machines2026

January 7, 2026

0.1 PAO25-25 - Python 01

Nombre:*Gabriel Salguero*

0.2 Introducción a Python

0.2.1 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 critico, hay lengujaes más apropiados
- Multiparadigma: orientado a objetos, procedural y funcional
- Indentación para eliminar bloques de código
- Gran libreria con módulos para múltiples tareas.
- Multiplataforma

0.2.2 ¿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 succint 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, facilmente mantenible
- Fácil integración con herramientas y otros lenguajes.

- Versátil en la tipología de programación
- Es multiplataforma, portable
- Compilable para mejorar ejecución
- Trabajo de memoria con grna cantida de datos Stack Overflow Developer Survey 2023 The Top Programming Languages 2024 Tiobe Index HackerRank Developer Skills Report.

0.2.3 ¿Qué puedo hacer con Python?

- Herramientas shell (adminitración 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 an retrieval

0.2.4 Filosofía Python

[31]:
```
import this
```

0.2.5 Documentación

- Python Enhancement Proposals (PEP)
  - Index
  - Purpose and Guidelines
- Python Style Guide

0.3 Intérprete de Python y ejecución de scripts

0.3.1 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

0.3.2 ¿Qué es un interprete?

- 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).

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

[32]:
```
#Imprimir en pantalla hello
print ('Hello World')
```

```
Hello World
```

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

0.3.4 Perspectiva del desarrollador

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

0.3.5 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.

0.3.6 ¿Cómo puedes ejecutar tus scripts?

0.3.7 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.

0.3.8 REPL

Sistema interactivo para comunicarse con el ordenador en un leguaje, 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.

0.3.9 Ficheros

- Permite almacenar programas.
- Ficheros de texto con instrucciones Python.
  - No olvidar *shebang* en Linux-> `#!/usr/bin/env python3`
- Terminología (varía segun 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 the Python 3.3) es:

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

- O incluso:

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

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

0.3.10 Instalación de librerías

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

[33]:
```
# Preguntar de forma interactiva
# print?
# Usar shift + tab para hint con ayuda
# Comentar una linea # """
# 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 subproceso

```
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

[34]:
```
ls
```

```
 El volumen de la unidad C es Asus Sistema
 El número de serie del volumen es: 76CC-8DFF

 Directorio de C:\Users\Reromash\Desktop\SalgueroGabriel\machines2026

07/01/2026  23:30    <DIR>          .
07/01/2026  23:11    <DIR>          ..
07/01/2026  23:16    <DIR>          .ipynb_checkpoints
07/01/2026  23:27    <DIR>          images
07/01/2026  23:30            15.014 machines2026.ipynb
               1 archivos         15.014 bytes
               4 dirs  27.529.928.704 bytes libres
```

[35]:
```
%%js
var asignatura = 'Materia de Machine E-learning 1';
var codigo = 1;
alert("Esta asinatura tiene el código: " + codigo)
```

```
<IPython.core.display.Javascript object>
```

0.3.11 Performance Sample, NumPy vs Plain Python

[36]:
```
import numpy as np

my_arr = np.arange(10000000) # Utilizando NumPy Arrays

my_list = list(range(10000000)) # Utilizando Listas
print(my_list[:10])
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[37]:
```
%time for _ in range(10): my_arr2 = my_arr * 2
%time for _ in range(10): my_list2 = [x * 2 for x in my_list]
```

```
CPU times: total: 375 ms
Wall time: 406 ms
CPU times: total: 11.7 s
Wall time: 12.4 s
```

0.4 Referencias

González Duque, R. Python para todos. Licencia Creative Commons.

Boschetti, A. y Massaron, L (2016). Python Data Science Essentials, Second edition. Birminghan-Munbai: 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-guide.org/writing/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.

Github: https://github.com/reromash1/machines2026

[ ]: