EL GRAN LIBRO DE PYTHON

Autor:

Marco Buttu

Python es un lenguaje de programación multiplataforma, consistente y maduro, en el cual confían con éxito las Empresas y organizaciones mundiales mas prestigiosas: Google, la NASA, YouTube, Intel y Yahoo! Su exito esta vinculado tanto al hecho de que favorece la productividad, haciendo mas sencillo el desarrollo de sistemas de software sin tener en cuenta su complejidad, como al hecho de que tiene múltiples entornos de uso: aplicaciones web, juegos y multimedia, interficies graficas, networking, aplicaciones científicas, inteligencia artificial y programacion de sistemas, entre muchos otros.

El gran libro de Python es el mas completo, moderno y detallado de entre los volúmenes dedicados a Python que pueden encontrarse actualmente en el mercado. Actualizado a la versión 3.4 del lenguaje, lanzada en enero de 2014. Su composición es muy detallada y sigue un curso gradual elaborado en torno a una amplia serie de ejemplos y ejercicios: parte de las bases del lenguaje, sin dar nada por sabido, hasta llegar a los argumentos considerados mas difíciles, incluso por los programadores mas experimentados.
Soporte al libro disponible online: http://code.google.com/p/the •]phytonic •]way/
Aspectos destacados:

- • Introduccion al lenguaje Python, a su sintaxis, a sus construcciones fundamentales y a la libreria estandar.
- Funciones y modos de emparejamiento de argumentos, generadores, corrutinas, archivos, comodines y expresiones
  regulares.
- • Modulos y paquetes, entornos y espacios de nombres, ambientes virtuales, instalacion y distribucion de aplicaciones.
- Prueba de validacion de cadenas de documentacion y desarrollo guiado por pruebas.
- Programacion orientada a objetos en Python: clases, herencia, gestion de las excepciones, patron y antipatron,
  propiedades y decoradores.
- Modelo a objetos de Python, atributos magicos, descriptores y metaclases.

Versiones de Python

Guido van Rossum ideó el lenguaje Python a finales de los 80 y comenzó a implementarlo en diciembre de 1989. 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. hasta 2018, el desarrollo de Python lo lleva a cabo un colectivo de programadores dirigido por Guido van Rossum y bajo el paraguas de la fundación Python Software Foundation. En julio de 2018 Guido van Rossum anunció que dejaría de dirigir el desarrollo de Python y a principios de 2019 se pondrá en marcha un consejo director de cinco miembros elegidos entre los desarrolladores de Python.

Las versiones de Python se identifican por tres números X.Y.Z, en la que:

- X corresponde a las grandes versiones de Python (1, 2 y 3), incompatibles entre sí:

  Los principales cambios introducidos en Python 2 fueron las cadenas Unicode, las comprensiones de listas, las asignaciones aumentadas, los nuevos métodos de cadenas y el recolector de basura para referencias cíclicas.

  Los principales cambios introducidos en Python 3 fueron la separación entre cadenas Unicode y datos binarios, la función print(), cambios en la sintaxis, tipos de datos, comparadores, etc.

  Por el momento, no hay planes de crear una nueva versión Python 4, incompatible con las anteriores.

- Y corresponde a versiones importantes en las que se introducen novedades en el lenguaje pero manteniendo la compatibilidad (salvo excepciones).

  Desde hace unos años, las versiones X.Y se publican aproximadamente cada año y medio y se mantienen durante cinco años, excepto la versión 2.7, que se mantendrá por lo menos durante diez años, hasta 2020.

- Z corresponde a versiones menores que se publican durante el período de mantenimiento, en las que sólo se corrigen errores y fallos de seguridad.

  Normalmente, se publica una última versión X.Y.Z justo antes de que una versión X.Y deje de mantenerse. Algunas empresas comerciales ofrecen el mantenimiento de versiones antiguas una vez acabado el mantenimiento oficial.

La imagen siguiente muestra la fecha de publicación de las versiones principales de Python, en cada una de las tres grandes versiones, Python 1, Python 2 y Python 3. Las versiones indicadas con punto rojo se consideran obsoletas, las versiones indicadas con punto negro siguen publicando actualizaciones, las versiones indicadas con punto blanco corresponden a versiones futuras con las fechas de publicación previstas.

Transición de Python 2 a Python 3

La transición de Python 2 a Python 3 ha resultado mucho más costosa de lo esperado, debido a que Python 3 introdujo muchos cambios en el

lenguaje y obligaba a reescribir prácticamente todos los programas (aunque se han creado herramientas para ayudar en ese proceso).

La intención inicial era haber terminado Python 2 con la versión 2.6, pero en 2010 se tuvo que publicar la versión 2.7, incorporando parte de las novedades de Python 3. Además, el período de mantenimiento de Python 2.7 se tuvo que duplicar de los cinco años habituales a diez, hasta 2020.

