



# Python Essentials



# Introducción

# Modulo 1

- Introducción
- Los fundamentos de la programación de computadoras
- Configurar su entorno de programación
- Compilación vs Interpretación
- Introducción a Python

# Modulo 2

- Tipos de datos y métodos básicos de formateo, conversión, entrada y salida de datos;
- Literales
  - Enteros, flotantes, cadenas, booleano, octal, hexadecimal
- Operadores
  - + (suma), - (resta), \* (multiplicación), / (división clásica), % (módulo), \*\* (exponenciación), // (división entera)
  - Expresiones, operadores unarios, operadores binarios, jerarquía de prioridades, exponenciación
- Variables.

# Modulo 3

- Valores **Booleanos**
- instrucciones **if-elif-else**
- Los bucles (loops) **while** y **for**
- Operaciones **lógicas** y **bit a bit** (bitwise)
- **>, <, >=, <=, ==, >>, <<**
- Y (and), o (or), no (not), xor -> **&** (ampersand), **|** (barra), **~** (tilde), **^** (circunflejo)
- **Listas y matrices**
- Ordenar, operaciones en lista

# Modulo 4

- Definición y uso de funciones
  - incorporadas, preinstaladas, definidas por el usuario, lambda
- Diferentes formas de paso de argumentos
  - Parámetros, argumentos, retorno
- Ámbitos (scope) del nombre
- Tuplas y diccionarios

# Modulo 5

- Módulos de Python: su razón de ser, función, cómo importarlos de diferentes formas y presentar el contenido de algunos módulos estándar proporcionados por Python;
- La forma en que se acoplan los módulos para crear paquetes.
- El concepto de una excepción y la implementación de Python de la misma, incluida la instrucción **try-except**, con sus aplicaciones y la instrucción **raise**.
- Cadenas y sus métodos específicos, junto con sus similitudes y diferencias en comparación con las listas.

# Modulo 6

- El enfoque orientado a objetos – fundamentos
- Clases, métodos, objetos y las características estándar del objeto
- Manejo de excepciones
- Trabajando con archivos



# Programación

## Conceptos básicos

# Lenguajes naturales vs. lenguajes de programación

- El lenguaje es un **medio** (y una herramienta) para expresar y registrar **pensamientos**:
  - **Lenguaje corporal**: es posible expresar tus sentimientos más profundos con mucha precisión sin decir una palabra.
  - **Lengua materna**: que utilizas para manifestar tu voluntad y pensar en la realidad.
- Las computadoras también tienen su propio lenguaje, llamado **lenguaje de máquina**, que es muy rudimentario.
  - Un conjunto completo de **comandos conocidos** se denomina **Lista de instrucciones** (LI)
  - Los diferentes tipos de computadoras pueden variar según el tamaño de sus LI y las instrucciones pueden ser completamente diferentes en diferentes modelos.

# ¿Qué hace un idioma?

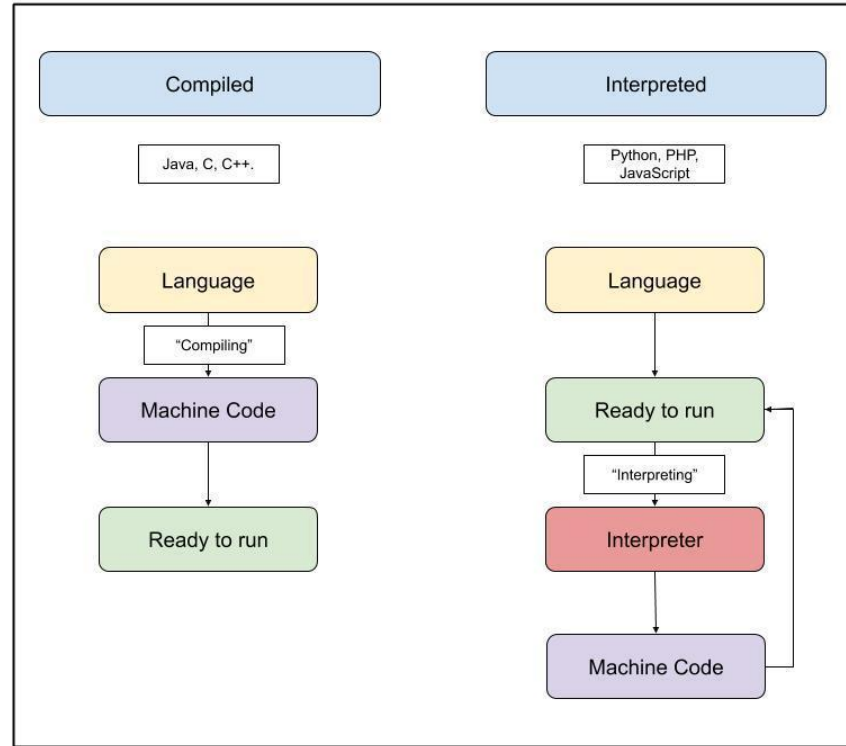
- **UN ALFABETO** Un conjunto de símbolos
- **UNA SINTAXIS** Un conjunto de reglas (formales o informales)
- **SEMÁNTICA** Un conjunto de reglas que determinan si una determinada frase tiene sentido.

