

Carpyntería

Primeros Pasos con Python

Gimenez, Christian

Universidad Nacional del Comahue
Facultad de Economía y Administración - Facultad de Informática

29 de Marzo del 2019



Expositor

- Lic. Christian Gimenez

Coordinadores

- Mg. Sc. Gustavo Gimenez
- Dr. Pablo D. Reeb

Material de Este Curso

<http://crowd.fi.uncoma.edu.ar/wiki/doku.php/christian:home>

O acceder a “My Wiki Page” desde:

<http://crowd.fi.uncoma.edu.ar/~christian.gimenez>



Web Page

<https://www.anaconda.com>

¿Qué es Anaconda?

Es una distribución para Data Science.

- Es un gestor de paquetes.
- Un gestor de ambientes.
- Una distribución de Python y/o R.
- Más de 1500 paquetes Open Source.

¿Una distribución?

Un conjunto de paquetes seleccionados y mantenidos por una comunidad para objetivo específico.

Un paquete puede contener: programal, bibliotecas(*libraries*), códigos, datos. **Información.**



Web Page

<https://conda.io>

¿Qué es Conda?

Un gestor de paquetes, dependencias y ambientes para varios lenguajes: Python, R, Ruby, Lua, Scala, Java, JavaScript, C/C++, FORTRAN.

- Es un programa que puede descargar e instalar paquetes.
- Hay paquetes que **dependen** de otros para poder funcionar.
 - Conda descarga todos los necesarios para que funcione el que le pedimos.



Web Page

<http://python.org>

¿Qué es Python?

Es un lenguaje de programación interpretado que:

- Es de propósito general.
- Es de alto nivel (nos abstraemos del lenguaje máquina)
- Enfatiza en la buena legibilidad del código.
- Tiene tipado dinámico y gestiona automáticamente la memoria.
- Soporta varios paradigmas: Orientada a objetos, imperativo, funcional, procedural.
- Posee una amplia biblioteca estándar.

¿Por qué Python?

Language Rank	Types	Spectrum Ranking
1. Python	  	100.0
2. C++	  	98.4
3. C	  	98.2
4. Java	  	97.5
5. C#	  	89.8
6. PHP		85.4
7. R		83.3
8. JavaScript	 	82.8
9. Go	 	76.7
10. Assembly		74.5

¿Por qué Python?

- Sus objetivos y características.
- ¿Se necesita más eficiencia?
 - Se puede crear módulos en C (foreign libraries).
 - Existe un compilador Just-in-time: PyPy.
 - Cython: C en estilo Python.
- Comunidad muy activa.
- Muchos paquetes para variados propósitos (ver <https://pypi.org>).
- No sólo se usa para Data Science, también para desarrollo Web, de aplicaciones de escritorio, sistemas embebidos.

Zen of Python: <https://www.python.org/dev/peps/pep-0020/>

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Paquetes de Python (PyPi y pip)

Paquetes

Colección de módulos con objetos y definiciones que pueden ser cargadas en Python para su posterior uso.

Python tiene sus propios paquetes y se pueden instalar usando el comando `pip` desde terminal.

Anaconda sugiere: si ya existen en su distribución lo instalemos desde `conda` o el `navigator`.

Buscar e Instalar

Buscar: `pip search numpy`

Instalar: `pip install numpy`



Web Page

<https://jupyter.org>



¿Qué es Jupyter?

Es una aplicación Web que permite crear y compartir documentos que contiene “código vivo” (*live code*), ecuaciones, visualizaciones y texto narrativo.

Requerimientos

- Sistema operativos soportados: Windows, macOS y GNU/Linux.
- Arquitecturas de 32-bits y 64-bits.
- Recomendamos: GNU/Linux de 64-bits para Data Science y desarrollo en general.

Descargar

<https://www.anaconda.com/distribution/#download-section>

En GNU/Linux

Descargar el archivo (supongamos que se llama Anaconda3-2018.12-Linux-x86_64.sh).

```
cd ~/Downloads
```

```
wget https://repo.anaconda.com/archive/Anaconda3-2018.12-Linux-x86_64.sh
```

```
chmod +x Anaconda3-2018.12-Linux-x86_64.sh
```

```
./Anaconda3-2018.12-Linux-x86_64.sh
```

¿Qué pasa luego de ejecutar el instalador?

