Prontuario di Python per il laboratorio di calcolo

Introduzione

Python è un linguaggio di programmazione *interpretato*. Significa che il codice sorgente che scriviamo viene passato ad un altro programma, detto *interprete*, che lo esegue.

In generale, Python è un linguaggio di più alto livello rispetto a C. Il concetto di "linguaggio di programmazione di alto livello" non è rigoroso, ed ha senso solo in un contesto relativo: il linguaggio X è di livello più alto rispetto a Y significa che con X è più difficile (o, in alcuni casi, impossibile) interagire con le primitive del sistema operativo e/o dell'hardware rispetto ad Y. Nel caso specifico di Python *vs.* C, le due principali differenze (per quanto concerne questo corso) sono:

1. In Python non esistono i puntatori
2. La memoria in cui risiedono le variabili che utilizziamo è gestita (*managed*) direttamente da Python (in maniera cosiddetta trasparente). Non esiste quindi una distinzione tra *stack* e *heap*, né funzioni per allocare/deallocare dinamicamente la memoria (cioè funzioni equivalenti a `malloc` e `free` in C).

Nel corso utilizzeremo Python principalmente come uno strumento per creare grafici (più o meno interattivi) e per semplici analisi dati.

Eseguire un programma scritto in Python

I programmi scritti in Python (spesso definiti *script*) devono essere passati ad un interprete per essere eseguiti. Sui sistemi che utilizzate questo si riduce a passare lo script come primo argomento del programma `python3`. Ad esempio, se il nostro script si chiama `prova.py`, per lanciarlo dovremo eseguire il seguente programma:

```
$ python3 prova.py
```

L'importanza di indentare correttamente

Sarete sorpresi di scoprire che i codici Python non possono essere indentati come vi pare. In Python, infatti, il cosiddetto *whitespace* (spazi e tablature) è parte integrante del codice. In Python, infatti, i blocchi di codice (che in C sono delimitati dalle parentesi graffe) devono essere *indentati*. In effetti, non basta che istruzioni appartenenti allo stesso blocco siano indentate, ma devono anche essere indentate allo stesso modo, cioè con la stessa quantità di *whitespace*: se utilizzate due spazi per la prima riga, dovrete usare due spazi anche per le seguenti.

Ecco un esempio in C:

```c
int a = 1;
if(a == 1) {
    printf("a è diverso da zero!\n");
    a = 0;
}
```

Che in Python diventa:

```python
a = 1
if a == 1:
    print("a è diverso da zero!")
    a = 0
```

Mentre l'indentazione in C è opzionale (ma raccomandata: migliora la leggibilità del codice), in Python è obbligatoria: provate a toglierla e vedrete che l'interprete se ne lamenterà!

Nella maggior parte degli script che vedremo nelle esercitazioni non ci sono blocchi di codice oltre a quello principale, e quindi non avremo la necessità di indentare alcunché (con alcune eccezioni).

Il comando `import`

All'inizio di quasi ogni script Python si trovano alcune linee del tipo

```
import os, sys
import numpy as np
```

Il comando `import` serve per *importare* delle librerie nel nostro script in modo da poterle utilizzare. Se vogliamo importare più librerie è sufficiente scrivere i nomi di tutte le librerie uno dopo l'altro, separandoli con delle virgole.

In generale funzioni, variabili o classi di una libreria possono essere utilizzate subito dopo che questa è stata importata utilizzando la sintassi `mylib.mystuff`, dove `mylib` è la libreria importata e `mystuff` il nome dell'oggetto cui si vuole accedere. Ad esempio, dopo la prima riga dell'esempio sopra è possibile accedere alla versione di Python che stiamo utilizzando con il comando `sys.version_info`.

Alcune volte, soprattutto nel caso di librerie a cui si accede spesso, può far comodo importare una libreria dandole un altro nome. Per far ciò si utilizza la particella `as`. Nell'esempio sopra si accederà alle funzioni, variabili e classi della libreria `numpy`utilizzando l'alias `np` (ad esempio, `np.zeros(100)` invece di `numpy.zeros(100)`).