

II Congreso Colombiano de
**Investigación
Operativa**

ASOCIO 2017

Una introducción a Analytics con Anaconda Python

JUAN DAVID VELÁSQUEZ HENAO, MSc, PhD

Profesor Titular

Departamento de Ciencias de la Computación y la Decisión
Facultad de Minas
Universidad Nacional de Colombia, Sede Medellín

 jdvelasq@unal.edu.co

 @jdvelasquezh

 <https://github.com/jdvelasq>

 <https://goo.gl/prkjAq>

 <https://goo.gl/vXH8jy>

¿Qué es y qué NO es Analytics?

Big Data

Área relacionada con la infraestructura computacional requerida para el almacenamiento de grandes volúmenes de datos y la ejecución de algoritmos de forma paralela y distribuida.

Data Science

Área relacionada con los procesos y sistemas para la extracción de conocimiento de datos almacenados electrónicamente (¿para la toma de decisiones? ¿para probar hipótesis?)

Analytics

Proceso científico de transformación de datos en conocimiento para mejorar el proceso de toma de decisiones [Informs].

- Problema organizacional
- Transformación en un problema de analytics
- Datos
- Selección de la metodología
- Desarrollo del modelo
- Puesta en marcha de la solución
- Gestión del ciclo de vida del modelo

Data Science and Data Scientists: What's in a Name?

Business Intelligence Practitioner

Combinación de negocios + tecnología con el fin de proveer información a las unidades de negocios para toma de decisiones

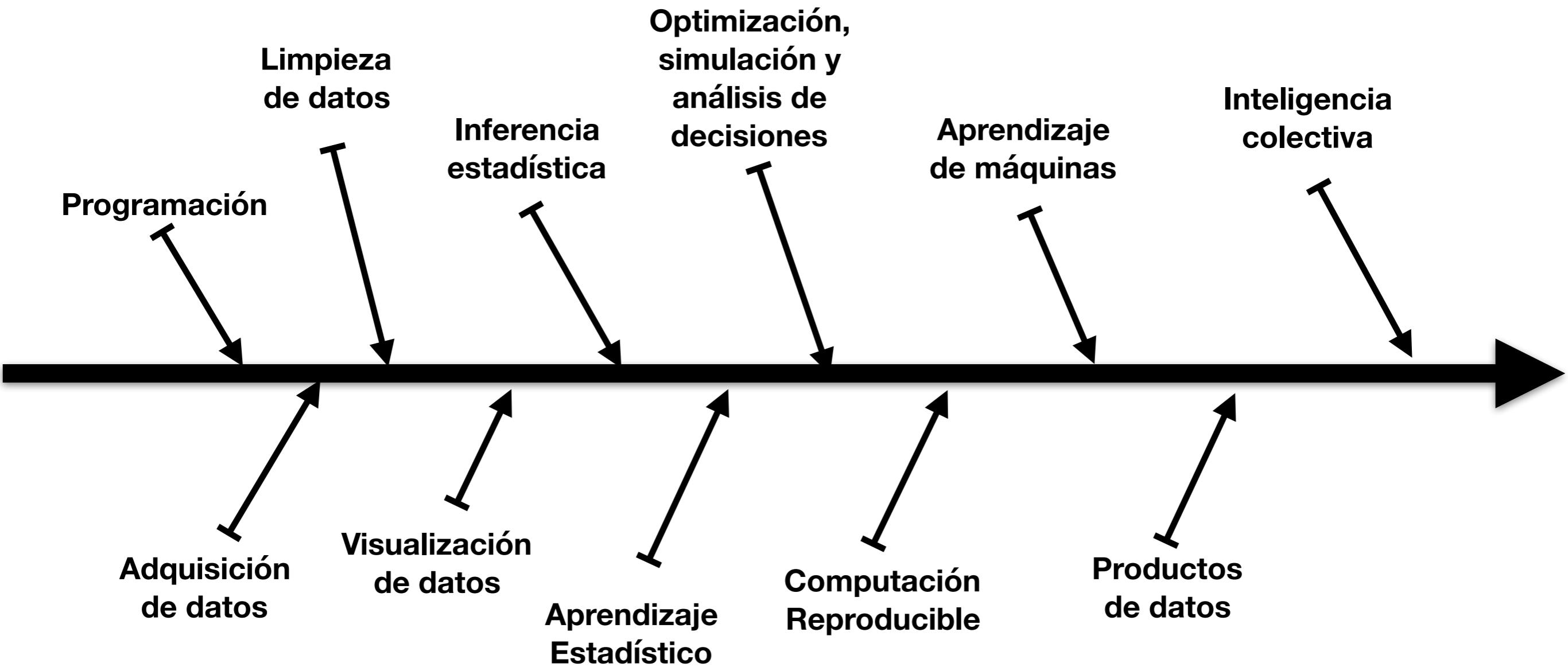
Data Scientist

Habilidades en la programación de computadores para manejo de datos y modelado predictivo (estadística, aprendizaje de máquinas, minería de datos, etc.).

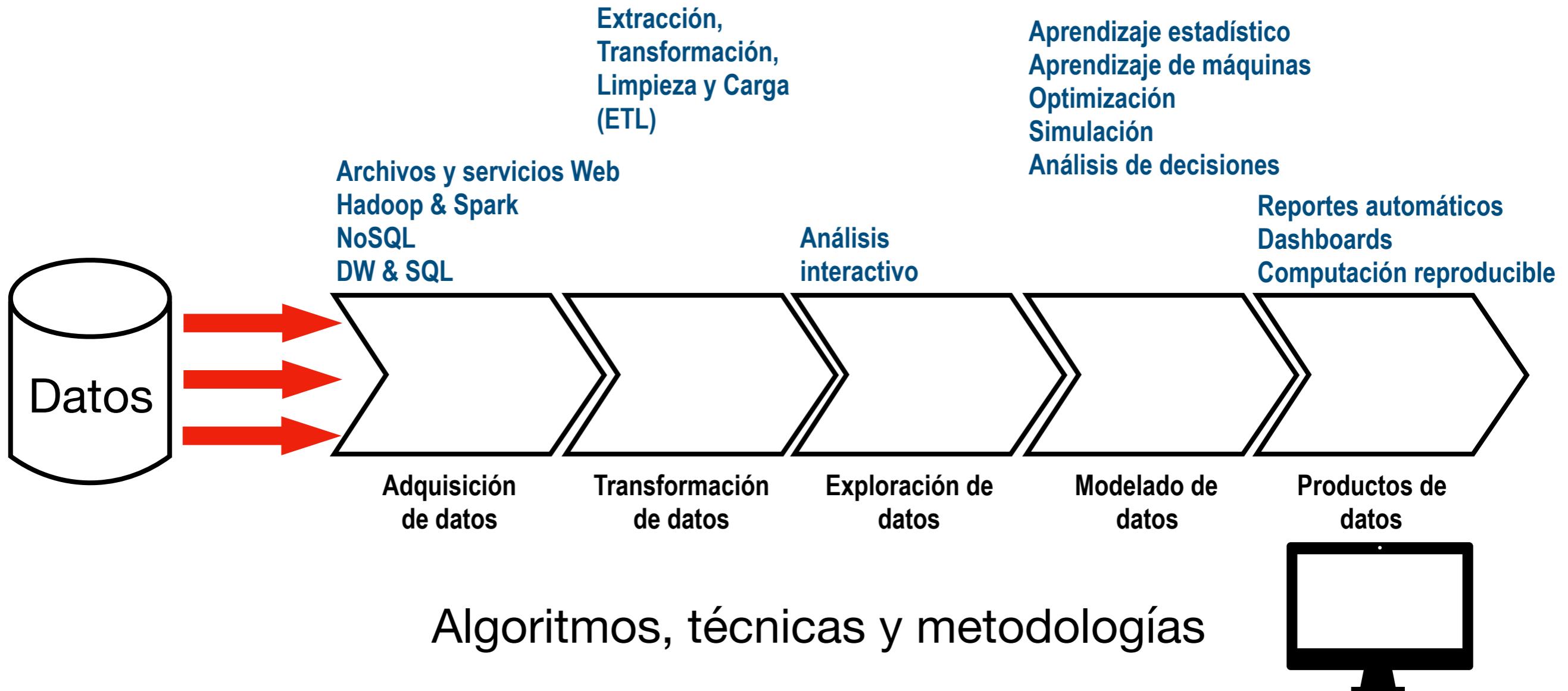
Analytics Practitioner

Data Science + Inteligencia de Negocios + Optimización + Simulación + Toma de Decisiones