Tanto en GNU/Linux como en Windows, preguntará por lo mismo.

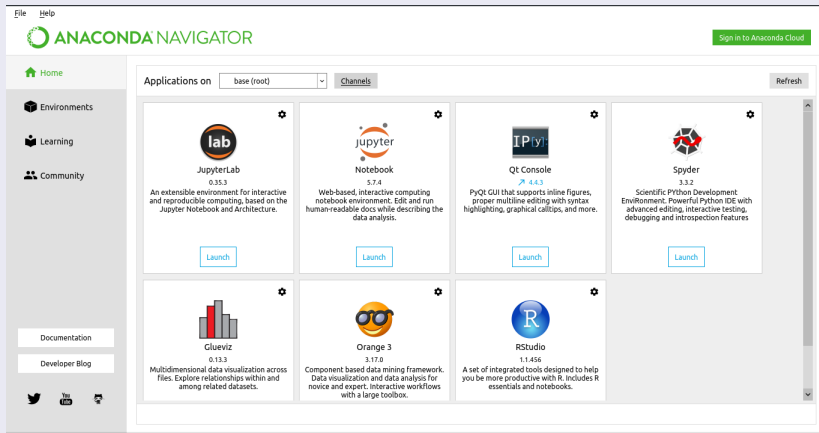
Pasos del Instalador

(En GNU/Linux que es más completo)

- Te dará la bienvenida y pedirá que presiones ENTER.
- Mostrará la licencia. Presionar Abajo hasta el final, escribir “yes” y presionar ENTER.
- Preguntará dónde instalarlo. Por defecto será en `/home/USUARIO/anaconda3`, presionar ENTER.
- Comenzará el proceso de instalación.
- Preguntará si agregar comandos de inicio para anaconda en el archivo de inicio `.bashrc`. Escribir “yes” y ENTER.

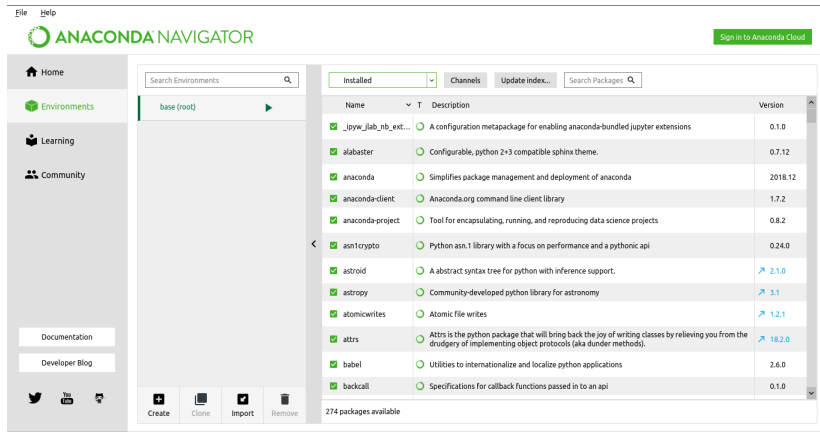
¿Cómo instalar paquetes?

Anaconda Navigator



Clic en “Install” sobre el paquete que se desea instalar una aplicación.
Clic en “Launch” para iniciar una de sus aplicaciones.

Paquetes Python o R



The screenshot shows the Anaconda Navigator application window. The top bar includes 'File' and 'Help' menus, the 'ANACONDA NAVIGATOR' logo, and a 'Sign in to Anaconda Cloud' button. The left sidebar contains navigation links for 'Home', 'Environments', 'Learning', and 'Community', along with 'Documentation' and 'Developer Blog' buttons and social media icons. The main area is divided into three panes. The left pane shows the 'base (root)' environment. The middle pane has a search bar and a list of installed packages. The right pane displays a table of installed packages.

