Unidad 01
Introducción

Curso de geoprocesamiento de datos con Python
2016

Cayetano Benavent Viñuales
Analista GIS en Geographica
cayetano.benavent@geographica.gs

Unidad 01 - Sumario de contenidos

¿Qué es Python?

Definición:

“Python is an interpreted, interactive, object-oriented programming language.

It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes.

Python combines remarkable power with very clear syntax.”

(Python Software Foundation, https://www.python.org/)

¿Qué es Python?

Definición:

"It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++.

It is also usable as an extension language for applications that need a programmable interface.

Finally, Python is portable: it runs on many Unix variants, on the Mac, and on PCs under MS-DOS, Windows, Windows NT, and OS/2."

(Python Software Foundation, https://www.python.org/)

Python Software Foundation (PSF)

La PSF, creada en creada el 6 de marzo de 2001, tiene por propósito:

"(...) Promote, protect, and advance the Python programming language, and to support and facilitate the growth of the international community of Python programmers."

Cualquier persona puede ser miembro y participar activamente en la comunidad Python.

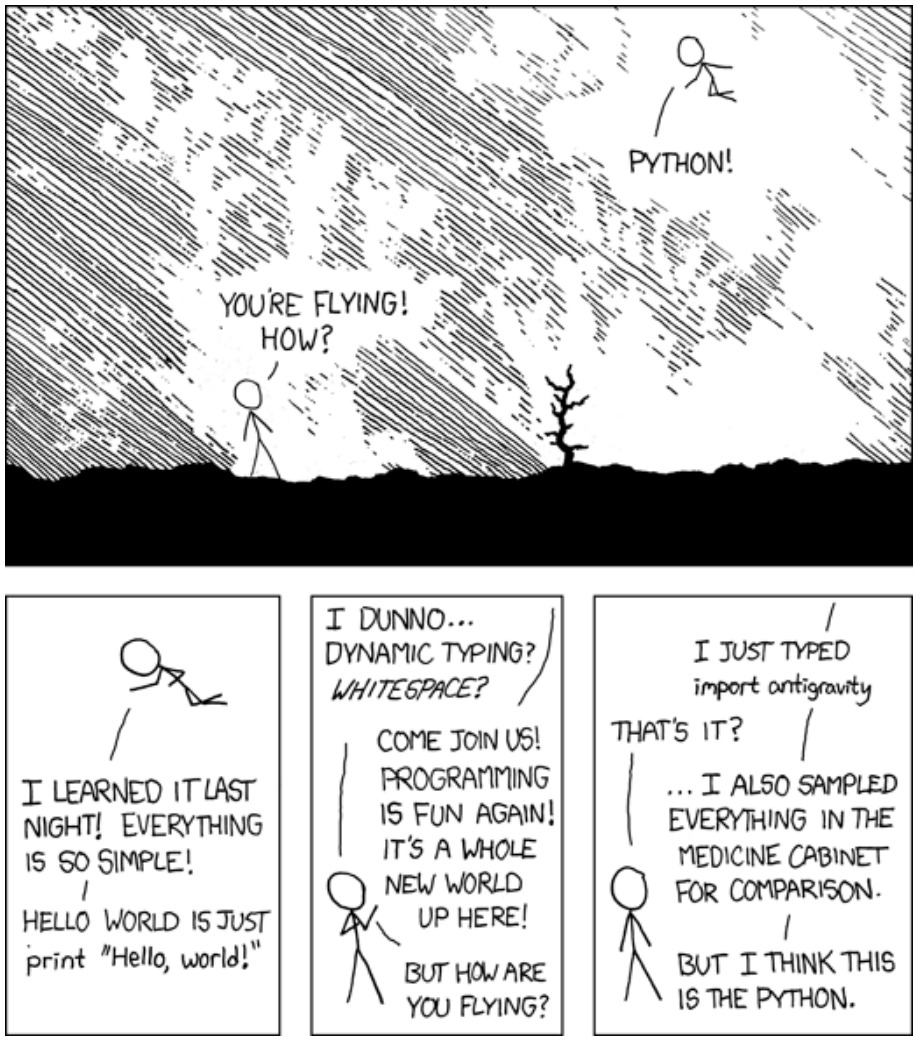
Para conocer más acerca de la PSF:

https://www.python.org/psf-landing/

¿Por qué usar Python para geoprocesamiento de datos?

- *You are flying! How?*

- *Python!*



https://xkcd.com/353/

¿Por qué usar Python para geoprocesamiento de datos?

Hay **muchos** motivos. Algunos de los principales son:

- Python es *free and open source*.

Python es liberado bajo los términos de The Python Software Foundation License (PSFL)1, que es una licencia BSD-style (permisiva y compatible con la GNU General Public License (GPL)).

https://docs.python.org/3/license.html

¿Por qué usar Python para geoprocesamiento de datos?

- Python es un lenguaje de propósito general.

Python es un lenguaje que encaja en casi cualquier tipo de aplicación, y se usa actualmente en prácticamente todos los sectores:

https://www.python.org/about/apps/

- Python es fácil de aprender.

Su elevada legibilidad, el ser interpretado, su carácter dinámico (en cuanto a tipado), etc. hacen que sea, en conjunto, un lenguaje muy asequible de asimilar.

¿Por qué usar Python para geoprocesamiento de datos?

- Python es multiplataforma.

Python funciona en los principales sistemas operativos:

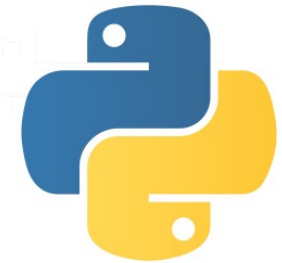
Linux

OSX

Windows

Podemos usar escribir un programa y ejecutarlo en cualquier sistema sin variarlo (con algunas excepciones).

¿Por qué usar Python para geoprocesamiento de datos?



- Python es uno de los lenguajes más importantes.

Python está entre los 5 principales lenguajes de programación:
(según el último índice TIOBE - Marzo 2016)

Mar 2016	Mar 2015	Change	Programming Language	Ratings	Change
1	**2**	⌃	**Java**	**20.528%**	**+4.95%**
2	1	⌄	C	14.600%	-2.04%
3	4	⌃	C++	6.721%	+0.09%
4	5	⌃	C#	4.271%	-0.65%
5	8	⌃	Python	4.257%	+1.64%
6	6		PHP	2.768%	-1.23%
7	9	⌃	Visual Basic .NET	2.561%	+0.24%
8	7	⌄	JavaScript	2.333%	-1.30%
9	12	⌃	Perl	2.251%	+0.92%
10	18	⌃⌃	Ruby	2.238%	+1.21%

http://www.tiobe.com/tiobe_index

¿Por qué usar Python para geoprocesamiento de datos?

- Python trae “baterías incluidas”.

Python mantiene desde sus inicios la filosofía de “baterías incluidas”. Significa que la distribución principal del lenguaje cuenta con un conjunto por defecto **enorme** de librerías y utilidades preinstaladas.

Ello queda materializado en The Python Standard Library (PSL).

https://docs.python.org/2/library/

https://docs.python.org/3/library/

¿Por qué usar Python para geoprocesamiento de datos?

- Catálogo de librerías enorme (fuera de la PSL).

Python, más allá de la PSL, tiene disponible un catálogo de librerías disponibles realmente grande. Abarcan prácticamente todos los ámbitos y temáticas.

El catálogo oficial es el Python Package Index (PyPI):

https://pypi.python.org

Otro catálogo a destacar es el provisto por Anaconda (de la empresa Continuum Analytics):

http://anaconda.org

¿Por qué usar Python para geoprocesamiento de datos?

- Stack computación científica maduro y potente.

Uno de los mayores éxitos de Python es la madurez, potencia y facilidad de uso de su stack de librerías para computación científica.

La principal materialización es SciPy Stack*:*

https://www.scipy.org

¿Por qué usar Python para geoprocesamiento de datos?

- Comunidad Python.

La comunidad Python se caracteriza por ser:

- Enorme.
- Muy activa.
- Muy diversa.
- Creciente...

En el mundo científico la comunidad es especialmente grande (en los últimos años ha crecido de una manera espectacular).

¿Por qué usar Python para geoprocesamiento de datos?

- Por último... porque programar es divertido con Python...

```
def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):
    print("-- This parrot wouldn't", action, end=' ')
    print("if you put", voltage, "volts through it.")
    print("-- Lovely plumage, the", type)
    print("-- It's", state, "!")
```

Fuente: http://www.theguardian.com / https://docs.python.org

¿Por qué usar Python para geoprocesamiento de datos?

- Por último... porque programar es divertido con Python...

```
(…)

LUKE: Is Perl better than Python?

YODA: No... no... no.  Quicker, easier,
      more seductive.

LUKE: But how will I know why Python is
      better than Perl?

YODA: You will know.  When your code you
      try to read six month
```

https://www.python.org/doc/humor/

¿Cuál debo utilizar?

Actualmente hay dos **versiones** estables de Python:

- Python 2.7
- Python 3.5

Según la PSF:

"Python 2.x is legacy, Python 3.x is the present and future of the language".

https://wiki.python.org/moin/Python2orPython3

¿Cuál debo utilizar?

Se debe aprender a manejar Python 3.X, pero hay que conocer bien las **diferencias principales** con Python 2.X. Todavía estamos en un periodo (¡¡muy largo!!) de transición entre versiones. Algunos sistemas y/o librerías importantes todavía usan Python 2.X.

"But wouldn't I want to avoid 2.x? (…)

Well, not entirely. Some of the less disruptive improvements in 3.0 and 3.1 have been backported to 2.6 and 2.7, respectively."

https://wiki.python.org/moin/Python2orPython3

Principales diferencias entre Python 2 y Python 3

Hay muchas diferencias en el core del lenguaje.

No obstante, a nivel “superficial” se han limado bastante en las última versiones de Python 2.

Tener en cuenta estas diferencias es clave si necesitamos escribir código Python 3 compatible con Python 2 (o viceversa).

Principales diferencias entre Python 2 y Python 3

Función Print

Es la diferencia más conocida, y la más fácil de resolver.

En Python 3 `print` pasa a ser una función.

En Python 2:

```
>>> print 'Hola, mundo'
```

En Python 3:

```
>>> print('Hola, mundo')
```

En Python 2 podemos (y debemos) siempre usar la forma Python 3. En Python 3 no podemos usar la forma Python 2.

Principales diferencias entre Python 2 y Python 3

División de enteros

La división entre enteros es otra de las diferencias clave. Si dividimos dos enteros en Python 2, el resultado es un entero.

En Python 2:

```
>>> 1 / 2
0
```

En Python 3:

```
>>> 1 / 2
0.5
```

Principales diferencias entre Python 2 y Python 3

División de enteros

Si en Python 2 dividimos un `float` por un `integer`, el resultado sí es `float`:

```
>>> 1.0 / 2
0.5
```

La solución óptima en Python 2 es usar en los scripts:

```
>>> from __future__ import division
```

Principales diferencias entre Python 2 y Python 3

Hay otras diferencias que merecen ser conocidas.

Se listan algunas de las más interesantes:

- Ámbito de variables en bucles FOR.
- Excepciones.
- Rangos con range y xrange.
- Soporte de Unicode.

En este magnífico artículo de S. Raschka puede profundizarse más en este asunto:

http://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html

Una documentación más exhaustiva aún en:

http://python-notes.curiousefficiency.org/en/latest/python3/index.html

Jupyter-notebook

“The Jupyter Notebook is a web application for interactive data science and scientific computing.”

http://jupyter.org/

Todos los ejercicios del curso están contenidos en Jupyter notebooks. Siempre que veamos este logo que aparece abajo, tendremos que abrir un jupyter-notebook.

Jupyter Notebook

Jupyter-notebook

Para abrir la aplicación (que se ejecuta en un navegador web), debemos dirigirnos a la carpeta dónde tenemos los notebooks que queremos editar o abrir, o la carpeta dónde queremos generar nuevos notebooks. Una vez allí, escribimos en la línea de comando:

```
$ jupyter-notebook
```

Git

Git es una de las herramientas de desarrollo de software colaborativo más importantes.

