BME
BOLSAS Y MERCADOS ESPAÑOLES
a SIX company

Introducción a la programación en

Sesión 1

Rafael Caballero Roldán (rafacr@ucm.es)

¿Cómo aprovechar este curso?

- Jugar, experimentar
- Practicar sin miedo a equivocarnos
- ¡Estamos aprendiendo un lenguaje!

Sesión 1: Entorno y estructuras

1.- Introducción

- ¿Qué es Python?
- ¿por qué Python? ¿Por qué no R?
- ¿Para qué se utiliza Python? Bibliotecas más comunes
- ¿Cómo vamos a trabajar con Python? Entorno Jupyter Notebooks

2.- Elementos básicos

- Constantes, variables y Expresiones
- Importación de bibliotecas
- Caso práctico: carga de ficheros con la biblioteca Pandas (Excel, CSV, PDF)

Sesión 1: Entorno y estructuras

3.- Datos complejos

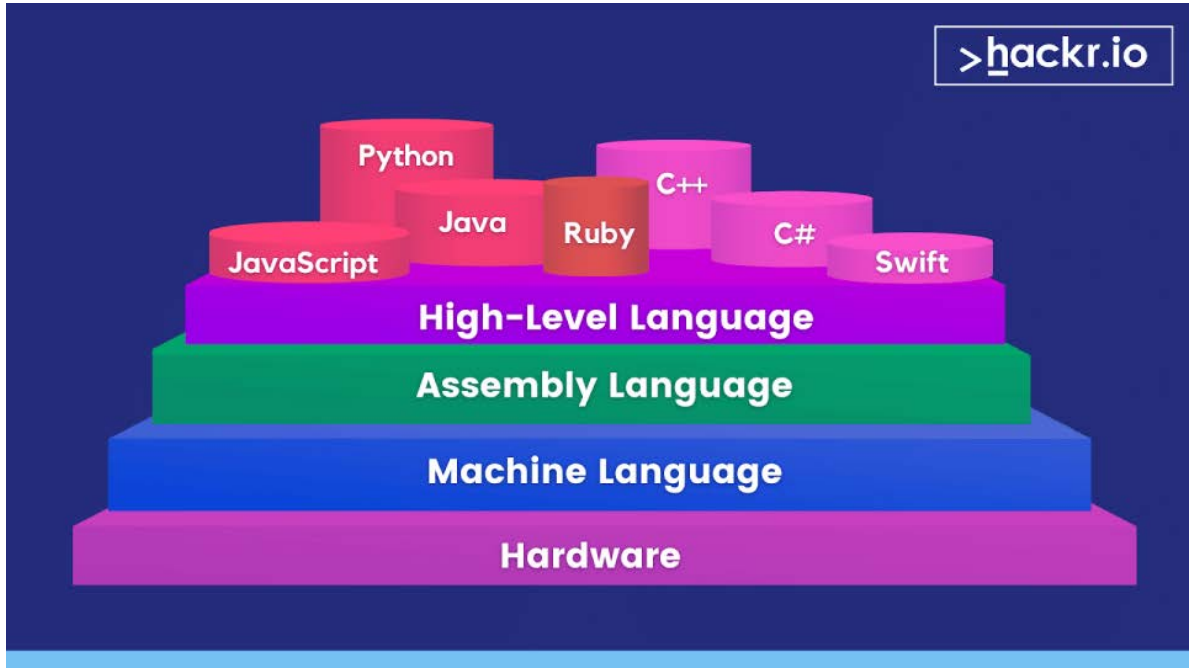
Secuencias: listas, tuplas

Conjuntos y diccionarios

Caso práctico: manejo básico de dataframes con pandas

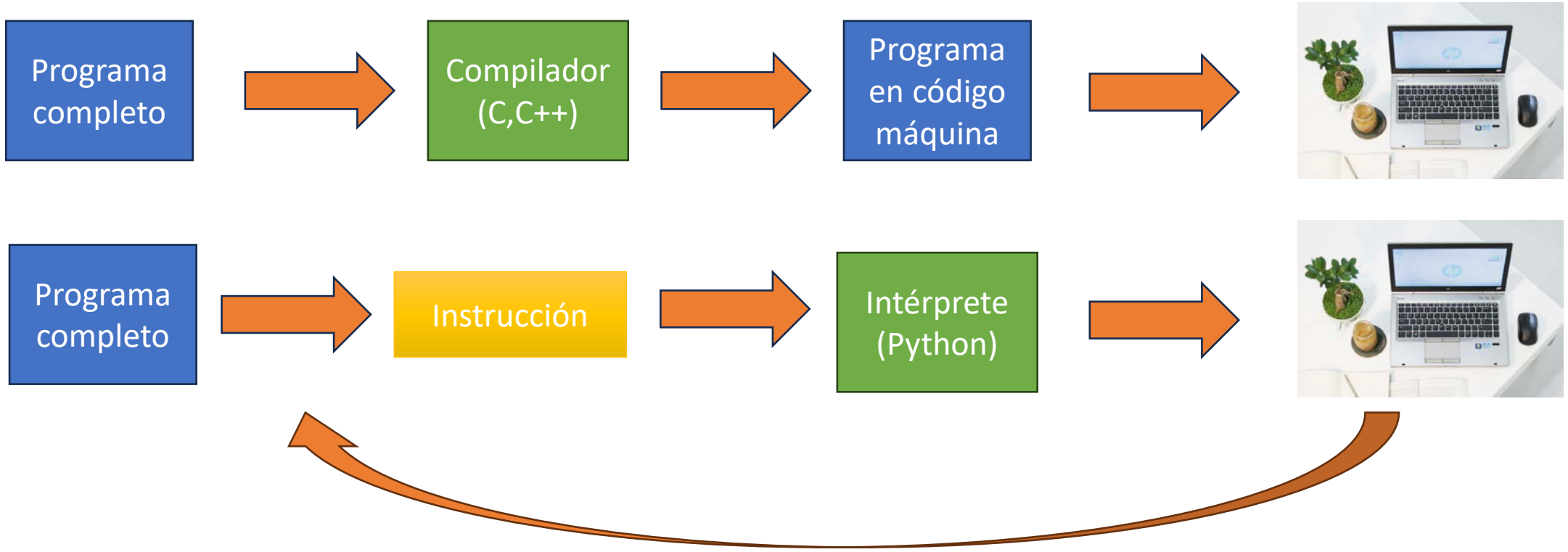
¿Qué es Python?

- Un lenguaje de programación…pero ¿qué es un lenguaje de programación?
- Una forma de dictar instrucciones a un ordenador
- Python → Lenguaje de alto nivel
  - Aun así hay que ser muy cuidadoso con la sintaxis



¿Cómo se “traduce” el código de un lenguaje de programación?

2 formas:



Analogía

- Compilar: llevar un texto a un traductor para que nos devuelva el texto traducido

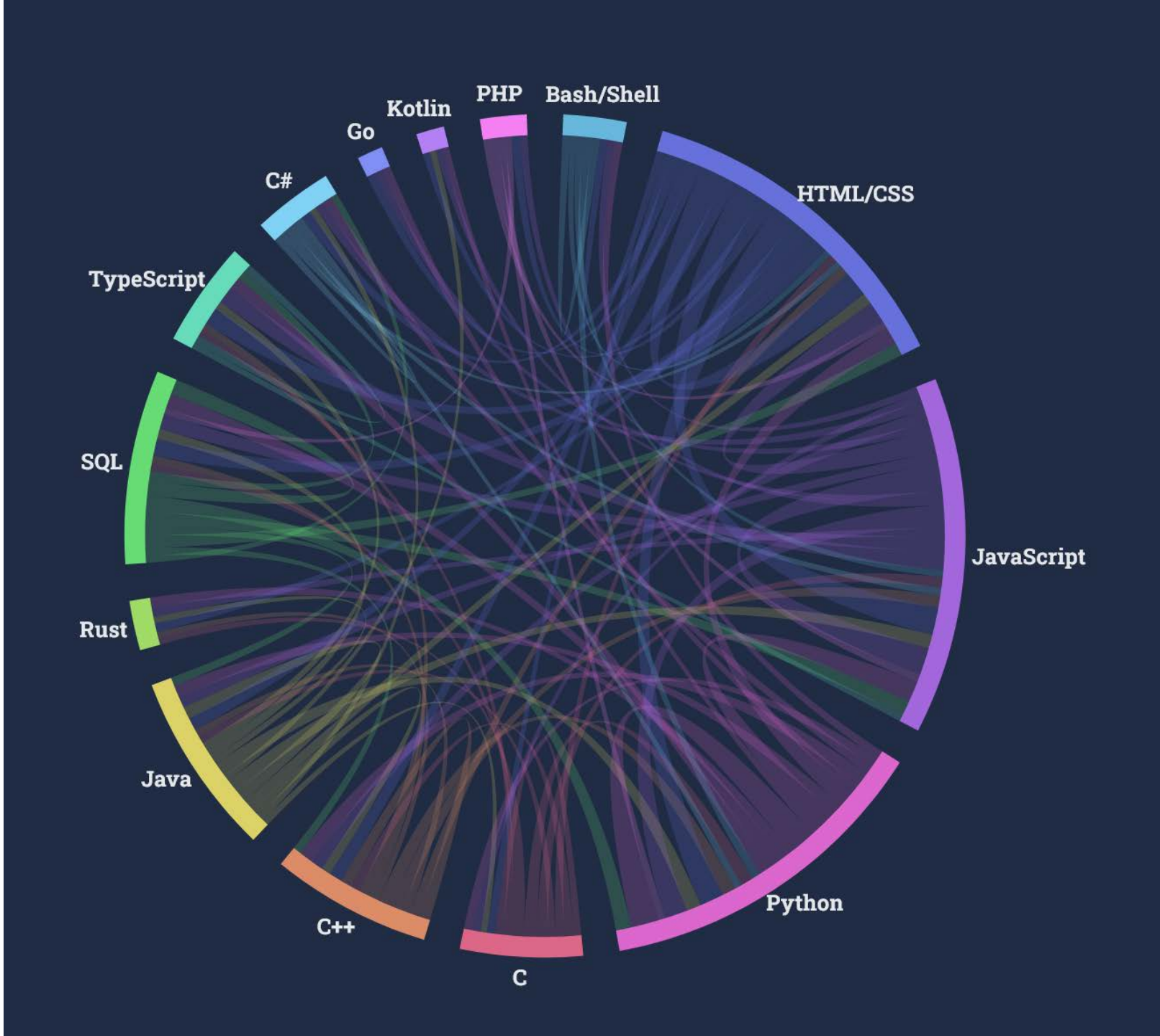
- Interpretar: disponer de un intérprete que va traduciendo cada cosa que decimos sobre la marcha

¿Por qué Python?

- Lenguaje de programación creado en 1991 por Guido van Rossum
- Libre, mantenido la asociación sin ánimo de lucro Python Software Foundation
- Ampliamente utilizado en ciencia de datos

Lenguaje popular (programadores)

[Fuente]



Lenguaje popular

- Stack Overflow: foro de preguntas más utilizado en programación

ZDNET

tomorrow
belongs to those who embrace it
today

trending tech innovation business security advice buying guides

/ business

Home / Business / Developer

Programming languages: Developers now ask more questions about Python than JavaScript on Stack Overflow

Rise in queries driven by interest in Python and community of new programmers.

Written by **Liam Tung,** Contributing Writer on April 23, 2019

Lenguaje popular

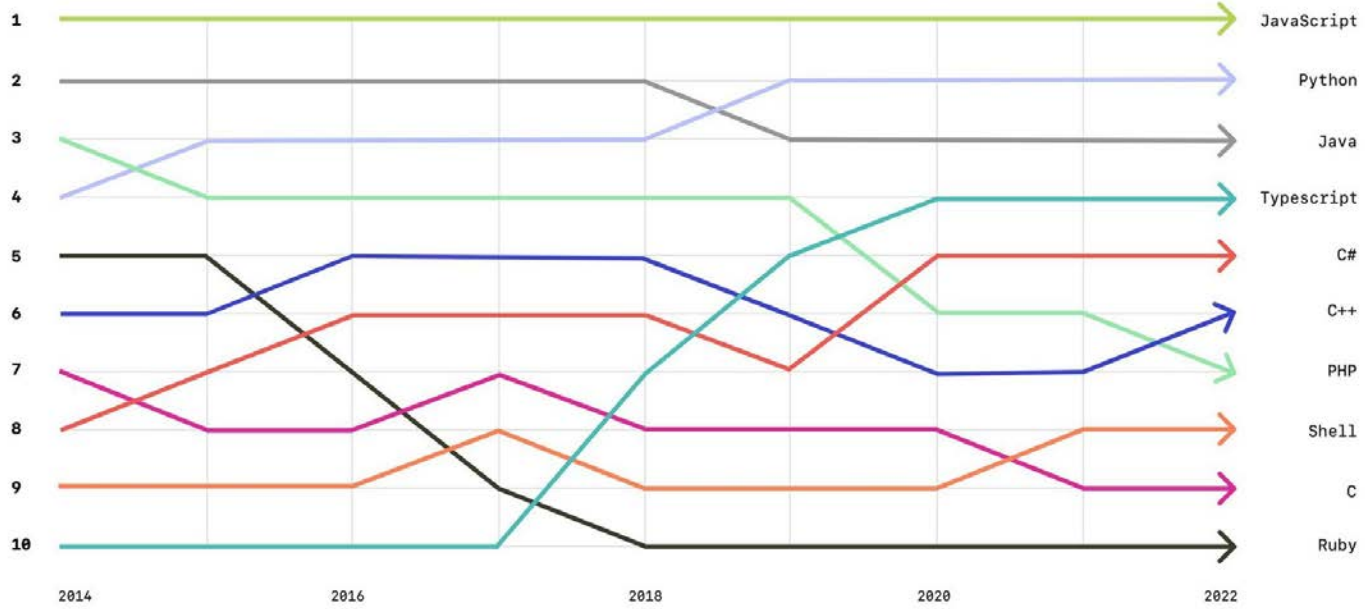
Fuente: PYPL

(PopularitY of Programming Languages)
Mar 2023

Worldwide, Mar 2023 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.91 %	-0.6 %
2		Java	16.58 %	-1.6 %
3		JavaScript	9.67 %	+0.6 %
4		C/C++	6.93 %	-0.5 %
5		C#	6.88 %	-0.5 %
6		PHP	5.19 %	-0.6 %
7		R	4.23 %	-0.2 %
8	↑	TypeScript	2.81 %	+0.6 %
9	↑	Swift	2.28 %	+0.2 %
10	↓↓	Objective-C	2.26 %	+0.0 %
11	↑↑↑	Rust	2.03 %	+1.0 %
12	↑	Go	1.93 %	+0.7 %
13	↓	Kotlin	1.82 %	+0.2 %
14	↓↓↓	Matlab	1.66 %	-0.3 %
15	↑	Ruby	1.1 %	+0.3 %
16	↑↑	Ada	1.01 %	+0.4 %
17	↓↓	VBA	1.01 %	+0.1 %