Name	Description	Version
✓ _ipyw_jlab_nb_ext...	A configuration metapackage for enabling anaconda-bundled jupyter extensions	0.1.0
✓ alabaster	Configurable, python 2+3 compatible sphinx theme.	0.7.12
✓ anaconda	Simplifies package management and deployment of anaconda	2018.12
✓ anaconda-client	Anaconda.org command line client library	1.7.2
✓ anaconda-project	Tool for encapsulating, running, and reproducing data science projects	0.8.2
✓ asn1crypto	Python asn.1 library with a focus on performance and a pythonic api	0.24.0
✓ astroid	A abstract syntax tree for python with inference support.	2.1.0
✓ astropy	Community-developed python library for astronomy	3.1
✓ atomicwrites	Atomic file writes	1.2.1
✓ attrs	Attrs is the python package that will bring back the joy of writing classes by relieving you from the drudgery of implementing object protocols (aka dunder methods).	18.2.0
✓ babel	Utilities to internationalize and localize python applications	2.6.0
✓ backcall	Specifications for callback functions passed in to an api	0.1.0

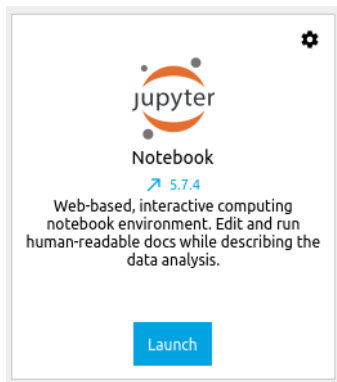
274 packages available

En “installed” lo cambiamos por “not installed”, buscamos el paquete que deseamos, luego lo tildamos y hacemos clic en “Apply”



Iniciar Jupyter Notebook

Iniciar Jupyter Notebook desde Anaconda Navigator. Clic en “Launch”.



Después se abrirá el Explorador Web apuntando a la URL <http://localhost:8888/tree>.

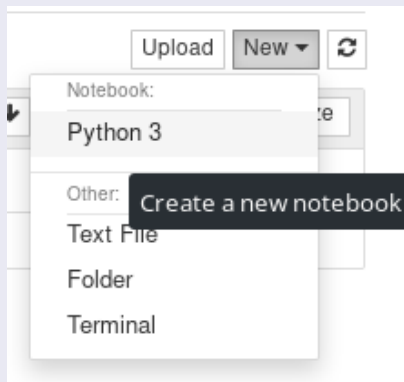
- Si no inicia: abrir el Explorador Web y escribir la URL en la barra de direcciones.
- El puerto (8888) puede cambiar si éste está ocupado por otra aplicación (probar 8889, 8890, etc.)
- Esta página sólo es accesible por la máquina local.

¿Notebooks?

¿Qué son las Notebooks?

Son documentos que poseen celdas (o bloques) de código y texto.

Crear un nuevo notebook



Para escribir código Python, se debe usar una celda de tipo “Código”.

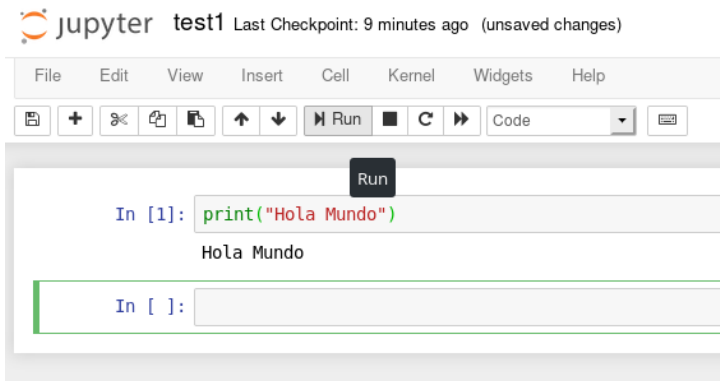
 jupyter test1 Last Checkpoint: 8 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

       Run    Code 

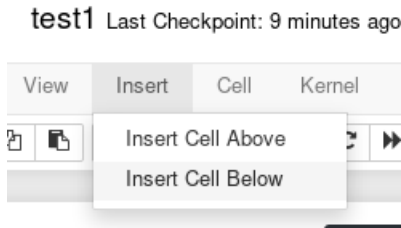
```
In [ ]: print("Hola Mundo")|
```

Al hacer clic en **Run** se ejecuta el código y se muestra el resultado debajo.

