Laboratorio 3

Informatica@DSS 2019/2020 — (E-N)

Massimo Lauria <massimo.lauria@uniroma1.it>*

Lunedì, 14 Ottobre 2019

Lo scopo del laboratorio è di esercitarsi e misurare la propria preparazione: gli esercizi non sono troppo difficili, se si sono seguite le lezioni. Non vi viene comunque messo alcun voto.

Modalità di lavoro: gli studenti devono lavorare in autonomia o in piccoli gruppi, senza disturbare i colleghi. Il lavoro di gruppo è fruttuoso solo se tutti partecipano e se ognuno scrive una propria soluzione per tutti gli esercizi.

Il docente cercherà per quanto possibile di non occupare il tempo del laboratorio per introdurre materiale nuovo, anche se a volte questo sarà necessario. Il docente è a disposizione per aiutare gli studenti, che possono iniziare a lavorare anche prima che il docente arrivi in aula, se lo desiderano

Raccomandazioni leggete bene il testo degli esercizi prima di chiedere chiarimenti. In ogni caso sarò in aula con voi. Alla fine della lezione per favore rispondete al questionario, disponibile al link alla fine di questo documento.

Altri consigli

Attraverso l'uso delle funzioni `input` (alla quale potete passare un argomento opzionale) e `print`, potete rendere un po' più comodi da usare tutti programmi che avete scritto oggi.

Scrivete il codice in maniera modulare: separate la funzionalità logica dal programma quanto possibile. Ad esempio se un programma deve effettuare un calcolo o una manipolazione, questa potrà essere effettuata

*http://massimolauria.net/courses/informatica2019/

da una funzione, ma i dati in input e i messaggi in output possono essere fatti dal programma esterno.

1 Autoapprendimento: assegnamenti multipli

Questo esercizio serve a farvi studiare da soli gli assegnamenti multipli in `python`. Normalmente un assegnamento ha la forma `varname = expr`, ovvero il valore di **un'espressione** è associato ad **una variable** `varname`.

In `python` è possibile effettuare assegnamenti multipli in una singola istruzione. Ad esempio se voglio assegnare a tre variabili x, t, r il valore di tre espressioni `expr1`, `expr2`, `expr3` posso scrivere

```
x, t, r = expr1, expr2, expr3
```

Notate che il numero di nomi a sinistra deve essere **tassativamente** uguale al numero di espressioni sulla destra, altrimenti `python` da errore. Per capire meglio come funzionano gli assegnamenti multipli eseguite il codice seguente. Riuscite a capire l'evoluzione dei valori nelle variabili?

```
import math

x , y = 40, 13    # primo esempio
print(x)
print(y)

# segue il secondo esempio
a, b, c  = math.sin(math.pi/2), 12, math.cos(0.0) + 4
print(a)
print(b)
print(c)
```

Fate attenzione ai prossimi due esempi! Notate differenze nel'output? Se si, perché?

```
v1 = "casa"
v2 = 1001

v2 = v1     # assegnamenti in sequenza
v1 = v2
print(v1,v2)
```

```
u1 = "casa"
u2 = 1001

u1, u2 = u2, u1   # assegnamenti multipli
print(u1,u2)
```

2 Autoapprendimento: return di più valori

L'assegnamento multiplo va a braccetto con il fatto che una funzione possa restituire non solo un singolo valore, ma anche una sequenza di valori. Vediamo il codice che segue:

```
def moduloresto(N,D):

    quoziente = N // D
    resto = N % D

    return quoziente, resto

q, r = moduloresto(28,5)
print(q)
print(r)
```

```
5
3
```

Vedete che una funzione può restituire, ad esempio, due valori e questi possono essere assegnati a due variabili utilizzando un assegnamento multiplo. Osservate che, al contrario degli esempi precedenti, l'assegnamento contiene due nomi di variabili sulla sinistra e una sola espressione sulla destra. Questo non è un problema per `python` perché l'espressione sulla destra è una chiamata a funzione che restituisce due valori.

Una funzione può restituire più valori con l'istruzione

```
return expr1,expr2, ...,exprN
```

ovvero restituisce una sequenza di valori, ottenuti valutando ad una ad una le espressioni nell'istruzione. Espressioni che devono essere separate da una virgola. Ad esempio:

```
def ordina2(A,B):
    if A <= B:
        return A,B
    else:
        return B,A

x,y = ordina2(100,20)
print(x,y)
```

```
20 100
```

Il `return` con valori multipli e gli assegnamenti multipli possono esservi utili nel prossimo esercizio.

3 Conversione in ore, minuti e secondi.

Scrivete un programma che data una quantità arbitraria di secondi, stampi l'equivalente in formato `hh:mm:ss`.

Esempio: se in input vengono dati `11537` secondi, allora il programma dovrà stampare qualcosa di simile a

```
I secondi inseriti corrispondono a 03:12:17
```

Esempio: se in input vengono dati `435797` secondi, allora il programma dovrà stampare qualcosa di simile a

```
I secondi inseriti corrispondono a 121:03:17
```

Esempio: se in input vengono dati `780` secondi, allora il programma dovrà stampare qualcosa di simile a

```
I secondi inseriti corrispondono a 00:13:00
```

Notate che

- le ore, i minuti e i secondi devono essere stampati sempre con almeno due cifre. Ad esempio 9 è `09` e 7 è `07` e 0 è `00`.
- le ore possono utilizzare un numero arbitrario di cifre (ma sempre almeno due).

Suggerimento: vi conviene separare il programma in pezzi.

1. Un pezzo potrebbe essere una funzione che dato un numero di secondi totali restituisce tre numeri corrispondenti a ore, minuti e secondi. (Rileggete gli esercizi 1 e 2 di questo documento per vedere come restituire tre valori con una funzione, e come usarli nel programma chiamante).
2. Un'altra funzione potrebbe gestire la conversione da numero a stringa, con in più l'aggiunta dello zero iniziale, quando questo zero iniziale dovesse servire.
3. Il programma finale potrebbe leggerà l'input, e usare le funzioni scritte sopra per costruire il testo da stampare.

4 Espressione

Quanto vale l'espressione seguente?

```
not -5//2**4 < -1 and 3 ** 2 ** (5 + - 3) >= 2*4                    1
```

Per scoprirlo può essere utile

1. capire quali siano le precedenze degli operatori;
2. costruire l'albero che indichi la struttura della valutazione dell'espressione, come abbiamo visto in aula;
3. calcolare il valore che corrisponda ad ogni sottoespressione dal basso verso l'altro.

5 Equazione di secondo grado

Scrivete una funzione `eqsecondogrado(A,B,C)` che calcoli e restituisca la soluzione **più grande**, tra quelle dell'equazione di secondo grado $Ax^2 + Bx + C = 0$.

Se l'equazione non ha soluzione allora la funzione deve restituire la stringa `'nessuna soluzione'`. Altrimenti deve restituire il valore numerico.

Una volta scritta, e testata abbondantemente, la funzione `eqsecondogrado`, scrivete un programma che legga i valori A, B, C da tastiera, e poi usi la funzione per calcolare il risultato e stamparlo a video.

Suggerimento: ricordate che se A è zero dovrete trattare l'equazione come se fosse di primo grado. Ricordate che potete utilizzare `math.sqrt` per calcolare le radici quadrate, dopo aver inserito `import math` all'inizio del programma.

Questionario: rispondete **dopo** aver fatto gli esercizi.

http://bit.ly/INFO2019-LAB03