Facultad de Ciencias de la Administración

TECNICATURA UNIVERSITARIA EN DESARROLLO WEB

# PROGRAMACIÓN I

## Unidad II – Python

Características. Historia. Versiones

Tecnicatura Universitaria en Desarrollo Web

Facultad de Ciencias de la Administración

Universidad Nacional de Entre Ríos

- **Objetivos**
  - Conocer las principales características de Python.
  - Realizar una instalación completa del ambiente de desarrollo.
  - Ejecutar programas simples.
- **Temas a desarrollar:**
  - **Python.**
    - Historia. Características. Versiones.
  - **Instalación.**
    - Entornos de desarrollo y editores de código. Ejemplos.
  - **Ejecución de programas.**
    - Sentencias simples y definición de Comentarios.

# Python

- **Python** es un lenguaje de programación interpretado de alto nivel.
- En 1989, **Guido van Rossumm** (holandés de 24 años) comenzó como hobby el desarrollo de **Python**, con el objetivo de mejorar la interfaz de usuario del Sistema Operativo Amoeba.
- En un principio, **Python** iba a ser un lenguaje de programación pequeño que sucedería al lenguaje **ABC** que desarrollaban en **CWI** (Centrum Wiskunde & Informatica) (instituto donde se desempeñaba **van Rossum**) incorporando algunas características adicionales y que ayudara a interactuar mejor con el sistema operativo.
- La primera versión de **Python** fue lanzada en febrero de 1991 con el número de versión 0.9.0.
- El nombre del lenguaje proviene de la afición que tenía **van Rossum** a la serie de televisión ***Monty Python´s Flying Circus*** y no de algo relacionado con el mundo de los reptiles.
  - Entre 2005 y finales de 2012 **van Rossum** trabajó en Google, entre otros proyectos, contribuyendo en el desarrollo de **Python**.
  - Entre 2012 y 2019 en Dropbox.
  - En 2019 se jubiló y en 2020 volvió a trabajar incorporándose a Microsoft.

# Características - Ventajas

- **Python** es un lenguaje de alto nivel, de propósito general, multiparadigma principalmente imperativo, orientado a objetos y funcional, de tipado dinámico y fuertemente tipado a nivel de lenguaje de programación.
- Ventajas:
  - **Sintaxis sencilla, simple y clara:** permite desarrollar programas de forma ***intuitiva***.
    - Esta característica hace que leer un programa escrito en **Python** sea muy parecido a leer un texto anglosajón.
    - Ejemplos que motivan la claridad son:
      - Los bloque lógicos se definen utilizando ***identación*** en vez de utilizar caracteres de apertura y cierre de bloque como **{** y **}** ( usado en lenguajes como Java, JavaScript o C).
      - Las expresiones simples no necesitan uso de paréntesis como pasa en lenguajes como JavaScript.
      - Para la separación de instrucciones se utiliza el ***salto de línea*** en vez del comúnmente utilizado carácter ";", aunque también permite utilizarlo.
      - Posee un ***sistema de recolección de basura*** que permite que el desarrollador se despreocupe de la gestión de memoria y que el lenguaje se pueda centrar en otros aspectos de alto nivel.

# Características – Ventajas (2)

- **Interpretado**: significa que no es necesario compilar los programas cada vez que se hace un cambio en el código.
  - Esto presenta una gran ventaja frente a los lenguajes compilados (C o C++) y ***aumenta considerablemente la velocidad de desarrollo de aplicaciones***.
  - Por otro lado, el ser un lenguaje interpretado permite que el código sea *independiente del hardware* en el que se ejecuta, y ayuda a que el lenguaje sea multiplataforma gracias al uso de su máquina virtual.
- **Baterías incluidas:** posee multitud de herramientas en la librería estándar que ayudan a realizar un sinfín de aplicaciones sin necesidad de utilizar bibliotecas de terceros.
  - **Python** también permite la **integración con otros lenguajes de programación**, ya sea importando código dentro de otros lenguajes o permitiendo ejecutar código de otros lenguajes en **Python**.
  - Así, podemos tener código **Python** ejecutando código C, C++, .Net o Java, y viceversa. Usando diferentes técnicas, el código **Python** se puede transcompilar en otro lenguaje (como JavaScript) u otros lenguajes pueden ejecutar código **Python** haciendo uso de subprocesos u otras técnicas.