Un primer obstáculo en el proceso de transición de Python 2 a Python 3 fue la propia disponibilidad de Python 3 en las distribuciones GNU/Linux. Muchas herramientas internas de las distribuciones están escritas en Python y su conversión de Python 2 a Python 3 no era fácil, por lo que las distribuciones no podían pasar simplemente de incluir una versión a otra.

Hasta 2015, Python 2 siguió siendo la versión predeterminada de Python en la mayoría de distribuciones GNU/Linux (aunque se podía instalar Python 3 sin problemas en ellas). Felizmente, esta situación está en vías de solución:

- Fedora hizo la transición a Python 3 en Fedora 23 (noviembre de 2015) incluyendo Python 3.4 [wiki de Fedora]. Posteriormente, Fedora 24 (junio de 2016) incluyó Python 3.5, Fedora 26 (julio de 2017) incluyó Python 3.6 y Fedora 29 (octubre de 2018) incluyó Python 3.7. RedHat Linux 8 (mayo de 2019) usa Python 3.6, ya que está basada en Fedora 28.
- Ubuntu hizo la transición a Python 3.5 en Ubuntu 16.04 (abril de 2016) [wiki de Ubuntu] e incluye Python 3.6 en Ubuntu 18.04 (abril de 2018) [wiki de Ubuntu].
- Debian hizo la transición a Python 3 en Debian 10 (julio de 2019) incluyendo Python 3.7. Lo planes de transición se empezaron a discutir en abril de 2015 [artículo de Linux Week News del 29/04/15] y Debian 9 (junio de 2017) ya incluyó Python 3.5.3 (aunque la versión predeterminada seguía siendo Python 2.7.13.
- OpenStack, una importante plataforma de virtualización, adoptó Python 3 (concretamente, Python 3.5) en la versión Pike, publicada en agosto de 2017

Aunque las nuevas versiones de las distribuciones ya incluyan Python 3, las distribuciones que incluyen Python 2 seguirán instaladas en servidores durante bastantes años. Ese es el motivo por el que la fundación Python prolongó el período de publicación de actualizaciones de seguridad de Python 2.7 hasta finales de 2019. A partir de 2020 serán las distribuciones las que mantengan Python 2.7 durante el

tiempo que se mantengan las propias distribuciones (por ejemplo, RedHat 7 se mantendrá hasta el año 2024, como mínimo, y RedHat 8 o Debian 10 todavía permite instalar Python 2.7).

Un segundo obstáculo en el proceso de transición de Python 2 a Python 3 (y que ha afectado además a todos los sistemas operativos) ha sido la disponibilidad de las bibliotecas.

Python cuenta con un gran número de bibliotecas, cuyo repositorio oficial es PyPI (Python Package Index), que facilitan la programación de aplicaciones complejas. Cuando se publicó Python 3, la inmensa mayoría de bibliotecas sólo estaban disponibles para Python 2 y, lógicamente, si un programa necesitaba alguna biblioteca que sólo estaba disponible para Python 2, el programa no se podía pasar tampoco a Python 3.

Poco a poco, la mayoría de bibliotecas de Python han ido publicando versiones para Python 3, por lo que este problema también está en vías de solución.

En marzo de 2016, un estudio de empleados de Microsoft señalaba que por aquel entonces algo más del 50% de las bibliotecas estaban disponibles tanto para Python 2 como para Python 3, un 25% estaban disponibles sólo para Python 2 y un poco menos del 25% estaban disponibles sólo para Python 3, pero la tendencia parecía indicar que a mediados de 2016 Python 3 pasaría a ser la versión más popular.

Varias páginas web hacen un seguimiento de la compatibilidad con Python 3 de las bibliotecas más populares

- Python 3 Readiness y Python 3 Wall of Superpowers (parece que desde abril de 2018 esta web ya no actualiza los datos) muestran cómo la inmensa mayoría de las bibliotecas más populares ya están disponibles en Python 3
- la distribución GNU/Linux Fedora lleva el seguimiento de los paquetes de Python que forman parte de la distribución: paquetes y evolución.
- Drop Python muestra la proporción de las bibliotecas más populares que han dejado de dar soporte a las versiones obsoletas de Python (Python 2.6, Python 3.2 y Python 3.3).

La década de 1990: la era de Internet[editar]

El rápido crecimiento de Internet en la década de 1990 fue el siguiente gran acontecimiento histórico para los lenguajes de programación. Con la apertura de una plataforma totalmente nueva para los sistemas informáticos, Internet creó una oportunidad adoptar nuevos lenguajes. En particular, el lenguaje de programación JavaScript se hizo popular debido a su pronta integración con el navegador web Netscape Navigator, y varios lenguajes de scripting alcanzaron un amplio uso en el desarrollo de aplicaciones personalizadas para servidores web. La década de 1990 no vio ninguna novedad fundamental en los lenguajes imperativos, pero sí mucha recombinación y la maduración de viejas ideas. Esta era comenzó la difusión de los lenguajes funcionales. Una filosofía de conducción grande era la productividad del programador. Surgieron muchos lenguajes de "aplicaciones de desarrollo rápido" (RAD), los cuales usualmente venían con un IDE, recolector de basura, y eran descendientes de lenguajes anteriores. Todos estos lenguajes eran orientados a objeto. Estos incluían Object Pascal, Visual Basic y Java. Java, en particular, recibió mucha atención. Pero más radicales e innovadores que los lenguajes de RAD eran los nuevos lenguajes de script. Estos no descendían directamente de otros lenguajes y ofrecieron nuevas sintaxis e incorporación más liberal de otras características. Muchos consideran estos lenguajes de script más productivos que los lenguajes de RAD, aunque esto se debe a menudo a que es más difícil escribir y mantener largos programas que pequeños programas simples. Sin embargo, no es menos cierto que los programas de script llegaron para convertirse en los más prominentes en la conexión con la Web.

Algunos lenguajes importantes que se desarrollaron en este período son:

- 1990 - Haskell
- 1991 - Python
- 1991 - Visual Basic
- 1991 - HTML (lenguaje de marcado de hipertexto)
- 1993 - Ruby
- 1993 - Lua
- 1994 - CLOS (parte del ANSI Common Lisp)
- 1995 - Java
- 1995 - Delphi (Object Pascal)
- 1995 - JavaScript
- 1995 - PHP
- 1996 - WebDNA
- 1997 - Rebol
- 1999 - D

Tendencias actuales[editar]

La evolución de los lenguajes de programación continúa, tanto en la industria como en investigación. Algunas de las tendencias actuales incluyen:

- Aumentar el soporte para la programación funcional en lenguajes importantes utilizados comercialmente, incluida la programación funcional pura para hacer el código más fácil de razonar y de paralelizar (tanto en macro como en micro-niveles).
- Construir lenguajes para apoyar la programación concurrente y distribuida.
- Mecanismos para añadir al lenguaje verificación en cuanto a seguridad y confiabilidad: chequeo sintáctico extendido, control de flujo de información, seguridad de hilos.
- Mecanismos alternativos de modularidad: mixins, delegados, aspectos.
- Desarrollo de software orientado a componentes.
- Metaprogramación, la reflexión o el acceso al árbol de sintaxis abstracta.
- Mayor énfasis en cuanto a distribución y movilidad.

- Integración con bases de datos, incluyendo XML y bases de datos relacionales.
- Soporte para Unicode para que el código fuente (texto del programa) no se limite sólo a los caracteres contenidos en el conjunto de caracteres ASCII, permitiendo, por ejemplo, el uso de caracteres no latinos basados en guiones o signos de puntuación extendidos.
- XML para interfaz gráfica (XUL, XAML).
- El código abierto como una filosofía de desarrollo de lenguajes, incluyendo la colección de compiladores de GNU y lenguajes recientes, como Python, Ruby, y Squeak.
- Programación Orientada a Aspectos (AOP).
- Lenguajes que soporten programar sobre los procesadores de la GPU en paralelo, como OpenCL.

Algunos lenguajes importantes desarrollados durante este período incluyen:

- 2000 - ActionScript
- 2001 - C#
- 2001 - Visual Basic .NET
- 2002 - F#
- 2003 - Groovy
- 2003 - Scala
- 2003 - Factor
- 2005 - Scratch
- 2007 - Clojure
- 2009 - Go
- 2011 - Dart
- 2014 - Swift

Importancia de los Lenguajes de Programación

Los lenguajes de programación refieren a distintos tipos de expresiones y reglas de estructuración

lógica que sirven para generar taras recurrentes y sistemáticas. Los mismos son de gran importancia porque permiten generar distintos sistemas que sirven para tareas que satisfacen las necesidades de los usuarios. Existe una gran variedad de estos lenguajes y en cualquier caso los mismos se orientan a distintos tipos de requerimientos; no obstante, puede dividirse esta pluralidad en dos grandes grupos, los **lenguajes de programación** estructurada y los lenguajes orientados a objetos. Casi todo lo que podemos utilizar en informática se debe en buena medida a los **lenguajes de programación**.

Un ordenador permite realizar operaciones de cálculo a una gran velocidad; no obstante, el mismo es totalmente inservible si no fuera por los programas que se ejecutan en el mismo. Estos programas son de diversa índole, siendo quizá el **sistema operativo** el más importante de todos, sistema sobre el que corren las distintas **aplicaciones**. Para el desarrollo de los mismos siempre son necesarios los **lenguajes de programación**, cada uno de los cuales tiene **características** específicas para su implementación. Además cada lenguaje de programación se escribe a su vez con otro de más bajo nivel, permitiendo que exista una pluralidad de respuestas a las necesidades existentes.

