Informatica

Corso di Laurea in Scienze Geologiche

Terza Parte

Ugo Dal Lago

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

informatiques mathématiques
Inria

Anno Accademico 2015-2016

Sezione 1

`Python`: e il Resto?

C'è Altro?

- Il frammento di `Python` che abbiamo studiato finora è certamente **sufficiente** allo sviluppo di piccoli programmi.
- Nondimeno, tanti altri costrutti e istruzioni sono disponibili, alcuni dei quali risultano utili quando la dimensione dei progetti diventa **non banale**.
- Non abbiamo tempo per parlarne, ma vale la pena dare una scorsa, molto veloce, ad *alcuni* dei costrutti che non abbiamo visto.
- Maggiori informazioni su questi argomenti possono essere recuperate:
  - Sul **libro di testo**;
  - In **rete**.

Dizionari

- I **dizionari** sono generalizzazioni delle liste nei quali gli indici non sono necessariamente numeri naturali, ma oggetti modificabili qualunque.
- **Esempio**:

```
>>> x={}
>>> x['a']=3
>>> x
{'a': 3}
>>> x[True]=6
>>> x
{'a': 3, True: 6}
>>> x.keys()
['a', True]
>>> x.values()
[3, 6]
```

Classi

- In `Python` è addirittura possibile creare **nuovi tipi di dato**, attraverso il comando `class`.
- In questo modo è possibile definire tipi di dato che corrispondano direttamente alle entità del problema che si vogliono modellare.
- I metodi che è possibile chiamare sugli oggetti di una certa classe parte della definizione stessa.
- **Esempio**: se stessimo scrivendo un programma per la gestione di dati di tipo geografico, potremmo optare per una classe `island` dotata di metodi come `extension()`, `name()`, etc.
- L'uso delle classi non è necessario se il programma `Python` che si sta scrivendo è semplice.
  - Tutto questo vale, ad esempio, per tutti i programmi che abbiamo scritto e che scriveremo in questo corso.

Sezione 2

Sviluppo di Applicazioni

Commentare il Codice — I

- Commentare il codice `Python` lo rende più comprensibile.
- In particolare, è spesso utile usare i commenti per:
  - Descrivere il ruolo delle variabili;
  - Spiegare quale sia il tipo atteso di un parametro formale in una funzione.
- I commenti si possono inserire in *due modi diversi*:
  - Se occupano **una sola riga**, si possono inserire tramite il simbolo `#`. Tutto ciò che segue tale carattere nella riga corrente è considerato commento, e quindi scartato.
  - I commenti che occupano **più di una riga**, sono invece delimitati da tre virgolette.

Commentare il Codice — II

```
def scalmult(v,c):
  i=0
  while i<len(v):
    v[i]=v[i]*c
```

```
def scalmult(v,c):
""" Assumiamo che v sia
lista e che c sia float """
  i=0 # Contatore
  while i<len(v):
    v[i]=v[i]*c
  # Qui non c'è bisogno
  # del return
```

Testare il Codice

- Una volta finito di scrivere il nostro programma `Python`, il nostro lavoro non è certo finito.
- Prima di mettere all'opera il programma per fare i nostri calcoli, è necessario **testarlo**, in modo da scoprire eventuali problemi prima che sia troppo tardi.
- Si può per esempio procedere testando tutte le funzioni che il nostro programma definisce.
- Un testing *esaustivo* di ciascuna funzione è impossibile.
  - Ciò equivale ad osservare il comportamento della funzione su **ogni input possibile**.
- Occorre quindi costruire una cosiddetta *testing suite*, ovvero un insieme di input limitato, ma rappresentativo.
- Il libro di testo contiene un capitolo dedicato all'argomento.