# ¿Qué hace un idioma?

- El lenguaje informático está lejos de ser una lengua materna humana. Necesitamos (tanto computadoras como humanos) un lenguaje o un puente entre dos mundos diferentes.
- Un **lenguaje** en el que los **humanos** pueden escribir sus **programas** y un lenguaje que las **computadoras** pueden usar para **ejecutar** los programas.
- Estos lenguajes son lenguajes de **programación de alto nivel**. Utilizan símbolos, palabras y convenciones legibles para los humanos. Estos lenguajes permiten a los humanos **expresar comandos** a las computadoras.
- Un programa escrito en un lenguaje de programación de alto nivel se denomina código fuente (en contraste con el código de máquina ejecutado por computadoras) o archivo fuente.

# Compilación vs. interpretación

Hay dos formas diferentes de **transformar** un programa de un lenguaje de programación de **alto nivel** a un **lenguaje máquina**:



# Video: Compilación vs. interpretación (cont.)

- <https://www.youtube.com/watch?v=hPYCSu-LPc&t=343s>



# Compilation vs. interpretation

	COMPILACIÓN	INTERPRETACIÓN
VENTAJAS	<ul style="list-style-type: none"><li>La <b>ejecución</b> del código traducido suele ser más <b>rápida</b>.</li><li>Solo el usuario debe <b>tener</b> el compilador; el usuario final puede usar el código <b>sin</b> él.</li><li>El <b>código</b> traducido se almacena en lenguaje de máquina, es probable que sus propios inventos y trucos de programación <b>sigan siendo su secreto</b>.</li></ul>	<ul style="list-style-type: none"><li>Puede <b>ejecutar el código</b> tan pronto como lo complete; <b>no hay fases adicionales</b> de traducción;</li><li>El código se almacena usando el lenguaje de programación, no el de la máquina; esto significa que se puede <b>ejecutar</b> en computadoras que usan diferentes <b>lenguajes de máquina</b>; no compila su código por separado para cada arquitectura diferente.</li></ul>
DESVENTAJAS	<ul style="list-style-type: none"><li>La compilación en sí puede ser un proceso que <b>consume mucho tiempo</b>; es posible que no pueda ejecutar su código inmediatamente después de cualquier modificación;</li><li>Debe tener tantos <b>compiladores como plataformas de hardware</b> en las que desee que se ejecute su código.</li></ul>	<ul style="list-style-type: none"><li>No espere que la interpretación acelere su código a alta velocidad; su <b>código</b> compartirá el poder de la computadora con el <b>intérprete</b>, por lo que no puede ser realmente rápido;</li><li>Tanto usted como el usuario final deben tener el <b>intérprete</b> para ejecutar su <b>código</b>.</li></ul>

# ¿Qué hace que Python sea especial?

- **Fácil de aprender**
- **Fácil de enseñar**
- **Fácil de usar**
- **Fácil de entender**
- **Fácil de obtener, instalar e implementar**

