Python 303

"giuliof"

GOLEM

3 luglio 2018

Cosa abbiamo fatto la volta scorsa?

- Le funzioni;
- Cicli for e while;
- Operazioni sulle stringhe;

Link alla lezione precedente:
`https://golem.linux.it/wiki/Ore_del_GOLEM#19_Giugno:_Python_101_x2`

Le liste (o array)

```
alcuni_numeri = [8,15,7,18,65]
print alcuni_numeri

media = 0
conta = 0
for i in alcuni_numeri:
  media += i
  conta += 1
media = media / conta
print media

parole = ["abc", "123", "qwerty"]
print parole
parole += ["ins"]
print parole
print parole[3]   # Richiama direttamente un elemento
```

Le liste (o array), qualche funzione interessante

```
rubrica = ["tizio", "caio", "sempronio", "pinco", "pallino", "caio"]

cerca = raw_input();

if cerca in rubrica:
  print "Trovato!"
  numero_occorrenze = rubrica.count(cerca)
  if numero_occorrenze > 1:
    print "Inoltre, ti posso dire che è ripetuto", numero_occorrenze,"volte"
else:
  print "Non c'e'"
```

- `in` e `not in` permettono anche di verificare se un elemento è presente all'interno di una lista!
- `.count()` permette, come per le stringhe, di contare le occorrenze

Funzioni per ogni variabile

```
>>> a = 5
>>> a = "pippo"
>>> dir(a)
[...]
'capitalize', 'center', 'count', 'decode', 'encode', 'endswith',
'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha',
'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join',
'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind',
'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title',
'translate', 'upper', 'zfill'
```

`dir` ci dà tutte le funzioni associate ad un determinato tipo di variabile (o, più in generale, qualsiasi oggetto)

Esercizi

- Leggere da tastiera delle parole e controllare ogni volta se fanno parte di una vostra lista a piacere. Occhio alle maiuscole!
- Leggere da tastiera una lista di 10 parole e poi scrivere se e quante volte sono state ripetute

I dizionari

```
dizionario = {
"cane" : "mammifero␣a␣4␣zampe␣bla␣bla",
"squali" : "pesci␣noti␣per␣essere␣squelli"
}

minime_gennaio = {
'roma' : 4,
'firenze' : 5,
'torino' : 40
}
```

In un dizionario possiamo inserire dati accoppiati da una *chiave*

I dizionari

```
dizionario = {
"cane" : "mammifero␣a␣4␣zampe␣bla␣bla",
"squali" : "pesci␣noti␣per␣essere␣squelli"
}

minime_gennaio = {
'roma' : 4,
'firenze' : 5,
'torino' : 40
}

# leggo il dizionario
print dizionario['cane']
print "A␣Firenze␣la␣minima␣era" , minime_gennaio['firenze']
```

In un dizionario possiamo inserire dati accoppiati da una *chiave*

Operazioni sui dizionari

```
# scrivo o aggiungo cose al dizionario
minime_gennaio['torino'] = 0
dizionario['GOLEM'] = 'Gruppo␣Operativo␣Linux␣EMpoli'

# Sfoglio l'intero dizionario
for indice in dizionario:
  print "La␣voce",indice,"contiene",dizionario[indice]

# Svuoto una voce (ci metto una stringa vuota)
dizionario['cane'] = ''
# Elimino una voce
del minime_gennaio["torino"]
```

```
La voce GOLEM contiene Gruppo Operativo Linux EMpoli
La voce cane contiene mammifero a 4 zampe bla bla
La voce squali contiene pesci noti per essere squelli
```

Dizionari nidificati

```
miei_film = {
  'Inception' : {
    'genere' : 'Thriller',
    'anno'   : 2010,
    'voto'   : 4.2
  },
  'Matrix' : {
    'genere' : 'Fantascienza',
    'anno'   : 1999,
    'voto'   : 3.5
  }
}
```

È possibile annidare più dizionari, ma anche combinare liste e dizionari insieme!

Dizionari nidificati

```
for film in miei_film:
  print film,"e' uscito nel",miei_film[film]['anno']
```

```
Inception e' uscito nel 2010
Matrix e' uscito nel 1999
```

Leggere e scrivere i file

- Python permette di leggere e scrivere file di testo.
- Prima di poter fare operazioni su un file bisogna aprirlo: `open('nomefile.txt', ...)`.
- Per concludere le modifiche il file deve poi essere chiuso: `close()`

Scriviamo qualcosa

```
f = open('pippo.txt','w')
f.write('Verba␣volant,\n')
f.write('scripta␣manent')
f.close()
```

- Specificando `'w'` alla funzione `open` possiamo scrivere su un file.
  Achtung: tutto il contenuto del file sarà eliminato! Per continuare a scrivere su un file già esistente utilizzare `'a'` (append).
  `open` da' indietro una variabile *maniglia* (handler) che permette di fare operazioni sul file;
- La funzione `write` permette di scrivere variabili o stringhe nel file. Non aggiunge il ritorno a capo, che va inserito manualmente con `\n`.

Rileggiamo qualcosa

```
f = open('pippo.txt','r')
dati = f.read()
f.close()
print dati
```

- Specificando `'r'` alla funzione `open` possiamo leggere su un file;
- La funzione `read` legge l'intero file e lo inserisce in una variabile.

Rileggiamo qualcosa

```
f = open('pippo.txt','r')
linea = f.readline()
while len(linea)>0:
  print linea
  linea = f.readline()
f.close()
```

È possibile leggere anche una sola riga per volta

Esercizi

Scrivere un programma che...

- Stampa le righe di un file di testo aggiungendo un numero davanti ad ogni riga
- Stampa le righe di un file di testo solo se iniziano con #
- Salva le righe di un file di testo su un altro file di testo, trasformandole tutte in maiuscole
- Ci dice se un file di testo contiene una parola e a quale riga

Un accenno alle librerie

Calcoliamo la radice quadrata di 9 (*square root* in inglese)

```
>>> sqrt(9)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'sqrt' is not defined
```

La funzione radice quadrata non esiste, dobbiamo aggiungere una libreria!

Un accenno alle librerie

```
import math
num = input("Inserisci un numero naturale ")
print math.sqrt(num)
```

Con `import` si richiama una specifica libreria per il programma. Le nuove funzioni richiedono il prefisso `nome_libreria.`

Varianti di import

- Richiamare le funzioni in modo breve. Utile quando il nome della libreria è lungo

```
import math as mt
print mt.sin(mp.pi)
```

- Incorporare solo alcune funzioni scelte, senza bisogno di citare la libreria

```
from math import exp
print exp(2)
```

- Incorporare tutte le funzioni senza dover citare la libreria

```
from math import *
print log10( sin(pi/2)/3.7 )
```

Ancora sui numeri...

- Quanto fa `sqrt(-1)`?
- Python gestisce nativamente i numeri complessi!
- Per poter fare operazioni più "complesse" è consigliata la libreria cmath (matematica complessa).

Ancora sui numeri...

- Quanto fa `sqrt(-1)`?
- Python gestisce nativamente i numeri complessi!
- Per poter fare operazioni più "complesse" è consigliata la libreria cmath (matematica complessa).

Ancora sui numeri...

- Quanto fa `sqrt(-1)`?
- Python gestisce nativamente i numeri complessi!
- Per poter fare operazioni più "complesse" è consigliata la libreria cmath (matematica complessa).

Python2 vs Python3

- `print "Hello World"`
- `raw_input()` legge le stringhe
- `input()` legge i numeri
- `1/2` è una divisione intera. `1.0 // 2.0` forza una divisione intera con i *float*

- `print ("Hello World")`
- `input()` legge le stringhe
- `int(input())` legge i numeri
- `1/2` è una divisione con virgola. La divisione intera si fa con `1 // 2` e funziona anche con i *float*

Python2 vs Python3

- `print "Hello World"`
- `raw_input()` legge le stringhe
- `input()` legge i numeri
- `1/2` è una divisione intera. `1.0 // 2.0` forza una divisione intera con i *float*

- `print ("Hello World")`
- `input()` legge le stringhe
- `int(input())` legge i numeri
- `1/2` è una divisione con virgola. La divisione intera si fa con `1 // 2` e funziona anche con i *float*

Python2 vs Python3

- `print "Hello World"`
- `raw_input()` legge le stringhe
- `input()` legge i numeri
- `1/2` è una divisione intera. `1.0 // 2.0` forza una divisione intera con i *float*

- `print ("Hello World")`
- `input()` legge le stringhe
- `int(input())` legge i numeri
- `1/2` è una divisione con virgola. La divisione intera si fa con `1 // 2` e funziona anche con i *float*

Python2 vs Python3

- `print "Hello World"`
- `raw_input()` legge le stringhe
- `input()` legge i numeri
- `1/2` è una divisione intera. `1.0 // 2.0` forza una divisione intera con i *float*

- `print ("Hello World")`
- `input()` legge le stringhe
- `int(input())` legge i numeri
- `1/2` è una divisione con virgola. La divisione intera si fa con `1 // 2` e funziona anche con i *float*

Link utili

- Note sulla migrazione a Python 3
  `http://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html`
- Installazione di Python su Windows e altri trucchetti interessanti
  `https://gist.github.com/mallegrini/`
- Quick reference python3 `https://perso.limsi.fr/pointal/_media/python:cours:mementopython3-english.pdf`

Grazie per l'attenzione!

GOLEM - Gruppo Operativo Linux Empoli

GOLEM – Gruppo Operativo Linux Empoli
presso "La Vela – Margherita Hack"
via Magolo, 32 – 50053 Empoli (FI)
tutti i martedì sera dalle 21.30 alle 24.00

Crediti & licenza

Questa serata è offerta da giuliof (`giulio@glgprograms.it`)
a partire dal materiale di Fiore (FabLab Toscana) e con contributi di lucam
utilizzando un template LaTeX a cura di giomba.
Materiale rilasciato sotto GPL3 presso `golem.linux.it`