Analytics se refiere al proceso integral, no a sus partes



Proceso de modelado en Analytics



Infraestructura computacional

{ Un procesador
Muchos procesadores

{ Computación en máquinas locales
Computación en la nube

El lenguaje Python

Python es un lenguaje de alto nivel desarrollado originalmente por Guido van Rossum en 1989 como un proyecto de programación por hobby; Python sería sucesor del lenguaje ABC, que fue diseñado para prototipado y enseñanza.

Su nombre fue inspirada en la serie de televisión *Monty Python's Flying Circus* transmitida originalmente entre 1969 y 1973.

Python es un lenguaje interpretado y multi-paradigma que soporta completamente la programación orientada a objetos y la programación estructurada. Hay soporte parcial para metaprogramación (los programas son tratados como objetos) y programación orientada a aspectos.

El lenguaje Python

El lenguaje da soporte parcial a la programación funcional: implementa las funciones map(), reduce() y filter(); comprehensions para listas, diccionarios y conjuntos; y generadores.

```
S = [2 * x for x in range(101) if x ** 2 > 3]
```

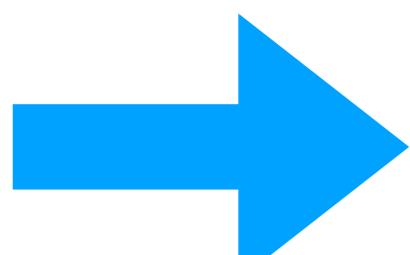
Su diseño se basa en la claridad y la simplicidad (pythonic):

“Es mejor una sola forma obvia de hacer las cosas”.

Es un lenguaje sencillo, con sintaxis inspirada en los lenguajes ALGOL, C, C++, Haskell, Java, ec. al donde la funcionalidad se obtiene a partir de un core sencillo y una gran librería estándar.

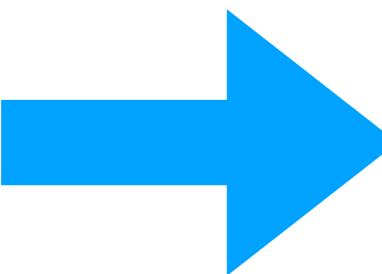
Esto permite que sea ideal para el desarrollo de prototipos.

The 2015 Top Ten Programming Languages (IEEE Spectrum)



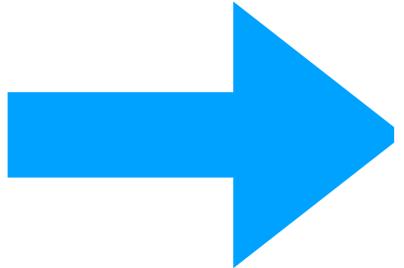
Language Rank	Types	Spectrum Ranking	Spectrum Ranking
1. Java	🌐📱💻	100.0	100.0
2. C	📱💻💻	99.9	99.3
3. C++	📱💻💻	99.4	95.5
4. Python	🌐💻	96.5	93.5
5. C#	🌐📱💻	91.3	92.4
6. R	💻	84.8	84.8
7. PHP	🌐	84.5	84.5
8. JavaScript	🌐📱	83.0	78.9
9. Ruby	🌐💻	76.2	74.3
10. Matlab	💻	72.4	72.8

The 2016 Top Ten Programming Languages (IEEE Spectrum)