Combinada con GitHub, una plataforma en la nube para alojar repositorios Git, podemos alcanzar gran productividad desarrollando software.

Git

Todos los Jupyter notebooks que usaremos están alojados en GitHub.

Para poder usarlos como notebooks en nuestra propia máquina, seguiremos los pasos que se dictan en el Notebook unit01_01.

Instalación de Python

Python es multiplataforma.

La instalación en cada sistema es diferente:

Windows

https://www.python.org/downloads/windows/

OSX

https://www.python.org/downloads/mac-osx/

Linux

Depende totalmente de la distro usada (suele venir por defecto instalado). Para actualizar, seguir los consejos de cada distro.

El intérprete de Python

El intérprete de Python es una de sus mayores riquezas ya que permite lo que se conoce como “modo interactivo”.

Para entender mejor su funcionamiento, realizaremos un ejercicio práctico siguiendo el Notebook unit01_02.

Ejecución de scripts de Python en la línea de comandos

Para ejecutar un programa Python fuera del modo interactivo, debemos primero escribo el código fuente en un fichero con extensión .py, y posteriormente ejecutarlo desde la línea de comandos.

Para entender mejor su funcionamiento, realizaremos un ejercicio práctico siguiendo el Notebook unit01_03.

IPython: un intérprete avanzado

Uno de los potenciales de Python, el modo interactivo, se ve MUY mejorado con IPython. Está especialmente diseñado para computación científica, y pertenece al SciPy Stack (que veremos en la unidad 04).

Para entender mejor su funcionamiento, realizaremos un ejercicio práctico siguiendo el Notebook unit01_04.

Creando repositorios en GitHub

Ahora vamos a crear el repositorio de trabajo personal en GitHub para almacenar todo el trabajo que realicemos durante el curso.

Debemos abrir y seguir el Notebook unit01_05.

Instalación de librerías del PyPi con pip

El Python Package Index (PyPi) es el repositorio oficial de librerías de terceros de Python (*official third-party software repository*).

El PyPI es mantenido por la PSF, aunque cada librería individual es mantenida por la persona/s que lo ha subido.

Subir librerías al PyPI es libre, aunque debemos registrarnos previamente. Descargar librerías del PyPI es libre. Únicamente debemos tener en cuenta los términos legales de cada librería.

https://pypi.python.org/pypi

Instalación de librerías del PyPi con pip

El cliente oficial para instalación de paquetes desde el PyPI es pip.

Pip se maneja desde la línea de comandos. Su uso es tremendamente sencillo:

```
$ pip install nombrepaqueteainstalar
```

Documentación oficial:

https://pip.pypa.io

Bibliografía

Documentación oficial de la PSF:

** Referencia oficial del lenguaje (poco asequible para principiantes):*

- PSF (2016): “The Python Language Reference”. Python Software Fundation.
    - Python 3: https://docs.python.org/3/reference/
    - Python 2: https://docs.python.org/2/reference/

- PSF (2016): “The Python Standard Library”. Python Software Fundation.
    - Python 3: https://docs.python.org/3/library/
    - Python 2: https://docs.python.org/2/library/

** Introducción informal (asequible para principiantes):*

- PSF (2016): “The Python Tutorial”. Python Software Fundation.
    - Python 3: https://docs.python.org/3/tutorial/
    - Python 2: https://docs.python.org/2/tutorial/

Bibliografía

Selección bibliográfica:

- Allen B. Downey (2015): “Think Python. How to think like a computer scientist (2nd Edition)”. O'Reilly. http://www.greenteapress.com/thinkpython2/index.html

- Zed Shaw (2014): “Learn Python the Hard Way (3rd Edition)”. Addison Wesley.

- Alex Martelli (2009): “Python in a Nutshell (2nd Edition)”. O'Reilly.

- Mark Lutz (2015): “Learning Python (5th Edition)”. O'Reilly.

- David Beazley, Brian K. Jones (2013): “Python Cookbook (3rd Edition)”. O'Reilly.

- Luciano Ramalho (2015): “Fluent Python”. O'Reilly.

- David Beazley (2009): “Python Essential Reference (4th Edition)”. Addison Wesley.