Como hemos dicho existen dos grandes grupos de **lenguajes de programación**, los que tienen una programación estructurada y los que son orientados a objetos. En el primer caso el lenguaje consiste en un conjunto de instrucciones y órdenes como asimismo en un conjunto de reglas de aplicación delas mismas; la lógica proposicional tiene un rol fundamental en este caso y todas las operaciones pueden estar supeditadas a relaciones lógicas. En el caso de un lenguaje de programación orientado a objetos, en cambio, existe una intención de reflejar en el lenguaje distintas circunstancias de la existencia real; así, se hará referencia a objetos, a clases, a herencias, a atributos, etc.

Los **lenguajes de programación** también pueden dividirse entre lenguajes interpretados y lenguajes compilados; en el primer caso el lenguaje debe ser interpretado por otro programa, mientras que en el

segundo caso debe ser traducido mediante un compilador, creándose en el proceso un archivo que se denomina ejecutable.
El **conocimiento** de algún lenguaje de programación puede ser muy importante para desarrollar capacidades en lo que respecta a resolución de problemas y automatización de tareas. Ciertamente este tipo de saber sumado a algunas nociones básicas de **algoritmos** puede abrir todo un nuevo panorama de posibilidades laborales.

Python, el rey de los lenguajes de programación

Python está de moda. Y no es para menos. Este lenguaje de programación se ha convertido en muy poco tiempo en uno de los más populares.
A día de hoy, y según el índice TIOBE, su uso no ha parado de crecer y actualmente se encuentra en el tercer puesto de la lista, tan solo superado por Java y C.
Según TIOBE, si el crecimiento de Python sigue el mismo ritmo, es muy probable que en 3 ó 4 años desbanque tanto a Java como a C. Pero, ¿qué le hace tan especial?
En Paradigma, Python es uno de los lenguajes que más usamos en nuestros proyectos. Además, en nuestras formaciones internas y meetups, ha sido en muchas veces el tema estrella.
Recopilamos nuestro mejor contenido con este lenguaje como protagonista.

¿Es Python el lenguaje del futuro?

Python es uno de los lenguajes que más usamos en Paradigma junto con Java y Node.js. Desde hace años, hemos llevado a cabo exitosos proyectos desarrollados en este lenguaje.
Hace algunas semanas se publicó en el blog de Stackoverflow, la principal página de preguntas y respuestas sobre programación a nivel mundial, un interesante artículo reflexionando sobre el increíble crecimiento del uso del lenguaje de programación Python.

A raíz de este artículo, que ha tenido gran repercusión, queremos hacer un análisis más profundo sobre las causas que han llevado a Python a ser uno de los lenguajes más usados y qué perspectivas de futuro se plantean.

Los orígenes

El lenguaje Python surgió a principios de los 90 e inicialmente fue desarrollado por Guido Van Rossum, un ingeniero holandés que trabajaba en ese momento en el CWI de Amsterdam, el Centro de Investigación de Ciencias de la Computación holandés.

Python surgió como un hobby para Guido y su nombre, Python, fue tomado del grupo cómico británico Monty Python, del que Guido era un gran fan. Desde sus comienzos, Python nació como un proyecto de software libre y posiblemente deba parte de su éxito a la decisión de hacerlo código abierto.

Actualmente, la evolución del lenguaje Python es gestionada por la Python Software Foundation, una sociedad sin ánimo de lucro dedicada a dar difusión al lenguaje y apoyar su evolución. Guido sigue totalmente involucrado en el desarrollo y en la toma de decisiones de diseño.

Python está licenciado bajo licencia PSFL, derivada de BSD y compatible con GPL. Muchas empresas y organizaciones, como Google, Microsoft o Red Hat, hacen un gran uso de Python y tienen influencia en su evolución, pero ninguna ejerce un control sobre el mismo. Esto diferencia a Python de otros lenguajes.

Características diferenciales

Python tiene una serie de características que lo hacen muy particular y que, sin duda, le aportan muchas ventajas y están en la raíz de su uso tan extendido.

Python es un lenguaje multiparadigma, esto significa que combina propiedades de diferentes paradigmas de programación.

Principalmente es un lenguaje orientado a objetos, todo en Python es un objeto, pero también incorpora aspectos de la programación imperativa, funcional, procedural y reflexiva.

Una de las características más reseñables de Python es que es un lenguaje interpretado, esto significa que no se compila a diferencia de otros lenguajes como Java o C/C++, sino que es interpretado en tiempo de ejecución. Además, es de tipado dinámico, aunque opcionalmente desde la versión 3.5 podemos hacer uso de tipado estático.

Python es *cross plataforma*, es decir, podemos ejecutarlo en diferentes sistemas operativos como Windows o Linux simplemente usando el intérprete correspondiente.
Algunos le achacan a Python que es más lento en tiempo de ejecución que otros lenguajes compilados como Java o C/C++. Y es cierto, al tratarse de un lenguaje interpretado, Python es más lento.
Sin embargo, esto no es un gran problema, las diferencias en velocidad son pequeñas y hoy en día el cuello de botella en los proyectos de desarrollo de software no está en la CPU. Gracias a avances como la computación en la nube, hoy en día disponemos de gran capacidad de cómputo a un coste muy asequible. El desafío está en acortar los tiempos de desarrollo, mejorando la mantenibilidad y calidad del código. Python pone foco en esto, facilitando la vida a los desarrolladores.
Los principios de diseño del lenguaje están guiados por una serie de aforismos recogidos en el "Zen de Python". En estos principios podemos ver que la legibilidad del código y favorecer la simplicidad del mismo son partes esenciales del diseño del lenguaje desde el principio. Estas ideas han ayudado mucho a que la curva de aprendizaje de Python sea baja respecto a otros lenguajes.

Python como lenguaje de scripting

Tradicionalmente Python ha tenido un uso muy extendido como herramienta de scripting, sustituyendo a scripts escritos en bash, otros lenguajes de script más limitados o herramientas como AWK o sed. Por ello, Python siempre ha sido un buen compañero de los administradores de sistemas y los equipos de operaciones.
Hoy en día, muchas de las herramientas punteras para gestión de despliegues e infraestructura usan o se basan en Python. Algunas de las más destacadas son Ansible, Salt o Fabric.
Otra área en la que Python es pionero es en el mundo del scraping y el crawling, donde podemos extraer información de páginas web gracias a técnicas de "scraping", herramientas de Python como Scrapy son muy usadas en este contexto.

Python en el desarrollo web

Otro de los campos en los que Python ha brillado en los últimos años es en el desarrollo de aplicaciones web, principalmente gracias a

frameworks de desarrollo web muy potentes como Django, un framework completo o Flask, un microframework.
Sin embargo, en el ecosistema de desarrollo web existen muchas alternativas y frameworks muy maduros y asentados como Symfony para PHP, Spring para Java, Grails para Groovy o Rails para Ruby. Todos estos frameworks están continuamente tomando ideas entre ellos, inmersos en ofrecer las mejores alternativas para los desarrolladores.

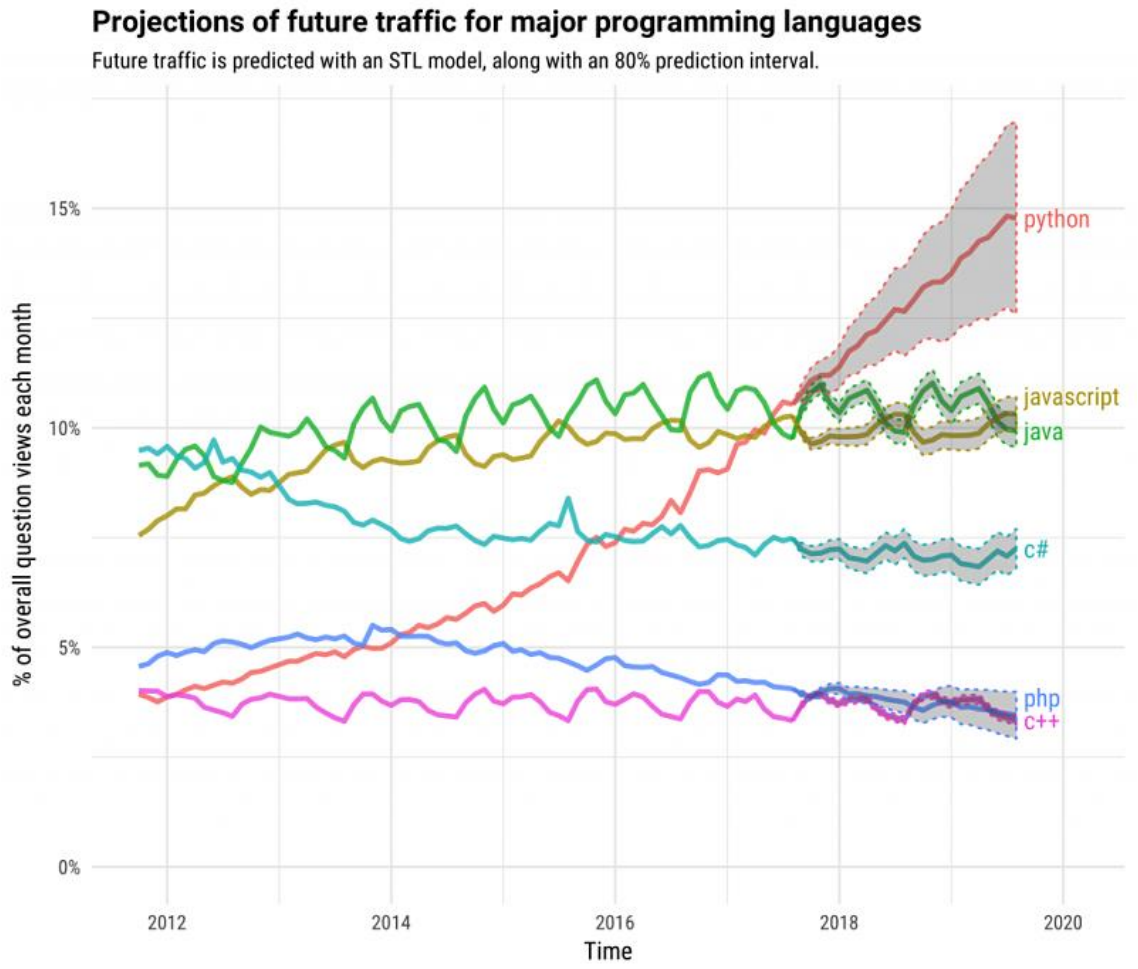
Big Data, Data Science, AI: el boom de Python

