Il mondo di Python (autore: Vittorio Albertoni)

Premessa

Il luogo di massima concentrazione di informazioni scritte in bella lingua italiana su Python è il sito

www.python.it

Per imparare il linguaggio, oltre a quanto ci offre il sito di Python, su Internet si trovano innumerevoli manuali e manualetti in formato PDF.

Per chi preferisca una guida cartacea che in poco spazio contenga quanto necessario, sempre per imparare il linguaggio, suggerisco:

- per acquisire le basi: il tascabile di Marco Beri, Python 3, edito da Apogeo;
- per acquisire anche qualche cosa in più: il testo di Kenneth A. Lambert, Programmazione in Python, edito da Maggioli per Apogeo education.

A parte queste produzioni a pagamento, come mai tanta documentazione di libero accesso su Internet?

La risposta l'abbiamo dalla home page del sito di Python, dove leggiamo il titolo "Python - Sito ufficiale della comunità italiana" e l'affermazione che Python è distribuito con licenza Open-Source: siamo cioè in presenza di software libero e, come succede in questo mondo, si creano comunità di persone che contribuiscono liberamente e gratuitamente al perfezionamento e al diffondersi di conoscenza dei progetti.

Nel caso di Python, almeno per l'evoluzione e l'arricchimento del linguaggio di programmazione, tutto avviene comunque sotto l'occhio vigile del creatore del linguaggio, Guido van Rossum, amabilmente chiamato, nella comunità, il benevolo Dittatore a vita.

In questa piccola guida non ripeterò cose che, meglio di come possa fare io, sono già state scritte e facilmente reperibili. Vorrei solo presentare alcuni strumenti per entrare e muoversi nel mondo che si è creato attorno a Python, strumenti che nei manuali per apprendere il linguaggio sono un po' fuori tema e le informazioni sui quali non sono facilmente e organicamente reperibili in lingua italiana.

Indice

1 Quale Python

Sono tuttora distribuite due versioni di Python, la 2 e la 3.

La versione 2, giunta alla 2.7.13, è la versione del passato ed è stata soppiantata dalla versione 3, giunta alla 3.6.

Purtroppo tra le due esistono differenze tali che uno script creato in una versione non viene eseguito nell'altra, per cui, per utilizzare programmi scritti per la versione 2, occorre disporre di Python 2, salvo apportare le necessarie modifiche per poterli utilizzare con l'interprete Python 3.

Per chi ha la fortuna di utilizzare il sistema operativo Linux, o il suo cugino di discendenza Unix Mac OS X, vi sono modi per gestire senza confusioni entrambe le versioni, in quanto gli eseguibili degli interpreti hanno nomi diversi e c'è modo di indicare l'eseguibile nello script[1]. Per chi usa il sistema operativo Windows, purtroppo, l'interprete si chiama python.exe in entrambi i casi e la differenza la fa la directory in cui si trova l'interprete stesso (Python2 o Python3), con tutte le complicazioni che derivano dalla gestione dei path nelle variabili d'ambiente.

Bello, soprattutto per chi usa Windows, sarebbe avere la sola versione 3, quella del futuro, su cui far girare programmi scritti per la versione 2 opportunamente adattati. Chiunque può provvedere all'adattamento tenendo presenti le differenze che riepilogo nel paragrafo che segue.

1.1 Da Python 2 a Python 3

Importazione di moduli

- Il modulo `tkinter` si importa con `from tkinter import *`, con la t minuscola (nella versione 2 Tkinter è scritto con la T maiuscola),
- Il modulo `sqlite3`, per la verità fin dalla versione 2.6, ha sostituito il vecchio modulo `PySQlite` e si importa con `import sqlite3`; la connessione si stabilisce poi con `mioDatabase = sqlite3.Connection(<''path e nome del database''>)`.