# Características - Ventajas (3)

- **Multiplataforma:** lo que permite que se pueda ***ejecutar y programar en multitud de plataformas***, desde los sistemas operativos más tradicionales de ordenadores personales, como Windows, Linux o Mac OSX, hasta dispositivos electrónicos más exóticos como teléfonos o relojes inteligentes, pasando por consolas de videojuegos.
- **Libre, de Código abierto y gratuito:** es decir que sin licencias restrictivas **Python** puede ser usado, copiado, estudiado, y modificado de cualquier forma.
  - El código de **Python** es compartido libremente y se alienta a la comunidad de programadores que mejoren el diseño del software.
  - Las distintas versiones de **Python** puede descargarse desde el sitio Web oficial: https://www.python.org/downloads/source/
- **Comunidad:** Python cuenta con extensa documentación y una enorme comunidad.
  - La comunidad oficial **"Python Software Foundation"** (PSF): https://www.python.org/psf/
  - La evolución de **Python** se rige por la aprobación/implementación de propuestas de mejora conocidas con el nombre de **PEP**. que siguen un proceso de aprobación que participa e involucra a los miembros de la comunidad.

# Características - Debilidades

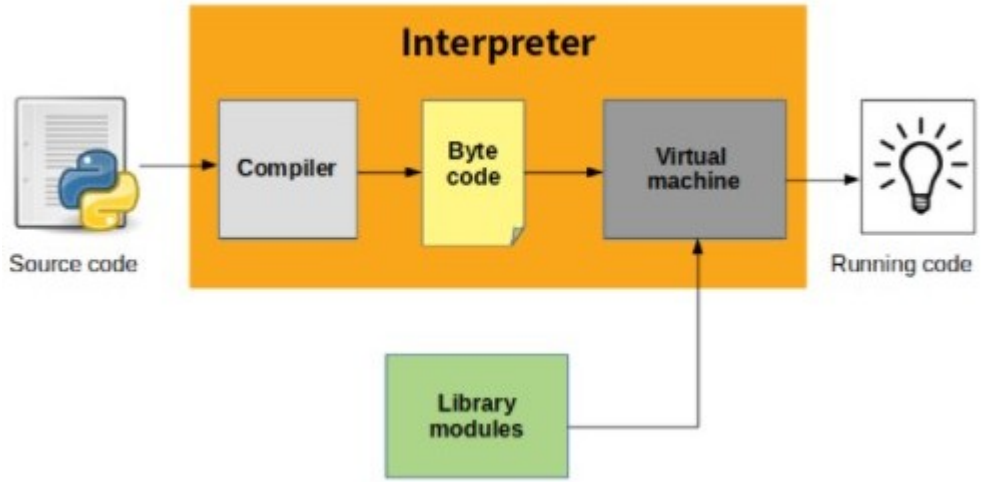
- Al igual que cualquier otro lenguaje de programación, **Python** también tiene **puntos débiles:**
  - **Lentitud**: en muchas ocasiones se considera que los programas desarrollados en **Python** son ***"lentos"*** en comparación con los tiempos de ejecución de lenguajes compilados.
    - El origen de esta afirmación reside en que se trata de lenguaje interpretado y no poseer por defecto un compilador JIT (del inglés Just in – Time), lo que haría que se compilase el programa escrito en **Python** y optimizasen más los tipos de datos.
    - Aún así, en **Python 3** se han hecho muchas mejoras de rendimiento de los tipos de datos y se ha mejorado notablemente este aspecto.
    - No obstante, existen librerías que permiten marcar porciones de código para ser compiladas en tiempo de ejecución o la opción de utilizar **CPython**, que permite escribir código **C** compatible con **Python** e integrarlo de forma natural para mejorar la velocidad de procesamiento.

# Versiones

- A lo largo de su historia **Python** ha sufrido numerosos cambios y hoy en día sigue recibiéndolos continuamente a través de las **PEP**. A continuación, se nombran las principales versiones con los cambios más destacados:
  - **Versión 0.9 (1991):** primera versión de **Python** publicada por **van Rossum**. Contaba con muchos componentes actuales como listas, diccionarios, conceptos de orientación a objetos, cadenas de caracteres y otras características.
  - **Versión 1.0 (1994):** se introducen conceptos de programación funcional.
  - **Versión 1.6 (2000):** se añade licencia compatible con GPL (GNU General Public License).
  - **Versión 2.7 (2010):** Es la última versión de la rama 2.X y se incluyen algunas de la ya empezada a desarrollar versión3.X. En noviembre de 2014 se anuncia la versión2.X y que dejará de tener soporte a principios de 2020, invitando a los usuarios a migrar activamente a la versión 3.
  - **Versión 3.0 (2008):** se hacen cambios en cuestiones principales del lenguaje, quitando redundancia de código e introduciendo grandes incompatibilidades con la versión 2.
  - **Versión 3.9 (2020):** se añaden múltiples funcionalidades y se borran algunas presentes por retrocompatibilidad con la versión 2.
  - **Versión 3.10.2 (13 de Enero de 2022):** última versión del lenguaje.

# Intérprete de Python

- **Python** es un lenguaje de programación interpretado, lo que significa que el código fuente no necesita ser compilado al código máquina específico del hardware donde se ejecuta, sino que se ejecuta directamente en cualquier sistema que tenga instalada la ***máquina virtual de Python.***
- Cuando se instala **Python** en una máquina, este tiene, como mínimo dos componentes:
  - El **intérprete** y
  - La **librería estándar** (módulo, funciones, constantes, tipos, tipos de datos, excepciones, etc.).
- Dependiendo de la implementación de **Python**, el intérprete puede estar escrito en **C, Java, .net**, etc.




Fuente: https://python-kurs.github.io/sommersemester_2019/units/S01E01.html

# Intérprete de Python (2)

- Se podría definir **el intérprete** como un **programa** que ***se encarga de ejecutar otros programas***.
- A continuación se ahondará en ello:
  - El **código fuente**, que se compone de archivos de texto plano que tienen una gramática específica (que se denomina lenguaje **Python**), con una extensión concreta (.py) y estructurados de una forma precisa.
  - Por otro lado, se encuentran los ficheros de **byte code**, que son el resultado de una compilación rápida que se efectúa justo antes de comenzar la ejecución. El código escrito en **byte code** está listo para ser ejecutado en cualquier máquina virtual de **Python**.
  - Por último, se encuentra la **máquina virtual de Python (PVM – Python Virtual Machine**), que es la encargada de ejecutar los archivos que tienen el byte code. Por lo tanto, la parte que ***sí es dependiente del hardware utilizado es la máquina virtua***l.
  - Lo que se denomina intérprete de **Python** es el programa completo que analiza el código fuente, genera los ficheros compilados y ejecuta el código usando la máquina virtual.

# Intérprete de Python (3)

- Cabe destacar algunas peculiaridades del **byte code**:
  - Los archivos que contienen el **byte code** tienen una extensión **.pyc (Python compiled)**.
  - Los archivos ***no son necesarios para la ejecución del programa***, dado que, si no se pueden generar por algún motivo (por falta de espacio o de permisos de escritura), el **byte code** será **generado e insertado en memoria directamente**, sin crearse en ficheros guardados en el sistema operativo.
  - Un programa en **Python** que tenga los archivos **.pyc** generados no necesita tener el código fuente, por lo que se puede ahorrar espacio de disco borrando los códigos fuente solo ejecutando los **.pyc.** (solo se recomienda en sistemas con grandes restricciones de espacio)
  - El **intérprete de Python** es inteligente a la hora de generar los **.pyc**, y ya se han generado los ficheros **.pyc** y no ha habido cambios en el fichero fuente, no realiza ninguna compilación, simplemente ***usa los ficheros ya compilados***, agilizando así el proceso de iniciar la aplicación.

# Implementaciones de Python

- Cuando se habla de la implementación de **Python**, normalmente se hace referencia a la implementación usando C denominada ***CPython***.
  - No obstante, el intérprete puede ser implementado en otros lenguajes. Las diferencias entre implementaciones están, principalmente, en la habilidad de usar librerías escritas en algún lenguaje específico.
- CPython:
  - Es la implementación original del lenguaje y la más utilizada, programada en ANSI C.
  - El intérprete genera **byte codes** desde archivos de código fuente para ejecutarlos en la máquina virtual de **Python**, y esta los ejecuta.
  - Si un sistema operativo tiene una versión de **Python** preinstalada, lo más seguro es que la implementación sea **CPython**.
  - Es el estándar, siempre se mantiene actualizada, soporta la interoperabilidad con librerías escritas en C y normalmente es muy rápida en tiempo de ejecución comparada con las demás implementaciones.
  - Otras implementaciones: **Jython, PyPy, IronPython**.

# Bibliografía


- Óscar Ramírez Jiménez: ***"Python a fondo"*** 1era Edición. Ed. Marcombo S.L.. 2021.
- Allen Downey. ***"Think Python"***. 2Da Edición. Green Tea Press. 2015.
- Eirc Matthes: ***"Python Crash Course"***. 1era Edición. Ed. No Starch Press. 2016.
- Zed A. Shaw: ***"Learn Python 3 the Hard Way"***. 1era Edición. Ed. Addison-Wesley. 2017.