Esercizi in Laboratorio

Informatica@SEFA 2017/2018 - Laboratorio 3

Massimo Lauria <massimo.lauria@uniroma1.it>
http://massimolauria.net/courses/infosefa2017/

Lunedì, 16 Ottobre 2017

Formattazione delle stringhe

Metodo `format`

Ogni stringa ha un **metodo** per la formattazione di dati.

```
print( "Ho {} mele e {} pere. {}".format(10,5,"Evviva!") )     1
print( "Ho {0} mele e {1} pere. {2}".format(10,5,"Evviva!") ) 2
print( "{2} Ho {1} pere e {0} mele.".format(10,5,"Evviva!") ) 3
print( "{2} Ho {1} pere e {0} mele. Sì ho detto {0} mele.".   4
    format(10,5,"Evviva!") )
```

```
Ho 10 mele e 5 pere. Evviva!
Ho 10 mele e 5 pere. Evviva!
Evviva! Ho 5 pere e 10 mele.
Evviva! Ho 5 pere e 10 mele. Sì ho detto 10 mele.
```

Tantissimi tipi di formattazione

Al segnatosto `{0}` si possono aggiungere dei modificatori per la formattazione dopo i due punti.

```
print( "{0} è {0:b} in binario e {0:x} in esadecimale".format 1
    (12132) )
```

```
12132 è 10111101100100 in binario e 2f64 in esadecimale
```

Allineamento

Si può forzare python ad usare una certa quantità di spazio per inserire una stringa di testo ed ad allineare a destra o a sinistra.

```
print( "| {: <30} |".format("ciao") )   1
print( "| {: >30} |".format("ciao") )   2
print( "| {:*<30} |".format("ciao") )   3
print( "| {:*>30} |".format("ciao") )   4
```

```
| ciao                           |
|                           ciao |
| ciao************************** |
| **************************ciao |
```

Documentazione di `str.format`

Imparate a leggere documentazione

- anche se non capite tutto
- cercate di trovare le informazioni
- fate prove ed esperimenti

`https://docs.python.org/3/library/string.html#format-string-syntax`

Esercizi

Liste vs tuple vs sequenze

Tutte le soluzioni agli esercizi che chiedono una lista in input devono funzionare anche su tuple (o su qualunque sequenza immutabile).

Esercizio 11

Scrivere una funzione

```
max_mod(lista,base)
```

Che data una lista o una sequenza restituisca il massimo tra gli elementi nelle posizioni `0`, `base`, `2*base`, ecc…

Struttura di una tabella

codice	esercizio	percentuale voto
A1001	calcolo dello sconto	10
A1002	formattazione strighe	20
A1003	ghms2	20
A1004	operazioni su tabelle	15
A1005	interprete assembler	35

Una tabella di 3 colonne può essere realizzata attraverso un dizionario on 3 chiavi (e.g. `codice`, `esercizio`, `percentuale voto`). Dove ad ogni chiave è associata una lista di valori.

Nota: le liste devono avere la stessa lunghezza.

Esempio

```
tabella={}                                                          1
tabella['codice']=["A1001"   ,"A1002", "A1003" ,"A1004", "          2
    A1005"]
tabella['esercizio']=[                                              3
    "calcolo dello sconto",                                         4
    "formattazione strighe",                                        5
    "ghms2",                                                        6
    "operazioni su tabelle",                                        7
    "interprete assembler"]                                         8
tabella['percentuale voto']=[10,20,20,15,35]                        9
print(tabella)                                                      10
```

```
{'codice': ['A1001', 'A1002', 'A1003', 'A1004', 'A1005'],
'esercizio': ['calcolo dello sconto', 'formattazione strighe',
              'ghms2', 'operazioni su tabelle', 'interprete assembler'],
'percentuale voto': [10, 20, 20, 15, 35] }
```

Esercizio 12

Formattazione delle righe di una tabella

```
formatta_riga(tabella,indice_riga,vista)
```

- la tabella è data nel formato descritto in precedenza
- `indice_riga` quale riga deve essere stampata (0 è la prima riga)
- `vista` è una sequenza di triple dove per ogni tripla
  - nome di una colonna da stampare
  - l'ampiezza della colonna
  - `'l'` o `'r'` per indicare allineamento a sx o dx

Possono esserci ripetizioni tra le colonne e alcune colonne possono essere omesse

Esercizio 12 (II)

- separatori tra le colonne
- uno spazio tra separatore e il testo della colonna

Esempio:

```
formatta_riga(tabella,2,
              [('codice',6,'l'),
               ('esercizio',20,'l'),
               ('percentuale voto',6,'r')
               ('percentuale voto',5,'r')])
```

restituisce la striga

```
"| A1003  | ghms2                |     20 | 20    |"
```

Esercizio 13

Formattate un'intera tabella

```
formatta_tabella(tabella,colonne)
```

- una riga contenente i nomi delle colonne
- una riga separatrice (con dei + agli incroci)
- allineamento a sinistra per i campi alfanumerici
- a destra per quelli **numerici**
- e se non ci sono righe?
- calcolate voi lo spazio necessario

```
| codice | esercizio              | percentuale voto |
|--------+------------------------+------------------|
| A1001  | calcolo dello sconto   |               10 |
| A1002  | formattazione strighe  |               20 |
| A1003  | ghms2                  |               20 |
| A1004  | operazioni su tabelle  |               15 |
| A1005  | interprete assembler   |               35 |
```

Soluzione Esercizi `lab02`

Esercizio 8

Migliorare tutte le funzioni degli esercizi 1–7 con il controllo dei parametri.

```
def volume_parallelepipedo_rettangolo(altezza,larghezza,      1
    profondità):
    if altezza<0 or larghezza<0 or profondità<0:              2
        raise ValueError("le dimensioni devono essere non     3
    negative")
    return altezza*larghezza*profondità                       4
```

Esercizio 9 (I)

Costruire una funzione

```
ghms2(secondi)
```

simile a quella di `lab01`, ma che produca stringhe più sensate. Ad esempio.

input	output
0	`0 secondi.`
2348	`39 minuti e 8 secondi.`
3840	`1 ora e 4 minuti.`
122456	`1 giorno, 10 ore e 56 secondi.`

- attenzione ai plurali e singolari.
- attenzione alla punteggiatura e all'uso di 'e'
- controllare la correttezza degli input
- fate un bel respiro e aiutatevi con il file di test

Esercizio 9 (II)

```
def ghms2(secondi):                                                    1
    if secondi<0:                                                      2
        raise ValueError("mi aspetto un input non negativo")           3
    if secondi == 0:                                                   4
        return "0 secondi."                                            5
                                                                       6
    valori =[0,0,0,0]                                                  7
                                                                       8
    valori[0] = secondi // (24*60*60)                                  9
    secondi   %= 24*60*60                                              10
                                                                       11
    valori[1] = secondi // (60*60)                                     12
    secondi %= 60*60                                                   13
                                                                       14
    valori[2] = secondi // 60                                          15
    valori[3] %= 60                                                    16
```

Esercizio 9 (III)

```
# continua la definizione di 'ghms2'
# 'valori' contiene le informazioni [gg,hh,mm,ss]
    singolare=['giorno', 'ora', 'minuto', 'secondo']
    plurale  =['giorni', 'ore', 'minuti', 'secondi']
    finale=[]

    for i in len(valori):
        if valori[i] == 1:
            finale.append( "1 " + singolare[i] )
        elif valori[i] > 1:
            finale.append( str(giorni) + " " + plurale[i])

    if len(finale)==1:
        return finale[0] + '.'
    else:
        return ", ".join(finale[:-1]) + " e " + finale[-1] + "
    ."
```

Esercizio 10

Verificare se una lista è ordinata in modo crescente

- solleva `ValueError` se nella lista ci sono sia numeri che stringhe
- restituisce `True` se sono ordinati
- restituisce `False` se non sono ordinati

```
def ordinati(lista):                                                    1
    if len(lista)<2:                                                    2
        return True                                                     3
    first = lista[0]                                                    4
    for second in lista[1:]:                                            5
        if (type(first)==str) ^ (type(second)==str):                    6
            raise ValueError("Lista di tipo non omogeneo")              7
        elif first > second:                                            8
            return False                                                9
        else:                                                           10
            first = second                                              11
    return True                                                         12
```