19*a + b - d - g + 15) % 30
j = c // 4
k = c % 4
m = (a + 11*h) // 319
r =  (2*e+2*j-k-h+m+32) % 7
n = (h-m+r+90) // 25
p = (h-m+r+n+19) % 32
print("Pasqua è il giorno", p,
      "del mese", n,
      "dell'anno", y)
```

```
Pasqua è il giorno 21 del mese 4 dell'anno 2030
```

Esercizio 0-4

Scrivere il programma `printTimeInterval.py` che chieda all'utente due orari nel formato "24 ore", ciascuno di quattro cifre (ad esempio, 0900 oppure 1730), con il secondo orario successivo al primo visualizzi il numero di ore e di minuti (separatamente) che intercorrono fra i due orari come nell'esempio seguente

```
Inserire il primo orario (es. 0930): 0900
Inserire il secondo orario (successivo al primo): 1730
Tempo trascorso: 8 ore e 30 minuti
```

Verificare il corretto funzionamento del programma in casi diversi.

Come si comporta il programma nel caso in cui i dati d'ingresso non rispettino le specifiche, cioè, ad esempio, il secondo orario sia inferiore al primo? Fare qualche esperimento e cercare di capire perché il programma si comporta così.

Soluzione

```
first = int(input("Inserire il primo orario: "))
second = int(input("Inserires il secondo orario: "))

first_hours = first // 100
first_minutes = first % 100
second_hours = second // 100
second_minutes = second % 100

first = first_hours * 60 + first_minutes
second = second_hours * 60 + second_minutes

duration = second - first
duration_hours = duration // 60
duration_minutes = duration % 60
print("Tempo trascorso", duration_hours, "ore e", duration_minutes, "minuti")
```

Esercizio 0-5

Scrivere il programma `printTimeInterval2.py`, modificando il programma `printTimeInterval.py` visto in precedenza in modo che funzioni correttamente anche se il secondo orario è inferiore al primo (cioè per un intervallo di tempo che comprenda la mezzanotte), come in questo esempio di funzionamento:

```
Inserire il primo orario (es. 0930): 1730
Inserire il secondo orario: 0900
Tempo trascorso: 15 ore e 30 minuti
```

Verificare il corretto funzionamento del programma in casi diversi.

Soluzione

È sufficiente aggiungere un'istruzione condizionale che controlla se il secondo orario è precedente al primo: in questo caso al secondo orario vengono aggiunti i minuti di un giorno intero.

```
first = int(input("Inserire il primo orario: "))
second = int(input("Inserires il secondo orario: "))

first_hours = first // 100
first_minutes = first % 100
```

```
second_hours = second // 100
second_minutes = second % 100

first = first_hours * 60 + first_minutes
second = second_hours * 60 + second_minutes

if second < first:
    # roll over one day
    second = second + 60*24 # minutes in a day

duration = second - first
duration_hours = duration // 60
duration_minutes = duration % 60
print("Tempo trascorso", duration_hours, "ore e", duration_minutes, "minuti")
```