Sezione 3

Alcuni Moduli

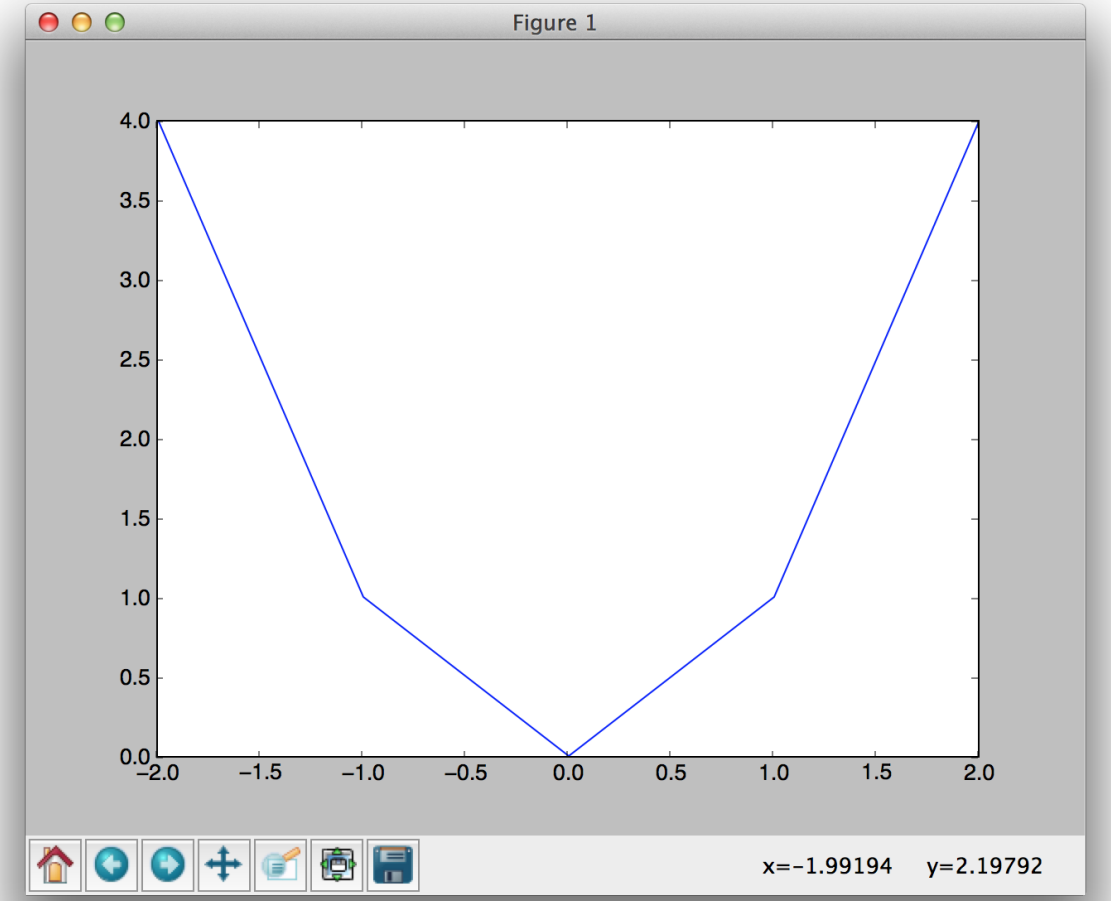
Il Modulo pylab — I

- Il modulo pylab permette di disegnare dei grafici.
- Una descrizione abbastanza intuitiva (ma dettagliata!) del modulo si può trovare nel libro di testo.
- **Esempio**:

```
import pylab
pylab.figure(1)
pylab.plot([-2,-1,0,1,2],[4,1,0,1,4])
pylab.show()
```

  - Il risultato è quello alla pagina successiva.

Il Modulo pylab — II



Il Modulo `random`

- L'esecuzione di un certo programma `Python` fissato, se le condizioni a contorno sono le stesse, è sempre la stessa.
- Talvolta, però, risulta necessario generare **dati casuali**.
  - Ogniqualvolta, ad esempio, occorra simulare fenomeni fisici tramite il calcolatore.
- Questo è precisamente che permette di fare il modulo `random`.
- **Esempio**:

```
>>> import random
>>> random.choice([True,False])
False
>>> random.choice([1,2,3,4,5,6])
3
>>> random.choice([1,2,3,4,5,6])
1
```

Il Modulo `urllib2`

- Talvolta risulta utile poter aprire dei file che non risiedano nel nostro elaboratore, ma sulla rete.
- Per questo specifico scopo, il modulo `urllib2` mette a disposizione specifiche funzionalità.
- Il file remoto verrà visto come un file handler, e potremo accedere ad esso solo in lettura.
  - Oltre alla possibilità di vedere il file handler come una sequenza e usare `for`, possiamo ora usare il metodo `readlines`.
- **Esempio**:

```
import urllib2
s='http://www.cs.unibo.it/~dallago'
filehandle = urllib2.urlopen(s)
x=filehandle.readlines()
```

Tutto qui?

- Certamente no!
- La quantità di moduli disponibili per il linguaggio `Python` è **enorme**.
- Esempi:
  - Calcolo numerico: `numpy`
  - Accesso a database.
  - GIS: `geopy`
  - ...
- Un buon modo per rendersi conto se esiste un modulo che faccia esattamente ciò di cui ho bisogno è documentarsi:
  - Sul **libro di testo**.
  - Sulla **rete**.