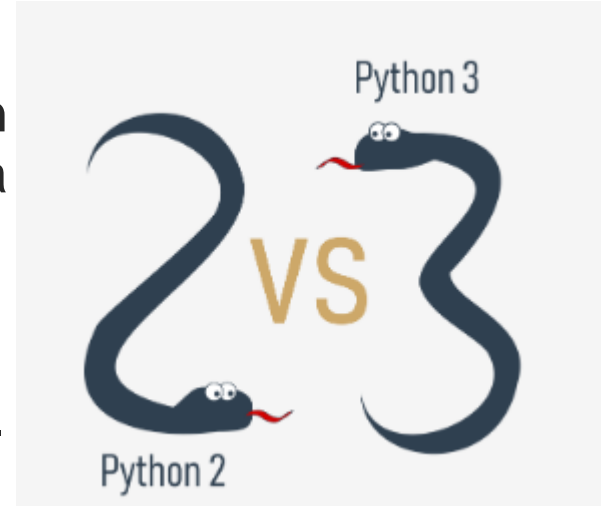


# Cuando no usar Python

- **Programación de bajo nivel** (a veces llamada programación "cercana al metal" / "lenguaje máquina" / "lenguaje ensamblador"): si desea implementar un controlador o motor gráfico **extremadamente efectivo**, **no usaría** Python;
- **Aplicaciones para dispositivos móviles**: aunque este territorio todavía está **esperando** ser **conquistado** por Python, lo más probable es que algún día suceda.

# Hay más de un Python

- **Python 2** es una versión **anterior** del Python original. Desde entonces, su desarrollo se ha **detenido** intencionalmente, sin embargo, las **actualizaciones** se publican de **forma regular**, pero no tienen la intención de modificar el lenguaje de manera significativa.



- **Python 3** es la versión **actual** del lenguaje. Está pasando por su **propio** camino de evolución, creando sus propios estándares y hábitos.