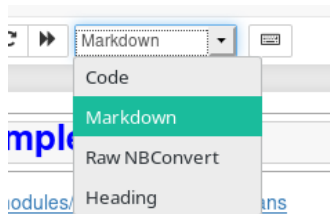


Texto con formato: Markdown

Se puede insertar una celda nueva arriba (above) o abajo (below) de la actual.



Y se le puede cambiar el tipo a Markdown para escribir texto.



Texto con formato: Markdown

Markdown nos permite escribir texto con un poco de formato.
(¡Sólo formatos simples!)

Su autor es **John Gruber** (Daring Fireball) junto a **Aaron Swartz** (RSS, CreativeCommons, Reddit) Ver su sitio:

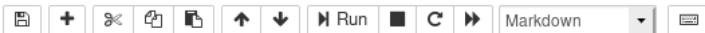
<https://daringfireball.net/projects/markdown/>



Texto con Formato: Markdown

 **jupyter** test1 Last Checkpoint: 12 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help



In [1]: `print("Hola Mundo")`

Hola Mundo

In []:

Más allá de un Hola Mundo

Vamos a intentar hacer algo ****más complejo****.

******¡Estén preparados!******

(Entrada y) Salida

En Jupyter la entrada y salida de datos se realiza sobre la página. Intente los siguientes ejemplos en una celda diferente de tipo “código”.

Example (Salida)

```
print("hello world")  
print(1 + 1)  
print(2 + 2 * 2)
```

Con print se muestra varios resultados.

Example (Retorno)

```
"hello world"  
1 + 1  
2 + 2 * 2
```

Sólo se muestra la última expresión.

No es usual usar `input` para pedir información. Normalmente, todo lo necesario está escrito en el código.

Pero sin embargo, podemos intentarlo...

Example (Entrada de Datos)

```
dato1 = input("ingrese su nombre:")  
dato2 = int(input("ingrese un num. entero:"))  
dato3 = float(input("ingrese un num. real:"))
```

Variables

- Las variables son pequeñas unidades de memoria que pueden contener números, strings, objetos, etc.
- Se crean simplemente usando un nombre y asignándoles un valor inicial.

Los nombres pueden contener:

- Letras del abecedario inglés (mayúsculas y minúsculas), números o guiones bajos. Ejemplos: `mivar`, `miVar`, `mi_var`, `_var`, `var2`
- No pueden empezar con números. Contraejemplos: `2var`, `105_mi_var`.
- No se pueden usar acentos y letras que no pertenezcan al alfabeto inglés. Contraejemplos: `nom_función`, `índice`, `fecha_mañana`.

Sugerencia

Se sugiere que los nombres sean descriptivos de su función o de lo que van a contener.

Example (Variables)

```
cant_ejemplos = 2  
saludo = "hola mundo"  
total_variables_aqui = 3
```

Example (Uso de las Variables)

```
print(saludo)  
print(cant_ejemplos + 1)  
  
total_variables_aqui = cant_ejemplos * 2  
print(total_variables_aqui)
```

¿Qué clase de información podemos usar?

Referencia: <https://docs.python.org/3/library/stdtypes.html>

Booleanos True, False (son instancias)

Texto str (Strings)

Numéricos

Enteros int

Flotantes float (Reales)

Complejos complex

Secuenciales

Listas list

Tuplas tuple

Rango range

Valen la pena mencionar

`dict`, `bytes`, `bytesarray`, `set`

Provenientes de Módulos y Paquetes

Algunos módulos y paquetes proveen de clases y objetos que fueron programados en Python.

El Paquete Pandas

Pandas agrega “Estructuras de Datos” muy útiles para Data Science:
`Series` y `DataFrames`

Operaciones

Cada uno de estos tipos tiene varias operaciones aritméticas y de comparación que se pueden utilizar.