Lenguaje popular

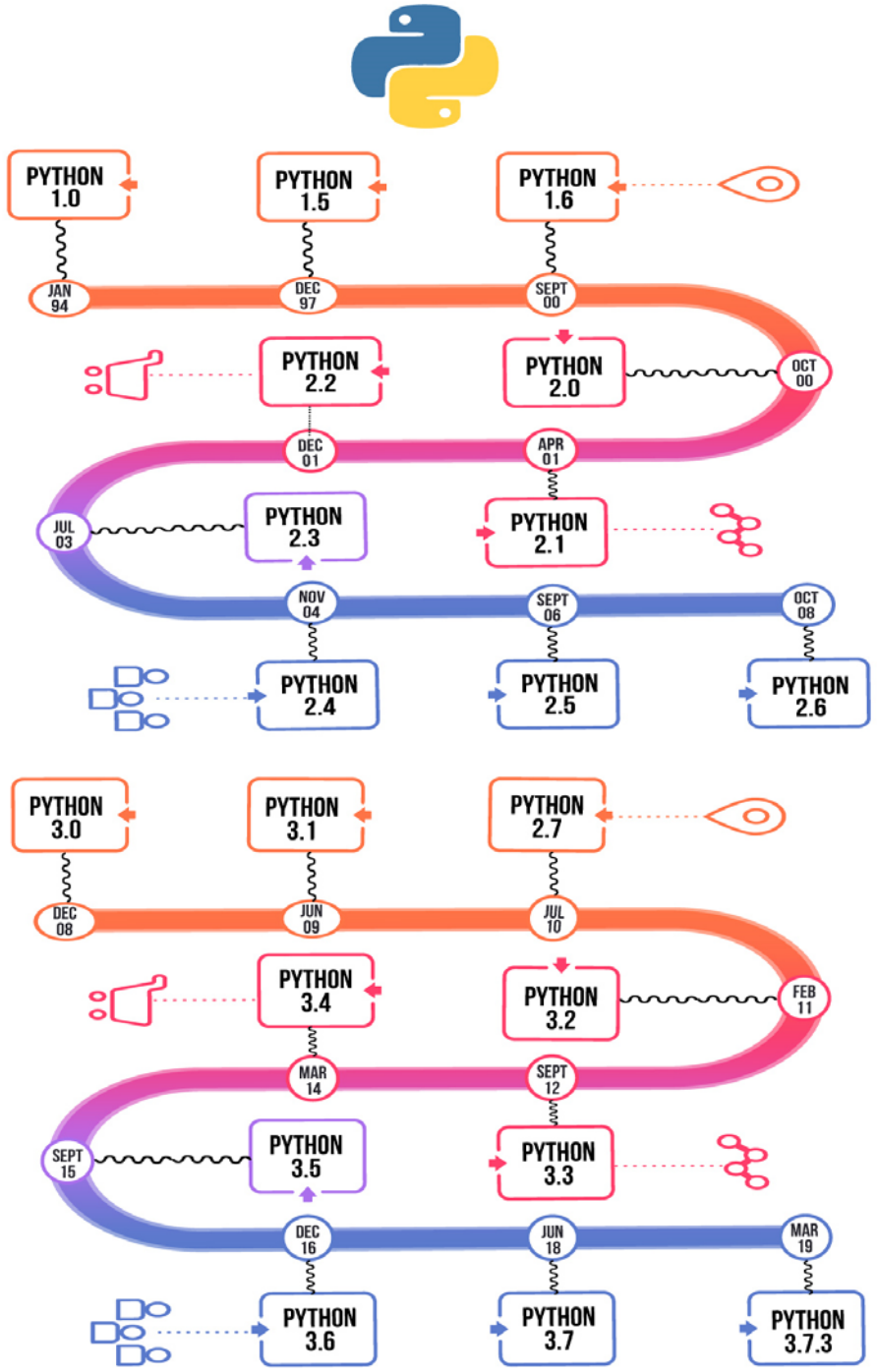


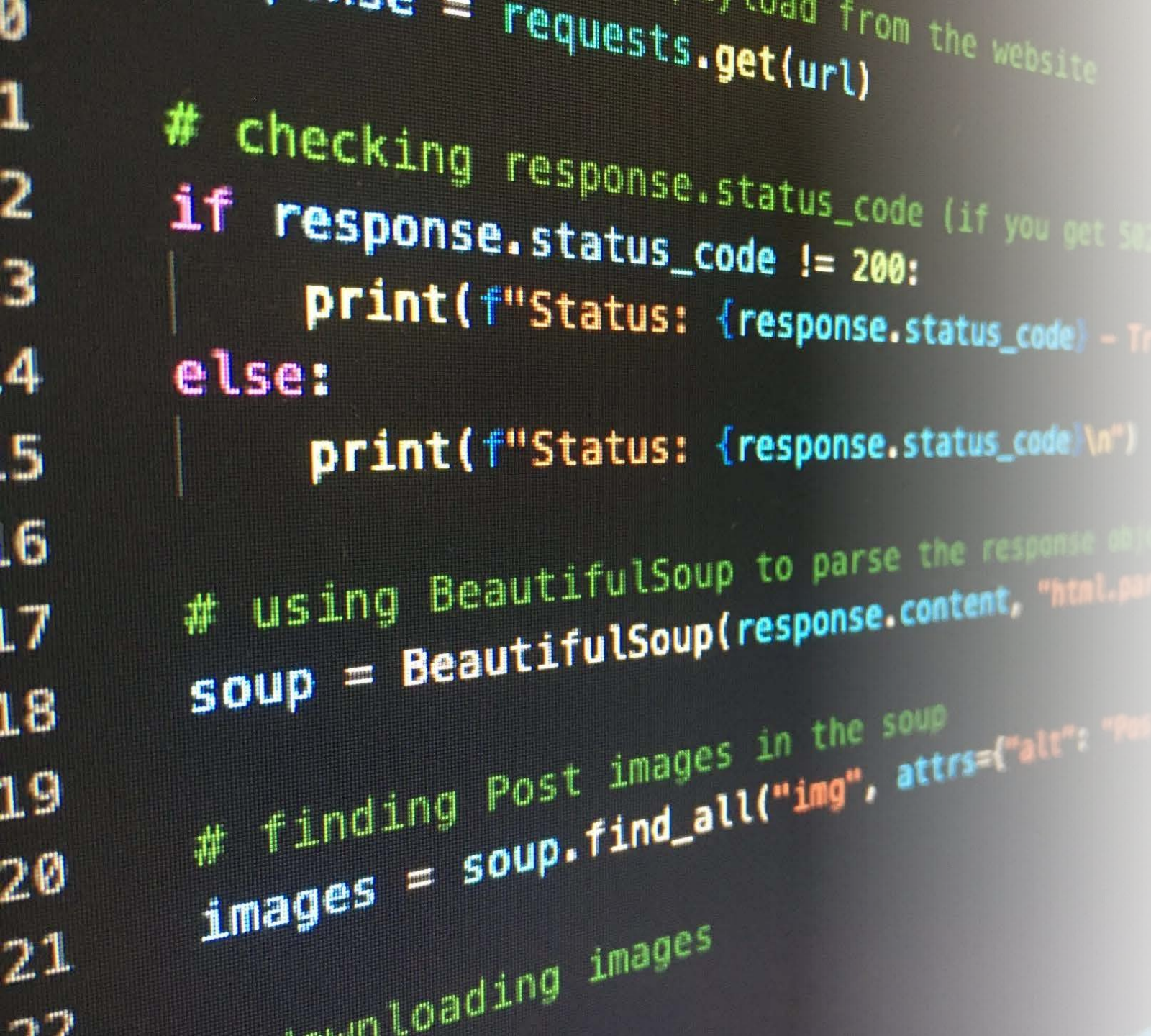
Lenguajes más utilizados en GitHub (fuente: GitHub)

Orígenes

- Lenguaje de programación creado en 1991 por Guido van Rossum (BDFL entre 1997 y 2018)
- Libre, mantenido por la asociación sin ánimo de lucro Python Software Foundation
- Ampliamente utilizado en ciencia de datos

Evolución


PYTHON 1.0
PYTHON 1.5
PYTHON 1.6
JAN 94
DEC 97
SEPT 00
PYTHON 2.2
PYTHON 2.0
OCT 00
DEC 01
APR 01
JUL 03
PYTHON 2.3
PYTHON 2.1
NOV 04
SEPT 06
OCT 08
PYTHON 2.4
PYTHON 2.5
PYTHON 2.6
PYTHON 3.0
PYTHON 3.1
PYTHON 2.7
DEC 08
JUN 09
JUL 10
PYTHON 3.4
PYTHON 3.2
FEB 11
MAR 14
SEPT 12
SEPT 15
PYTHON 3.5
PYTHON 3.3
DEC 16
JUN 18
MAR 19
PYTHON 3.6
PYTHON 3.7
PYTHON 3.7.3

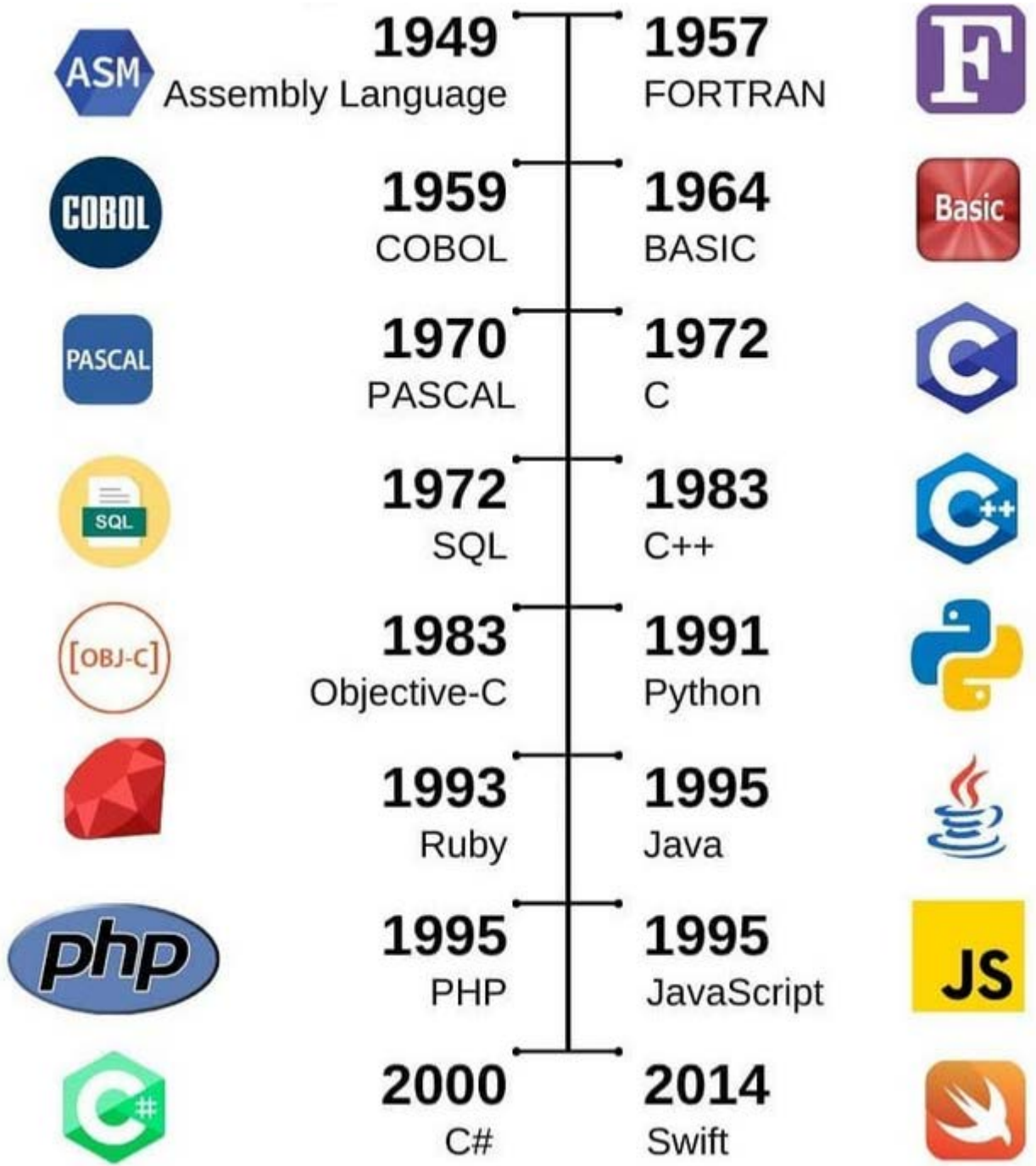


Razones para aprender Python

- Excelente como primer lenguaje
- Simple, fácil de aprender de forma incremental
- Bibliotecas: seguramente el lenguaje con más bibliotecas actualmente
- Interpretado: fácil hacer pruebas, o de ejecutar bloques de pocas líneas

¿Por qué ahora? ¿Por qué no antes?

Evolución de los lenguajes de programación



Hola Mundo en C#

```
using System;


namespace Program
{
    class Program
    {
        public static void Main(string[] args)
        {
            System.Console.WriteLine("Hello, World!");
        }
    }
}
```

Helllo World en Java

```
class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }

}
```

Hello World en Python

```
print("Hello, World!")
```

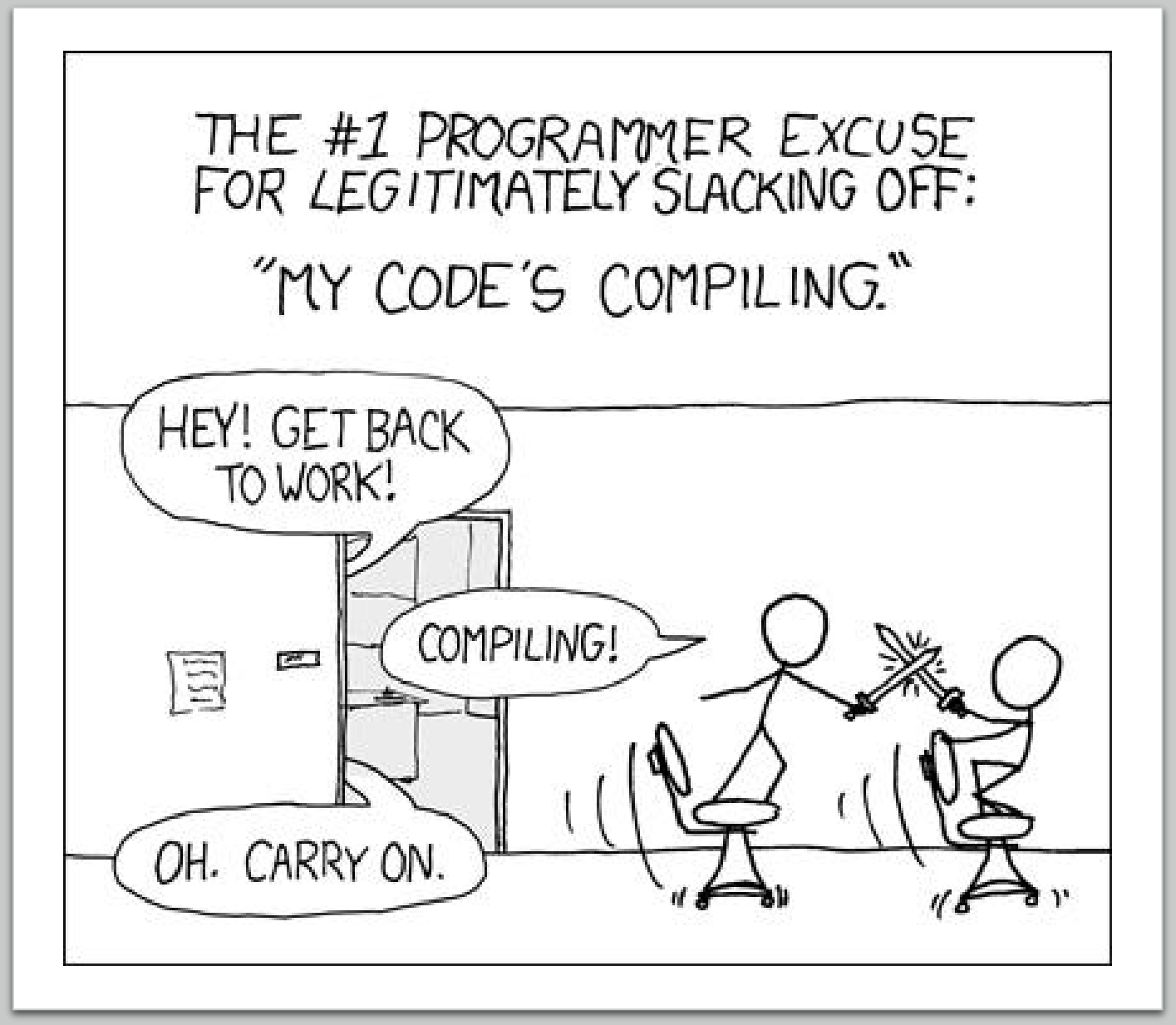
```
>>> import this
The Zen of Python, by Tim Peters

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!
```

Cuando NO utilizar Python

- Proyectos grandes con muchas piezas y muchos individuos involucrados: C#, Java…
- Velocidad: mejor lenguajes compilados de bajo nivel (C,C++) (pero…ver más adelante)
- Tipos creados por el usuario: el usuario no puede definir sus propios tipos simples fácilmente

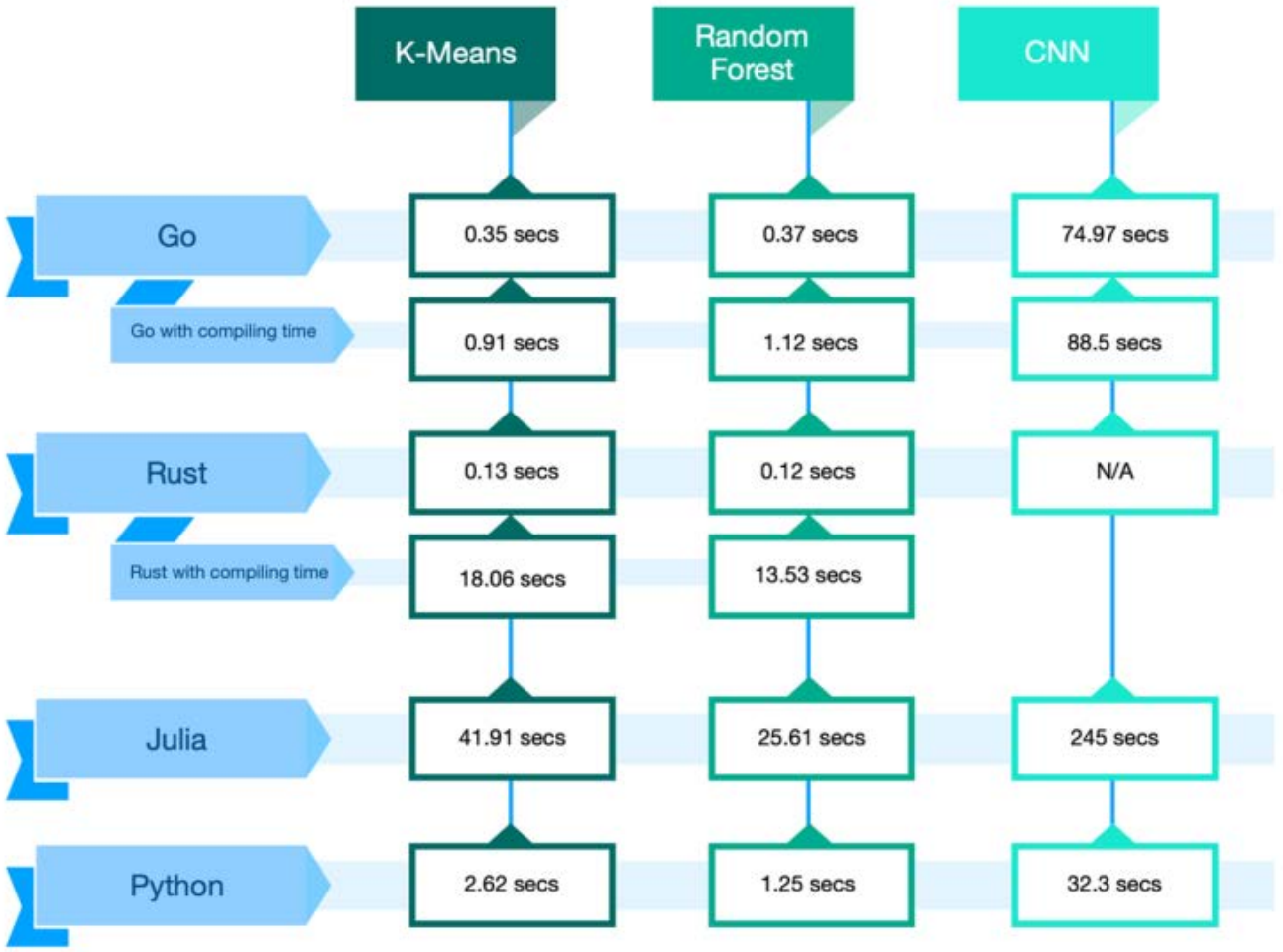
¿Velocidad?

- Lenguaje de programación de propósito general interpretado y no compilado (igual que R)
  - Interpretado: cada instrucción es traducida y ejecutada de forma independiente
  - Compilado: el programa entero se traduce, generando un código que a continuación puede ser ejecutado



Realmente no tan lento…

- La razón es que hace “trampa”
  - Las bibliotecas a menudo están hechas en otro lenguaje
  - Python solo sirve the interface para esas bibliotecas



Versatilidad

- Distintos estilos de programación → Multiparadigma
  - Programación orientada a objetos
  - Programación funcional
  - Programación imperativa
  - Programación procedural
- Combina bien con otros lenguajes
  - Bibliotecas
  - Integrando lenguajes (por ejemplo de bases de datos)

Pero ¿Por qué para el tratamiento y análisis de datos?

