1 Python - Introducción a Python

Introducción a Python

1.1 Sumario

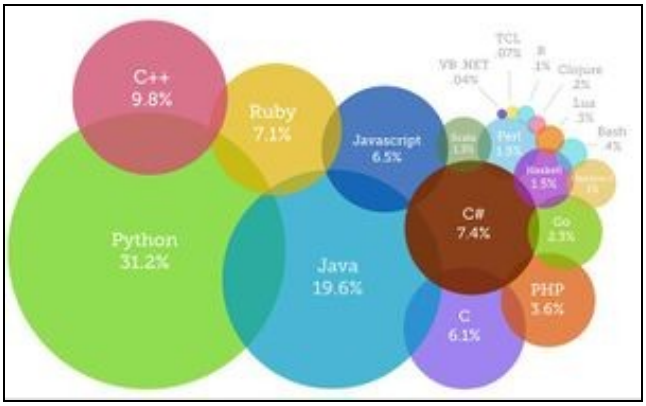
1.2 ¿Qué es Python?

Python es un lenguaje de programación interpretada que intenta favorecer una escritura de código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, emplea tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada **Python Software Foundation License**, que es compatible con una Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

1.3 ¿Por qué Python?



Programming Languages 2020

En el mundo de la informática nos podemos encontrar con un gran número de lenguajes de programación diferentes, un número tan elevado que es prácticamente imposible que un programador sea capaz de controlar todos ellos a la perfección. Lo que sí se puede conseguir es llegar a dominar aquellos lenguajes más importantes y que más opciones ofrecen para encontrar un buen trabajo.

Los usuarios de Phyton dicen que puede hacerlo casi todo. Aplicaciones web, interfaces de usuario, análisis de datos, estadísticas: sea cual sea el problema, hay probablemente un framework para Python.

Además, Python, es la opción más popular entre los principiantes en programación debido a sus bajas barreras de entrada y su facilidad de uso. Por el contrario, este detalle lleva a que los programadores ya experimentados se frustren cuando lo están aprendiendo.