# Versiones de Python

Además de Python 2 y Python 3, hay más de una versión

- Python aka Cpython



- Cython



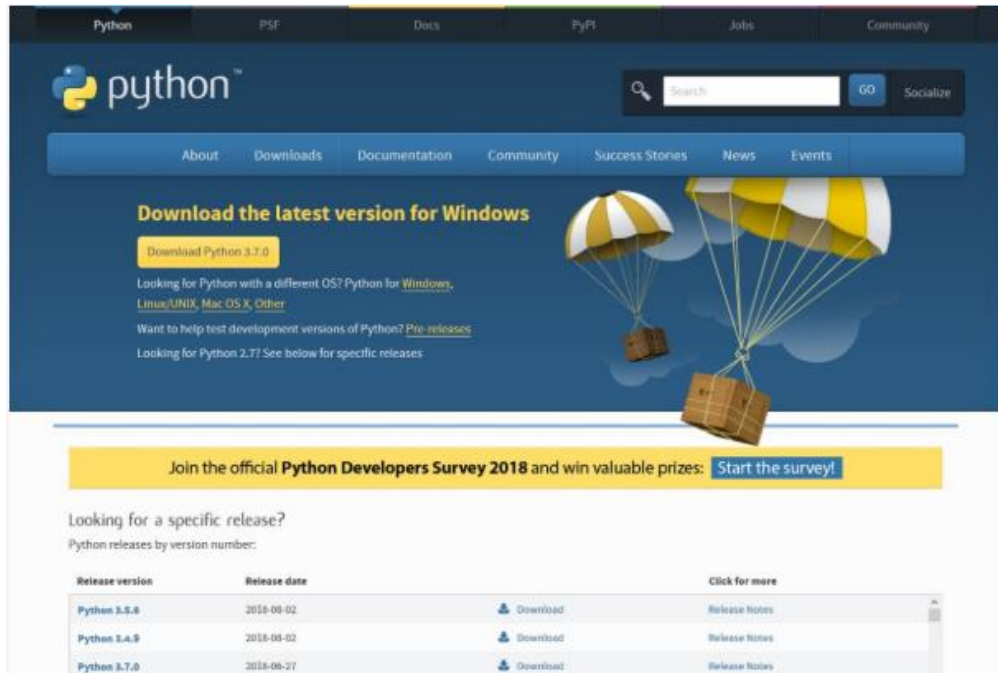
- Jython



- PyPy and RPython



# Cómo obtener Python y cómo llegar a usarlo

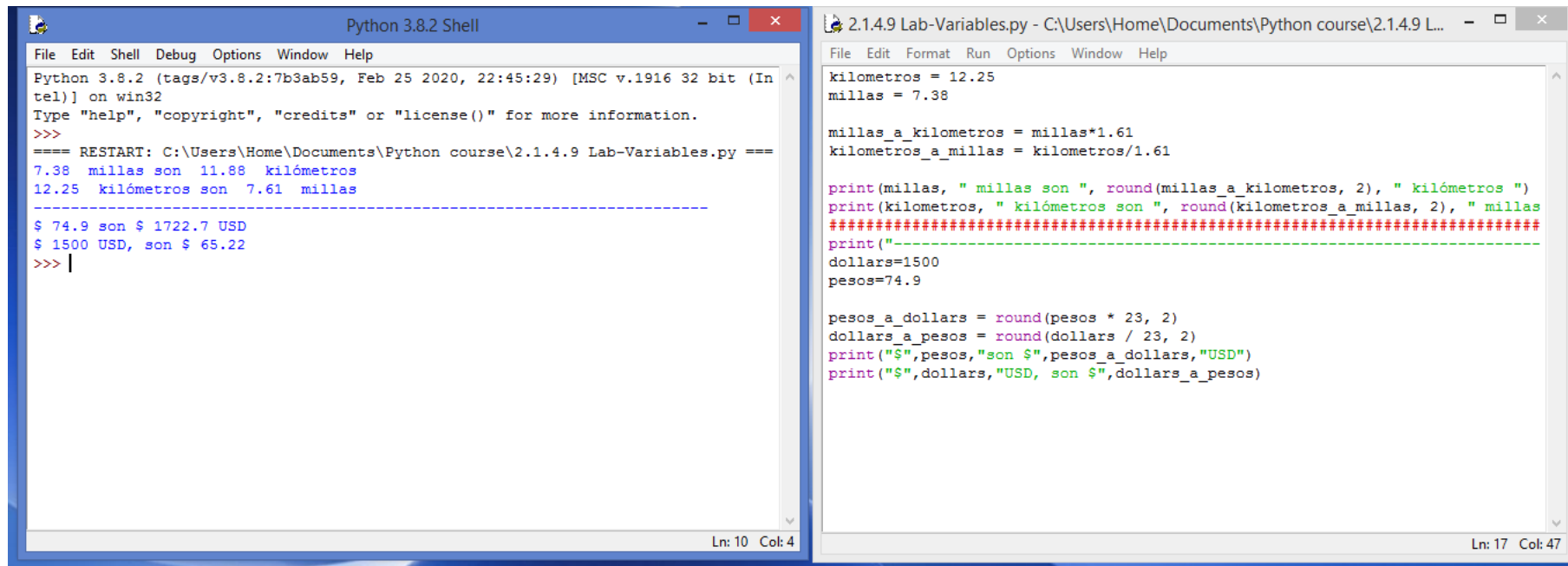


The screenshot shows the Python.org website. At the top, there's a navigation bar with links: Python, PSF, Docs, PyPI, Jobs, and Community. Below this is a search bar and a 'Socialize' button. A secondary navigation bar contains links: About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a large banner with the text 'Download the latest version for Windows' and a button 'Download Python 3.7.0'. Below the button, there are links for other operating systems: 'Python for Windows', 'Linux/UNIX', 'Mac OS X', and 'Other'. There's also a link for 'Pre-releases' and a note about Python 2.7. Below the banner is a yellow box with the text 'Join the official Python Developers Survey 2018 and win valuable prizes: Start the survey!'. At the bottom, there's a section titled 'Looking for a specific release?' with the text 'Python releases by version number:'. This section contains a table with three columns: 'Release version', 'Release date', and 'Click for more'.

Release version	Release date	Click for more
Python 3.5.8	2018-09-02	<a href="#">Download</a> <a href="#">Release Notes</a>
Python 3.6.8	2018-08-02	<a href="#">Download</a> <a href="#">Release Notes</a>
Python 3.7.0	2018-08-27	<a href="#">Download</a> <a href="#">Release Notes</a>

<https://www.python.org/downloads/>

# Demostración: cómo usar Python IDLE



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\Home\Documents\Python course\2.1.4.9 Lab-Variables.py ====
7.38 millas son 11.88 kilómetros
12.25 kilómetros son 7.61 millas
-----
$ 74.9 son $ 1722.7 USD
$ 1500 USD, son $ 65.22
>>> |
```

```
2.1.4.9 Lab-Variables.py - C:\Users\Home\Documents\Python course\2.1.4.9 L...
File Edit Format Run Options Window Help

kilometros = 12.25
millas = 7.38

millas_a_kilometros = millas*1.61
kilometros_a_millas = kilometros/1.61

print(millas, " millas son ", round(millas_a_kilometros, 2), " kilómetros ")
print(kilometros, " kilómetros son ", round(kilometros_a_millas, 2), " millas ")
#####
print("-----")
dollars=1500
pesos=74.9

pesos_a_dollars = round(pesos * 23, 2)
dollars_a_pesos = round(dollars / 23, 2)
print("$",pesos,"son $",pesos_a_dollars,"USD")
print("$",dollars,"USD, son $",dollars_a_pesos)
```