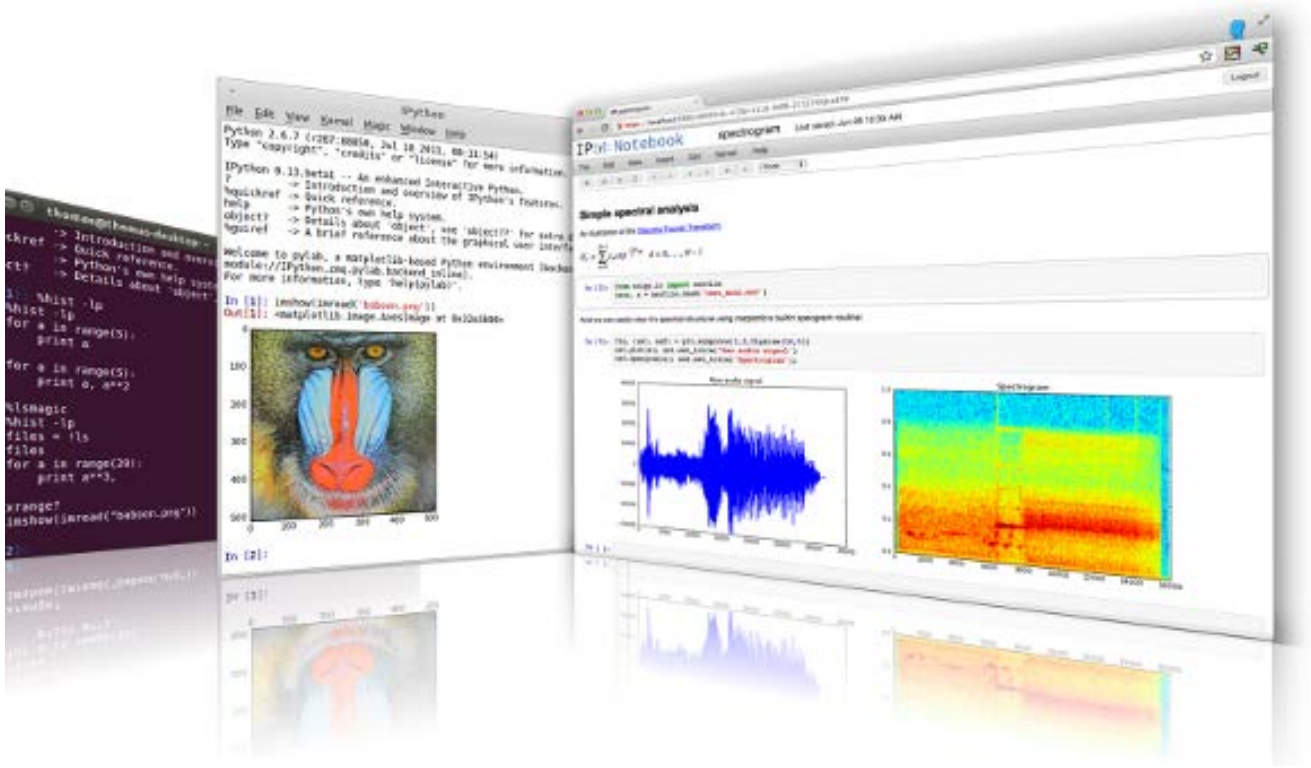
Vale, asumimos que Python es

- Muy utilizado
- Con muchas bibliotecas disponibles

Eso significa que puede ser adecuado para muchos propósitos

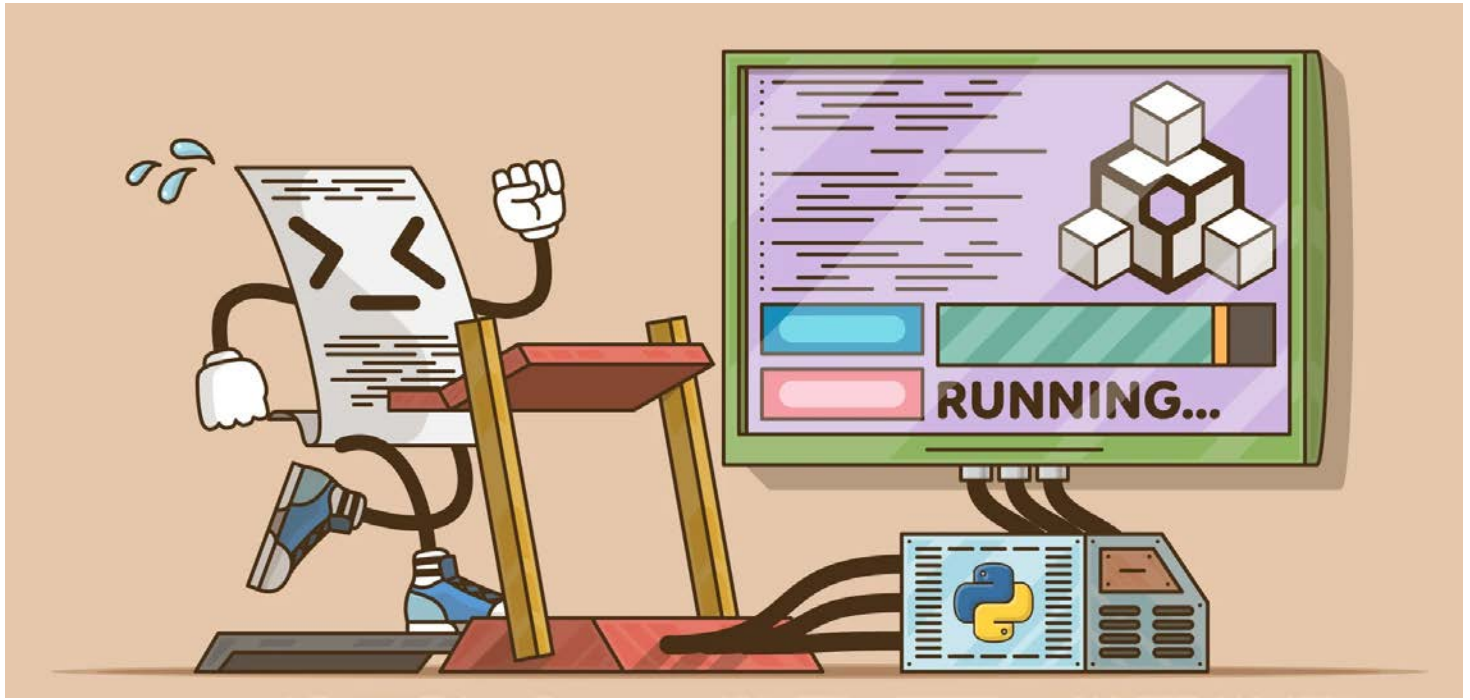
¿Por qué el tratamiento y análisis de datos en particular?

Razón 1: facilidad para realizar experimentos



- Especialmente gracias a entornos tipo Jupyter Notebooks es fácil realizar múltiples pruebas dejando los resultados documentados
  - No es necesario escribir un programa completo, los resultados se ven inmediatamente permitiendo aceptar o descartar en el momento
  - Hay que generar poco código
  - Los resultados van quedando acumulados en el propio *notebook*

Razón 2: scripts

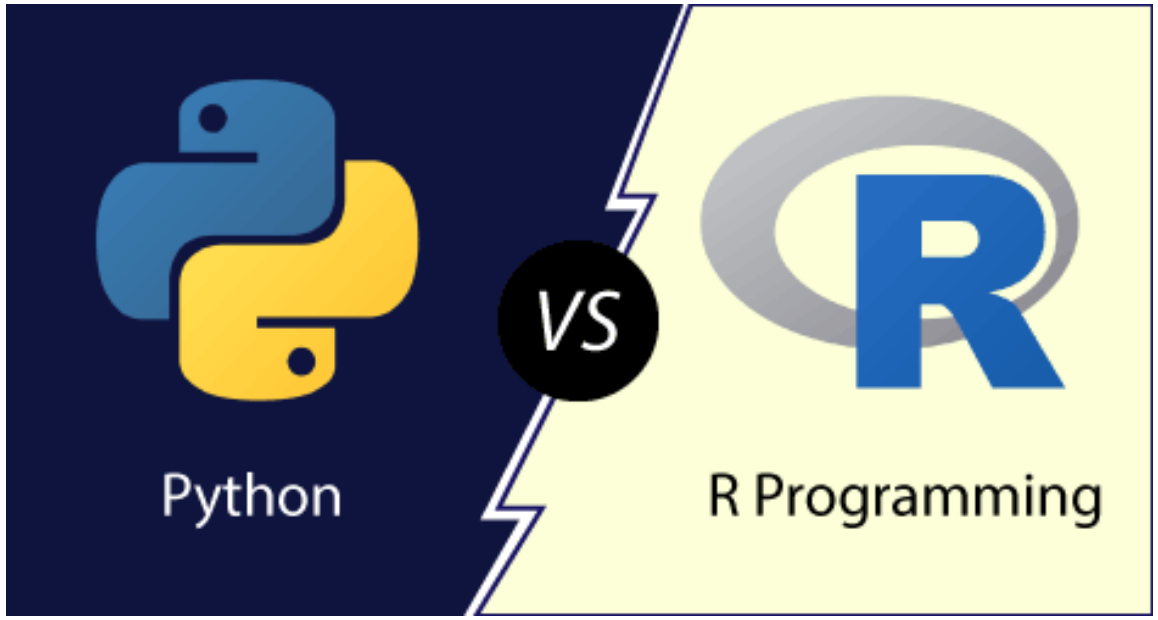


- El tratamiento de datos a menudo se estructura en pasos que se ejecutan secuencialmente
- Tareas que explicitadas en pasos que se repiten a menudo: scripts
- Por eso a menudo se dice que Python es un lenguaje de scripting

Vale, ¿pero por qué no R?

- Otro lenguaje con algunas características similares
- También apto para tratar datos

¿Cuál elegir?



Diferencias

Lenguaje de propósito general

Análisis de datos

Lenguaje Orientado a estadística

Diferencias

- Orientado a ingeniería de datos
  - Más rápido
  - Mejor manejo de la memoria
  - Mejores interfaces con bases de datos y big data
  - Más métodos de aprendizaje automático
  - Lenguaje de uso más general: más complejo y potente (ej. Desarrollo web)

- Orientado a estadística
  - Mejores tests estadísticos
  - Más fácil hacer pruebas rápidas
  - Gráficos más sencillos e integrados
  - Bibliotecas para un acceso más sencillo a los datos
  - Más orientado a no programadores

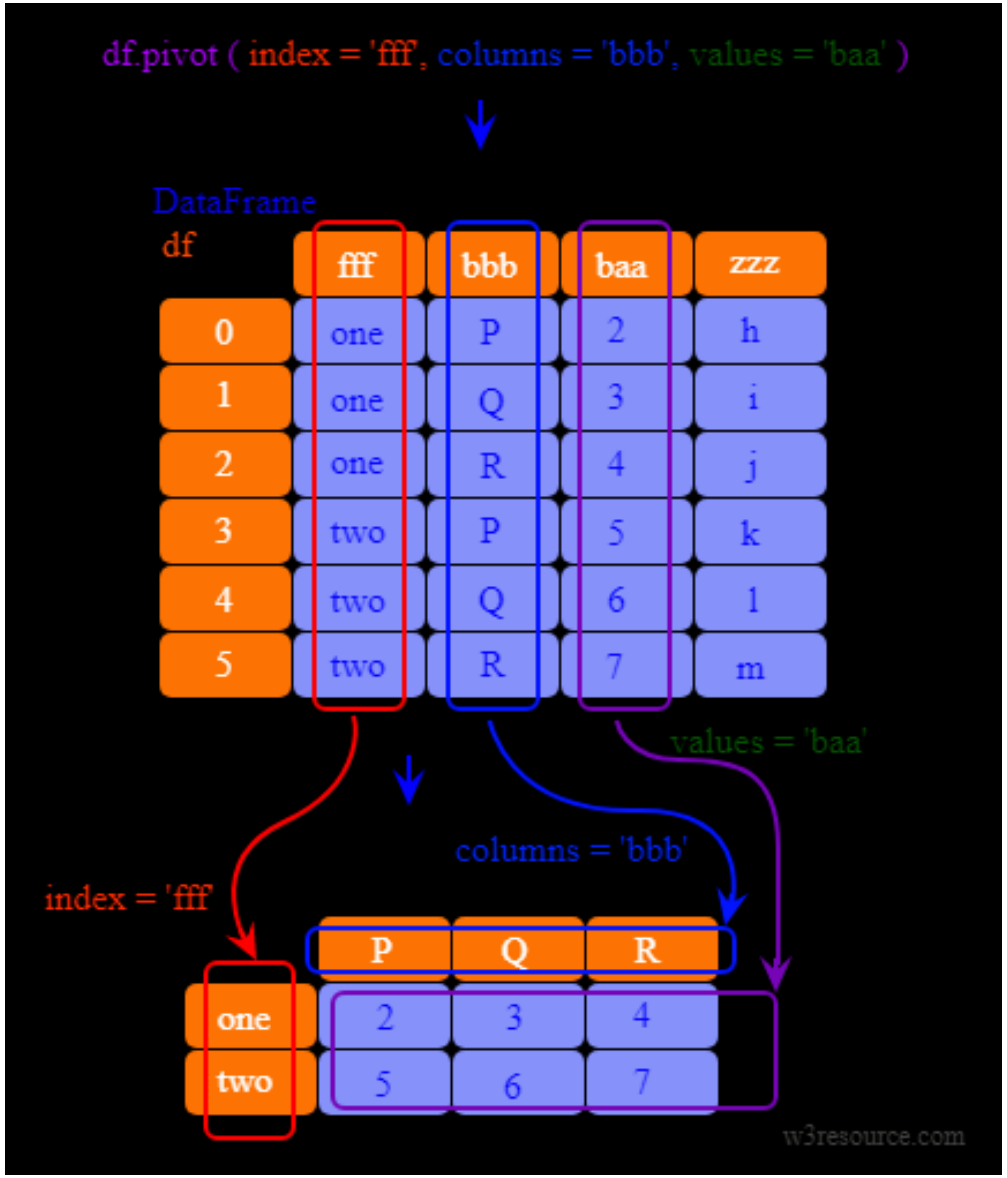
python™

¿para que? campos en los que se aplica

Gestión

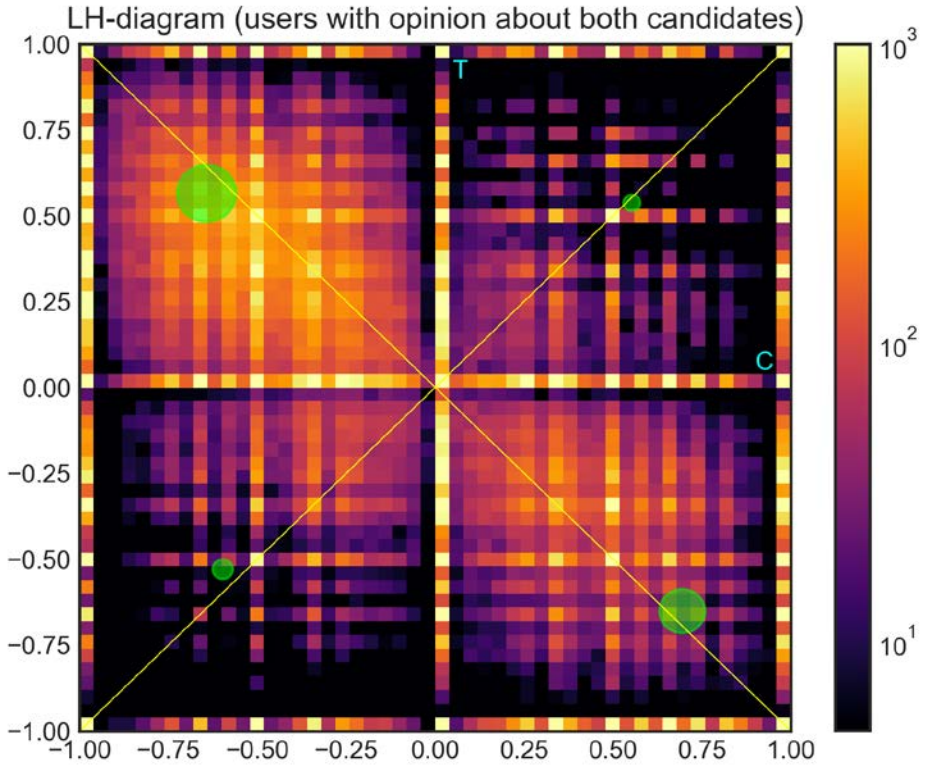
- Especialmente útil para tareas que queramos automatizar y repetir a menudo con un solo click
- También tareas complejas difíciles de realizar por ejemplo en Excel
  - División de un fichero en varios
  - Operaciones complejas con ficheros

- Ejemplos: gestion1,gestion2,gestion3



Gráficos y estadísticas

- Entender bien los datos
- Diferentes visualizaciones
- Grabar el gráfico en disco
- Detectar outliers, tipos de distribuciones ,etc
- Ejemplo: graficas_y_estadísticas



Captura de datos web

- Tomar datos de páginas de forma automática
- 2 casos principales:
  - Los datos están directamente en la página (biblioteca BeautifulSoup)
  - Hay que interaccionar con la página (biblioteca selenium)

- Ejemplos: scraping1, scraping2

Análisis de textos

- Intentar detectar si un texto tiene sentido positivo o negativo (análisis de sentimiento)
- Detectar los nombres de los que habla
- Nubes de palabras y otras visualizaciones
- Ejemplo: textos

Aprendizaje automático

- Supervisado: predecir una columna
  - Regresión: la columna tiene datos numéricos continuos
  - Clasificación: la columna toma una cantidad pequeña de valores