Sin embargo, al margen de todas las bondades que hemos comentado del lenguaje, en los últimos años ha ocurrido algo que ha revolucionado y extendido radicalmente el uso de Python.
La generalización del Big Data en los últimos años, seguida de la explosión de la Inteligencia Artificial, Machine Learning, Deep Learning y el surgimiento de la ciencia de datos o data science como un nuevo área de trabajo con especialistas propios, ha revolucionado el panorama.
Y es que muchas de las nuevas herramientas que han surgido, y que son explotadas por los ingenieros de datos y los científicos de datos, han sido desarrolladas en Python o nos ofrecen Python como la forma predilecta de interactuar con ellas.
Podemos hablar de tecnología para Big Data como PySpark, de herramientas para Data Science como Pandas, NumPy, Matplotlib o Jupyter. De herramientas del procesamiento del lenguaje natural como NLTK, y por último el área de machine learning que tanto interés está despertando con herramientas como Tensorflow, MXNet o scikit-learn.

Conclusión

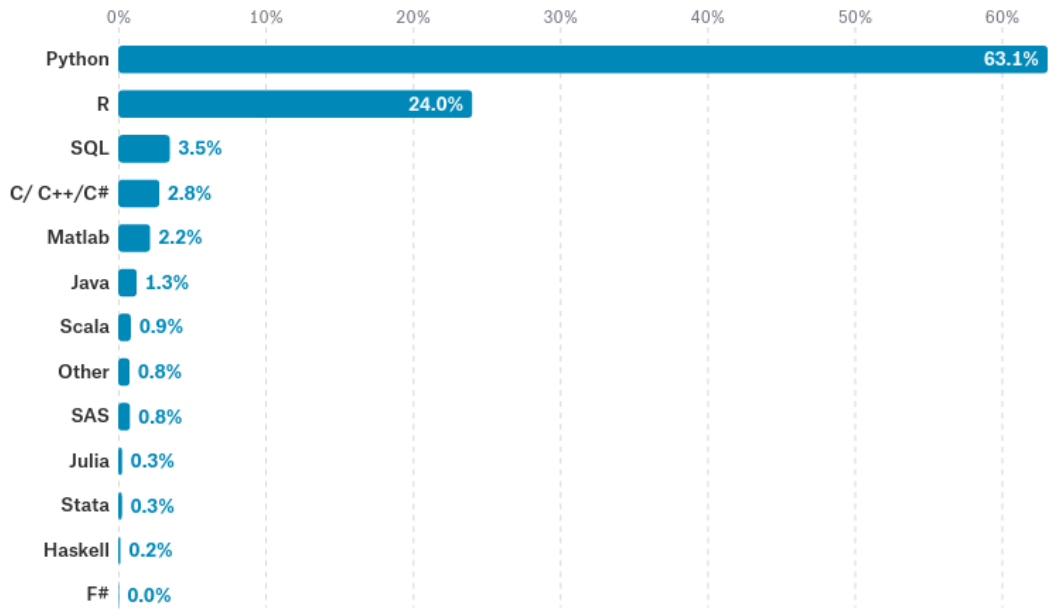
Podemos afirmar que Python es un lenguaje maduro, con una gran base de desarrolladores, documentación y proyectos en producción.
El crecimiento en el uso del lenguaje está siendo espectacular gracias, fundamentalmente, a las nuevas tecnologías de Data Science y Machine Learning, donde junto con el lenguaje R es el rey.
Sin embargo, R es un lenguaje más de nicho que proviene del mundo de la estadística. Python, por otro lado, es un lenguaje de propósito general y su uso está mucho más extendido.

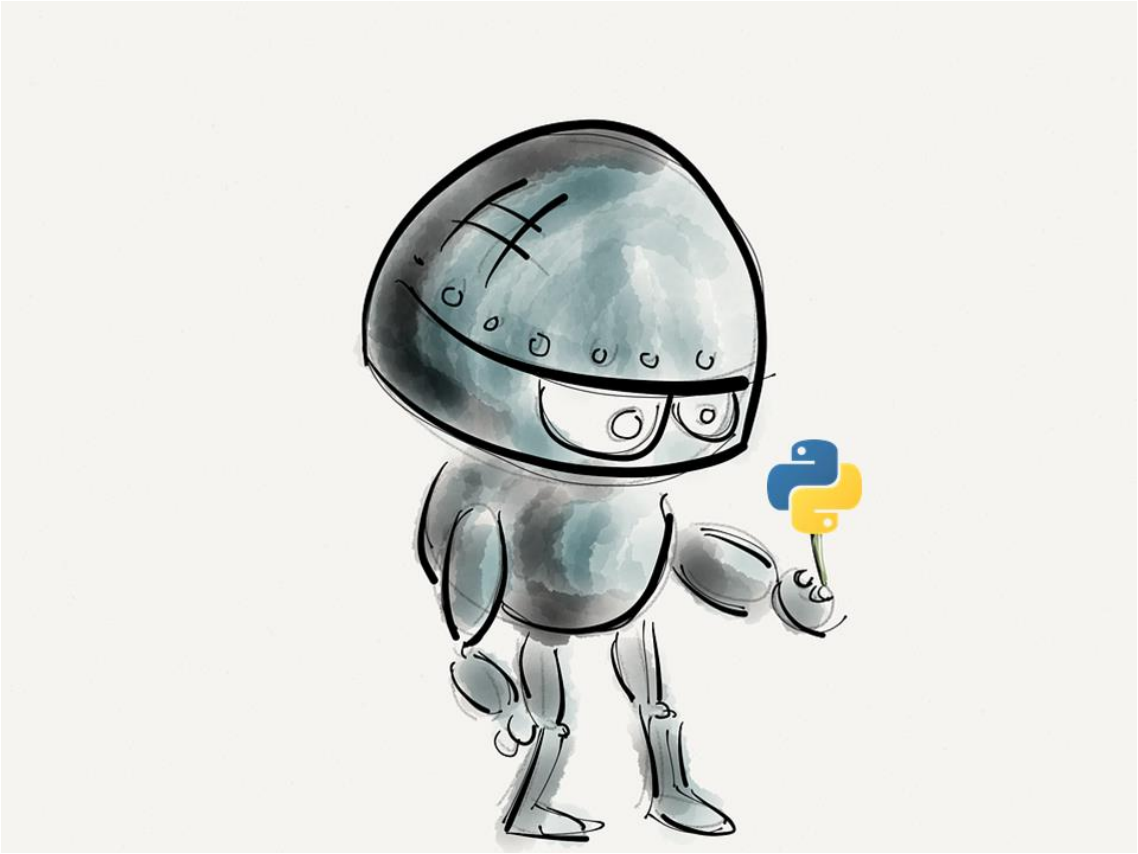
En la siguiente gráfica vemos una proyección para los próximos años de Stackoverflow sobre el número de visitas que espera recibir en función de los principales lenguajes de programación.



Fuente: Stackoverflow

Otro índice muy relevante, el PYPL, basado en la popularidad de los lenguajes de programación en Google, sitúa a Python como el segundo con un crecimiento del 10% en los últimos 5 años.
Otros rankings como el del IEEE a mediados de 2017, también sitúan a Python en los primeros puestos.
Un dato muy significativo es la respuesta a la pregunta: *¿Qué lenguaje de programación recomendaría a los nuevos científicos de datos aprender primero?* Más del 63% de los encuestados respondió Python.


0%
10%
20%
30%
40%
50%
60%
Python
63.1%
R
24.0%
SQL
3.5%
C/ C++/C#
2.8%
Matlab
2.2%
Java
1.3%
Scala
0.9%
Other
0.8%
SAS
0.8%
Julia
0.3%
Stata
0.3%
Haskell
0.2%
F#
0.0%



Guido Van Rossum, desarrollador del lenguaje Python

¿Qué es Python?

La sintaxis de Python es muy fácil de usar, de ahí que se ha ido haciendo cada vez más popular en los años recientes.

Es un lenguaje de programación, con grandes similitudes con el lenguaje Perl (en principio se cogió de referencia el *lenguaje ABC*, que se utiliza en las escuelas).

Hay lenguajes que siguen siendo muy populares, como **Java** o **C++**, pero Python se está usando mucho más en varios campos, y sobretodo está ganando popularidad para la **programación de páginas html**, gracias a **Django**.

El creador del lenguaje de programación Python es Guido van Rossum, al final de la década de los 80 (en el año 91 se publicó la primera versión). ¿Para que fue creado?

Cuando se programa en entornos Unix, se utiliza sobretodo el lenguaje C, y este sistema no es el más fácil para desarrollar del mundo, así que lo inventó para que esas tareas de programación en éstos sencillas fueran mucho más sencillos.