Example (Comparación)

```
print(1 == 1)
print(1 < 2)
print(1 <= 1)
print(0.5 >= 0.3)
print(0.3 != 0.3)
print("hola" == "mundo")
```

Example (Operaciones)

```
print(1 + 3.14)
print("hola" + " " + "mundo")
print(True and False)
print(not (True and True or False))
```

¡Ojo con los Espacios!

¡Tengan cuidado con el dos puntos y el espaciado!

¡Usen cuatro espacios!

Example (if ...)

```
num = 20
if num >= 15:
    print("num es mayor o igual que 15")
```

Example (if ... elif ... else)

```
a = False
if a:
    print("a es true")
else:
    print("a es falso")

if not a:
    print("not a es verdadero")
else:
    print("not a es falso")
```

Example (Cuerpos con más de una linea)

```
if 1 + 3 == 5:
    print("resultado: 5")
    print("Es menor...")
elif 1 + 3 == 4:
    print("resultado correcto")
    print("¡Muy bien!")
elif 1 + 3 == 6:
    print("¿6?")
    print("¿Estás seguro?")
    print("¡6 es muy grande!")
else:
    print("No es 5, 4 ni 6")
```

Repetitivas

Repetir una porción de código varias veces.

¡Cuidado con los Espacios!

¡Usar cuatro espacios!

Example (while)

```
while i <= 10:  
    print(i)  
    i = i + 1
```

Example (for ... in)

```
for i in range(1, 10):  
    print(i)
```

Repetitivas

Cuando necesitamos los índices:

Example (For sobre una lista)

```
lista = ['Alan Turing', 'Ada Lovelace', \
        'Von Neumann', 'Denis Ritchie']
for elt in lista:
    print(elt)
```

Si usamos un map:

Example (For sobre un mapeo)

```
mymap = {'nombres': 'Alan Mathison', 'apellido': 'Turing',
        'edad': 41, 'fecha_nac': '23 Junio 1912'}
for key, val in mymap:
    print(key, ': ', val)
```

List Comprehensions

Otra forma de crear listas.

```
[EXPRESSION for VARIABLE in RANGO_O_LISTA]
```

Example (X^2)

```
l = [x**2 for x in range(10)]  
print(l)
```

```
l2 = [abs(x) for x in range(-10, 10)]  
print(l2)
```

Funciones (Subprogramas)

- Retazos de código que se les asigna un identificador.
- Se ejecutan cada vez que se los llama por el nombre.

Signatura

- Nombre de la función.
- Parámetros: Nombre, cantidad, orden y tipos de datos aceptado.
- Tipo de dato de retorno

Definir una Función

Sintaxis

```
def nombre_de_la_funcion(parametros):  
    "Texto que comenta qué hace la función"  
    #  
    # código de la función aquí  
    #  
    # si no hay código usar:  
    pass
```

Funciones: Ejemplos

Example (Definir una función)

```
def hola_mundo():  
    "Escribe un saludo"  
    print("Hola Mundo")
```

Example (Función con Parámetros)

```
def fib(n):  
    """Retorna el n-ésimo número de Fibonacci"""  
    a, b = 0, 1  
    i = 0  
    while a < n:  
        a, b = b, a + b  
        i = i + 1  
    return a
```

Funciones Recursivas

Son funciones que se llaman a sí mismas.

No siempre son más eficientes, pero el código es muy legible.

Example (Funciones Recursivas)

```
def fib2(n):  
    """  
    Retorna el n-ésimo número de Fibonacci.  
    """  
    if (n >= 2):  
        return fib(n-1) + fib(n-2)  
    else:  
        return n
```

- Usar nombres explicativos.
- Tratar de documentar sus funciones pensando “qué hace” y no “cómo lo hace”.
- Dividir el programa acorde a lo que debe hacer en cada paso.
- Se puede empezar construyendo sus programas:
 - Desde la idea general e ir hacia lo más específico (*top-down*).
 - Desde funciones específicas hacia la idea general (*bottom-up*).

Módulos y Paquetes

Módulos

Archivos con definiciones y sentencias que pueden ser cargadas en Python para su posterior uso.

Paquetes

Los paquetes son una colección de módulos.

Todos los paquetes de Python se pueden instalar con:

Anaconda Navigator Recomendado.

conda `https://conda.io` Ej.: `conda search numpy`

pip Se puede buscar paquetes en `https://pypi.org/` Ej.: `pip install pyfpgrowth`

Importar Paquetes

Example (Importar Selectivamente (Recomendado))

```
from math import pi, tan

print(pi)
```

Example (Importar Todo)

```
import math
import numpy as np

print(math.pi)
# Cuarto, quinto y sexto valor son inválidos.
ma = np.ma.array(range(6), mask=[0, 0, 0, 1, 1, 1])
print(ma.mean())
```

Example (Paquetes)

```
from scipy.stats.mstats import describe
import numpy as np
ma = np.ma.array(range(6), mask=[0, 0, 0, 1, 1, 1])
describe(ma)
```

Para Todo Programador

- Tratar de modularizar: Divide y vencerás
 - Separar el código en funciones
 - Separar un archivo en varios
- ¡Explicar su código!
 - Va a ser leído por muchas personas.
 - Muchas veces, van a tardar en leerlo mucho más del que se tarda en escribirlo.
 - Usen comentarios, identificadores explicativos y comenten sus funciones.
 - ¿Cuántas veces pasa que leemos un código que hicimos hace tiempo y no lo entendemos?
 - Personalmente: ¡Muchas veces!
- Compartan el código
 - Otras personas pueden encontrar errores y ayudar a repararlos
 - Brinda apoyo y motivación a la comunidad

<https://www.python.org/dev/peps/pep-0008/>

- No usar caracteres de tabulación (usar siempre espacios)
- Limitar las líneas a 79 caracteres
 - Se puede leer en dispositivos móviles.
 - Se pueden usar dos columnas de código.
- `class` y `def` deben estar separadas por dos líneas en blancos
- Los archivos deben estar en UTF-8
- Evitar líneas en blanco con espacios y espacios al final de las sentencias.

PEP8 En Emacs

```
anaconda

!!l = [x++2 for x in range(10)]
print(l)

!!l = [abs(x) for x in range(-10, 10)]
print(l)

!!def nombre_de_la_funcion(parametros):
    "Texto que comenta qué hace la función"
    #
    # # codigo de la función aquí
    #
    # # si no hay código usar:
    pass

!def hola_mundo():
    "Escribe un saludo"
    ! print("Hola Mundo")
!

!def fib(n):
    W291 trailing whitespace
    """
    Retorna el n-ésimo número de Fibonacci
    """
    !! a, b = 0, 1
    !! i = 0
    !! while i <= n:
    !!     a, b = b, a + b
    !!     i = i+1
    !! return a

!!def fib2(n:int) :
    """
    Retorna el n-ésimo número de Fibonacci.
    """
    !! if (n >= 2):
    !!     return fib(n-1) + fib(n-2)
    !! else:
    !!     return n

~/c/G/Carpynteria/anaconda.py 108:0 Bot LF UTF-8 Python 3.7.2 28
E101 indentation contains mixed spaces and tabs, W191 indentation contains tabs
```



PEP8 En Terminal

```
▶ flake8 anaconda.py ^ /dev/null
anaconda.py:52:11: E221 multiple spaces before operator
anaconda.py:59:7: F821 undefined name 'i'
anaconda.py:60:11: F821 undefined name 'i'
anaconda.py:61:9: F821 undefined name 'i'
anaconda.py:74:23: W291 trailing whitespace
anaconda.py:77:1: E741 ambiguous variable name 'l'
anaconda.py:80:1: E741 ambiguous variable name 'l'
anaconda.py:83:1: E302 expected 2 blank lines, found 1
anaconda.py:86:8: E114 indentation is not a multiple of four (comment)
anaconda.py:86:8: E116 unexpected indentation (comment)
anaconda.py:88:7: E114 indentation is not a multiple of four (comment)
anaconda.py:88:7: E116 unexpected indentation (comment)
anaconda.py:91:1: E302 expected 2 blank lines, found 1
anaconda.py:91:18: W291 trailing whitespace
anaconda.py:93:24: W291 trailing whitespace
anaconda.py:94:1: W293 blank line contains whitespace
anaconda.py:95:1: E302 expected 2 blank lines, found 1
anaconda.py:95:11: E203 whitespace before ':'
anaconda.py:99:11: E225 missing whitespace around operator
anaconda.py:101:12: E225 missing whitespace around operator
anaconda.py:102:13: E225 missing whitespace around operator
anaconda.py:106:1: E302 expected 2 blank lines, found 1
anaconda.py:106:11: E231 missing whitespace after ':'
anaconda.py:106:15: E202 whitespace before ')'
anaconda.py:106:19: E203 whitespace before ':'
anaconda.py:108:1: W191 indentation contains tabs
anaconda.py:108:1: E101 indentation contains mixed spaces and tabs
anaconda.py:109:1: E101 indentation contains mixed spaces and tabs
```



Literate Programming con Python y Markdown

Es un estilo de programación introducido por Donald Knuth (T_EX en 1972, WEB en 1992).

Se basa en desarrollar programas **en el orden fijado por la lógica y el flujo de pensamientos.**



En la implementación...

Escribir la explicación del algoritmo y sus partes junto con *snippets* de su código que lo implementa.

¡Empecemos!

Vamos a hacer tres bloques:

- El título de nuestro texto en Markdown.
- Un texto explicativo de dónde sacamos las cosas y qué vamos a hacer.
- Un código Python para importar las librerías.

Notebooks de ejemplo en: <http://crowd.fi.uncoma.edu.ar/wiki/doku.php/christian:home>

K-Means Example

<http://scikit-learn.org/stable/modules/clustering.html#k-means>

The data set is explained [\[at the source code\]](https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/datasets/descr/digits.rst)(<https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/datasets/descr/digits.rst>) and [\[at its official Web page\]](https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits)(<https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits>).

The following code, import needed libraries:

```
- **numpy** for importing all type of useful scientific objects and functions  
- **matplotlib** for graphic support  
- **sklearn** (SciKit-learn) is a kit with machine learning algorithms. K-Means is included!
```

```
In [1]: print(__doc__)  
  
from time import time  
import numpy as np  
import matplotlib.pyplot as plt  
  
from sklearn import metrics  
from sklearn.cluster import KMeans  
from sklearn.datasets import load_digits  
from sklearn.decomposition import PCA  
from sklearn.preprocessing import scale
```

¿Qué pasa si ejecutamos los bloques?

Ejecutamos el bloque con código haciendo clic en “Run”:

```
In [9]: print(__doc__)

from time import time
import numpy as np
import matplotlib.pyplot as plt

from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
```

Automatically created module for IPython interactive environment

Loads the digit dataset. `n_digits` is the amount of k the k-means is going to use.

¿Y con el bloque de Markdown?

K-Means Example

<http://scikit-learn.org/stable/modules/clustering.html#k-means>

The data set is explained [at the source code](#) and [at its oficial Web page](#).

The following code, import needed libraries:

- **numpy** for importing all type of useful scientific objects and functions
- **matplotlib** for graphic support
- **sklearn** (SciKit-learn) is a kit with machine learning algorithms. K-Means is included!

```
In [9]: print(__doc__)

from time import time
```

BY-ND

Imágenes en Markdown

Si queremos mostrar imágenes podemos usar la sintaxis `![Título](URL)` o adjuntarla.

¡Ver ejemplo completo en el Notebook [PersonajesCsComp.ipynb](#)!

Example (Imagen Remota)

(Todo en una línea)

```
![Alan Turing](https://upload.wikimedia.org/wikipedia/commons/3/3a/Alan_Turing_Age_16_Colorized.jpg)
```

Example (Imagen Local)

```
![Alonzo Church](./imgs/Alonzo_Church.jpg)
```

Example (Adjuntas)

En el menú “Editar” > “Insertar Imagen” se puede elegir la imagen a adjuntar.

```
![303px-Ada_Lovelace.jpg](attachment:303px-Ada_Lovelace.jpg)
```

¿Y los Gráficos?

Al utilizar matplotlib y se intenta programar un gráfico: Jupyter lo computa y muestra la imagen abajo.

Importar matplotlib:

```
from matplotlib.pyplot as plt
```

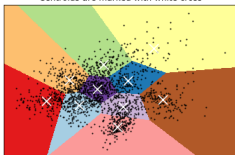
Programar el gráfico y mostrarlo con `plt.show()`.

```
plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)
# Plot the centroids as a white X
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)
plt.title('K-means clustering on the digits dataset (PCA-reduced data)\n'
          'Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

Al hacer clic en “Run”, se obtiene lo siguiente:

```
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross



- <https://docs.anaconda.com/>
- <https://conda.io/en/latest/>

Python

- Tutoriales, referencia del lenguaje y de las bibliotecas estándares:
<https://docs.python.org/3/>
- En español: <http://docs.python.org.ar/tutorial/>

¡Probar `help(EXPRESION)` en Jupyter o `pydoc EXPRESION` en terminal!

Paquetes de Python

- Scipy y Numpy: <https://docs.scipy.org/doc/>
- Pandas user guide y API:
<http://pandas.pydata.org/pandas-docs/stable/>
- SciKit-Learn API, user guide, tutorial, etc.:
<https://scikit-learn.org/stable/documentation.html>
- Matplotlib User's Guide:
<https://matplotlib.org/users/index.html>

- “Python para Todos” por Raúl González Duque:
<https://openlibra.com/en/book/python-para-todos>
- “Introducción a la Programación con Python” por Andrés Becerra Sandoval: <https://openlibra.com/en/book/introduccion-a-la-programacion-con-python-2>

- “Hands-on Python Tutorial” por Andrew N. Harrington:

<https://openlibra.com/en/book/hands-on-python-tutorial>

- “Python for Everybody: Exploring Data Using Python 3” por Charles Severance:

<https://openlibra.com/en/book/python-for-everybody-exploring-data-using-python-3>

¡Muchas Gracias!

¡Muchas Gracias por su Atención!

Licencia de Esta Presentación

Excepto en los lugares que se ha indicado lo contrario:

“Carpyntería 2019” por Christian Gimenez se distribuye bajo
Licencia Creative Commons Atribución-SinDerivadas 4.0 Internacional.



CC-BY-ND

Excepto las imágenes que se referencian en la diapositiva siguiente:

Esta obra está licenciada bajo la **Licencia Creative Commons Atribución-SinDerivadas 4.0 Internacional**. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-nd/4.0/>



Material usado con Otra Licencia



(Anaconda Inc.). Todos los derechos reservados.

Anaconda logo por www.anaconda.com



derechos reservados.

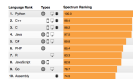
Conda logo por conda.io (Anaconda Inc.). Todos los



Jupyter logo por Cameron Oelsen está bajo la licencia BSD.



Python logo por www.python.org está bajo la licencia GNU General Public License 2 o posterior.



Language Rank obtenido desde la página Web

<https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>
el día 26 de Marzo del 2019 (captura: <http://archive.today/MSSEx>). IEEE todos los
derechos reservados.



Material usado con Otra Licencia



Foto de Don Knuth por Jacob Appelbaum está bajo la licencia CC-BY-Share Alike 2.5 Generic. Visite <https://en.wikipedia.org/wiki/File:KnuthAtOpenContentAlliance.jpg> para más información.



Foto de John Gruber por George del Barrio está bajo la licencia CC-BY-ShareAlike 3.0. https://en.wikipedia.org/wiki/File:John_Gruber.jpeg



Foto de Aaron Swartz por Sage Ross está bajo la licencia CC-BY-ShareAlike 2.0 Generic. https://commons.wikimedia.org/wiki/File:Aaron_Swartz_2_at_Boston_Wikipedia_Meetup,_2009-08-18.jpg