- No supervisado: no hay columna para predecir
  - Clustering

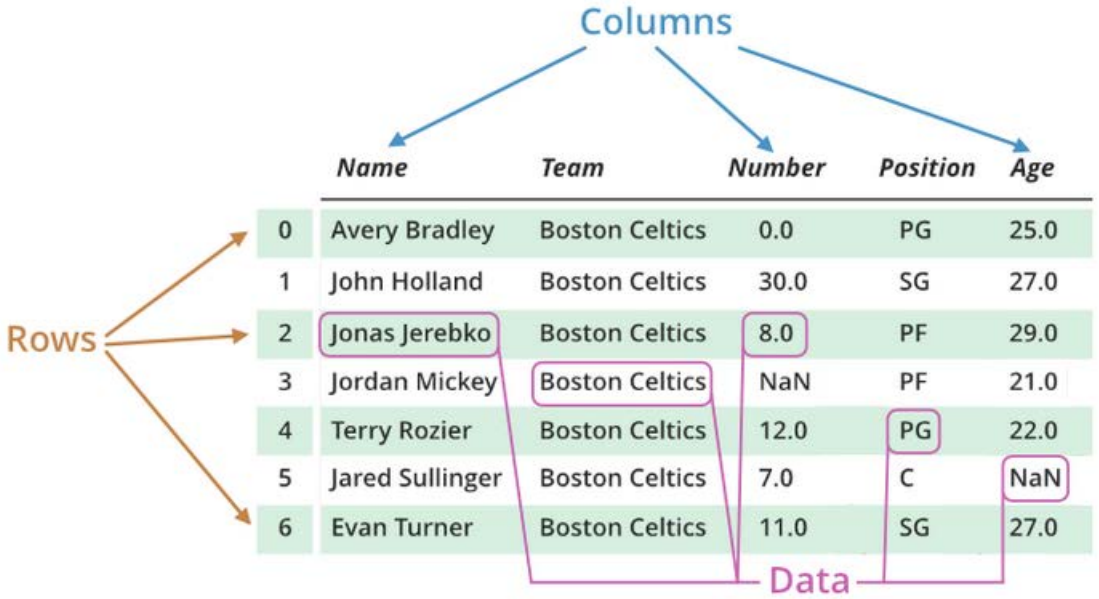
- Ejemplos: ml1,ml2, ml3, ml4

Bibliotecas más comunes (en el entorno de ciencia de datos)

Pandas

la biblioteca de tratamiento de datos por excelencia; en muchas empresas está sustituyendo a Excel para scripts tratamiento complejo de datos

- Tablas de datos (dataframes)
- Se basa en numpy para representar filas, columnas, etc.



fuente

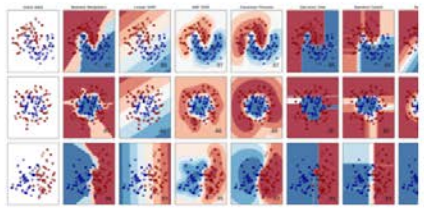
Scikit-learn

aprendizaje automático en Python: regresión, clasificación, clustering

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

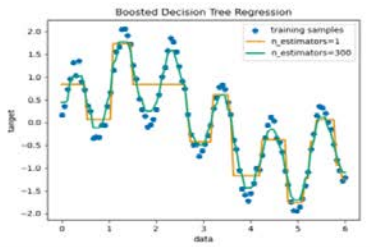


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

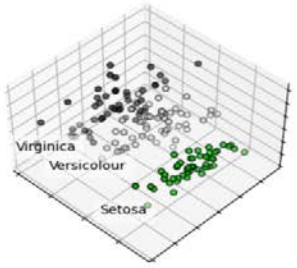
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization, and more...

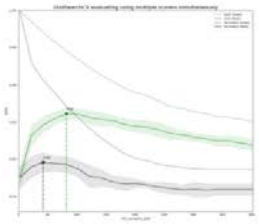


Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...

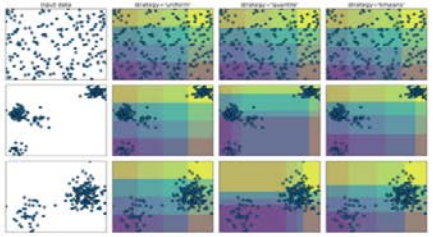


Examples

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...

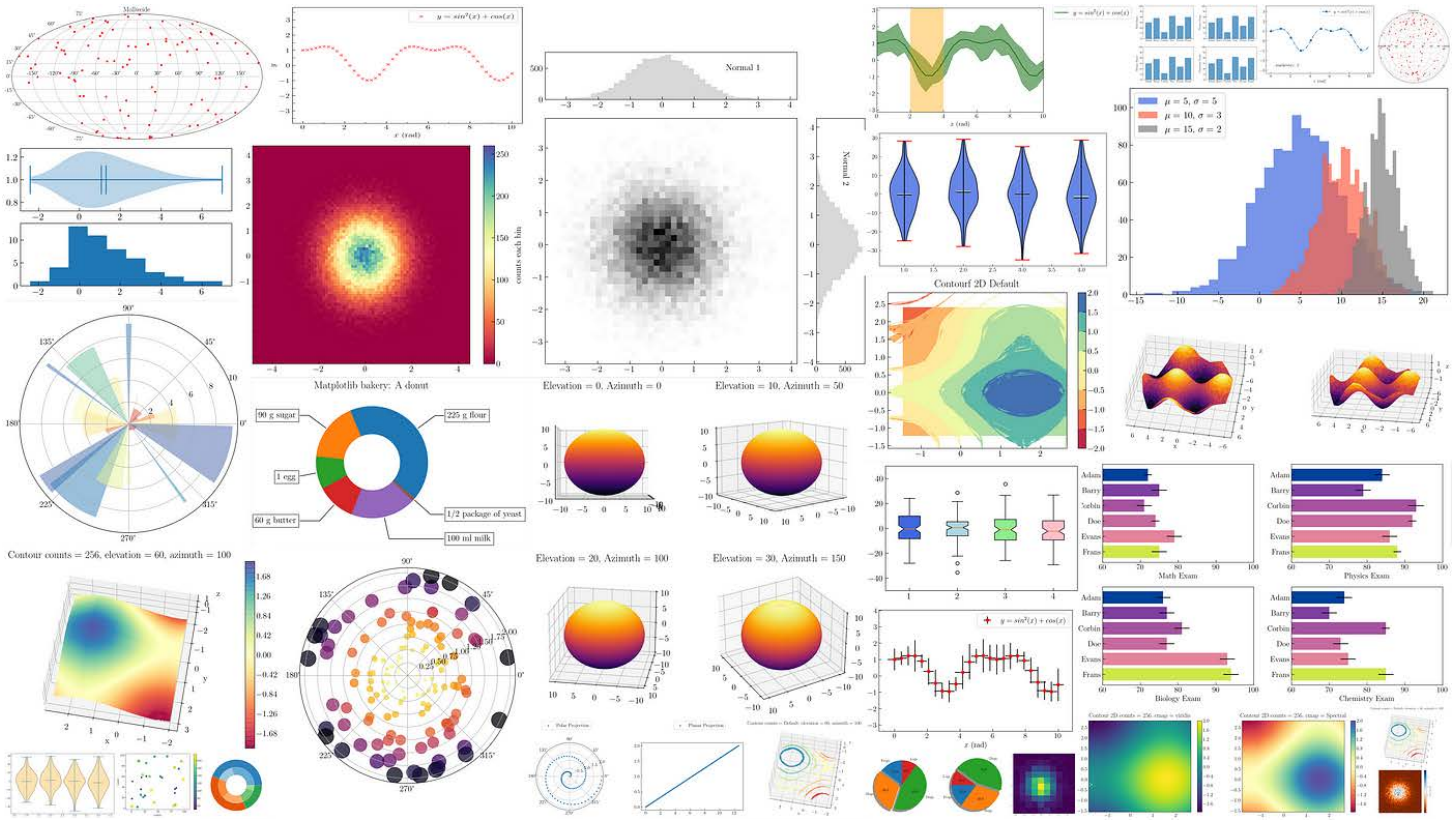


Examples

Matplotlib

Gráficas de todo tipo, muy versátil y adaptable

A menudo sustituida por Seaborn en ciencia de datos para gráficos más “elegantes”



Tensorflow

El entorno de Google para el desarrollo de modelos de aprendizaje automático

- Aprovechamiento de hardware para el procesamiento paralelo

- Bibliotecas de apoyo para deep learning como Keras

Pyspark

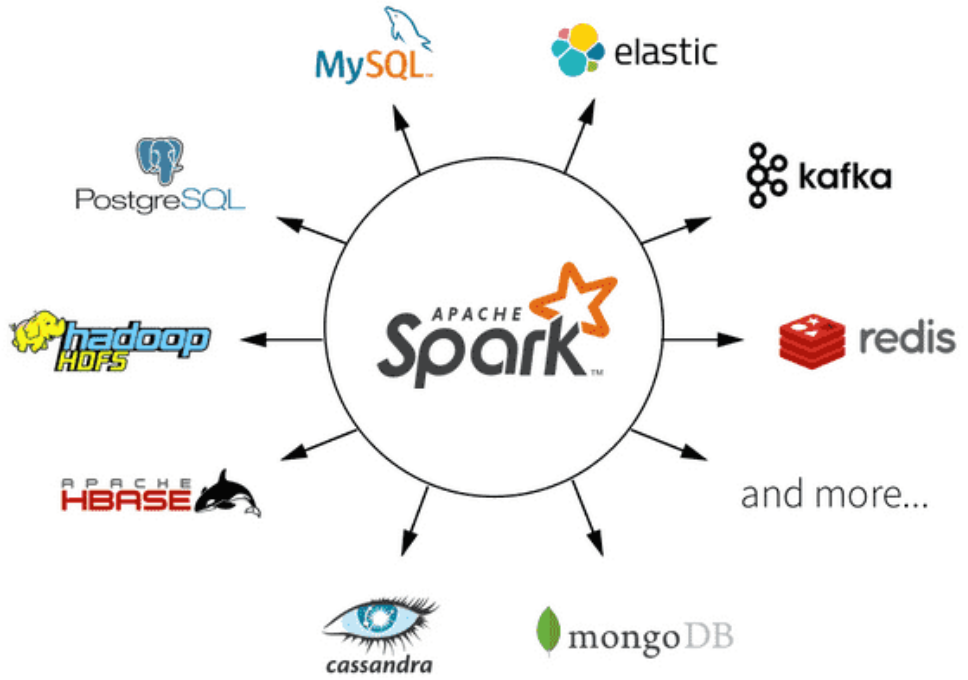
Cuando los datos no caben en memoria

Soporte para Big Data

Cómputo distribuido en clúster

Tanto preprocesamiento como aprendizaje automático

También disponible en Scala, Java, R



Otras bibliotecas

- Scypy: Estadísticas de todo tipo, álgebra lineal…
- Theano: cómputo científico a gran escala
- PyTorch : alternativa o complemento a TensorFlow
- LightGBM: modelos “gradient boosting”

¿Cómo programar en Python?: entornos

Python es el lenguaje, pero se puede utilizar en diferentes entornos

Específicos para Python

PyCharm → completo, versión gratuita y de pago, incluye depurador

Spyder → sencillo, orientado a ciencia de datos, parte de Anaconda

Thonny → Ideal para principiantes

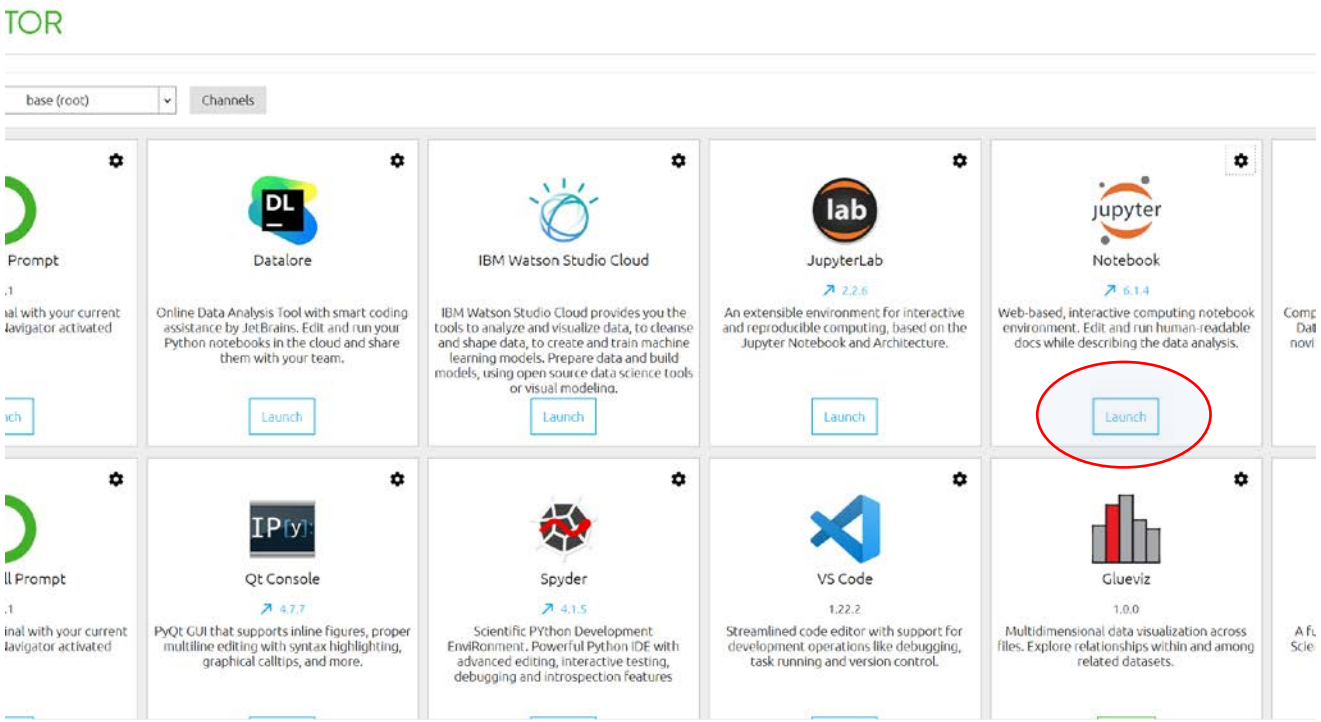
Generales

Eclipse + PyDev → editor muy empleado en entornos Java. Interesante si ya se está acostumbrado a su "filosofía". En otro caso hay mucho que aprener

Sublime → editor general muy utilizado. No es gratuito. Mejor para usuarios avanzados

Visual Code Studio: seguramente el editor más utilizado por los programadores junto con sublime por su variedad de posibilidades, y es gratis.

