



**INSTITUTO SUPERIOR
TECNOLÓGICO QUITO**

Excelencia en Educación Superior

Introduccion

Enero 7, 2026



Bryan Justin Antamba

2 Introduccion a Python

2 Introducción a Python

2.0.1 Principales características

- Lenguaje de propósito general
- Interpretado, no compilado. Más flexible y portable
- Tipado dinámico
- Fuertemente tipado
- Énfasis en la legibilidad
- Lenguaje de alto nivel
- Gestión automática de memoria. Cuando el rendimiento es crítico, hay lenguajes más apropiados

- Multiparadigma: orientado a objetos, procedural y funcional
- Indentación para eliminar bloques de código
- Gran librería con módulos para múltiples tareas
- Multiplataforma

2.0.2 ¿Por qué Python?

"Python is used in pursuits as diverse as data science, film-making, computer science education, IT management, and much more. There really is no computing field that Python has not touched(except maybe kernel development). Python is loved for its flexibility, beautiful and succinct 1 syntax, object-oriented purity, and bustling community. The strong community is important because it means Python is welcoming to newcomers and has a large ecosystem of available libraries for developers to build upon"

Kopec, D.(2019). Classic computer science problems in Python. Simon and Shuster.

- Uno de los lenguajes mas usados en todo el mundo.
- Perfecto para introducción a la programación.
- Ecosistema amplio con librerías estables para múltiples áreas
- Comunidad muy participativa y mucha documentación
- Incrementa la productividad del desarrollador
 - Menos programación. Código más compacto.
 - No largas compilaciones.
 - Código legible, fácilmente mantenible.
- Facil integracion con herramientas y otros lenguajes.
- Versatil en la tipologia de programacion
- Es multiplataforma, portable
- Compiable para mejorar ejecucion
- Trabajo de memoria con grna cantida de datos Stack Overflow Developer Survey 2023
The Top Programming Languages 2024 Tiobe Index HackerRank Developer Skills Report

2.0.3 ¿Que puedo hacer con Python?

- Herramientas shell (administración de sistemas)
- Manipulación de ficheros
- Ejecución de comandos.
- Desarrollo de interfaces Gráficas de Usuario(GUIs)
- Internet and network communication.
 - Generación y parseado de XML y JSON
 - Recuperación de webs por URL
 - COMunicación a través de sockets.
 - Transferencia de icheros por FTP.

- Programación de base de datos.
- Computación numérica (NumPy)
- Análisis lenguaje natural.
- Aprendizaje automático e inteligencia artificial.
- Programación multimedia.
- Tratamiento de datos.

- Linear algebra
- statistical modeling
- visualization
- computational linguistics
- graph analysis
- machine learning
- business intelligence
- data storage and retrieval

2.0.4 Filosofía Python

In [1]: `import this`

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!
```

2.0.5 Documentación

- Python enhancement Proposals (PEP)

- [Index](#)
- [Purpose and Guidelines](#)

- Python Style Guide

3 Interprete de Python y ejecución de scripts

3.0.1 Antes de empezar

Se puede utilizar diferentes formas para ejecutar código:

- Línea de comandos o terminal, Shell de python o interactivo, python/ipython
- IDE -> Eclipse, Pycharm, Sublime, Nano, VSCode, Atom, Spyder, ...
- Google Colab, Azure Notebooks, Jupyter

3.0.2 ¿Qué es un intérprete?

- Un programa que ejecuta otros programas.
- Una capa de software entre tu código y el hardware que lo ejecuta.
- Debe estar instalado en tu ordenador para poder ejecutar código Python.
- Para la especificación de Python, existen varias implementaciones:
 - CPython (implementación en C). Es el más común.
 - Jython (implementación en Java).
 - IronPython (implementación en .NET).

3.0.3 ¿Cómo se lleva a cabo la ejecución de scripts?

```
In [2]: #Imprimir en pantalla hello"
print("Hello World")
```

Hello World

Guia con detalles para Windows, Linux y macOS: <https://realpython.com/run-python-scripts/>

3.0.4 Perspectiva del desarrollador

- Un script en Python es un fichero de texto que:
 - Contiene instrucciones Python.
 - Tiene extension .py.
- Puedes ejecutar scripts:
 - Línea de comandos
 - IDE

3.0.5 Perspectiva de Python

1. Compilación de código fuente a *byte code*.
- Código byte code se ejecuta más rápido.

- Ficheros .pyc que se almacenan en caché.
- Permite saltarse el paso de compilación.

2. Python Virtual Machine (PVM)

- Ejecuta las instrucciones en *byte code*

3.0.6 ¿Cómo puedes ejecutar tus scripts?

3.0.7 Línea de comandos

- Ejecutar "py" o "python" para abrir una sesión interactiva del intérprete.
- También ejecutando la aplicación "Python" desde el menú inicio.
- Los caracteres ">>>" indican que estás en una sesión interactiva.
- Útil para experimentación y testing.