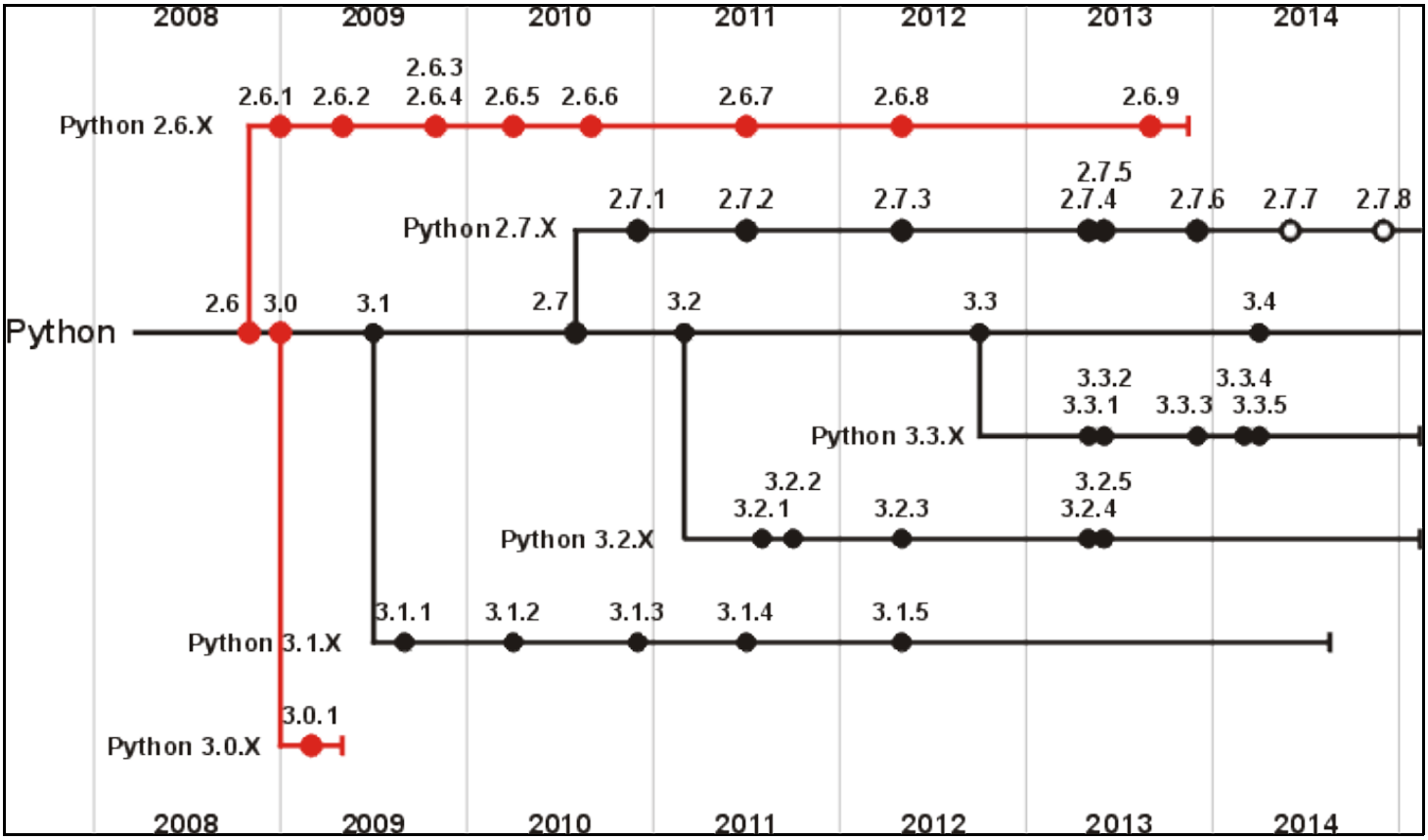
1.4 Historia de Python

- Python fue creado a finales de los años ochenta por Guido van Rossum en el Centro para las Matemáticas y la Informática **(CWI, Centrum Wiskunde & Informatica)**, en los Países Bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo **Amoeba**.

- El nombre de este lenguaje viene de la afición de su creador por los humoristas británicos Monty Python.
- En febrero de 1991 publicó la primera versión pública, la versión 0.9.0. La versión 1.0 se publicó en enero de 1994, la versión 2.0 se publicó en octubre de 2000 y la versión 3.0 se publicó en diciembre de 2008. El desarrollo de Python lo lleva a cabo un colectivo de programadores que trabaja bajo el paraguas de la fundación **Python Software Foundation**, pero Guido van Rossum sigue dirigiendo el desarrollo de Python.
- La mayoría de las distribuciones Linux, y tambien MAC, tienen instaladas dos versiones de Python. Por ejemplo Debian stretch tiene instalados Python 2.7 (versión por defecto) y Python 3.5.

  > La transición de Python 2 a Python 3 está resultando mucho más costosa de lo esperado, seguramente porque Python 3 introdujo muchos cambios en el lenguaje y obliga a reescribir prácticamente todos los programas (aunque se han creado herramientas para ayudar en ese proceso).

- El desarrollo de Python se lleva a cabo en gran parte a través del proceso de ***Python Enhancement Proposal* (PEP)**. El proceso de PEP es el mecanismo principal para proponer nuevas características importantes, para recopilar información de la comunidad sobre un problema y para documentar las decisiones de diseño que se han aplicado a Python. Los PEP sobresalientes son revisados ??y comentados por la comunidad de Python y por Van Rossum.
- Las versiones de Python y su cronología podemos verla en la siguiente imagen (Fuente Wikipedia):



1.5 Instalación y actualización de Python

Por defecto, en Linux y Mac ya vendrán instalados el Python 2.7.x y el Python 3.5.x. Sí tendremos que realizar la instalación en Windows. Para ver la versión de Python y Python3 instaladas en mi equipo ejecutaremos los comandos:

```
$ python -V
Python 2.7.13
$ python3 -V
Python 3.5.3
```