Jupyte Notebooks - Anaconda



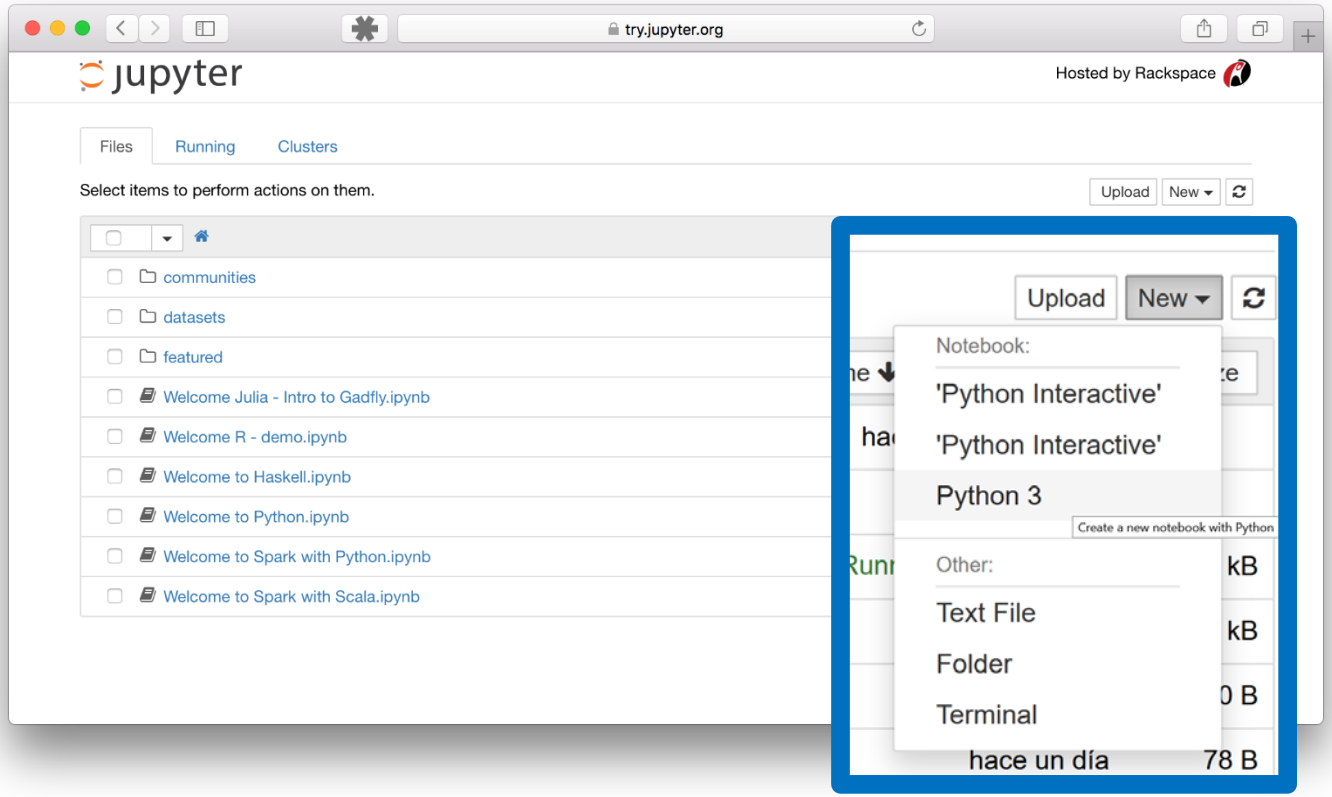
- Anaconda es un sistema con muchas aplicaciones para ciencia de datos
- Nosotros vamos a usar los Jupyter Notebooks

Entorno Jupyter

Al abrirse los Jupyter Notebooks veremos algo como

Aquí podremos abrir nuestros notebooks → combinaciones de código y texto

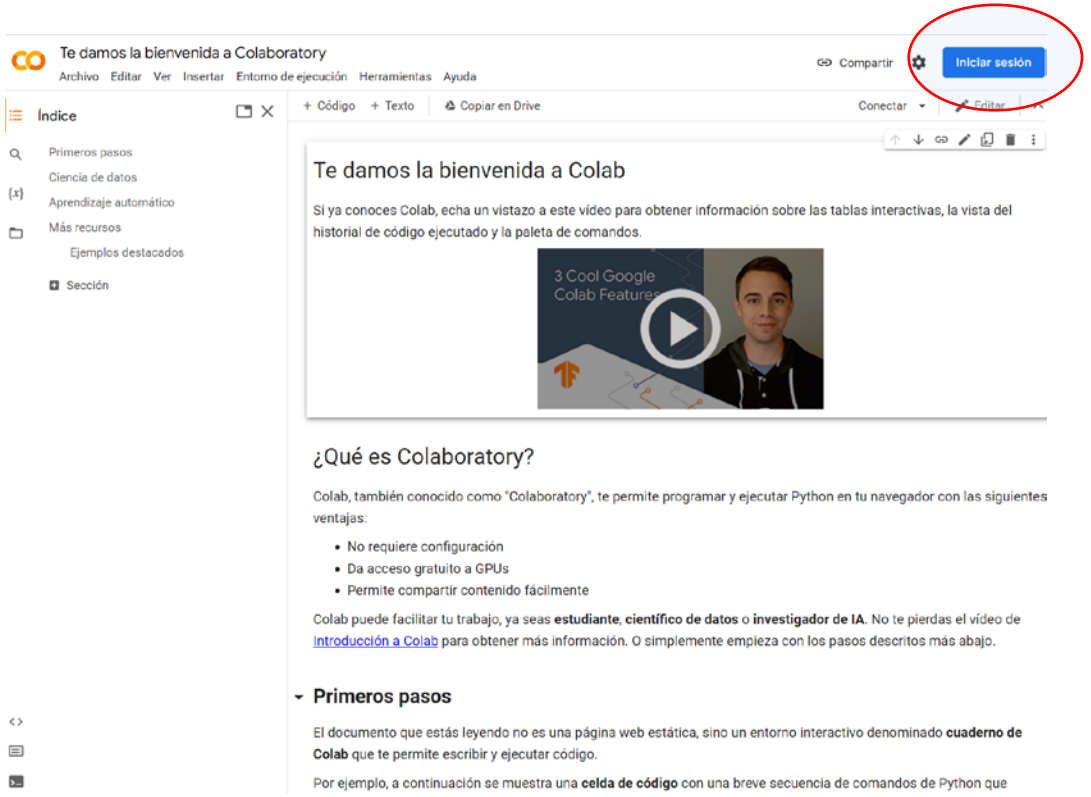
Elegiremos New + Python 3



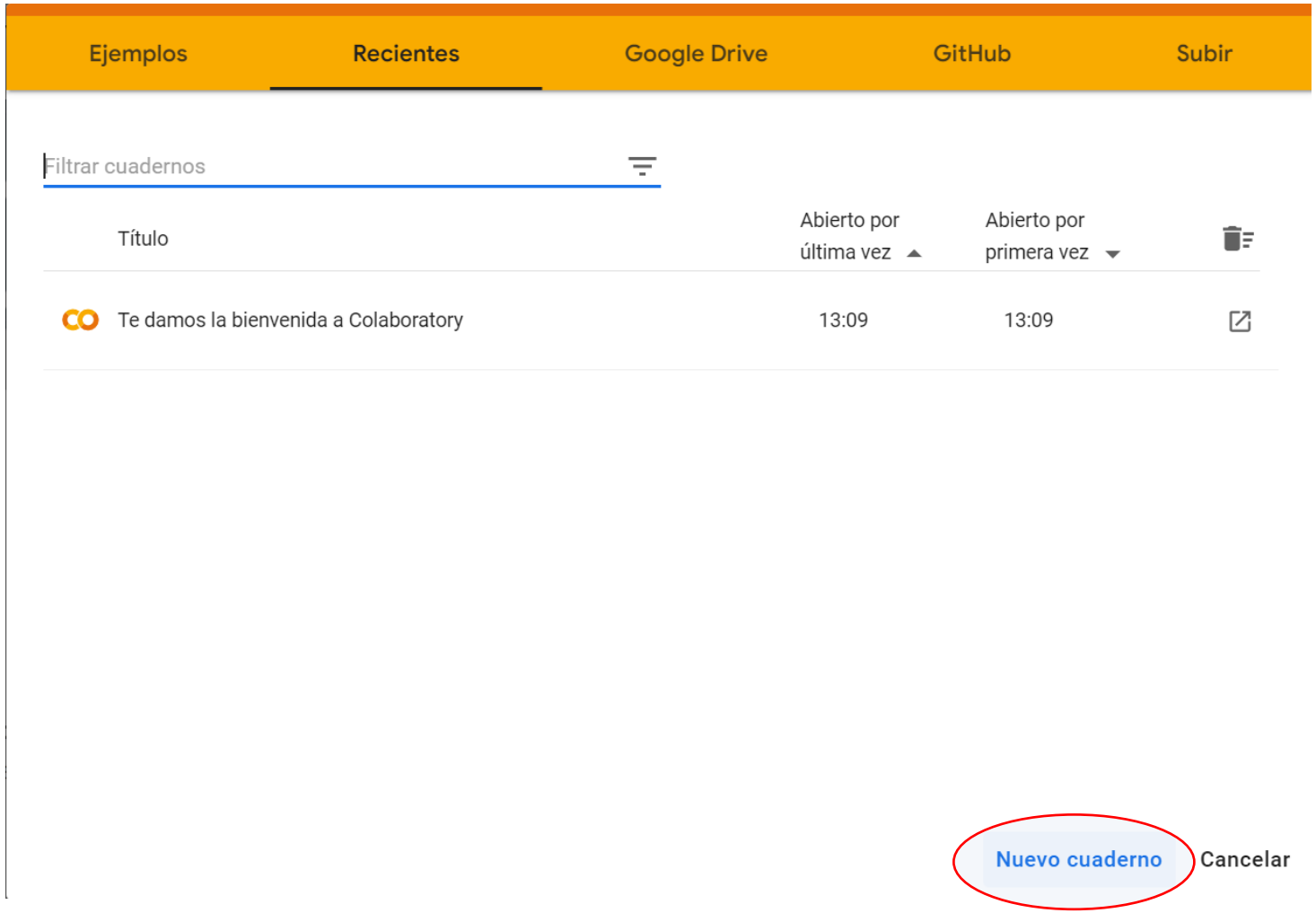
Jupyte Notebooks – Google Colab

- Desde un navegador accedemos a https://colab.research.google.com/

- Debemos iniciar sesión con una dirección de gmail

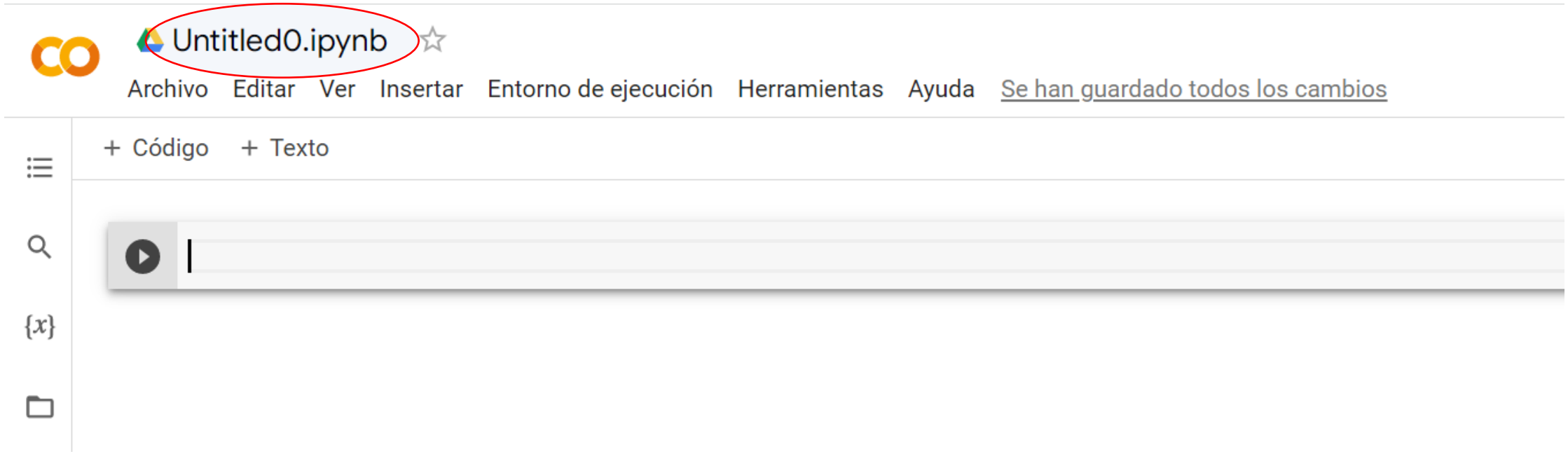


Creando un Notebook

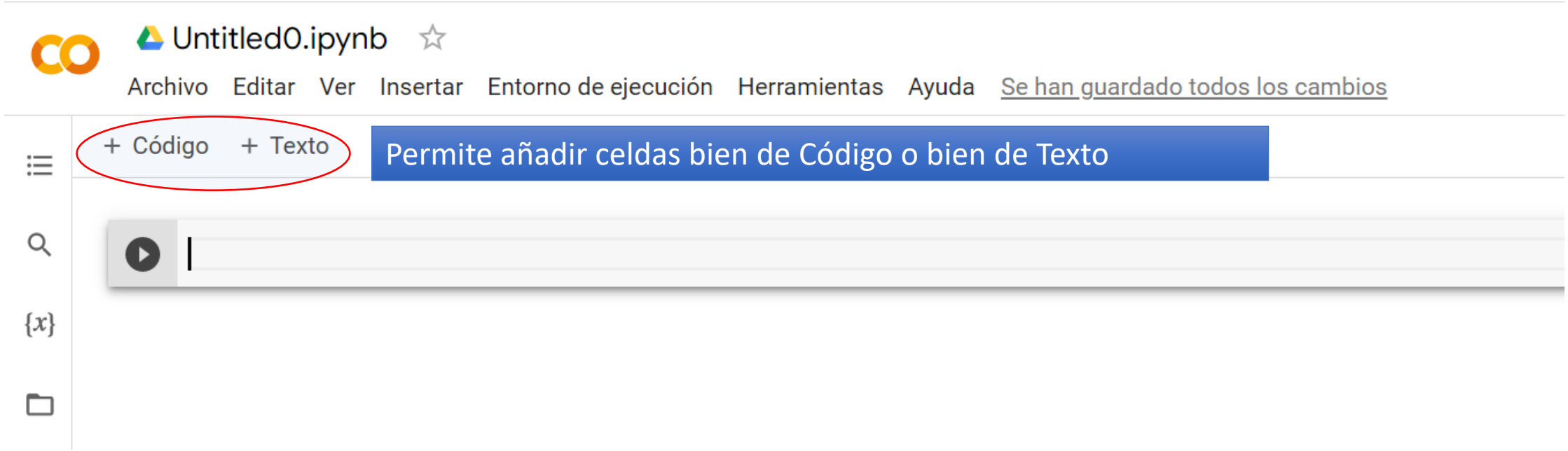


Notebook con colab

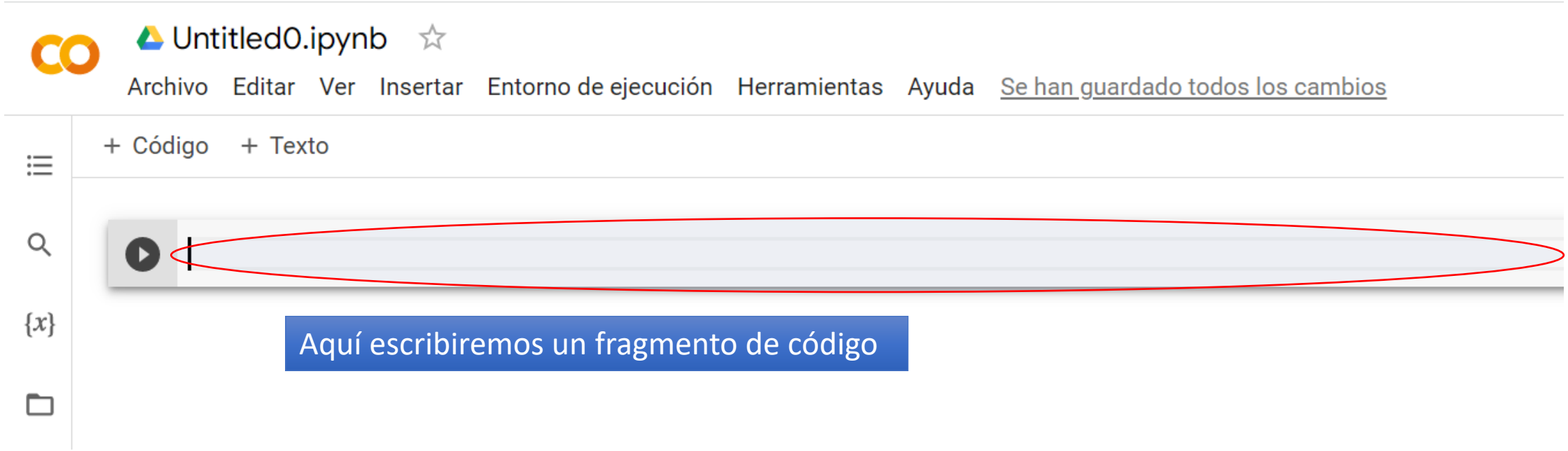
Nombre del cuaderno: se puede cambiar hacienda click encima



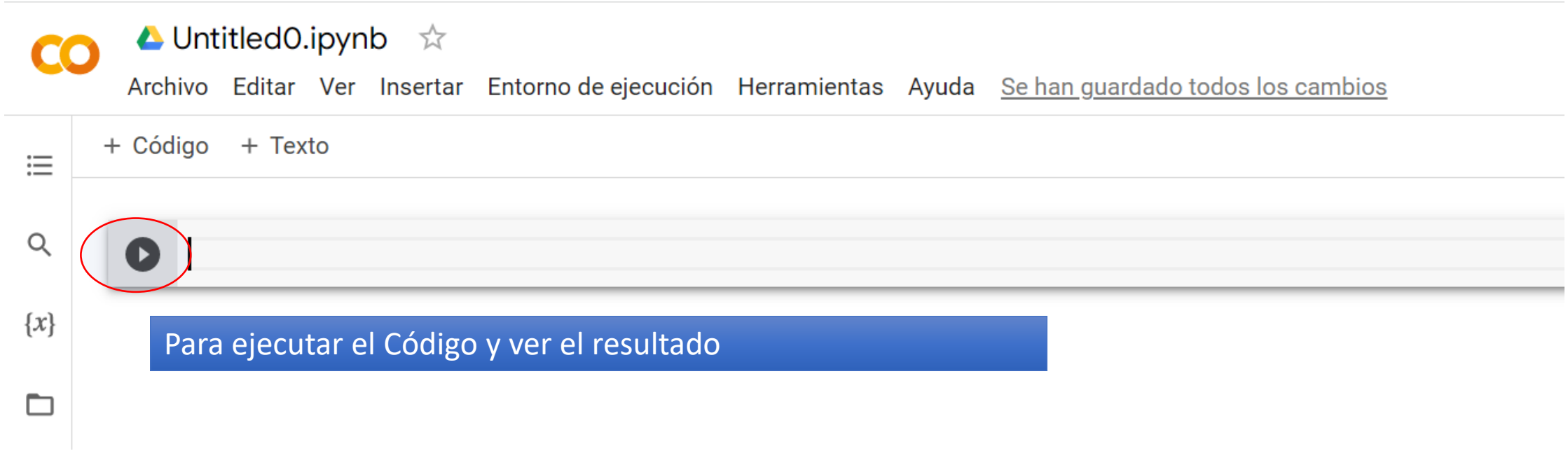
Notebook con colab



Notebook con colab

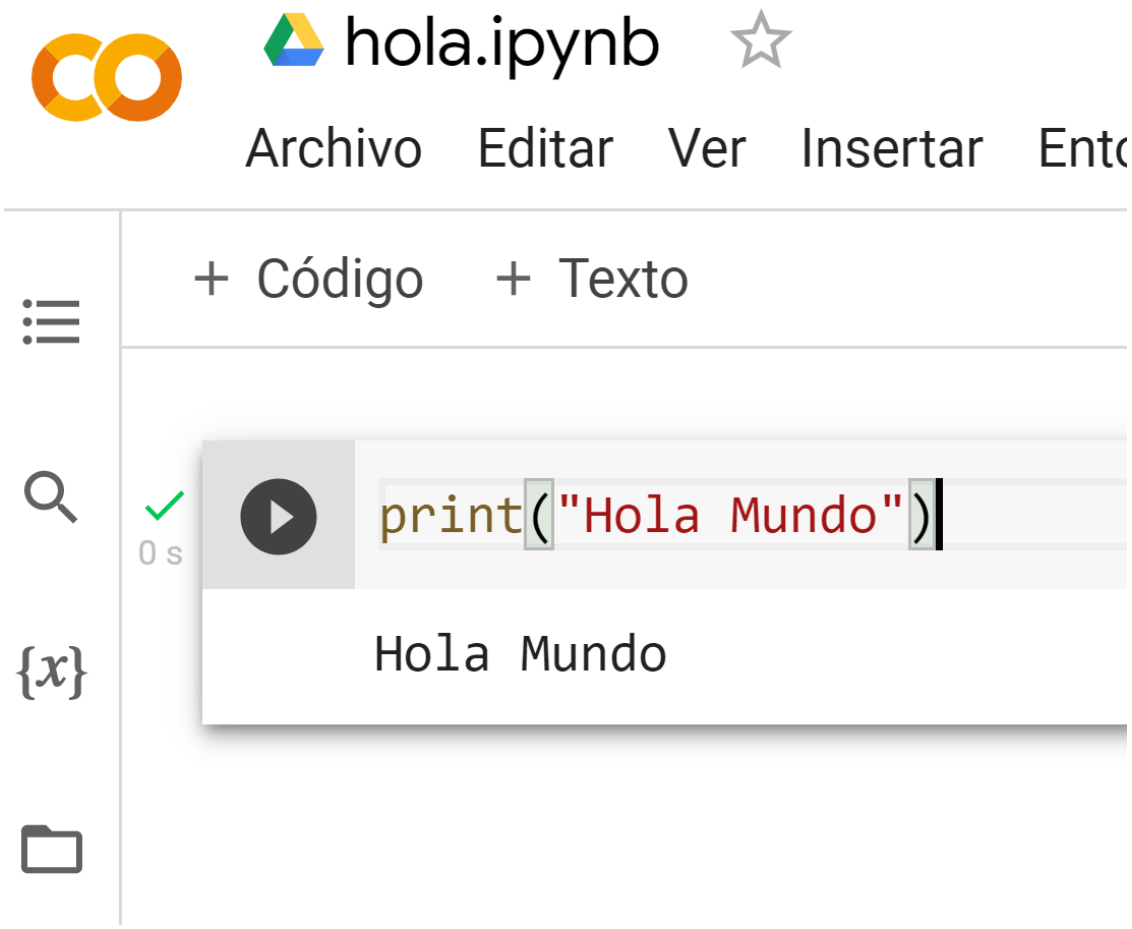


Notebook con colab



Nuestro primer programa

Nuestro primer programa



Programando con notebooks

- Un programa va a ser una combinación de
  - Casillas de texto: explicaciones, comentarios, etc. Formato markdown.
  - Casillas de código: un fragmento de programa
    - Influye lo que se haya ejecutado anteriormente
    - Se muestra el resultado de la última instrucción

Hola Mundo (nb)

Nuestro primer programa

```
print("Hola Mundo")
```

Mucho que entender en solo una línea

- print: función predefinida
- "Hola Mundo" un valor de tipo String que se pasa como argumento
- Las funciones tienen
  - Un *código* (que en este caso no vemos porque está predefinida)
  - Un *nombre* (print)
  - Unos argumentos (en este ejemplo "Hola Mundo") → su comunicación con el exterior
  - Devuelven valores, o efectúan acciones

Llamada a una función

- Aspecto general

f(e1, e2,…,en)

Efecto

1) Se evalúan los argumentos e1→v1, e2→v2, …, en→vn
2) Se ejecuta la función con v1, v2,…vn como argumentos

Ejemplos (para jugar, nb)

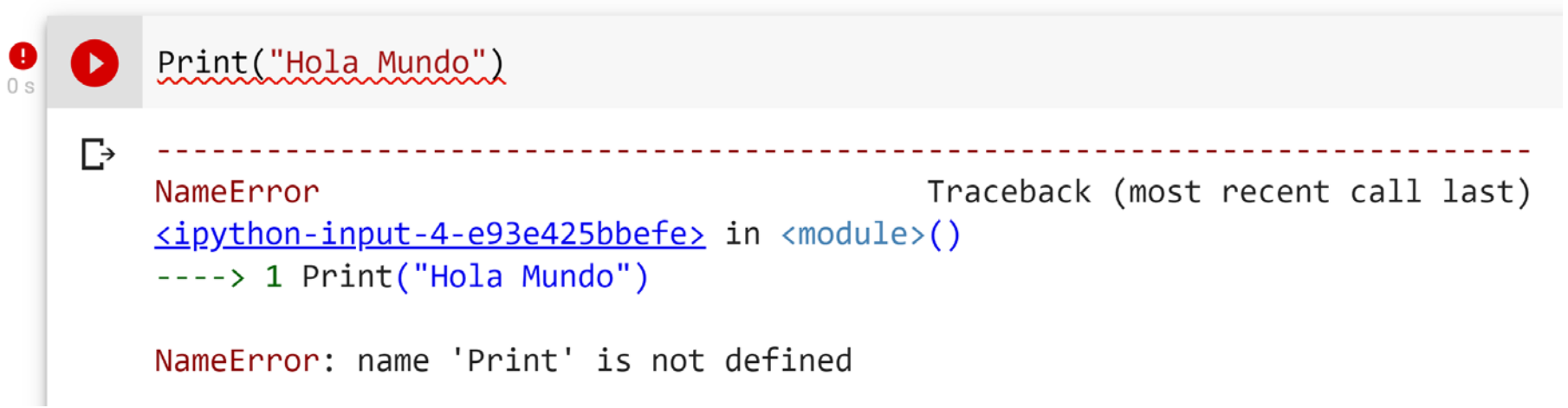
Probar y comentar el resultado

- `Print("Hola")`
- `print('hola')`
- `print(3+4)`
- `print("Tengo ",23+4, " años ")`
- `print(23+ " hola ")`
- `print(hola)`

Ejemplos (para jugar, nb)

