INTRODUCCIÓN

¿Qué es Python?

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses "Monty Python". Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible.

Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos.

Lenguaje interpretado o de script

Un lenguaje interpretado o de script es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados).

La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo los lenguajes interpretados son más flexibles y más portables.

Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi interpretado. En Python, como en Java y muchos otros lenguajes, el código fuente se traduce a un pseudo código máquina intermedio llamado bytecode la primera vez que se ejecuta, generando archivos .pyc o .pyo (bytecode optimizado), que son los que se ejecutarán en sucesivas ocasiones.

Tipado dinámico

La característica de tipado dinámico se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable,

sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.

Fuertemente tipado

No se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Por ejemplo, si tenemos una variable que contiene un texto (variable de tipo cadena o string) no podremos tratarla como un número (sumar la cadena `"9"` y el número `8`). En otros lenguajes el tipo de la variable cambiaría para adaptarse al comportamiento esperado, aunque esto es más propenso a errores.

Multiplataforma

El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios.

Orientado a objetos

La orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos.

Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.

¿Por qué Python?

Python es un lenguaje que todo el mundo debería conocer. Su sintaxis simple, clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, entre otros, hacen que desarrollar una aplicación en Python sea sencillo, muy rápido y, lo que es más importante, divertido.

La sintaxis de Python es tan sencilla y cercana al lenguaje natural que

los programas elaborados en Python parecen pseudocódigo. Por este motivo se trata además de uno de los mejores lenguajes para comenzar a programar.

Python no es adecuado sin embargo para la programación de bajo nivel o para aplicaciones en las que el rendimiento sea crítico.

Algunos casos de éxito en el uso de Python son Google, Yahoo, la NASA, Industrias Light & Magic, y todas las distribuciones Linux, en las que Python cada vez representa un tanto por ciento mayor de los programas disponibles.

Instalación de Python

Existen varias implementaciones distintas de Python: CPython, Jython, IronPython, PyPy, etc.

CPython es la más utilizada, la más rápida y la más madura. Cuando la gente habla de Python normalmente se refiere a esta implementación. En este caso tanto el intérprete como los módulos están escritos en C.

Jython es la implementación en Java de Python, mientras que IronPython es su contrapartida en C# (.NET). Su interés estriba en que utilizando estas implementaciones se pueden utilizar todas las librerías disponibles para los programadores de Java y .NET.

PyPy, por último, como habréis adivinado por el nombre, se trata de una implementación en Python de Python.

CPython está instalado por defecto en la mayor parte de las distribuciones Linux y en las últimas versiones de Mac OS. Para comprobar si está instalado abre una terminal y escribe `python`. Si está instalado se iniciará la consola interactiva de Python y obtendremos parecido a lo siguiente:

```
Python 2.5.1 (r251:54863, May 2 2007, 16:56:35)
[GCC 4.1.2 (Ubuntu 4.1.2-0ubuntu4)] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

La primera línea nos indica la versión de Python que tenemos instalada. Al final podemos ver el prompt (>>>) que nos indica que el intérprete está esperando código del usuario. Podemos salir escribiendo `exit()`, o pulsando Control + D.

Si no te muestra algo parecido no te preocupes, instalar Python es muy sencillo. Puedes descargar la versión correspondiente a tu sistema operativo desde la web de Python, en *http://www.python.org/download/*. Existen instaladores para Windows y Mac OS. Si utilizas Linux es muy probable que puedas instalarlo usando la herramienta de gestión de paquetes de tu distribución, aunque también podemos descargar la aplicación compilada desde la web de Python.

Herramientas básicas

Existen dos formas de ejecutar código Python. Podemos escribir líneas de código en el intérprete y obtener una respuesta del intérprete para cada línea (sesión interactiva) o bien podemos escribir el código de un programa en un archivo de texto y ejecutarlo.

A la hora de realizar una sesión interactiva os aconsejo instalar y utilizar iPython, en lugar de la consola interactiva de Python. Se puede encontrar en *http://ipython.scipy.org/*. iPython cuenta con características añadidas muy interesantes, como el autocompletado o el operador `?`. (para activar la característica de autocompletado en Windows es necesario instalar `PyReadline`, que puede descargarse desde *http://ipython.scipy.org/ moin/PyReadline/Intro*)

La función de autocompletado se lanza pulsando el tabulador. Si escribimos `fi` y pulsamos Tab nos mostrará una lista de los objetos que comienzan con fi (`file`, `filter` y `finally`). Si escribimos `file.` y pulsamos Tab nos mostrará una lista de los métodos y propiedades del objeto `file`.
El operador `?` nos muestra información sobre los objetos. Se utiliza añadiendo el símbolo de interrogación al final del nombre del objeto del cual queremos más información. Por ejemplo:

```
In [3]: str?
```

```
Type: type
Base Class:
String Form:
Namespace: Python builtin
Docstring:
str(object) -> string

Return a nice string representation of the object.
If the argument is a string, the return value is the same
object.
```

En el campo de IDEs y editores de código gratuitos PyDEV (*http://pydev.sourceforge.net/*) se alza como cabeza de serie. PyDEV es un plugin para Eclipse que permite utilizar este IDE multiplataforma para programar en Python. Cuenta con autocompletado de código (con información sobre cada elemento), resaltado de sintaxis, un depurador gráfico, resaltado de errores, explorador de clases, formateo del código, refactorización, etc. Sin duda es la opción más completa, sobre todo si instalamos las extensiones comerciales, aunque necesita de una cantidad importante de memoria y no es del todo estable.

Otras opciones gratuitas a considerar son SPE o Stani's Python Editor (*http://sourceforge.net/projects/spe/*), Eric (*http://die-offenbachs.de/eric/*), BOA Constructor (*http://boa-constructor.sourceforge.net/*) o incluso emacs o vim.

Si no te importa desembolsar algo de dinero, Komodo (*http://www.activestate.com/komodo_ide/*) y Wing IDE (*http://www.wingware.com/*) son también muy buenas opciones, con montones de características interesantes, como PyDEV, pero mucho más estables y robustos. Además, si desarrollas software libre no comercial puedes contactar con Wing Ware y obtener, con un poco de suerte, una licencia gratuita para Wing IDE Professional :)

MI PRIMER PROGRAMA EN PYTHON

Como comentábamos en el capítulo anterior existen dos formas de ejecutar código Python, bien en una sesión interactiva (línea a línea) con el intérprete, o bien de la forma habitual, escribiendo el código en un archivo de código fuente y ejecutándolo.

El primer programa que vamos a escribir en Python es el clásico Hola Mundo, y en este lenguaje es tan simple como:

```
print "Hola Mundo"
```

Vamos a probarlo primero en el intérprete. Ejecuta python o ipython según tus preferencias, escribe la línea anterior y pulsa Enter. El intérprete responderá mostrando en la consola el texto `Hola Mundo`.

Vamos ahora a crear un archivo de texto con el código anterior, de forma que pudiéramos distribuir nuestro pequeño gran programa entre nuestros amigos. Abre tu editor de texto preferido o bien el IDE que hayas elegido y copia la línea anterior. Guárdalo como hola.py, por ejemplo.

Ejecutar este programa es tan sencillo como indicarle el nombre del archivo a ejecutar al intérprete de Python

```
python hola.py
```

pero vamos a ver cómo simplificarlo aún más.

Si utilizas Windows los archivos .py ya estarán asociados al intérprete de Python, por lo que basta hacer doble clic sobre el archivo para ejecutar el programa. Sin embargo como este programa no hace más que imprimir un texto en la consola, la ejecución es demasiado rápida para poder verlo si quiera. Para remediarlo, vamos a añadir una nueva línea que espere la entrada de datos por parte del usuario.

```
print "Hola Mundo"
raw_input()
```

De esta forma se mostrará una consola con el texto `Hola Mundo` hasta que pulsemos Enter.

Si utilizas Linux (u otro Unix) para conseguir este comportamiento, es decir, para que el sistema operativo abra el archivo .py con el intérprete adecuado, es necesario añadir una nueva línea al principio del archivo:

```
#!/usr/bin/python
print "Hola Mundo"
raw_input()
```

A esta línea se le conoce en el mundo Unix como *shebang*, *hashbang* o *sharpbang*. El par de caracteres `#!` indica al sistema operativo que dicho script se debe ejecutar utilizando el intérprete especificado a continuación. De esto se desprende, evidentemente, que si esta no es la ruta en la que está instalado nuestro intérprete de Python, es necesario cambiarla.

Otra opción es utilizar el programa env (de environment, entorno) para preguntar al sistema por la ruta al intérprete de Python, de forma que nuestros usuarios no tengan ningún problema si se diera el caso de que el programa no estuviera instalado en dicha ruta:

```
#!/usr/bin/env python
print "Hola Mundo"
raw_input()
```

Por supuesto además de añadir el shebang, tendremos que dar permisos de ejecución al programa.

```
chmod +x hola.py
```

Y listo, si hacemos doble clic el programa se ejecutará, mostrando una consola con el texto `Hola Mundo`, como en el caso de Windows.

También podríamos correr el programa desde la consola como si tratara de un ejecutable cualquiera:

```
./hola.py
```