

Python

Introducción.

Python (<http://python.org>) es un lenguaje de programación de alto nivel, interactivo e interpretado, creado por Guido Van Rossum en 1991. El nombre del lenguaje es en honor al programa cómico de la televisión británica llamado "Monty Python Flying Circus".

Es de código abierto, multi-plataforma y se adecua a diversos paradigmas de programación (programación funcional, programación orientada a objetos, programación procesal, etc.).

Python se ha vuelto muy popular ya sea como un primer lenguaje de programación o como el lenguaje seleccionado por defecto por diversos proyectos de primer nivel tales como Openstack, Blender, Google App Engine, Django, Jupyter, etc.

Características de Python.

A lo largo de estos cursos se explorarán y aprovecharán las características que hacen de Python un lenguaje tan popular y poderoso.

- Sintaxis muy clara y legible.
- Fuerte capacidad de introspección.
- Orientación a objetos intuitiva.
- Expresión del código procedimental.
- Altamente modular, soporta paquetes jerárquicos.
- Enfocado en el uso de excepciones para el manejo de errores.
- Tipos de datos dinámicos de muy alto nivel.
- Extensa biblioteca estándar (<https://docs.python.org/3/library/>) (STL) y módulos de terceros para prácticamente todas las tareas.
- Extensiones y módulos fácilmente escritos en C, C++ (o Java para Jython, o .NET para IronPython).
- Integrable dentro de las aplicaciones como una interfaz de scripting.

Aplicaciones de Python.

Al ser un lenguaje multipropósito y altamente portable, Python se ha utilizado para desarrollar:

- Aplicaciones de escritorio.
- Aplicaciones web.
- Análisis de datos.
- Administración de servidores.
- Seguridad y análisis de penetración.
- Cómputo en la nube.
- Cómputo científico.
- Análisis de lenguaje natural.
- Visión artificial.
- Animación, videojuegos e imágenes generadas por computadora.
- Aplicaciones móviles *.
- Internet de las cosas (IoT).
- DevOps.

Python 2 y Python 3.

Debido a la continua evolución y revisión del lenguaje de programación, actualmente existen dos versiones de Python cuyo código no es compatible.

Python 3 es una versión revisada del lenguaje, la cual fue publicada en 2009 y que incluye modificaciones y mejoras que lo hacen incompatible con código de versiones previas; mientras que Python 2 es una versión que es compatible con código antiguo.

Python 2.7 es una versión "de transición" que permite usar algunos elementos sintácticos tanto de Python 2 como de Python 3.

Python 3 presenta mejoras notables con respecto a Python 2, sin embargo aún existe mucho código "heredado" compatible con Python 2.

Se prevé que la publicación y soporte de nuevas versiones de Python 2 cese a partir del año 2020.

NOTA: En este documento se utilizará código escrito en Python 3, pero se discutirá sobre las diferencias entre cada versión cuando se considere pertinente.

Implementaciones.

Python no sólo ha sido portado a diversos sistemas operativos, sino que existen implementaciones específicas del lenguaje. Algunas implementaciones más populares de Python son:

- CPython o es la implementación común de Python.
- [PyPy \(http://pypy.org/\)](http://pypy.org/) es una implementación de Python enfocada en optimizar su velocidad de ejecución y eficiencia en el uso de la memoria entre otras cosas.
- [Jython \(http://www.jython.org/\)](http://www.jython.org/) es una implementación de Python sobre la máquina virtual de Java (JVM).
- [IronPython \(http://ironpython.net/\)](http://ironpython.net/) es una implementación de Python para .NET.
- [MicroPython \(https://micropython.org/\)](https://micropython.org/) es una implementación de Python para microcontroladores (ESP32 y ESP8266).

Distribuciones.

Python ya viene con una gran biblioteca estándar, sin embargo existen algunas "distribuciones" que pretenden extender al lenguaje con propósitos particulares. Es posible consultar las diversas distribuciones de Python en <https://wiki.python.org/moin/PythonDistributions> (<https://wiki.python.org/moin/PythonDistributions>).

Instalación.

Las principales distribuciones de GNU/Linux, los sistemas *BSD, así como Mac OS X y la mayoría de los UNIX vienen al menos con Python 2 preinstalado. Del mismo modo, las principales distribuciones de GNU/Linux cuentan con paquetes de instalación de Python 3.

Las versiones más recientes de Python pueden ser descargadas desde el sitio principal, <https://python.org> (<https://python.org>), incluyendo binarios para Mac OS X y Windows e incluso es posible descargar el código fuente.

Nota: [Anaconda](https://www.anaconda.com/distribution/) (<https://www.anaconda.com/distribution/>) es una distribución de Python 2 y Python 3 especializada en cómputo científico, sin embargo es de muy fácil instalación y gestión tanto en Windows como en Mac OS X y GNU/Linux. Es una alternativa muy recomendable a las versiones oficiales de Python.

Breve introducción a los lenguajes de programación.

Un lenguaje es un conjunto de cadenas de símbolos con los que se pueden crear mensajes. De ese modo los mensajes son transmitidos de un emisor a un receptor. Aún cuando en la naturaleza se pueden identificar ciertos lenguajes, los seres humanos hemos desarrollado lenguajes de diversos tipos y gran complejidad.

Los lenguajes constan principalmente de la gramática, la cual trata sobre la construcción del lenguaje, y la semántica, la cual trata sobre el significado del lenguaje.

A su vez, la gramática consta de:

- Morfología: cómo se construyen las notaciones (género, tiempos, declinaciones).
- Sintaxis: cómo se deben escribir las notaciones (orden, estructura).

Durante el siglo XX, científicos como [Alan Turing](https://en.wikipedia.org/wiki/Alan_Turing) (https://en.wikipedia.org/wiki/Alan_Turing) y [Alonzo Church](https://en.wikipedia.org/wiki/Alonzo_Church) (https://en.wikipedia.org/wiki/Alonzo_Church) fundaron las bases del cómputo, la programación y sus lenguajes. Los lenguajes de programación actuales, a diferencia de los lenguajes humanos tienen una morfología rígida y simplificada con el fin de ejecutar instrucciones específicas en los sistemas de cómputo.

Lenguajes de alto y bajo nivel.

Los lenguajes de bajo nivel constan de un conjunto básico de instrucciones que son ejecutados directamente por la unidad de procesamiento de un sistema de cómputo, tal como es el caso del lenguaje ensamblador. Dichos lenguajes están ligados intrínsecamente al tipo de procesador que los ejecuta y resultan ser muy complicados de elaborar e interpretar por las personas.

Por su parte, los lenguajes de alto nivel son más accesibles para el ser humano e incluso menos dependientes del tipo de hardware, pero deben de ser a su vez traducidos a lenguaje de bajo nivel.

Lenguajes compilados e interpretados.

Los lenguajes de alto nivel interactúan con los sistemas de cómputo de dos formas.

- Mediante un compilador, el cual traduce el código de un programa a lenguaje de bajo nivel, dando por resultado un "archivo binario" el cual es susceptible de ser ejecutado.
- Mediante un intérprete, el cual ejecuta de inmediato las instrucciones que se ingresan.

Por lo general los lenguajes compilados son más rápidos y consumen menos recursos que los lenguajes interpretados en vista de que el archivo resultante es código de bajo nivel, mientras que los lenguajes interpretados deben seguir un proceso a través de varios niveles de abstracción hasta que las instrucciones son ejecutadas por el sistema.

Python es un lenguaje interpretado de alto nivel.

Entornos interactivos, guiones y 'Hola Mundo'.

Debido a que Python es un lenguaje interpretado, es posible utilizarlo mediante un entorno interactivo (shell) o mediante el uso de guiones (scripts).

El entorno interactivo.

El entorno interactivo (shell) de Python se ejecuta desde una terminal de texto.

En el caso de los sistemas basados en UNIX, como GNU/Linux *BSD y Mac OS X, es necesario abrir una terminal de texto e invocar el shell de la siguiente manera:

```
python
```

Es común que el intérprete de Python 2 y el de Python 3 se encuentren instalados en el mismo sistema. En ese caso, es común que el intérprete de Python 3 se invoque mediante la siguiente sintaxis.

```
python3
```

En el caso de Windows, se selecciona el lanzador de Python, el cual abrirá una terminal de texto corriendo el entorno interactivo.

El entorno interactivo se verá de forma similar a la siguiente:

```
Python 3.4.3+ (default, Oct 14 2015, 16:03:50)
[GCC 5.2.1 20151010] on linux

Type "help", "copyright", "credits" or "license" for more information.
>>>
```

NOTA: Para salir del entorno interactivo se debe teclear `exit()` y oprimir la tecla Intro.

'Hola Mundo' en el entorno interactivo.

Para desplegar el texto Hola Mundo desde el entorno interactivo sólo es necesario teclear lo siguiente:

```
>>> print("Hola, Mundo.")
```

Y el resultado es el siguiente:

```
Hola, Mundo.
>>>
```

'Hola Mundo' en la notebook de Jupyter.

Las notebooks de Jupyter funcionan mediante *kernels* los cuales son conexiones a un intérprete del lenguaje correspondiente.

Para ejecutar un comando de Python en una notebook de Jupyter con el kernel de Python 3, sólo es necesario escribir el código en una terminal de tipo `Code` y ejecutarla.

6to nivel A

Ejemplo:

```
In [1]: 1 print("Hola, Mundo.")
```

Hola, Mundo.

Ejecución de guiones (scripts).

Los scripts de Python son archivos de texto que contienen código que será leído y ejecutado por el intérprete de Python. Dichos archivos tienen la extensión `.py`. Algunos editores de texto cuentan con reconocimiento de la sintaxis de Python y utilizan colores para identificar sus componentes.

El script `codigo/holamundo.py`.

El documento `codigo/holamundo.py` ([codigo/holamundo.py](#)) contiene el siguiente código.

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

Ejecución mediante el intérprete.

Para ejecutar el script, es necesario que el intérprete acceda al archivo con una sintaxis similar a:

```
python <script>
```

Ejemplo:

Las notebooks de Jupyter permiten enviar comandos al sistema anteponiendo un signo de admiración `!` antes del comando.

- La siguiente celda ejecutará el script `codigo/holamundo.py` usando el intérprete de Python.

Nota: La celda sólo se ejecutará si el intérprete de Python esté en una ruta por defecto del sistema. En algunos casos el intérprete es `python3`.

```
In [2]: 1 !python codigo/holamundo.py
```

Hola, Mundo.

Ejecución de un script en Jupyter.

Jupyter puede ejecutar scripts mediante el "comando mágico" `%run`.

Más adelante se discutirán los comandos mágicos de IPython.

```
In [3]: 1 %run codigo/holamundo.py
```

Hola, Mundo.

Ejecución directa de un script en entornos basados en UNIX.

La ejecución directa del script en entornos basados en UNIX requiere que se indique la ruta en la que se encuentra el intérprete de Python mediante el texto:

```
#!/usr/bin/env python
```

Esta línea no afecta la ejecución del script en Windows, por lo que se recomienda incluirla siempre.

Si se desea ejecutar cualquier script en entornos basados en UNIX, es necesario que el script cuente con los permisos necesarios. Para asignar permisos de ejecución a un archivo en *NIX se utiliza el comando:

```
chmod +x <ruta del archivo>
```

Ejemplo:

- En caso de que esta notebook se encuentre corriendo en un sistema basado en UNIX como GNU/Linux o Mac OS X, la siguiente comando en la terminal asignará permisos de ejecución al script `codigo/holamundo.py`

Advertencia: Las siguientes celdas no funcionarán en Windows.

In [4]: 1 `!chmod +x codigo/holamundo.py`

"chmod" no se reconoce como un comando interno o externo, programa o archivo por lotes ejecutable.

- La siguiente celda se ejecutará en un entorno basado en UNIX si existe la ruta a un interprete configurado en la variable de entorno `PATH`.

In [5]: 1 `!codigo/holamundo.py`

"codigo" no se reconoce como un comando interno o externo, programa o archivo por lotes ejecutable.

Ejecución en Windows.

En el caso de Windows, el sistema relacionará a los archivos con la extensión `.py` con el intérprete de Python, por lo que con hacer doble click en el archivo, éste se ejecutará y tan pronto termine, cerrará la terminal.

Cuando se ejecute el script `holamundo.py` en Windows, se abrirá y cerrará una terminal de forma casi inmediata.

Ejemplo:

- La siguiente celda se ejecutará en un entorno basado en Windows si existe la ruta a un interprete de Python configurado.

Advertencia: Las siguientes celdas no funcionarán en entornos distintos a Windows.

In [6]: 1 `!codigo\holamundo.py`

El script `codigo/holamundo2.py`.

El script `codigo/holamundo2.py` ([codigo/holamundo2.py](#)) es similar a `codigo/holamundo2.py`, pero incluye una instrucción que no terminará su ejecución hasta que se oprima la tecla `Intro`.

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

In [7]: 1 %run codigo/holamundo2.py

Hola, Mundo.
Hola 6to A

Codificación de caracteres .

Python 3 utiliza la codificación [UTF-8](https://www.utf8-chartable.de/unicode-utf8-table.pl) (<https://www.utf8-chartable.de/unicode-utf8-table.pl>) por defecto.

Python 2 utiliza por defecto una codificación [ASCII](http://www.asciitable.com/) (<http://www.asciitable.com/>), por lo que desplegar caracteres especiales como la "ñ" y los acentos generan un mensaje de error.

Codificación *UTF-8* en Python 2.

Para indicarle al intérprete de Python 2 que utilice la codificación UTF-8 se debe incluir la siguiente línea al principio el script:

```
# -*- coding: utf-8 -*-
```

Es muy recomendable definir siempre el tipo de codificación UTF-8 en los scripts hechos para Python 2.

El script *hola_ninos.py* es un ejemplo de código en Python 2.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
print "Hola niños"
input()
```

Cabe hacer notar que el código de este script no es compatible con Python 3, ya que *print* requiere del uso de paréntesis en dicha versión.

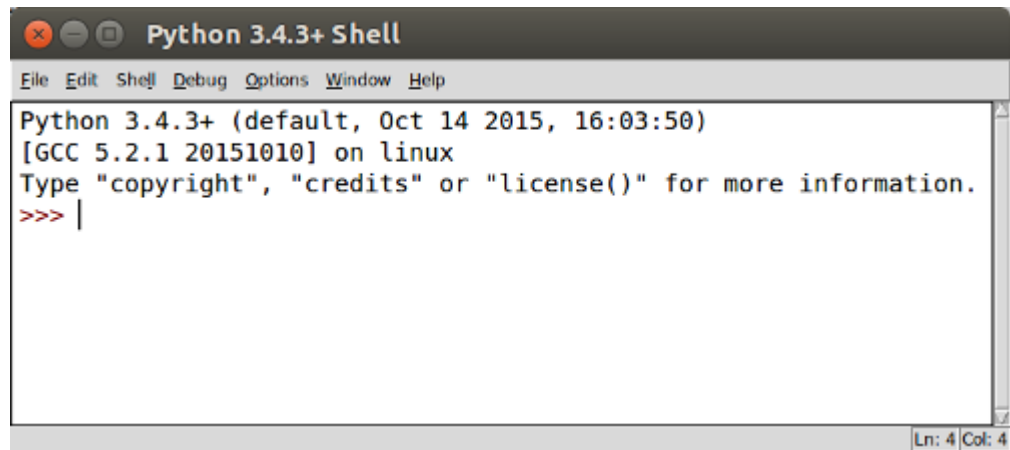
Entornos de Desarrollo Integrado.

Un Entorno de Desarrollo Integrado (IDE) es un conjunto de herramientas e interfaces que facilitan la programación y que están integrados por algunos de los siguientes elementos:

- Editor.
- Depurador.
- Compilador (no es necesario para Python).
- Interfaz de ejecución.
- Gestor de componentes.
- Control de versiones.

IDLE.

IDLE es el IDE que viene preinstalado con Python, el cual se ejecuta desde una ventana en el Escritorio de un entorno gráfico.



```
Python 3.4.3+ Shell
File Edit Shell Debug Options Window Help
Python 3.4.3+ (default, Oct 14 2015, 16:03:50)
[GCC 5.2.1 20151010] on linux
Type "copyright", "credits" or "license()" for more information.
>>> |
```

Existen varios editores e IDE que pueden soportar a Python. Algunos de ellos son:

- [Eclipse](https://www.eclipse.org/) (<https://www.eclipse.org/>).
- [XCode](https://developer.apple.com/xcode/) (<https://developer.apple.com/xcode/>).
- [Visual Studio Code](https://code.visualstudio.com/) (<https://code.visualstudio.com/>).
- [NetBeans](https://netbeans.org/) (<https://netbeans.org/>).
- [Atom](https://atom.io/) (<https://atom.io/>).
- [Sublime text](https://www.sublimetext.com/) (<https://www.sublimetext.com/>).
- [Komodo IDE](https://www.activestate.com/products/komodo-ide/) (<https://www.activestate.com/products/komodo-ide/>).
- [Vim](https://www.vim.org/) (<https://www.vim.org/>).
- [Emacs](https://www.gnu.org/software/emacs/) (<https://www.gnu.org/software/emacs/>).
- [PyCharm](https://www.jetbrains.com/pycharm/) (<https://www.jetbrains.com/pycharm/>).
- [Wing IDE](https://wingware.com/) (<https://wingware.com/>).
- [Ninja IDE](http://ninja-ide.org) (<http://ninja-ide.org>).
- [Spyder IDE](https://www.spyder-ide.org/) (<https://www.spyder-ide.org/>).

La comunidad de Python.

Python es un lenguaje que ha crecido gracias a una amplia y participativa comunidad. Aún cuando Guido Van Rossum tiene la última palabra en el desarrollo de Python, todos los comentarios son bienvenidos y existen cientos de proyectos en torno a este lenguaje de programación.

A los programadores de Python se le conoce como "pythonistas" y con el tiempo han desarrollado todo un compendio de mejores prácticas y códigos de estilo para programar en este lenguaje.

El Zen de Python.

El *Zen de Python* es una lista de aforismos que indican de forma general lo que se espera del código hecho en Python, y ha sido reconocido de tal forma por la comunidad que puede ser consultado desde el intérprete de Python de la siguiente manera.

In [1]:

```
1 import this
2
```

The Zen of Python, by Tim Peters

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Aquí la traducción de los aforismos.

- Bello es mejor que feo.
- Explícito es mejor que implícito.
- Simple es mejor que complejo.
- Complejo es mejor que complicado.
- Plano es mejor que anidado.
- Escaso es mejor que denso.
- La legibilidad cuenta.
- Los casos especiales no son tan especiales como para romper las reglas.
- Aunque lo práctico le gana a la pureza.
- Los errores no deben pasar en silencio.
- A menos que sean silenciados explícitamente.
- En caso de ambigüedad, resiste la tentación a adivinar.
- Debería haber una --y preferiblemente sólo una-- manera obvia de hacer las cosas.
- Aunque esa manera no parezca ser tan obvia a menos que seas Holandés.
- Ahora es mejor que nunca.
- Aunque nunca es normalmente mejor que ahora "mismo".
- Si la implementación es difícil de explicar, es una mala idea.
- Si la implementación es fácil de explicar, podría ser una buena idea.
- Los Espacios de Nombres son una magnífica idea -- ¡Hagamos más de esos!

Las PEP de Python.

Es muy probable que algún pythonista comparta alguna inquietud o tenga alguna propuesta relacionada con algún aspecto de Python y es por eso que existe un espacio que compendia una inmensa cantidad de propuestas e ideas relacionadas con el lenguaje.

[PEP \(http://www.python.org/dev/peps/\)](http://www.python.org/dev/peps/) es el acrónimo de "Propuesta de mejora de Python" y es el sitio en el que se puede proponer, consultar e incluso debatir sobre todas estas inquietudes de forma abierta y directa con los desarrolladores del lenguaje.

La [PEP-8 \(https://www.python.org/dev/peps/pep-0008/\)](https://www.python.org/dev/peps/pep-0008/) es particularmente interesante, ya que se refiere a las reglas de estilo que se sugieren para programar en Python.

El índice de paquetes de Python (PyPI).

Python es un lenguaje que viene con una gran biblioteca estándar, sin embargo existe una biblioteca en línea aún más grande y que sirve como repositorio de diversos proyectos publicados por sus desarrolladores para uso de la comunidad. El sitio <https://pypi.org> (<https://pypi.org>) no sólo enlista a dichos proyectos, sino que además cuenta con `pip`, una herramienta que puede instalar de forma local los paquetes correspondientes a uno o diversos proyectos contenidos en PyPI.



(<http://creativecommons.org/licenses/by/4.0/>)

Esta obra está bajo una [Licencia Creative Commons Atribución 4.0 Internacional](http://creativecommons.org/licenses/by/4.0/)

(<http://creativecommons.org/licenses/by/4.0/>).

© José Luis Chiquete Valdivieso. 2019.