¿De dónde viene la palabra Python? Aunque todos crean que puede ser por la serpiente, la pitón, en realidad es por uno de los hobbies de quién lo creó: a los humoristas Monty Python. Así que si un día te inventas un lenguaje, ya tienes una referencia 😉

Características del lenguaje de programación Python

Vamos a ver las principales propiedades del lenguaje Python, que son muy similares a lenguajes programación como Java o Ruby.

Lenguaje de propósito general

Eso significa que no está orientado a un fin concreto, como puede ser PHP, pensado sobretodo para hacer páginas de internet.

Con Python podrás crear páginas sin tener un alto conocimiento (con Javascript como un poderoso aliado), pero también hacer scripts o software para el sistema operativo Windows.

Aún no hay nada destacado para dispositivos móviles, pero se puede usar Kivy para este propósito.

Es multiparadigma

¿Y qué significa eso? ¿**Multiparadigma**?

Pues aunque su fuerte sea la programación orientada a objetos (es un lenguaje de alto nivel), existen otros paradigmas o estilos de programación para sus usuarios, como es la programación imperativa (con sentencias de bucle) o la programación funcional (con módulos y funciones).

Así que si no sabes nada de objetos y sólo sabes escribir código mediante métodos, puedes usar Python perfectamente, cosa que en otros lenguajes **hacer** eso es imposible.

Python es un lenguaje interpretado

Cuando programamos en Python, no compilamos el código fuente a código máquina, sino que hay un intérprete que es el que ejecutará el programa basándose en el código directamente.

Aunque esta propiedad hace pensar que los programas puedan ser más lentos, que en lenguaje Python no suele ser así, eso facilita el desarrollo para la siguiente característica.

Es multiplataforma

Al contrario que muchos lenguajes como **visual basic**, que principalmente solo puedes hacer cosas para Windows, con Python tienes la posibilidad de usarlo en muchos dispositivos y sistemas operativos, ya que se han creado intérpretes para Unix, Linux, Windows y sistemas Mac Os.

Es de tipado dinámico

Cuando declaramos una variable, no es necesario decirle de que tipos son los datos (si es **int**, **string**, **float**, etc.). La variable se adapta a lo que escribimos cuando se ejecuta el programa.

Antes esta característica siempre ha sido criticada en otros lenguajes, por la optimización de la memoria, errores a la hora de escribir código, etc. pero con Python el objetivo es que el lenguaje ayude a la creación de software, no tener que lidiar con peculiaridades propias del lenguaje.

Igualmente, Python es fuertemente tipado, por ejemplo, no podrás sumar números y texto (una variable del tipo int con una de tipos cadenas) porque daría error.

```
1  import random
2
3  def buscarElemento(lista, elemento):
4      for i in range(0,len(lista)):
5          if(lista[i] == elemento):
6              return i
7
8  def imprimirLista(lista,nombre):
9      for i in range(0,len(lista)):
10         print nombre + "[" + str(i) + "]=" + str(lista[i])
11
12 def leerLista():
13     lista=[]
14
15     i=0
16     while i < 10:
17         lista.append(int(random.randint(0, 10)))
18         i=i+1
19     return lista
20
21 A=leerLista()
22 imprimirLista(A,"A")
23 cn=int(raw_input("Numero a buscar: "))
24 print "A[" + str(buscarElemento(A,cn)) + "]"
```

Es orientado a objetos

Ya hemos dicho que podemos aplicar otro estilo de programación, hacer software orientado a objetos conlleva una serie de ventajas estándar, sobretodo a la hora de reutilizar los componentes gracias a la herencia y sus funciones de polimorfismo.

Otras propiedades de Python

Vamos a nombrar brevemente otras funciones o elementos propios de Python, como pueden ser:

- De *libre distribución*.
- Gracias a su popularidad, existen una gran cantidad de *librerías y funciones* ya hechas, que podemos utilizar gracias a su extensa biblioteca.
- Tiene soporte para múltiple variedad de *bases de datos*.
- Tiene un gran *soporte* gracias a su comunidad. Por ejemplo, la última versión de Visual Studio te permite desarrollar en Python, o la comunidad de la página oficial de Python, dónde vemos todas las actividades que hacen en el mundo.

Programas hechos con Python

Ahora vamos a nombrar algunos programas famosos que están hechos con Python, como por ejemplo:

- Calibre: el mejor gestor de e-books para todos los usuarios.
- GNU MailMan: un programa para manejar listas de correo.
- BitTorrent: programa para compartir ficheros de tipo torrent estándar.
- Odoo (antes OpenERP): un ERP y mucho más para la gestión de empresas, de software libre.

Aunque hemos hablado de programas, Python se usa en webs conocidas, como Youtube y software hecho por Google.

Y si te preguntas *¿Dónde puedo aprender Python?*, con el tutorial en español de Python y el curso de Udemy podrás entender y dominar este lenguaje que tiene más importancia en el desarrollo de aplicaciones.