Si vemos que no está instalado Python3 o no es la versión más actual, podemos realizar la descarga desde la web python.org, para, a continuación, hacer la instalación y así tener la última versión.

```
$ apt-cache policy python3
$ apt-get install python3
$ ls -lia /usr/bin | grep python3
```

En Windows, si tenemos Chocolatey.

```
#Para instalar:
PS> choco install python3
#Para actualizar:
PS> choco upgrade python3
```

1.5.1 Configurar nano como editor Python

Si se utiliza **nano** es posible activar la detección de la sintaxis en el editor, esa característica permite trabajar con distintos tipos de fichero para destacar ciertas palabras clave de la sintaxis en otros colores. Es una ayuda importante a la hora de editar código fuente de todo tipo de lenguajes, pero también para la edición de ficheros de texto pertenecientes a otros formatos como correos electrónicos o páginas de manual.

Para hacerlo simplemente hay que editar el fichero de configuración:

```
sudo nano /etc/nanorc
```

Esto hará que se abra el citado fichero de configuración de **nano**, en el que tendremos que añadir estas líneas (o descomentarlas quitando la almohadilla en esas líneas si están presentes):

```
## Nanorc files
include "/usr/share/nano/nanorc.nanorc"

## C/C++
include "/usr/share/nano/c.nanorc"

## HTML
include "/usr/share/nano/html.nanorc"

## TeX
include "/usr/share/nano/tex.nanorc"

## Quoted emails (under e.g. mutt)
include "/usr/share/nano/mutt.nanorc"

## Patch files
include "/usr/share/nano/patch.nanorc"

## Manpages
include "/usr/share/nano/man.nanorc"

## Perl
include "/usr/share/nano/perl.nanorc"

## Python
include "/usr/share/nano/python.nanorc"

## Ruby
include "/usr/share/nano/ruby.nanorc"

## Java
include "/usr/share/nano/java.nanorc"

## Assembler
include "/usr/share/nano/asm.nanorc"

## Bourne shell scripts
include "/usr/share/nano/sh.nanorc"
```

1.5.2 PyCharm

Para programar en Python podemos utilizar un editor de texto plano cualquiera como nano, vi, notepad, etc. Pero, sin duda, lo mejor es instalar un IDE como puede ser PyCharm.

> PyCharm es un IDE (Entorno de desarrollo integrado) desarrollado por la compañía Jetbrains, está basado en **IntelliJ IDEA**, el IDE de la misma compañía pero enfocado hacia *Java* y la base de *Android Studio*. *Pycharm* tiene cientos de funciones que lo puede ver como una herramienta muy pesada, pero que valen la pena ya que ayuda con el desarrollo del día a día.
> Características:

- Autocompletado, resaltador de sintaxis, herramienta de análisis y refactorización.
- Integración con *frameworks web* como: *Django*, *Flask*, *Pyramid*, *Web2Py*.
- Frameworks javascripts: jQuery, AngularJS.
- Debugger avanzado de Python y Javascript.
- Integración con lenguajes de plantillas: Mako, Jinja2, Django Template.
- Soporta entornos virtuales e intérpretes de Python 2.x, 3.x, PyPy, Iron Python y Jython.
- Compatibilidad con SQLAlchemy (ORM), Google App Engine, Cython.
- Soporte para modo VIM (Con plugin)
- Sistemas de control de versiones: Git, CVS, Mercurial.