```
>>> print ("Hello world")
Hello World
```

Inconveniente: los programas que ejecutas en la línea de comandos desaparecen tras ser ejecutados.

3.0.8 REPL

Sistema interactivo para comunicarse con el ordenador en un lenguaje, Python. Se debe cumplir: * Read. El ordenador pueda leer unidades como entrada * Evaluate. El código pueda ser procesado * Print. Los resultados puedan verse * Loop. Continuar con la conversación

3.0.9 Ficheros

- Permite almacenar programas.
- Ficheros de texto con instrucciones Python
 - No olvidar shebang en Linux -> `#!/usr/bin/env python3`
- Terminología (varía según fuentes):
 - Scripts o programas: programa principal.
 - Módulos: ficheros importados desde otros ficheros.
- Se puede lanzar pasando nombre de fichero a comando python.

```
> python ./script1.py
Hello world
```

- Otra alternativa (a partir de Python 3.3) es:

```
py ./script1.py
Hello world
```

- O incluso:

```
./script1.py
Hello world
```

- También es posible hacer doble-click sobre fichero .py.

3.0.10 Instalacion de librerias

- pip (built-in >Python3.4)
- pipenv (gestiona paquetes y entornos virtuales) o virtualenv

3.0.11 Jupyter

```
In [1]: # Preguntar de forma interactiva
# print?
# Usar shift + tab para hint con ayuda
# Comentar una línea
# """
# Se puede comentar un texto
# más grande para hacer descripciones detalladas con más de una línea
# """
import pandas as pd
print("Siempre podremos 'poner' los comentarios en forma de salida, para ver_resultados")
print('Siempre podremos "poner" los comentarios en forma de salida, para ver_resultados')
# universidad = 'ITQ'
# print(universidad)
```

Siempre podremos 'poner' los comentarios en forma de salida, para ver_resultados
 Siempre podremos "poner" los comentarios en forma de salida, para ver_resultados

Celdas Python - Marca el número de la ejecución - Marca si está en ejecución con * Kernel - Se puede resetear el kernel cuando haya problemas - Limpiar y resetear kernel Comandos especiales (IPython): - %run: ejecuta un script %run ./miprograma.py

- %time: ejecuta una línea de código y devuelve el tiempo de ejecución
- %%time: ejecuta una celda de código y devuelve el tiempo de ejecución
- !: ejecuta un comando de consola
- %%bash: ejecuta un comando en bash en un subprocesso

ls -la

- %%js: ejecuta javascript

```
%%js
```

```
var master = "Máster en Inteligencia Artificial";
```

```
var codigo = 1;
```

```
alert("Esta asinatura tiene el código: " + codigo)
```

Atajos de teclado (shortcuts): Documentación de Jupyter

In [2]: `ls`

El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 1C60-82D2

Directorio de C:\Users\TUF A15\Desktop\machine2026

```
07/01/2026  16:30    <DIR>          .
07/01/2026  14:17    <DIR>          ..
07/01/2026  14:32    <DIR>          .ipynb_checkpoints
07/01/2026  14:25    <DIR>          img
07/01/2026  16:30                15.211 IntroduccionPython.ipynb
07/01/2026  14:30                1.388 Untitled.ipynb
                2 archivos          16.599 bytes
                4 dirs   503.882.051.584 bytes libres
```

In [4]: `%%js`
`var` asignatura = "Materia de Machine E-learning 1";
`var` codigo = 1;
`alert`("Esta asinatura tiene el código: " + codigo)

3.0.12 Performance Sample, NumPy vs Plain Python

In []: `import` numpy `as` np
`my_arr` = np.arange(10000000) # Utilizando NumPy Arrays
`my_list` = list(range(10000000)) # Utilizando Listas
`print`(`my_list`)

In []: `%time` for _ in range(10): `my_arr2` = `my_arr` * 2

In []: `%time` for _ in range(10): `my_list2` = [x * 2 for x in `my_list`]

<https://github.com/BryanAntamba/machine2026-PrimerParcial>

4 Referencias

González Duque, R. Python para todos. Licencia Creative Commons. Boschetti, A. y Massaron, L (2016). Python Data Science Essentials, Second edition. Birmingham - Mumbai: Packt

Python Software Foundation. 2. Built-in Functions — Python 3.6.7 documentation. Recuperado el 16 noviembre 2018 de <https://docs.python.org/3.6/library/functions.html>

Tutorialspoint, Python Tutorial. Recuperado el 16 noviembre 2018 de <https://www.tutorialspoint.com/python/>

Kenneth Reitz (2018), Code Style — The Hitchhiker's Guide to Python. Recuperado el 16 noviembre 2018 de <https://docs.python.org/3/guides/style/>

Frank Hofmann (2018), Introduction to the Python Coding Style. Recuperado el 16 noviembre 2018 de <https://stackabuse.com/introduction-to-the-python-coding-style/>

Design At Large: Fernando Perez: The Architecture of Jupyter. Perez, Fernando (2017). <https://www.youtube.com/watch?v=dENc0gwzySc&t=131s> YouTube.

In []: