



Estructura de bases de datos y programación en Python

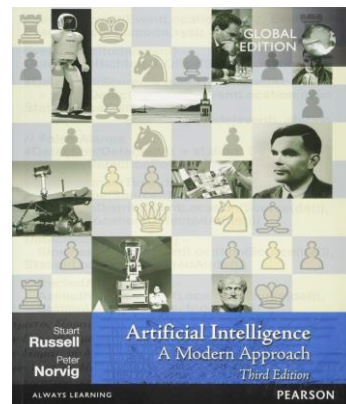
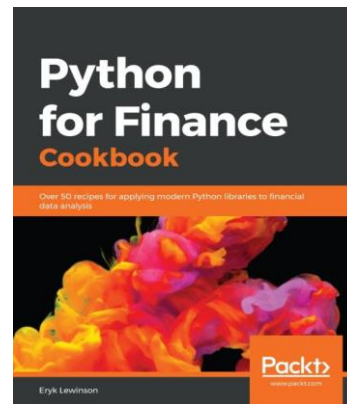
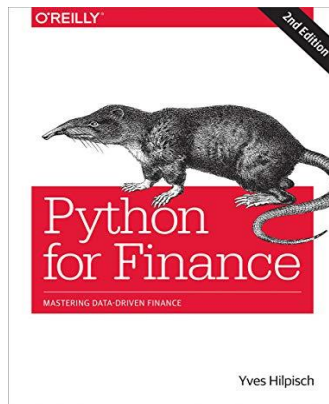
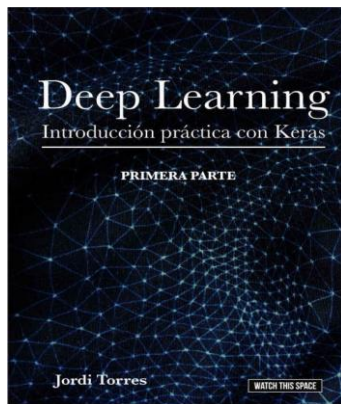
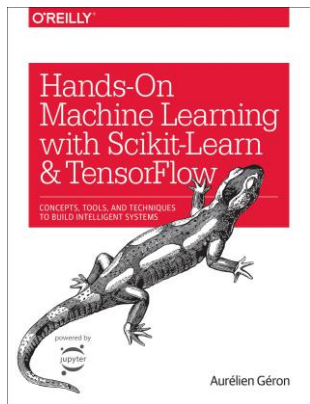
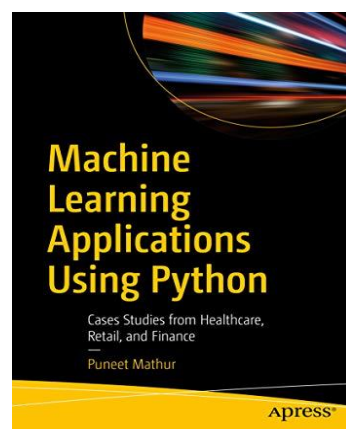
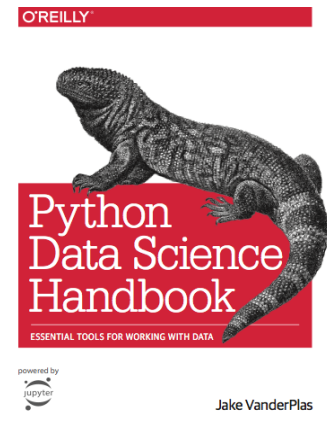
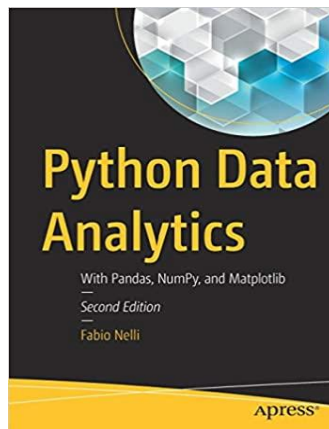
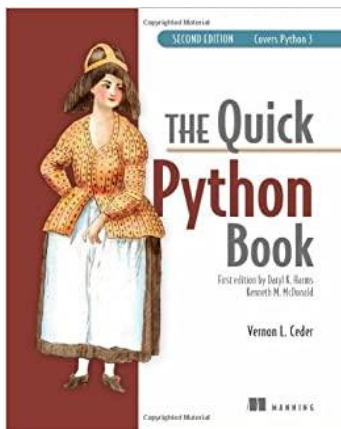
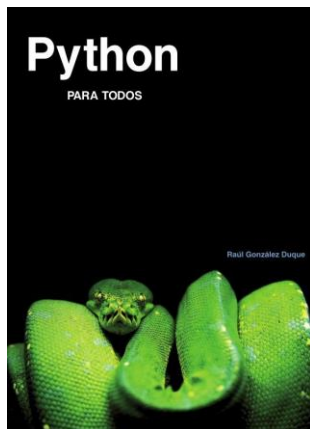
Mg. Heber Baldeón Paucar
heberjbaldeon@gmail.com

Python para Economistas

Índice

1. Introducción al Open Source
2. ¿Por qué Python?
3. Instalación de Anaconda
4. Tipos de datos y estructuras de datos nativas
5. Indexación y slicing
6. Gestión de paquetes: importación de bases de datos (DataFrame)
7. Paquetes, módulos, submódulos y funciones
 - Numpy
 - Sympy
 - Pandas
 - Matplotlib
8. Creación de bases de datos (DataFrame)
9. Introducción a la programación
 - Loops
 - Estructuras de control
10. Creación de funciones personalizadas

Bibliografía



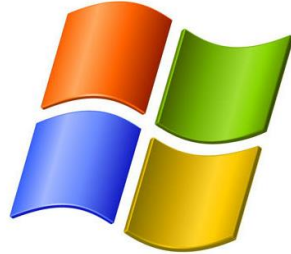
1. Proprietary software vs Open Source



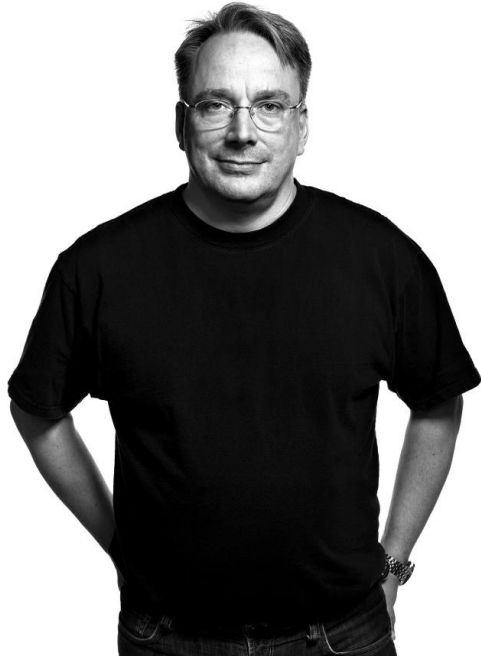
Proyecto GNU (1983)

En rechazo al software propietario e inicia el desarrollo del software libre basado en la idea de la cooperación entre los usuarios.

1. Open source para Economistas



1. Los rostros del *Open Source*



Linus Torvalds (1991)



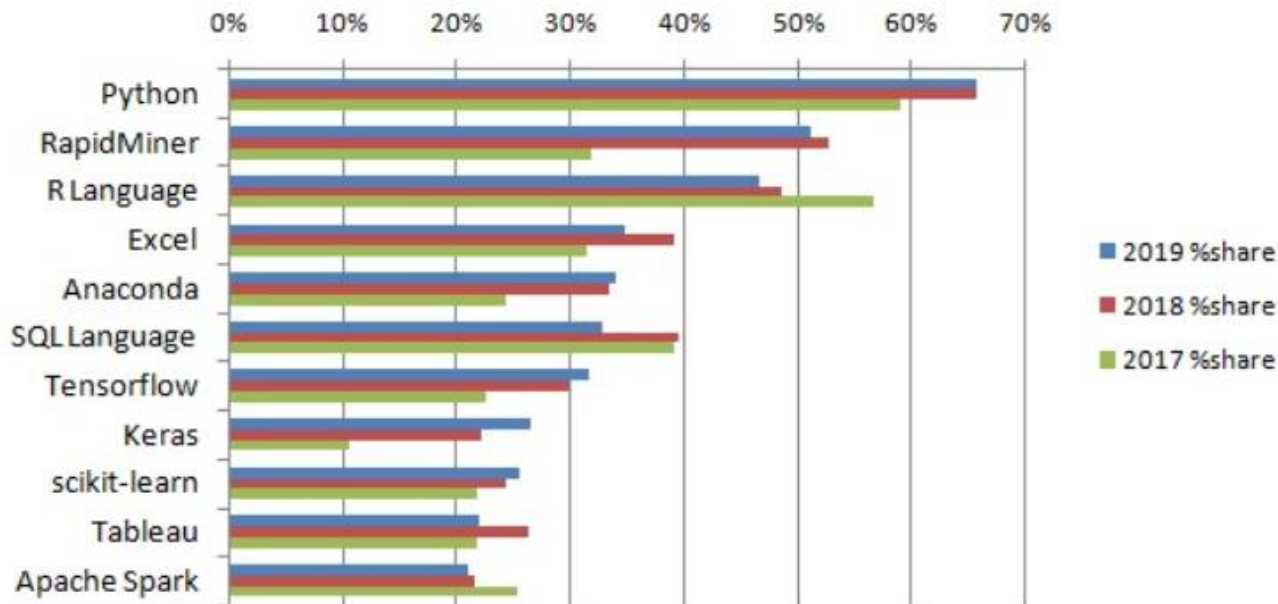
Guido van Rossum (1991)
versión 0.9.0



Christine Peterson (1998)

2. ¿Por qué Python?

Top Analytics, Data Science, Machine Learning Software 2017-2019, KDnuggets Poll



2. ¿Por qué Python?

Which Software Should I Choose?	Python	R	SAS	SQL
Best for:	General programming; Data analysis; Deep learning; Repeated tasks	Statistical analysis; Data analysis; Single passes of data	Statistical analysis; Data analysis	Database manipulating, updating, querying; Extracting, wrangling data
Availability	Free, open source	Free, open source	Paid (free for university edition); Closed source	Open and closed source versions available (free and paid)
Easy to learn?	Yes, especially for software engineers	Steep learning curve; Relatively easier if no prior coding experience	Yes, especially if you already know SQL	Relatively easy for basic level; Learning curve for more complex tasks
Advantages	Easy to deploy; General purpose language; Widely used by corporations	Minimal coding required for statistical models	Highly reliable, secure and stable	Very readable
Disadvantages	Requires rigorous testing	Very statistics oriented; Not a general-purpose program	Relatively expensive	Not general purpose: very specific, limited capability

2. ¿Qué es Python?

- Python es un **lenguaje de programación poderoso** creado por Guido van Rossum (1990's).
- Python es un programa:
 - **lenguaje interpretado** a través de scripts (vs lenguaje compilado),
 - **de tipado dinámico** (vs tipado estático),
 - **multiplataforma** (Windows, Mac, Linux), y
 - **multiparadigma**: programación estructurado, funcional y orientado a objetos (clases y objetos).
- Python permite escribir programas **compactos y legibles**. Los programas en Python son típicamente más cortos que sus programas equivalentes en C, C++ o Java por varios motivos:
 - **tipado dinámico** porque no es necesario declarar el tipo de variables.
 - la agrupación de instrucciones se hace **por sangría** en lugar de **llaves de apertura y cierre**
 - cuenta con **estructuras de datos eficientes** y de **alto nivel**
 - los tipos de datos de **alto nivel** permiten expresar operaciones complejas en una sola instrucción
- Página web: <https://www.python.org/>

2. ¿Qué es Python?



Class & Inheritance in Java :

```
class Animal{
    private String name;
    public Animal(String name){
        this.name = name;
    }
    public void saySomething(){
        System.out.println("I am " + name);
    }
}

class Dog extends Animal{
    public Dog(String name) {
        super(name);
    }
    public void saySomething(){
        System.out.println("I can bark");
    }
}

public class Main {
    public static void main(String[] args)
    {
        Dog dog = new Dog("Chiwawa");
        dog.saySomething();
    }
}
```



Class & Inheritance in Python :

```
class Animal():

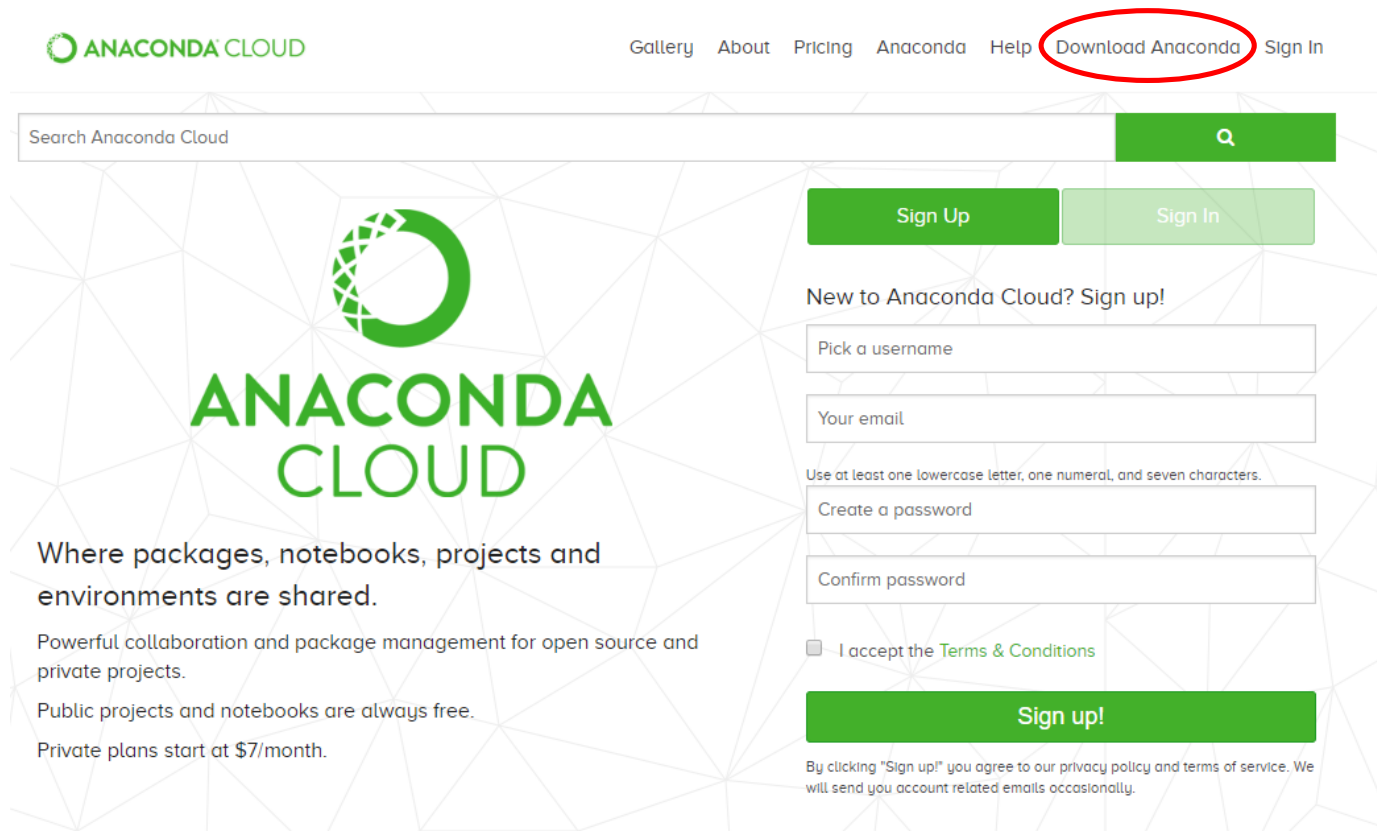
    def __init__(self, name):
        self.name = name

    def saySomething(self):
        print "I am " + self.name

class Dog(Animal):
    def saySomething(self):
        print "I am "+ self.name\
        + ", and I can bark"

dog = Dog("Chiwawa")
dog.saySomething()
```

3. Instalación de Python



The screenshot shows the Anaconda Cloud website. At the top, there is a navigation bar with links: Gallery, About, Pricing, Anaconda, Help, **Download Anaconda** (circled in red), and Sign In. Below the navigation bar is a search bar labeled "Search Anaconda Cloud" with a magnifying glass icon. The main content area features the Anaconda Cloud logo and the text: "Where packages, notebooks, projects and environments are shared." Below this, it says: "Powerful collaboration and package management for open source and private projects." and "Public projects and notebooks are always free." and "Private plans start at \$7/month." On the right side, there is a sign-up form. It includes a "Sign Up" button and a "Sign In" button. Below these buttons, it says: "New to Anaconda Cloud? Sign up!" followed by input fields for "Pick a username", "Your email", "Create a password" (with a note: "Use at least one lowercase letter, one numeral, and seven characters."), and "Confirm password". There is also a checkbox labeled "I accept the Terms & Conditions". At the bottom of the form is a large green "Sign up!" button. Below the button, it says: "By clicking 'Sign up!' you agree to our privacy policy and terms of service. We will send you account related emails occasionally."

ANAACONDA CLOUD

Gallery About Pricing Anaconda Help **Download Anaconda** Sign In

Search Anaconda Cloud

Sign Up Sign In

New to Anaconda Cloud? Sign up!

Pick a username

Your email

Use at least one lowercase letter, one numeral, and seven characters.

Create a password

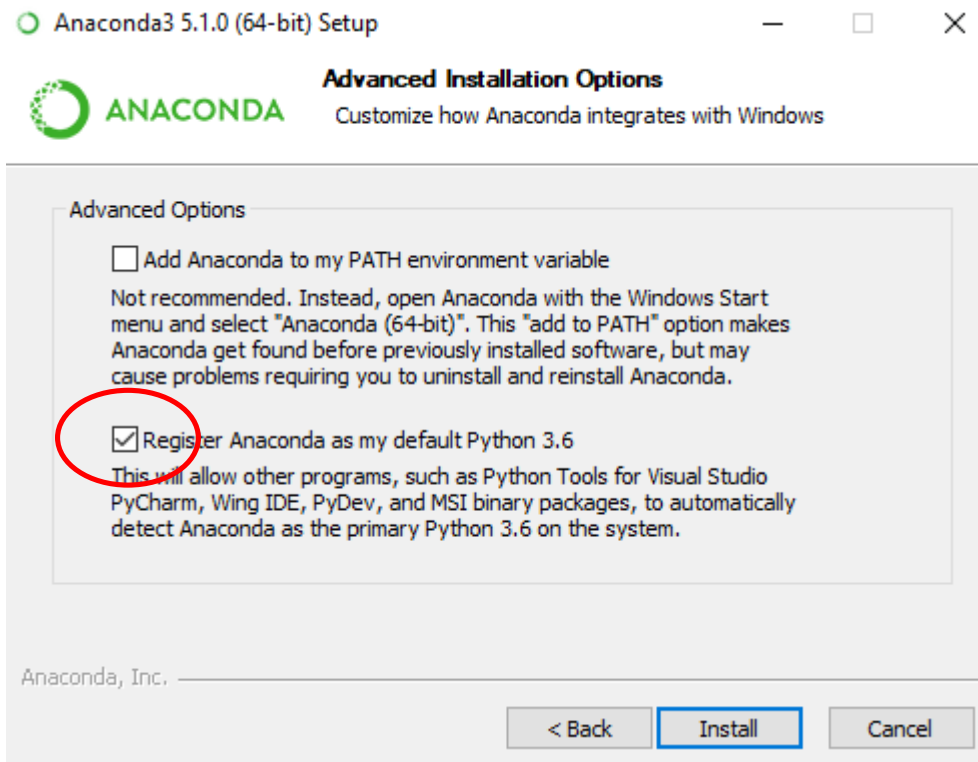
Confirm password

☐ I accept the Terms & Conditions

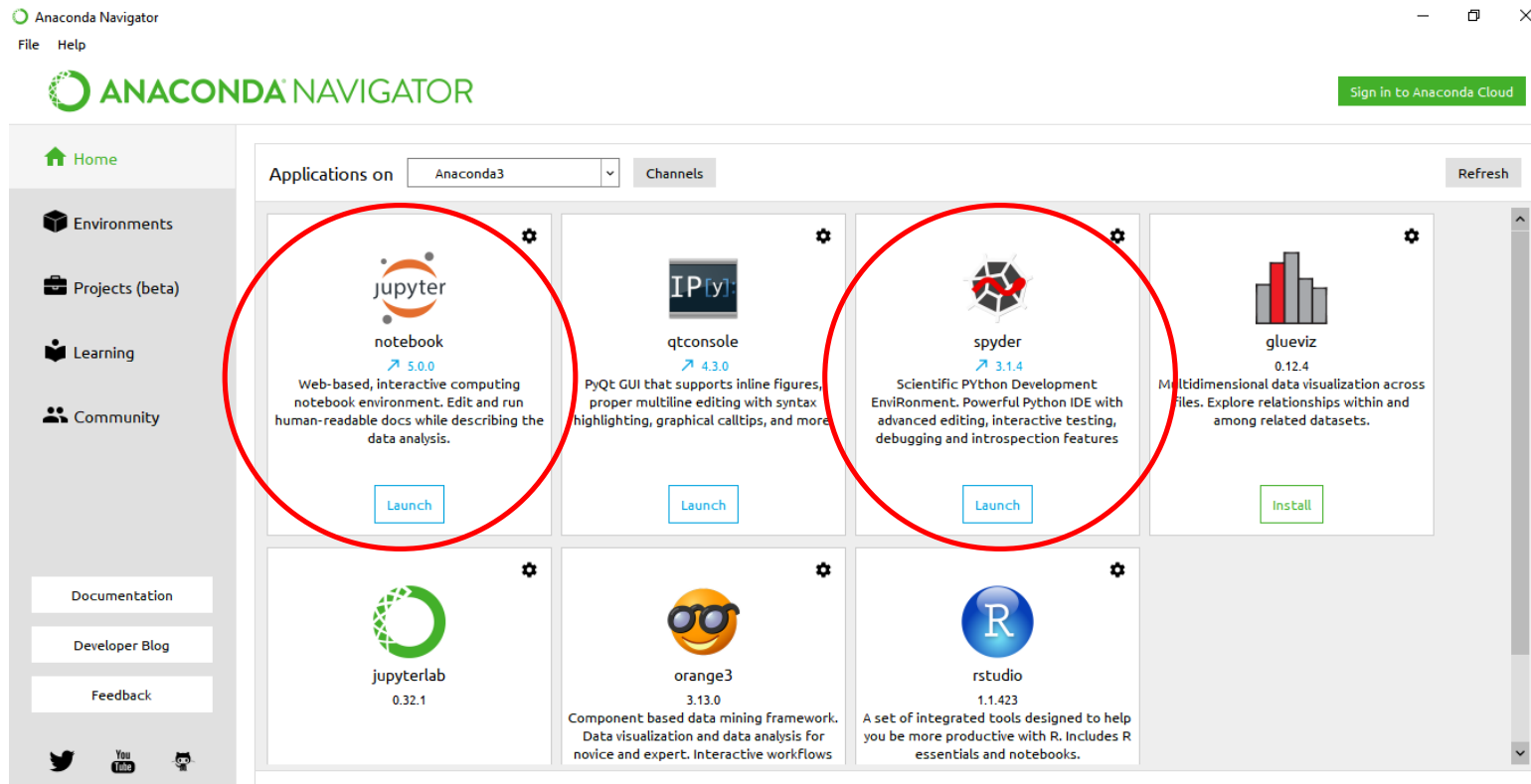
Sign up!

By clicking "Sign up!" you agree to our privacy policy and terms of service. We will send you account related emails occasionally.

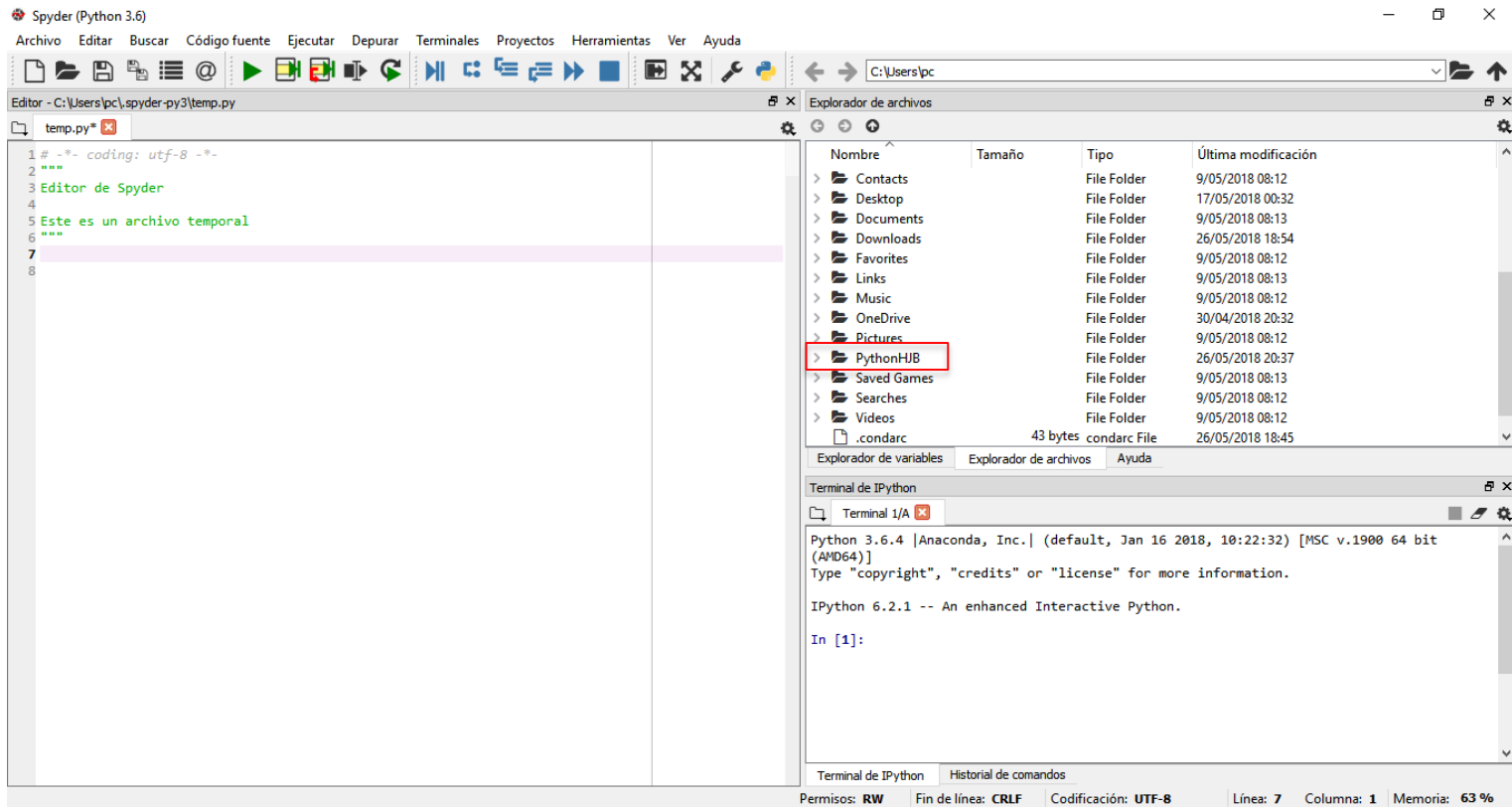
3. Instalación de Python



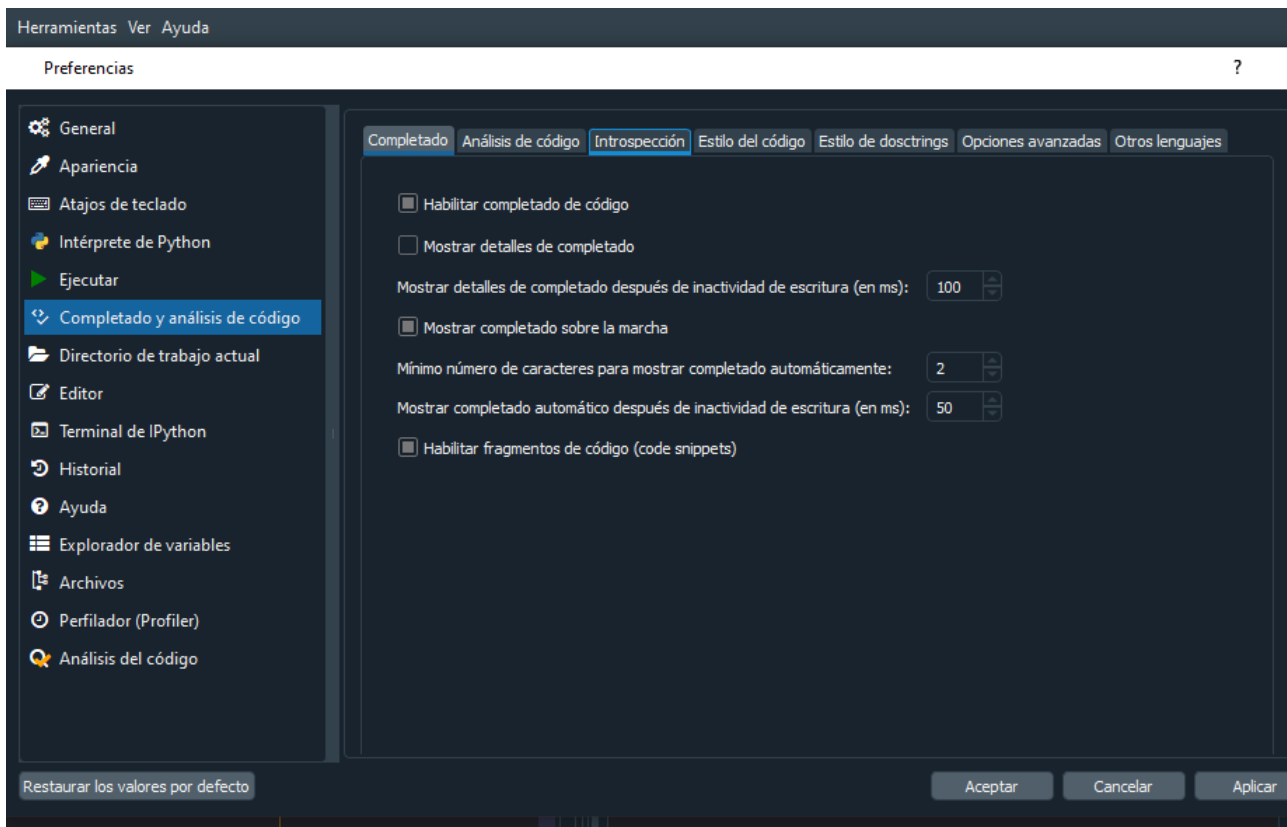
3. Instalación de Python



3. IDE Spyder (.py)



3. IDE Spyder (.py)



3. IDE Jupyter (.ipynb)

 Logout

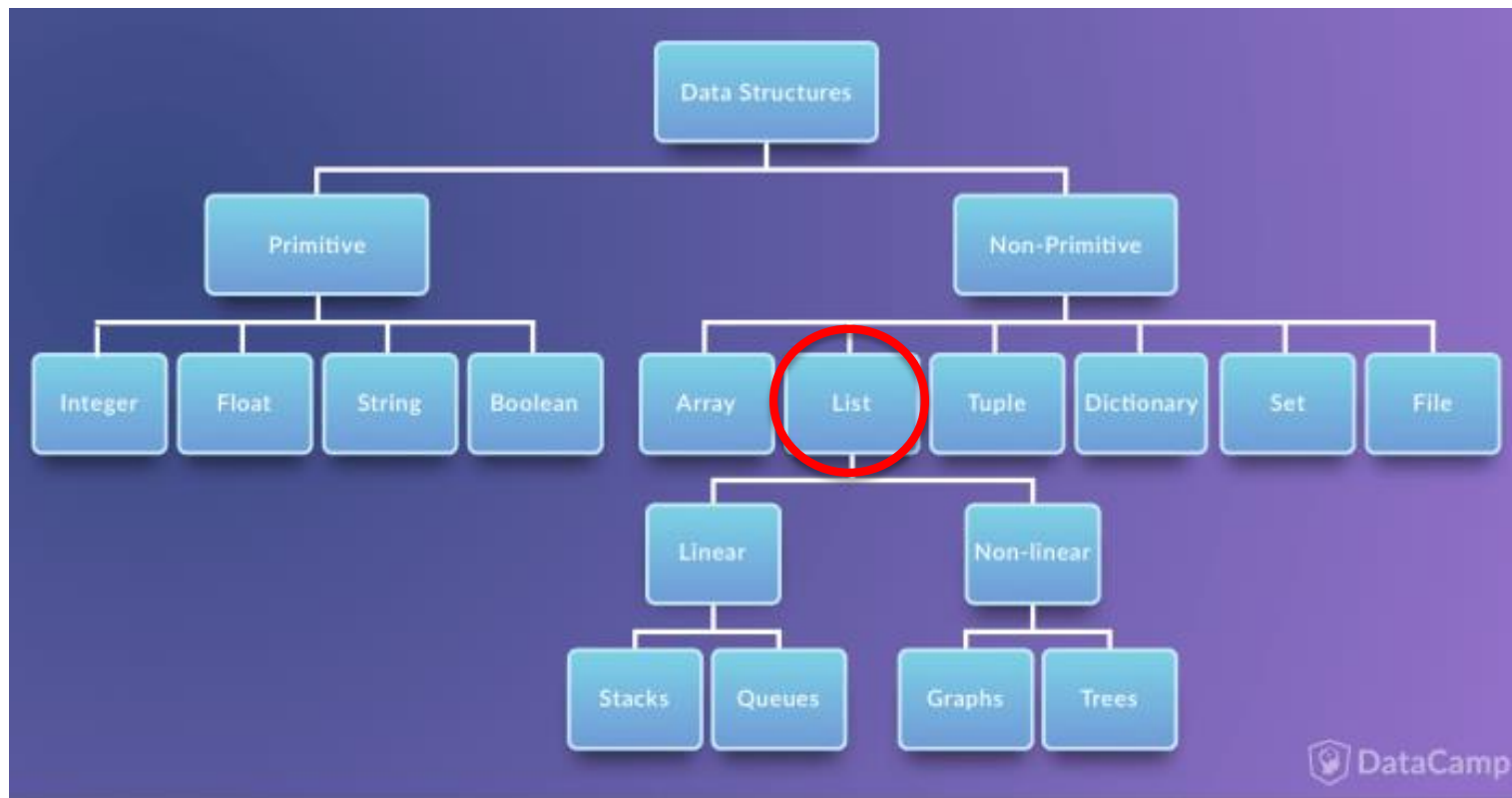
Files Running Clusters

Select items to perform actions on them.

Upload New ▾ ↺

<input type="checkbox"/> 0 ▾	 /	Name ▾	Last Modified
<input type="checkbox"/>	3D Objects		hace 18 días
<input type="checkbox"/>	Anaconda3		hace 2 horas
<input type="checkbox"/>	Application Data		hace 3 meses
<input type="checkbox"/>	Contacts		hace 18 días
<input type="checkbox"/>	Desktop		hace 10 días
<input type="checkbox"/>	Documents		hace 18 días
<input type="checkbox"/>	Downloads		hace una hora
<input type="checkbox"/>	Favorites		hace 18 días
<input type="checkbox"/>	Links		hace 18 días
<input type="checkbox"/>	Music		hace 18 días
<input type="checkbox"/>	OneDrive		hace un mes
<input type="checkbox"/>	Pictures		hace 18 días
<input type="checkbox"/>	PythonHJB		hace una hora
<input type="checkbox"/>	Saved Games		hace 18 días

4. Estructura de bases de datos



4. Estructura de bases de datos

Tipos de datos

1. NoneType
2. Numéricas:
 - Integer (enteros)
 - Float (reales)
 - Complejas: j
 - Operadores: +, -, *, **, /, //, %, %, %**
 - Operadores bit: &, |, ^, ~, <<, >>
3. String (cadena), Objeto secuencia
 - Character
 - Datetime
 - Operadores: +, *
4. Booleana
 - Operadores lógicos: and, or, y not
 - Operadores relacionales: ==, !=, <, >, >=, <=

Estructuras de datos (colecciones)

1. Tuple `()`: Array inmutable, pero ligera. Objeto secuencia
2. List `[]`: Array mutable. Objeto secuencia
3. Dict `{}` o dictionary: Es mutables y se organiza como una tabla Hash. Matrices asociativas (“clave” : “valor”). No se puede indexar o slicing (no tiene orden), en cambio se le llama por su clave. Objeto Mapping
4. Set & Frozen Set `{}`

4. Estructura nativas de datos

List

- General purpose
- Most widely used data structure
- Grow and shrink size as needed
- Sequence type
- Sortable

Tuple

- Immutable (can't add/change)
- Useful for fixed data
- Faster than Lists
- Sequence type

Set

- Store non-duplicate items
- Very fast access vs Lists
- Math Set ops (union, intersect)
- Unordered

Dict

- Key/Value pairs
- Associative array, like Java HashMap
- Unordered

4. Estructura de bases de datos

Tipo	Clase	Notas	Ejemplo:
int	Entero	Números enteros	30
float	Decimal	Coma o punto flotante	3.1416
str	Cadena (string)	Inmutable	'Hola'
list	Secuencia	Mutable	[1.0, 'Hola']
tuple	Secuencia	Inmutable	(1.0, 'Hola')
set	Conjunto	Mutable, sin orden, sin duplicados	Set([1.0, 'Hola'])
frozenset	Conjunto	Inmutable, sin orden, sin duplicados	Frozenset([1.0, 'Hola'])
dict	Mapping	Grupo de pares clave: valor	{'key 1':1.0, 'key 2': false }
bool	Booleana	Valor booleano verdadero o falso	True, False

4. Estructura de bases de datos

Tipos de operadores

Operation	Description
$x + y$	Addition
$x - y$	Subtraction
$x * y$	Multiplication
x / y	Division
$x ^ y$	Exponentiation
$x \% y$	Modular arithmetic
$x \%/ y$	Integer division
$x == y$	Test for equality
$x <= y$	Test for less than or equal to
$x >= y$	Test for greater than or equal to
$x \&\& y$	Boolean AND for scalars
$x y$	Boolean OR for scalars
$x \& y$	Boolean AND for vectors (vector x,y,result)
$x y$	Boolean OR for vectors (vector x,y,result)
$!x$	Boolean negation

5. Indexación y slicing

Arreglo
Unidimensional

Finito: Contiene N elementos.
Homogéneo: del mismo tipo.
Ordenado: con una posición.
Referenciado: con un índice.



PRIMERO	SEGUNDO	TERCERO	CUARTO	QUINTO	SEXTO	SÉPTIMO	...	ENÉSIMO
C_1	C_2	C_3	C_4	C_5	C_6	C_7	...	C_N
i_0	i_1	i_2	i_3	i_4	i_5	i_6	...	C_{N-1}

axis 0

axis 1

	0	1	2
0	0,0	0,1	0,2
1	1,0	1,1	1,2
2	2,0	2,1	2,2

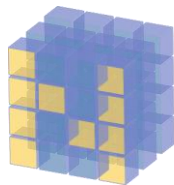
6. Instalación de módulos/paquetes

<u>pip</u> 	 <u>conda</u> ANACONDA
pip search pyserial	conda search pyserial
pip install pyserial	conda install pyserial
pip install pyserial --upgrade	conda update python
pip list	conda list

6. Instalación de Módulos/paquetes

Ananconda prompt

- conda install NumPy
 - Numerical Python
 - <http://www.numpy.org/>
- conda install pandas
 - Python Data Analysis Library
 - <https://pandas.pydata.org/>
- conda install matplotlib
 - Matplotlib: Python 2D plotting library
 - <https://matplotlib.org/>
- conda update pandas



NumPy



pandas

matplotlib

En caso no permita el instalado directo, habilitar ambiente:

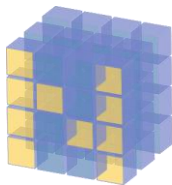
- conda install -c conda-forge pyreadstat

6. Instalación de módulos/paquetes

cmd prompt

```
cd C:\Users\pc\Anaconda3\Scripts
```

- pip install NumPy
 - Numerical Python
 - <http://www.numpy.org/>
- pip install pandas
 - Python Data Analysis Library
 - <https://pandas.pydata.org/>
- pip install matplotlib
 - Matplotlib: Python 2D plotting library
 - <https://matplotlib.org/>



NumPy



pandas

matplotlib

En caso no este instalado pip:

- pip install pandas --upgrade / --user /python get-pip.py

7. Importación de bases de datos

- **Formato de texto**

- Txt: `pd.read_table()`
- CSV: `pd.read_csv()`
- Excel: `pd.read_excel()`
- Stata: `pd.read_stata(filepath_or_buffer, convert_dates=True, convert_categoricals=True, index_col=None, convert_missing=False, preserve_dtypes=True, columns=None, order_categoricals=True, chunksize=None, iterator=False)`
- SPSS: `pd.read_spss(path: Union[str, pathlib.Path], usecols: Union[Sequence[str], NoneType]=None, convert_categoricals: bool=True)`
- Sas: `pd.read_sas(filepath_or_buffer, format=None, index=None, encoding=None, chunksize=None, iterator=False)`
- SQL: `pd.read_sql(sql, con, index_col=None, coerce_float=True, params=None, parse_dates=None, columns=None, chunksize=None) // pd.read_sql_query() // pd.read_sql_table()`
- DBF: `DBF(filename, load=False, encoding=None, char_decode_errors='strict', lowernames=False, ignorecase=True, parserclass=FieldParser, ignore_missing_memofile=False, raw=False)`

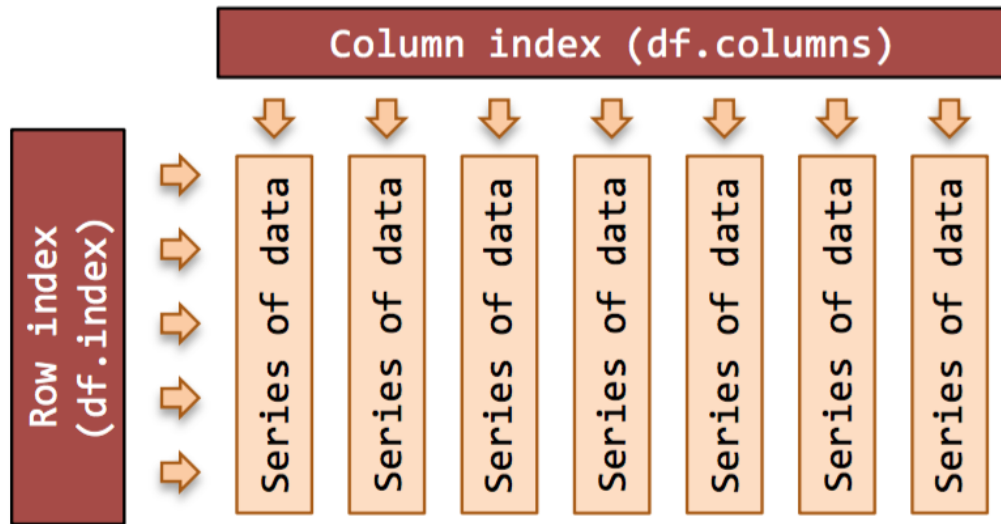
- **Web data**

- HTML data: `pd.read_html(io, match='.+', flavor=None, header=None, index_col=None, skiprows=None, attrs=None, parse_dates=False, thousands=',', encoding=None, decimal='.', converters=None, na_values=None, keep_default_na=True, displayed_only=True)`

8. Creación de bases de datos

Uso del paquete Pandas

- **Pandas Series (1D):** Las series son un array de una dimensión, pueden almacenar cualquier tipo de datos como valores discretos, continuos, cadenas y objetos Python.
- **Pandas Dataframe (2D):** Es una estructura de datos de 2 dimensiones de distinto tipos de datos, un data frame puede venir de las siguientes estructuras de datos: NumPy Array, Listas, Diccionarios, Series, 2D NumPy Array.
- **Panel Data (3D):** Es una estructura de datos de 3 dimensiones.
- Objetos Index
- Índices Jerárquicos



9. Introducción a la Programación

- El lenguaje **Python** cuenta con varios tipos de ciclos o repeticiones (loops), a saber:
 1. repeticiones por un número determinado de veces: for

for secuencia:
expresión1

2. repeticiones mientras se cumple una condición: while
3. repeticiones infinitas: repeat / break

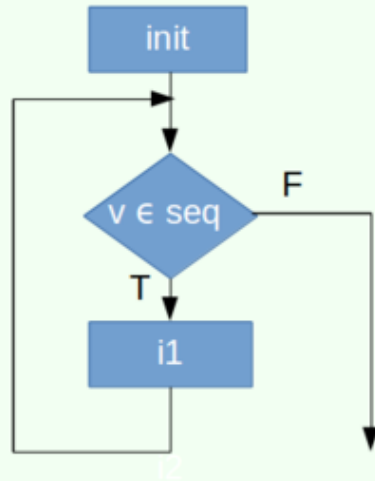
- En los lenguajes de programación (incluido el **Python**), se entiende por estructuras de control aquellas construcciones sintácticas del lenguaje que dirigen el flujo de la ejecución de un programa en una *dirección* o en otra dentro de su código: if / else

if condición:
expresión1
else:
expresión2

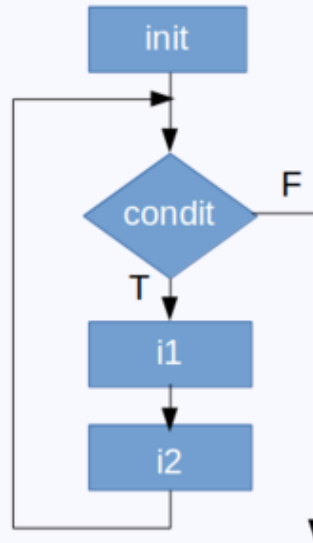
9. Introducción a la Programación

Tipos de Ciclos o repeticiones (loops)

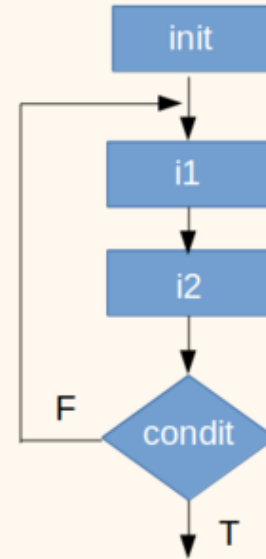
For loop



while loop



repeat loop



10. Funciones personalizadas

Anatomía de una función:

```
def nombre_1(arg1,arg2,arg3):  
    cuerpo(arg1,arg2,arg3)
```

```
def nombre_2(arg1,arg2,arg3):  
    A=cuerpo(arg1,arg2,arg3)  
    return A
```

Ejemplo:

```
def sumar(x, y): x+y  
    sumar(5,9)
```

Parámetros por defecto:

```
def sumar_1(arg1=1,arg2=2,arg3=3):  
    return arg1+ arg2+ arg3
```

*args en funciones:

```
def sumar_2(*args):  
    res=0  
    for val in args: res += val  
    return res
```

*kwargs en funciones:

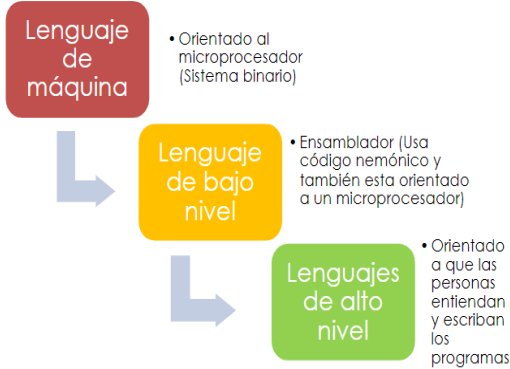
```
def sumar_3(**kwargs):  
    res=0  
    print(kwargs)  
    for key in kwargs: res += kwargs[key]  
    return res
```

Repaso

- ¿Cuáles son los tipos de datos primitivos?
- ¿Cuáles son las estructuras de datos primitivas?
- ¿Cuáles son los atributos usuales de un objeto?
- ¿Cuál es la función para instalar un paquete?
- ¿Cuál es la función para cargar una base de datos de stata?
- ¿Cuáles son las estructuras de control y las repeticiones?
- ¿Las funciones son objetos?
- ¿Cuáles son las 3 partes de una función?

Anexo 1. Tipos de Lenguaje de Programación

Clasificación de los lenguajes de programación



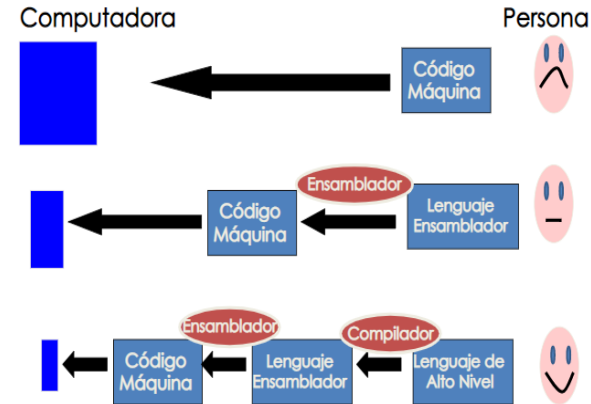
Codificación

Lenguaje de máquina: 0000001010111001010
0000001010111101010
00000011001100100110

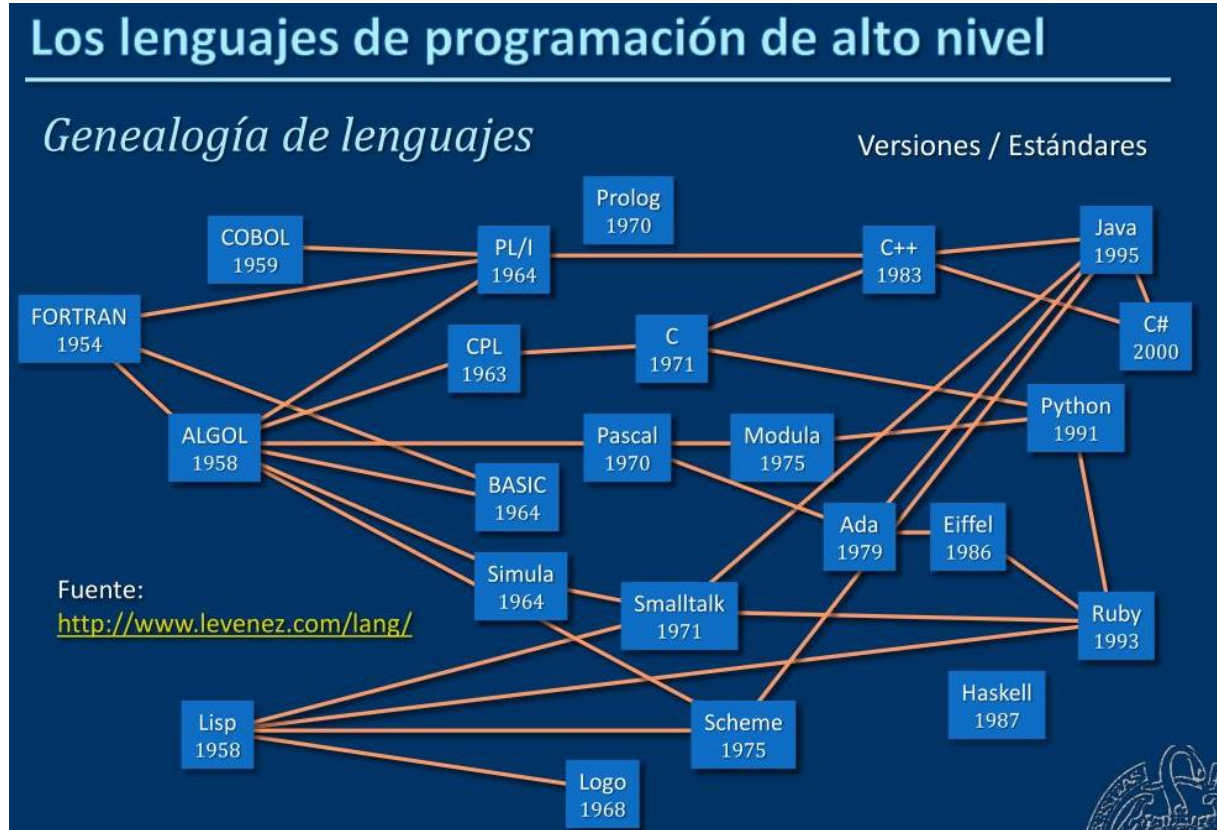
Lenguaje Ensamblador: Load I
Add J
Store K

Lenguaje alto nivel: $K = I + J$

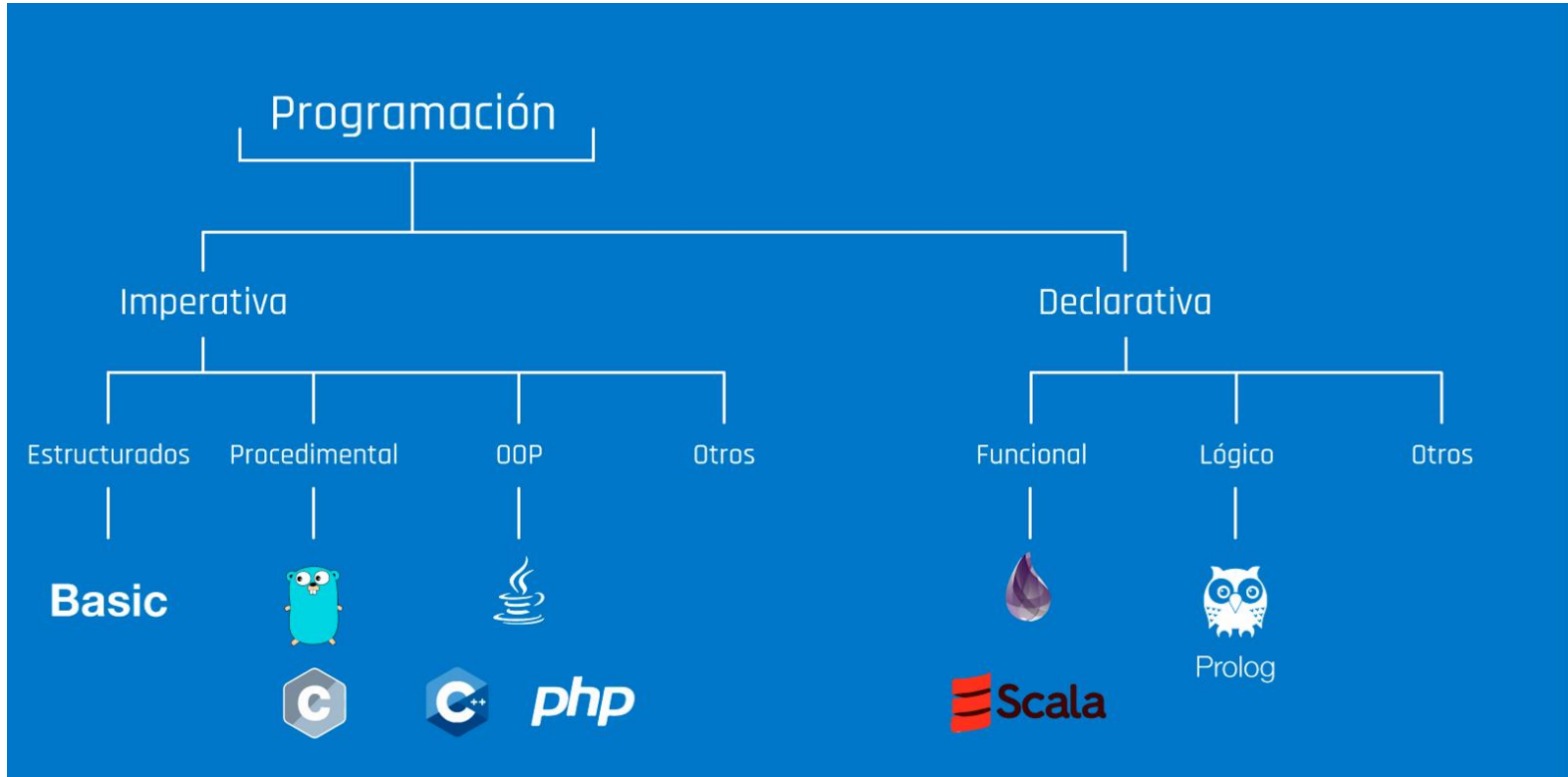
Compiladores e interpretes



Anexo 2. Paradigmas de Programación

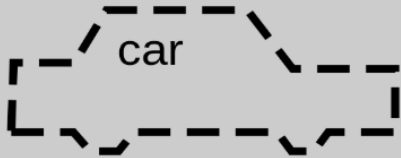


Anexo 2. Paradigmas de Programación



Anexo 3. Programación Orientada a Objetos

class



objects



```
class Car(object):
```

```
    def __init__(self, model, passengers, color, speed):
```

```
        self.model = model
```

```
        self.passengers = passengers
```

```
        self.color = color
```

```
        self.speed = speed
```

Atributos

```
    def accelerate(self):
```

```
        self.speed = self.speed + 2
```

```
        print (self.speed)
```

Métodos: es una función asociada a un objeto

```
bmw = Car("BMW", 4, "red", 5)
```

```
ferrari = Car("Ferrari", 2, "black", 10)
```

```
ford = Car("Ford", 6, "blue", 6)
```

Objetos

```
bmw.accelerate()
```

```
print (bmw.color)
```

```
ferrari.accelerate()
```

```
print (ferrari.color)
```

```
print (ford.passengers)
```

```
ford.accelerate()
```

Anexo 3. Programación Orientada a Objetos

- **Herencia:** Una clase puede heredar sus atributos y métodos a sus subclases. Lo que significa que una subclase, aparte de sus atributos y métodos propios, tiene incorporados los atributos y métodos heredados de la superclase. La herencia puede ser simple o múltiple.
- **Poliformismo:** Es la habilidad de cambiar el comportamiento en función a la instancia del objeto (tipo de dato en tiempo de ejecución). Los objetos de una clase responden en función a los parámetros o argumentos de entrada.
- **Encapsulamiento:** Para proteger a las variables de modificaciones no deseadas se encapsula. Los miembros de una clase se dividen en públicos y privados. En Python, el acceso a un atributo o método viene determinado por su nombre.
 - Si el nombre comienza con 2 guiones bajos (y no termina también con 2 guiones bajos) es un atributo o método privado.
 - Caso contrario, es un atributo o método público.