PyCharm es multiplataforma, hay binarios para: WIndows, Linux y Mac OS X. Existen dos versiones de PyCharm, una comunitaria y otra profesional.
Podemos ver, a continuación, la instalación de PyCharm Community Edition Linux:

```
$ sudo su
$ cd /
$ mkdir /opt
$ cd opt
$ mv /tmp/pycharm-community-4.5.4.tar.gz /opt
$ tar -xvzf ./pycharm-community-4.5.4.tar.gz
$ rm pycharm-community-4.5.4.tar.gz
$ cd pycharm-community-4.5.4/
$ cd bin/
$ ./pycharm.sh
```

Así de sencillo es desplegar la aplicación en nuestro sistema Linux, ahora sólo queda ejecutarla, para ello ejecutamos el archivo **sh** :

```
$ /opt/pycharm-community-5.0.1/bin/pycharm.sh
```

Pero claro, cada vez que queramos arrancarlo acordarnos de la ruta es un poco tedioso. Si queremos evitar este problema, haremos un enlace a /**usr**/**local**/**bin** :

```
$ ln -s /opt/pycharm-community-5.0.1/bin/pycharm.sh /usr/local/bin/pycharm
$ ln -s /opt/pycharm-community-5.0.1/bin/inspect.sh /usr/local/bin/inspect
```

A partir de ahora, podemos ejecutar PyCharm en consola:

```
$ pycharm
```

◊ Cambiar el tamaño de la letra en Pycharm

1.6 El Intérprete de Python

Python es un lenguaje interpretado, lo cual puede ahorrarte mucho tiempo durante el desarrollo ya que no es necesario compilar ni enlazar. En el caso de Python, "el intérprete" puede usarse interactivamente, lo que facilita experimentar con características del lenguaje, escribir programas descartables, o probar funciones cuando se hace desarrollo de programas de abajo hacia arriba.
También es interesante saber que el intérprete es también una calculadora de escritorio práctica.
Para iniciar el intérprete sólo es necesario escribir *python* en la linea de comandos y pulsar **Enter**. Luego ya podemos empezar a probar código...

```
$ python
Python 2.7.9 (v2.7.9:648dcafa7e5f, Dec 10 2014, 10:10:46)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 5                        # Definimos la variable "a"
>>> a                              # Así nos devuelve el valor de esa variable
5
>>> a + 2
7
>>> type (a)                     # Vemos qué tipo de variable es "a"
<type 'int'>                      # "a" es un 'entero'
>>> a = "hola"                 # Ahora "a" cambia de 'Tipo'
>>> a
'hola'
>>> type (a)
<type 'str'>                      # "a" es un 'string'
# Como es un string, ahora tiene un tipo de métodos y propiedades propias
```

```
>>> len(a)                      # Calculamos el número de caracteres de "a"
4
>>> a + len(a)                 # Este tipo de acciones da error
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>> a + str(len(a))           # Ahora sí estamos concatenando dos elementos del mismo tipo
'hola4'
# Python tiene muchas funciones definidas...
>>> cosa                       # Estamos llamando a una función que no existe
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cosa' is not defined
# Para salir del intérprete hay que pulsar Ctrl+D
>>> ^D
```

Como puede verse, es muy fácil experimentar con variables y operadores.
Como C++ y Java, Python también es sensible a las mayúsculas/minúsculas.
Al contrario de lo que existe en C++ y Java, Python no necesita terminar cada línea de código con un ";".
Los comentarios, tal y como podemos ver en el ejemplo, se escriben anteponiendo "#".

1.7 Los *scripts* en Python

Los scripts en Python usan la extensión ".py" y son llamados "módulos".
Podemos ver, a continuación, el módulo "hola.py", que podremos llamar desde la línea de comandos del modo "python hola.py Daniel", que llama al intérprete de Python para que ejecute el código existente en python.py pasándole el argumento "Daniel":

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

## Importamos el módulo "sys" para poder acceder a los parámetros pasados
import sys

# En la función main() se introduce el código principal del programa
def main():
    if len(sys.argv) > 1:
        print('Hola', sys.argv[1])
        # Los argumentos parados por linea de comandos son sys.argv[1], sys.argv[2] ...
        # sys.argv[0] es el nombre del script
    else:
        print('Introduce un argumento')

# Para empezar el programa llamamos del siguiente modo a la función "main()"
if __name__ == '__main__':
    main()
```

Si ejecutamos el programa desde la línea de comandos:

```
$ python3 proba.py Daniel
Hola Daniel
```

-- Volver