Probar y comentar el resultado

- `Print("Hola")`
- `print('hola')`
- `print(3+4)`
- `print("Tengo ",23+4, " años ")`
- `print(23+ " hola ")`
- `print(hola)`



Hola Mundo en Java

No tan fácil

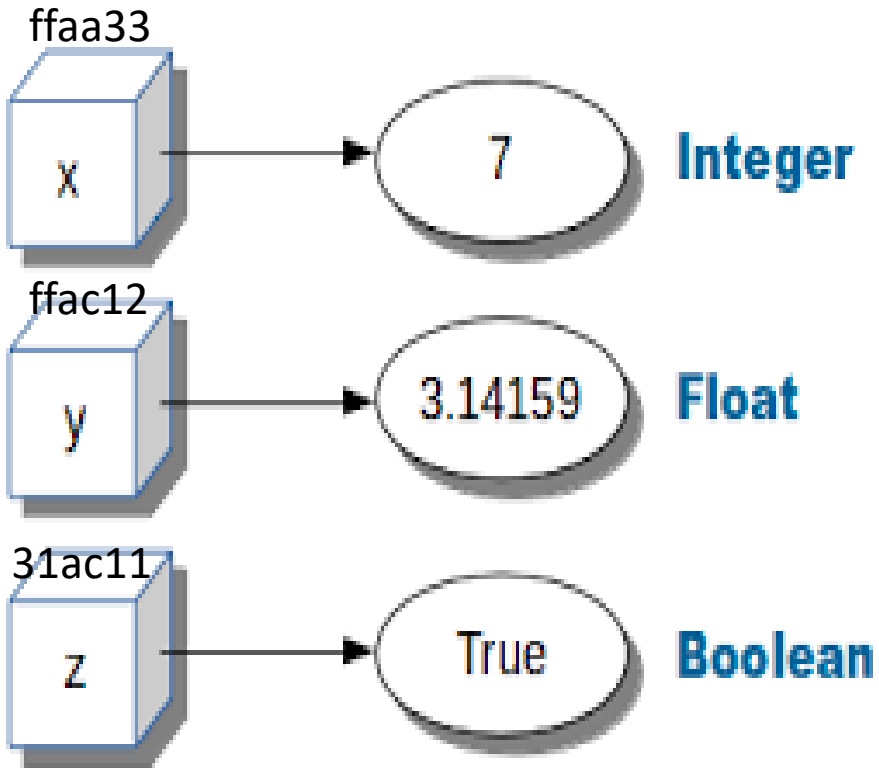
```
1.public class HolaMundo {
2.
3.  public static void main(String[] args) {
4.System.out.println("Hola Mundo");
5.}
6.
7.}
```

Variables

Una variable tiene 4 componentes

- Un nombre, que le ponemos nosotros
- Una dirección de memoria, que no vemos (ni nos suele importar)
- Un tipo que no vemos (y que sí nos suele importar)
- Un valor
- Al evaluar la variable, Python la convierte en su valor → si no tiene valor aun → error

- NB



Nombres de variables (identificadores)

- Mezclas de dígitos, letras (mayúsculas y minúsculas) y _

Ejemplos: contador, nombre, edad_3, SaldoFinal

- No pueden empezar por un dígito

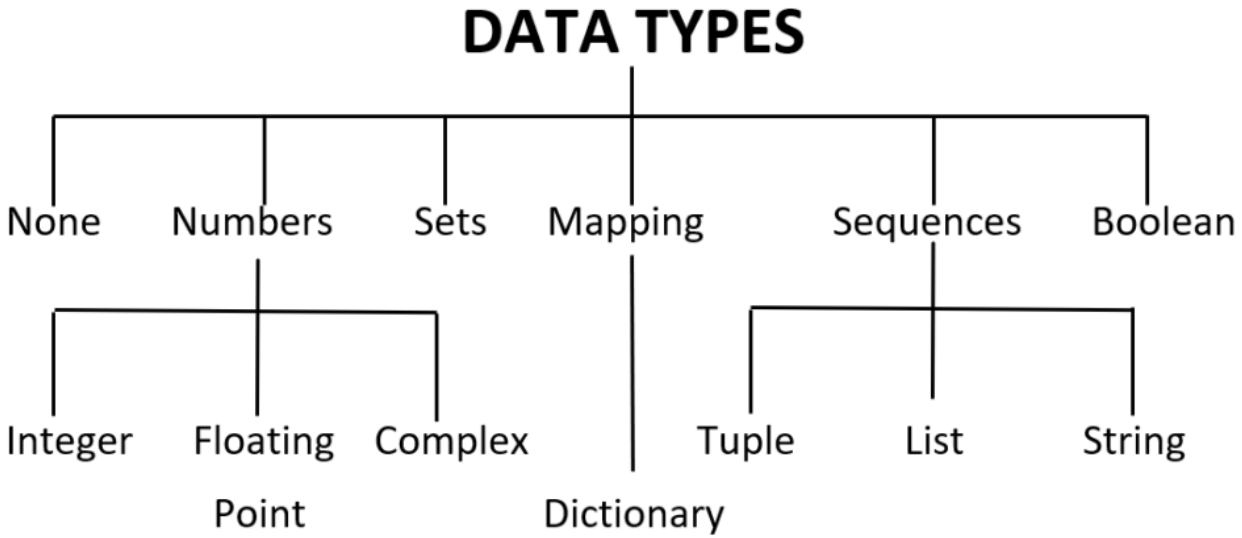
- Recordar que Python considera mayúsculas y minúsculas diferentes

- No puede utilizarse una palabra reservada

Palabras reservadas en Python

`False`	`class`	`finally`	`is`	`return`
`None`	`continue`	`for`	`lambda`	`try`
`True`	`def`	`from`	`nonlocal`	`while`
`and`	`del`	`global`	`not`	`with`
`as`	`elif`	`if`	`or`	`yield`
`assert`	`else`	`import`	`pass`	
`break`	`except`	`in`	`raise`	

Tipos en Python



- Se puede ver el tipo de una expresión con la función type (NB)
- Frecuente fuente de errores de ejecución

Python: librerías

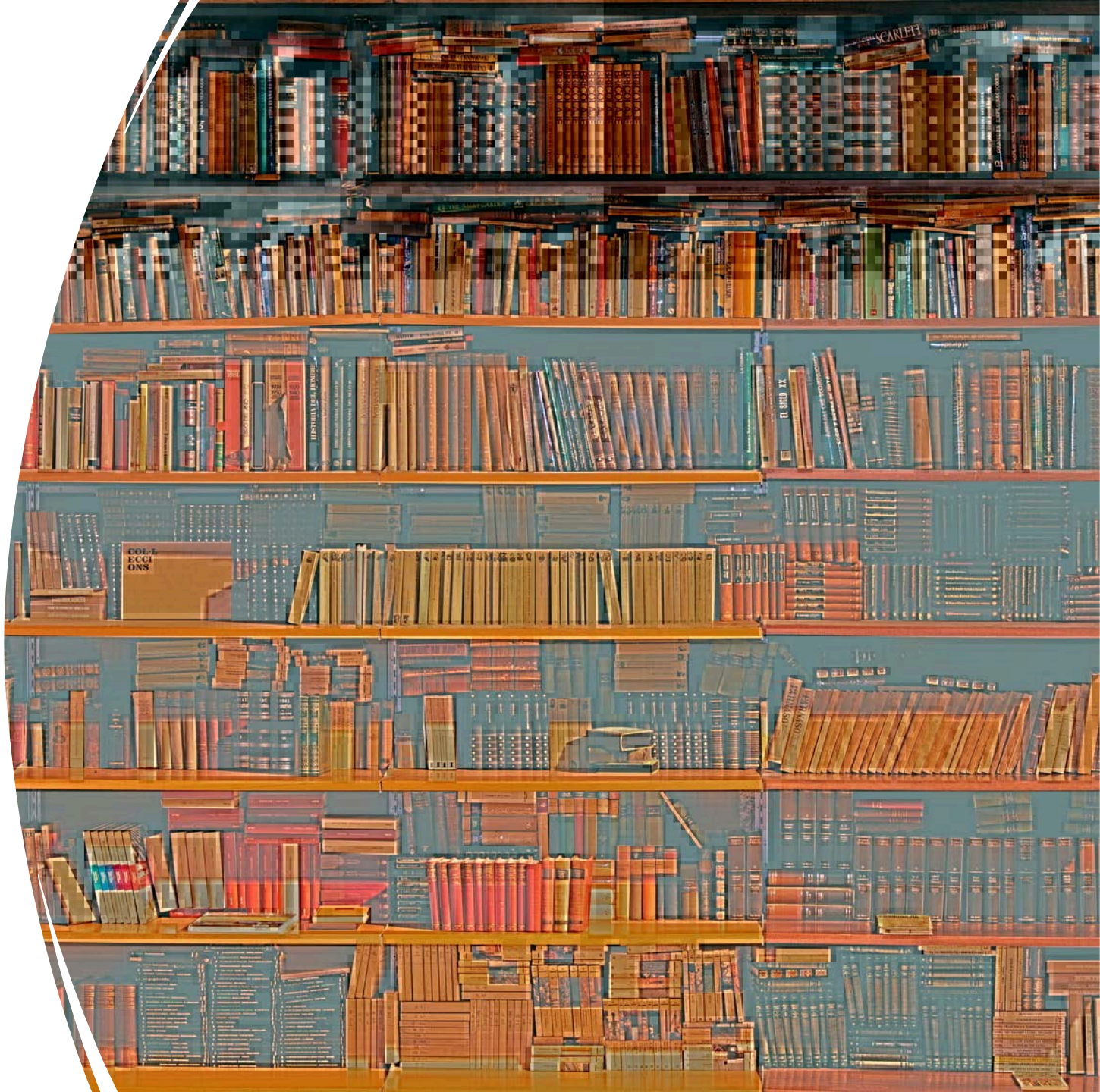
Las librerías deben ser

1) Descargadas si son nuevas(esto solo se hace una vez)

pip install libreria

2) Cargadas (esto se hace en cada sesión)

import pandas as pd



Carga de ficheros

Para ficheros CSV y Excel usaremos pandas, para otros formatos necesitaremos librerías adecuadas

Pandas read_csv, read_excel

- Permiten leer ficheros separados por un carácter y ficheros excel
- En ambos casos devuelven un *dataframe:* una tabla en Pandas que representa el conjunto de datos en memoria
- Algunos parámetros comunes de read_csv
  - sep: el separador, por defecto “,”
  - Header: para indicar si la primera línea contiene la cabecera (por defecto True)
  - Thousands, decimal: separadores de miles y de decimales
  - encoding: codificación de caracteres. Deber ser una codificaciones estándar

¡Gracias!

Disclaimer

This material has been prepared by Bolsas y Mercados Españoles, Sociedad Holding de Mercados y Sistemas Financieros S. A. (BME), its subsidiaries, affiliates and/or their branches (together, “BME”) for the exclusive use of the persons to whom BME delivers this material. This material or any of its content is not to be construed as a binding agreement, recommendation, investment advice, solicitation, invitation or offer to buy or sell financial information, products, solutions or services. The information does not reflect the firm positions (proprietary or third party) of the entities involved in the Spanish Securities Market. BME is under no obligation to update, revise or keep current the content of this material, and is subject to change without notice at any time. No representation, warranty, guarantee or undertaking – express or implied – is or will be given by BME as to the accuracy, completeness, sufficiency, suitability or reliability of the content of this material.

The opinions presented are theoretical and, therefore, the content hereof is intended for informational purposes only and should not be used for portfolio or asset valuations, or as the basis for any investment recommendations. Neither contributing Entities, nor Bolsas y Mercados Españoles, Sociedad Holding de Mercados y Sistemas Financieros S.A.(BME) nor any of its subsidiaries, accept responsibility for any financial loss or decision made based on the information contained in this material. In general, neither Bolsas y Mercados Españoles, Sociedad Holding de Mercados y Sistemas Financieros S. A. (BME) nor any of its subsidiaries, nor the contributing Entities, their directors, representatives, associates, subsidiaries, managers, partners, employees or advisors accept any responsibility for this information or unauthorised use of the same.

Este material ha sido preparado por Bolsas y Mercados Españoles, Sociedad Holding de Mercados y Sistemas Financieros S. A. (BME) y/o sus filiales (en conjunto, “BME”) para el uso exclusivo de las personas a las que a las que BME entrega este material. Este material o cualquiera de sus contenidos no debe interpretarse como un acuerdo vinculante, una recomendación, un consejo de inversión, solicitud, invitación u oferta de compra o venta de información financiera, productos, soluciones o servicios. Dicha información tampoco es un reflejo de posiciones (propias o de terceros) en firme de los intervinientes en el Mercado de Valores Español. BME no tiene ninguna obligación de actualizar, revisar o mantener al día el contenido de este material, y estará sujeto a cambios sin previo aviso en cualquier momento. Ninguna representación, garantía o compromiso -expreso o implícito- es compromiso -expreso o implícito- es o será dado por BME en cuanto a la exactitud, integridad, suficiencia, idoneidad o fiabilidad del contenido de este material.

Al reflejar opiniones teóricas, su contenido es meramente informativo y por tanto no debe ser utilizado para valoración de carteras o patrimonios, ni servir de base para recomendaciones de inversión. Ni las Entidades contribuidoras, ni Bolsas y Mercados Españoles, Sociedad Holding de Mercados y Sistemas Financieros S.A. (BME) ni de ninguna de sus filiales, serán responsables de ninguna pérdida financiera, ni decisión tomada sobre la base de la información contenida en este material. En general, Bolsas y Mercados Españoles, Sociedad Holding de Mercados y Sistemas Financieros S. A. (BME) ni ninguna de sus filiales, ni las Entidades contribuidoras, sus administradores, representantes, asociados, sociedades controladas, directores, socios, empleados o asesores asumen responsabilidad alguna en relación con dicha información, ni de cualquier uso no autorizado del mismo.