Tipi di dati predefiniti

- il tipo numerico `long` è unificato nel tipo `int`: nella versione 3 tutti gli interi sono `long`;
- il tipo `unicode` è unificato nel tipo `str`; nella versione 3 tutte le stringhe sono `unicode` e l'operatore `u` per renderle tali non c'è più;
- il tipo `basestring` è unificato nel tipo `str`;
- il tipo `xrange` è abolito e `xrange()` è rinominato e unificato in `range()`;
- `types.InstanceType` è abolito;
- `types.UnboundedMethodType` è abolito.

In generale

- il comando `print` della versione 2 diventa la funzione `print()` nella versione 3, sicché non si fa più `print 'ciao'` ma si fa `print('ciao');`
- per scrivere in un file non si usa più il comando `print>><nome_file>,...` ma si usa il metodo write dell'oggetto file: `<nome_file>.write();`
- il risultato della divisione con l'operatore `/` è sempre un float, anche se la divisione è fatta tra interi. Se serve che il risultato sia un intero occorre renderlo tale con il casting `int();`

[1] Ciò può avvenire creando nella prima riga dello script la così detta shebang line, cioè scrivendo all'inizio dello script i caratteri #! seguiti dall'indirizzo dell'interprete. Per esempio, volendo utilizzare l'interprete Python3, che si trova nella directory /usr, sottodirectory /bin, si scrive #! /usr/bin/python3 nella prima riga. Nel sistema operativo Windows questa riga non viene letta.

- la funzione `input()` incamera una stringa; per incamerare un numero, nella versione 3 occorre scrivere `eval(input())`;
- la funzione `raw_input()` è abolita e unificata nella funzione `input()`;
- la funzione `cmp()` è abolita e va sostituita utilizzando i confronti elementari `(x>y)`, `(x<y)`. Negli ordinamenti si devono usare le chiavi.

2 Il pacchetto base e la sua installazione

Sistema Linux

Il pacchetto base di Python è da sempre preinstallato sui sistemi operativi Linux.

La versione installata è l'ultima disponibile al momento dell'uscita della distro: in Debian e famiglia (Ubuntu e derivate, Mint) sono in genere preinstallate entrambe le versioni 2 e 3. In SuSE mi pare, nel momento in cui scrivo, sia installata solo la 2, come pure in Red Hat e Fedora.

Se la o le versioni installate non ci aggradano più con il passare del tempo e non vogliamo aggiornare il sistema operativo, possiamo procurarci il source del pacchetto che vogliamo installare dal sito *www.python.it* e installarlo, dopo averlo decompresso, con il solito procedimento

```
./configure
make
make test
sudo make install.
```

La dotazione di base Python che troviamo in Linux ha tutto ciò che serve per creare programmi anche complessi (nel caso di Python sarebbe più proprio chiamarli script, in quanto si tratta di istruzioni che vengono interpretate ogni volta per essere eseguite). Per quanto riguarda il calcolo, se non bastano i normali e più ricorrenti operatori per addizione, sottrazione, moltiplicazione, divisione ed elevamento a potenza, la dotazione di base comprende un modulo, chiamato **math**, che ci offre tutta una serie di funzioni precostituite.

In Python si chiama modulo una raccolta di funzioni riunite in una libreria.

Per utilizzare queste funzioni occorre importare nello script il modulo che le contiene. Per sapere quali sono le funzioni contenute in un modulo, dopo averlo importato, digitiamo la funzione `dir()` passandole come parametro il nome del modulo. Nel caso del modulo math, lavorando nella shell di Python:

```
>>> import math
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh',
'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor',
'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite',
'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf',
'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
>>>
```

Sistema Mac OS X

Dalla versione 10.2 vi è installata la versione Python 2: nel momento in cui scrivo la 2.7.13 compatibile con Mac OS X 10.5 e successive.

Per installare la versione 3 occorre procurarsi l'installer nella sezione *Download* del sito *www.python.it*: l'ultima versione Python 3.6, qui unica disponibile, richiede almeno la versione 10.6 di Mac OS X[2].

[2]Per trovare versioni precedenti alle ultime uscite che ci offre *www.python.it* occorre andare nella sezione *Download*s su *https://www.python.org*

La dotazione di base di Python che troviamo sul Mac, oltre, ovviamente, al modulo **math**, ci offre anche i moduli **NumPy** e **SciPy**, che arricchiscono enormemente la dotazione degli strumenti di calcolo e ai quali accennerò anche in seguito.

Sistema Windows

Windows non preinstalla Python.

Possiamo procurarci l'installer nella sezione *Download* del sito *www.python.it*: l'ultima versione Python 3.6 qui disponibile richiede almeno Vista. Se vogliamo installare su XP dobbiamo procurarci la versione 3.4[3].

La dotazione di base che installiamo su Windows, come quella per Linux, per quanto riguarda i calcoli contiene solo il modulo **math**, ma contiene, in più, la IDLE e il modulo **tkinter**.

* * * * *

Per lavorare bene con Python è consigliabile installare la IDLE (Integrated Development and Learning Environment). Su Windows, come ho appena detto, l'installazione avviene automaticamente quando lanciamo l'installer di Python. Su Linux e Mac OS X dobbiamo farlo noi ricorrendo al repository della nostra distro, nel caso di Linux, o, nel caso di Mac OS X all'indirizzo *https://www.python.org/download/mac/tcltk/*.

L'utilità di installare l'IDLE sta innanzi tutto nel fatto che, facendolo, automaticamente arricchiamo il pacchetto base del modulo **tkinter**, che ci consente di sviluppare in maniera abbastanza semplice applicazioni con interfaccia grafica. A parte questa utilità secondaria (il modulo lo potremmo installare altrimenti, come vedremo) l'IDLE ci offre una shell di Python con simpatiche e utilissime prestazioni di autoinserimento di funzioni e parametri molto utile per l'apprendimento del linguaggio. La IDLE contiene pure un editor per scrivere i programmi, ma questo non ci offre nulla in più di quanto ci offra un qualsiasi editor di testo.

3 Il repository di Python

La comunità Python ha sviluppato e continua a sviluppare programmi e librerie per Python.

Nel momento in cui scrivo, una catalogazione di quasi 100.000 oggetti di questo tipo la troviamo all'indirizzo

https://pypi.python.org/pypi

PyPI sta per Python Package Index.

Per esplorare il contenuto di questo enorme deposito possiamo cliccare su BROWSE PACKAGES in alto a sinistra della pagina web e scegliere le parole chiave nella successiva o successive pagine.

Per verificare se un determinato pacchetto si trova nel deposito possiamo scriverne il nome o parte del nome nella finestrella di search in alto a destra della pagina web.

Una volta individuato il package che ci interessa, cliccando sul suo nome apriamo una finestra in cui troviamo una più o meno ampia descrizione del pacchetto.

Questi pacchetti possono essere programmi che fanno qualche cosa una volta lanciati con l'interprete Python che abbiamo sul computer. Ad esempio, nel repository possiamo trovare programmi di utilità, giochi, ecc.

Di maggior interesse penso tuttavia siano le librerie (moduli) che ci consentono di espandere le potenzialità del pacchetto base.

Per arricchire il nostro Python di base con una libreria per il calcolo matriciale, per esempio, ci serve il modulo **NumPy**.

Se vogliamo un arricchimento per calcoli scientifici, altre a NumPy ci serve **SciPy**.

[3] vedi nota precedente

Come ho detto prima, questi due pacchetti li troviamo già nella versione preinstallata sul Mac.

Se vogliamo inoltrarci nel machine learning, oltre ai due moduli precedenti ci serve **scikit-learn** (importabile negli script come sklearn).

Di grande utilità, per sviluppare grafici 2D, il modulo **matplotlib**.

Per lavorare su serie temporali abbiamo il modulo **pandas**.

Installando tutte queste librerie attrezziamo il nostro Python per elaborazioni matematiche e statistiche di altissimo livello.

Se, più che alla scienza, siamo interessati al gioco, possiamo avere bisogno della libreria **python-chess**, utile per sviluppare applicazione di gioco a scacchi o la libreria **pgoapi** per sviluppare applicazioni pokemon, ecc.

Per sviluppare applicazioni dotate di interfaccia grafica, in alternativa o in aggiunta al già citato modulo **tkinter**, facile da usare ma alquanto spartano nei risultati estetici, troviamo **PyQt4** e **wxPython**.

Ovviamente, per usare i moduli aggiuntivi dobbiamo anche sapere come funzionano e il nostro indice PyPI, insieme ad una sommaria descrizione del pacchetto, ci indica il sito web di chi o della comunità che l'ha sviluppato, in modo che ci sia possibile trovare documentazione.

Per pacchetti dell'importanza di NumPy o SciPy, come per le librerie Qt4 e wxWidgets, troviamo parecchi manuali e manualetti navigando su Internet.

Il problema è l'installazione, a volte complicata, soprattutto per i programmi più che per le librerie, dal dover rispettare certe dipendenze, dal dover tener conto della versione Python con cui vogliamo utilizzare il modulo contenuto nel pacchetto, ecc.

Tutti questi problemi si possono superare utilizzando un installatore che si chiama **pip**.

Installazione e uso di pip

Sistema Linux

Dalle versioni Python 2.7.9 e Python 3.4 pip è installato di default.

Nelle distro Linux della famiglia Debian (Ubuntu e derivate, Mint) lo possiamo comunque installare sia come generico pip (che agisce sulla versione Python di default) sia come pip specializzato per la versione 3 di Python, con i comandi a terminale impartiti con collegamento Internet attivo

```
sudo apt-get install python-pip
sudo apt-get install python3-pip
```
[4].

Per installare un pacchetto contenuto in PyPI si usa il comando a terminale

```
pip install <nome_pacchetto>
```
oppure
```
pip3 install <nome_pacchetto>.
```

Per disinstallare un pacchetto precedentemente installato con pip si usa il comando a terminale

```
pip uninstall <nome_pacchetto>
```
oppure
```
pip3 uninstall <nome_pacchetto>.
```

Non dovrebbe essere richiesto di impartire questi comandi come superuser; in caso di difficoltà far precedere `sudo` ai comandi.

Con il comando

```
pip list
```
oppure
```
pip3 list
```
possiamo vedere l'elenco dei pacchetti catalogati in PyPI che abbiamo installato.

L'installazione di pip può avvenire anche scaricando il file **get-pip.py** da

[4]Nelle versioni più recenti di Linux non si usa più apt-get ma semplicemente apt.

https://bootstrap.pypa.io/get-pip.py
e, posizionati nella directory dove abbiamo messo il file scaricato, dando il comando a terminale
`python get-pip.py`
oppure
`python3 get-pip.py`.

In ogni caso l'upgrade di pip si fa con il comando a terminale
`pip install -U pip`
oppure
`pip install --upgrade pip`.

Sistema Mac OS X

Su Mac non è installato pip.

Oltre che con get-pip.py, nel modo appena visto per Linux, possiamo installare pip su Mac impartendo i seguenti comandi a terminale
`xcode-select --install`
`sudo easy_install pip`.

L'upgrade lo facciamo con
`sudo pip install --upgrade pip`.

Per l'installazione, la disinstallazione e la listatura dei pacchetti valgono gli stessi comandi visti per Linux.

Sistema Windows

Se installiamo Python con l'installer come abbiamo visto nel paragrafo precedente, ci troviamo l'eseguibile pip.exe nella directory C:\PythonXX\scripts e da lì possiamo lanciarlo per installare, disinstallare o listare i pacchetti con gli stessi comandi visti per Linux.

Rammento che quello che in Linux e Mac si chiama terminale per lanciare questi comandi, in Windows è la finestra che si apre con il comando `cmd`.

* * * * *

Alcuni dei principali pacchetti di moduli contenuti nel repository, come NumPy, SciPy, PyQt4, ecc. sono comunque scaricabili da Internet in archivi compressi e sono installabili anche nel modo che vediamo nel capitolo che segue.

4 Alternative al repository

Come ho appena detto, alcuni pacchetti che fanno parte del repository si trovano facilmente su Internet in archivi compressi che si prestano all'installazione manuale e altri pacchetti, che possiamo trovare su Internet, per i più svariati motivi, non sono catalogati nel repository.

In questi casi, procuratoci l'archivio che contiene il pacchetto (generalmente si tratta di un archivio compresso con estensione .tar.gz o .zip), lo scompattiamo e, con privilegio da superutente, lo trasferiamo nella directory adatta, che ora vedremo dove si trova, e da questo momento il modulo è importabile nei nostri script Python.

Unica avvertenza quella di sapere se il pacchetto è adatto alla versione Python che abbiamo installato.

La directory in cui mettere l'archivio si chiama `dist_packages` o `site_packages` e si trova ai seguenti indirizzi

- in Linux e Mac OS X `/usr/lib/pythonXX` oppure `/usr/local/lib/pythonXX`
- in Windows `C:\pythonXX\Lib`.

dove pythonXX sarà, per esempio, python3.5, python2.7, ecc., a seconda della versione Python installata.

Un simpatico modulo, utile a fini didattici, che non si trova nel repository PyPI è contenuto nel pacchetto pygraph di Daniele Zambelli e lo possiamo scaricare da
https://bitbucket.org/zambu/pygraph/downloads.

Il pacchetto non si trova in PyPI probabilmente per la sua omonimia con un altro pacchetto dedicato ai grafi, che è tutt'altra cosa, e la versione attualmente disponibile si chiama pygraph31.zip e richiede la versione 3 di Python.

L'archivio scompattato è costituito da una directory, che si chiama **pygraph**, all'interno della quale abbiamo le seguenti sottodirectory:
`doc`
`examples`
`test`
`pygraph`.

Le prime tre contengono documentazione, esempi e script di prova e servono per capire come funziona il modulo: all'interno di `doc`, in particolare, troviamo un ottimo manuale. Queste tre directory le possiamo tranquillamente trasferire in posizione più comoda da raggiungere, togliendole da dove sono ora.

La directory `pygraph` con ciò che rimane (file vari e la sottodirectory `paygraph`) costituisce il vero modulo Python e lo trasferiamo in `dist_packages`.

In questo modo, in maniera un po' più laboriosa di quanto ci avrebbe consentito pip, abbiamo arricchito il nostro Python di una libreria che ci consente di implementare nei nostri script una geometria della tartaruga (peraltro la tartaruga è già presente nel pacchetto Python di base), un piano cartesiano, un plotter di funzioni in una variabile e una geometria interattiva.

Questo ha voluto essere un esempio di come si possa installare un qualsiasi pacchetto senza fare ricorso a pip.

Non è detto, infine, che qualche modulo sia installabile, per chi usa linux, ricorrendo ai vari installatori (apt per Debian e famiglia Ubuntu e derivate, compreso Mint, yum per Fedora e Red Hat, YaST per SuSE) che attingono dai repository delle varie distro.

5 Il problema della trasferibilità degli script

Il programma Python, in realtà, è uno script, cioè è una serie di istruzioni scritte in linguaggio pseudo-umano, il linguaggio Python, che vengono eseguite dal computer essendo interpretate nel momento stesso dell'esecuzione. Sul computer dove vengono interpretate deve essere installato tutto ciò che serve per interpretarle, cioè il pacchetto Python di base e tutte le librerie cui eventualmente lo script faccia ricorso, quelle importate nella stesura dello script stesso. Uno script che fa ricorso al modulo NumPy può essere eseguito solo su un computer sul quale sia installato non solo il pacchetto base di Python ma anche il modulo NumPy, a sua volta scritto in linguaggio Python.

Tutti i vantaggi fruiti al momento della stesura dello script, con la possibilità di sperimentare le istruzioni nel momento stesso in cui le scrivevamo e con l'accesso ad una miriade di librerie di funzioni preconfezionate che ci facilitava il compito, li paghiamo nel momento in cui non ci accontentiamo di essere noi i soli utenti dello script, sul nostro computer, ma vogliamo trasferirlo ad altri, affinché lo possano utilizzare sul loro computer.

Un modo per superare il problema sarebbe quello di indurre il destinatario dello script ad attrezzarsi per poterlo eseguire: basterebbe che anche lui installasse le librerie necessarie. Né sarebbe difficile sapere quali debbano essere: basterebbe caricare lo script in un editor di testo e vedere, nelle prime righe, quali moduli esso importa.

Un modo più comodo per il destinatario dello script è quello di trasferirgli un file che contenga tutto quanto serve per eseguire lo script stesso. Alcuni chiamano questo file "eseguibile": propriamente non è un eseguibile in linguaggio macchina del tipo di quelli prodotti compilando programmi scritti in C o in Pascal; è più simile all'altrettanto così detto "eseguibile" archivio

.jar del linguaggio Java. Con la differenza, comunque, che l'archivio .jar richiede la presenza di Java sul computer, mentre il file che produciamo nel caso di Python non richiede nemmeno la presenza dell'interprete di base Python, per cui assomiglia veramente ad un eseguibile propriamente detto. Tutto ciò si paga con la pesantezza in termini di byte: ma, con le tecnologie di cui disponiamo ora, sia per trasferire sia per archiviare file, è un problema relativo.

Lo strumento principe per produrre questo file è **PyInstaller**, un pacchetto che, essendo contenuto nel PyPI, è installabile con pip con il comando a terminale
`pip install pyinstaller`.

Per produrre il file, posizionati nella directory che contiene lo script, si scrive a terminale il comando
`pyinstaller --onefile` <nome_script>`.py`
e nella stessa directory, con altre cose che possiamo cancellare, troviamo una nuova directory `dist` che contiene un file che ha lo stesso nome dello script, senza l'estensione .py, e che è il così detto "eseguibile" concentrato in un solo file.

L'indicazione dell'opzione `--onefile` è consigliabile in quanto, altrimenti, nella nuova directory `dist` non troveremmo solo il file che ci interessa, unico, ma una sottodirectory, con il nome dello script di partenza contenente un file "eseguibile" che deve viaggiare accompagnato da tutto quanto contiene la sottodirectory stessa, il che sarebbe parecchio scomodo da trasferire e da maneggiare.

Da tener presente che un file prodotto con PyInstaller su un sistema a 32 bit non gira su un sistema a 64 bit e viceversa. Così come un file prodotto con PyInstaller su un sistema Linux non gira su Windows e viceversa.

Altra avvertenza: su Linux il file "eseguibile" va reso tale sul computer di destinazione con `chmod +x` <nome_file> oppure attraverso il gestore file, cliccando destro sul file e agendo su PROPRIETÀ -> PERMESSI.

In alternativa a PyInstaller, che funziona su tutti i sistemi operativi Linux, Mac OS X e Windows, ricordo l'esistenza, per il sistema Windows, di **py2exe** e, per il sistema Mac OS X, di **py2app**, che fanno la stessa cosa.

Detto questo, devo segnalare l'esistenza di una corrente di pensiero che considera una cattiva soluzione quella del ricorso a questi software e che raccomanda, soprattutto nel caso di script impegnativi che utilizzino librerie importanti, l'installazione di quanto serve sul computer destinatario del trasferimento dello script.