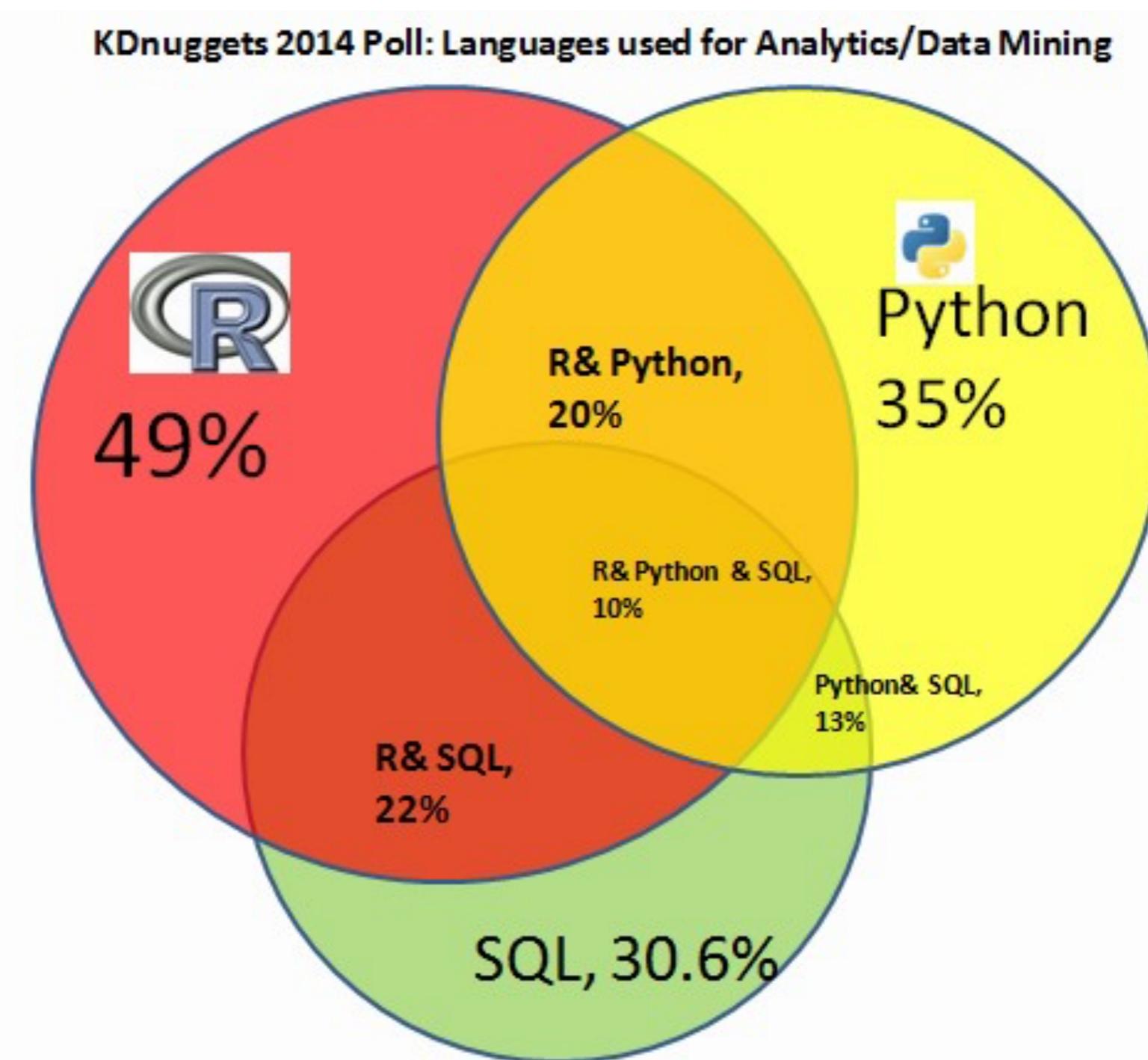
Language Rank	Types	Spectrum Ranking
1. C	📱💻CHIP	100.0
2. Java	🌐📱💻	98.1
3. Python	🌐💻	98.0
4. C++	📱💻CHIP	95.9
5. R	💻	87.9
6. C#	🌐📱💻	86.7
7. PHP	🌐	82.8
8. JavaScript	🌐📱	82.2
9. Ruby	🌐💻	74.5
10. Go	🌐💻	71.9

The 2017 Top Ten Programming Languages (IEEE Spectrum)



Language Rank	Types	Spectrum Ranking
1. Python	🌐💻	100.0
2. C	📱💻🧠	99.7
3. Java	🌐📱💻	99.5
4. C++	📱💻🧠	97.1
5. C#	🌐📱💻	87.7
6. R	💻	87.7
7. JavaScript	🌐📱	85.6
8. PHP	🌐	81.2
9. Go	🌐💻	75.1
10. Swift	📱💻	73.7

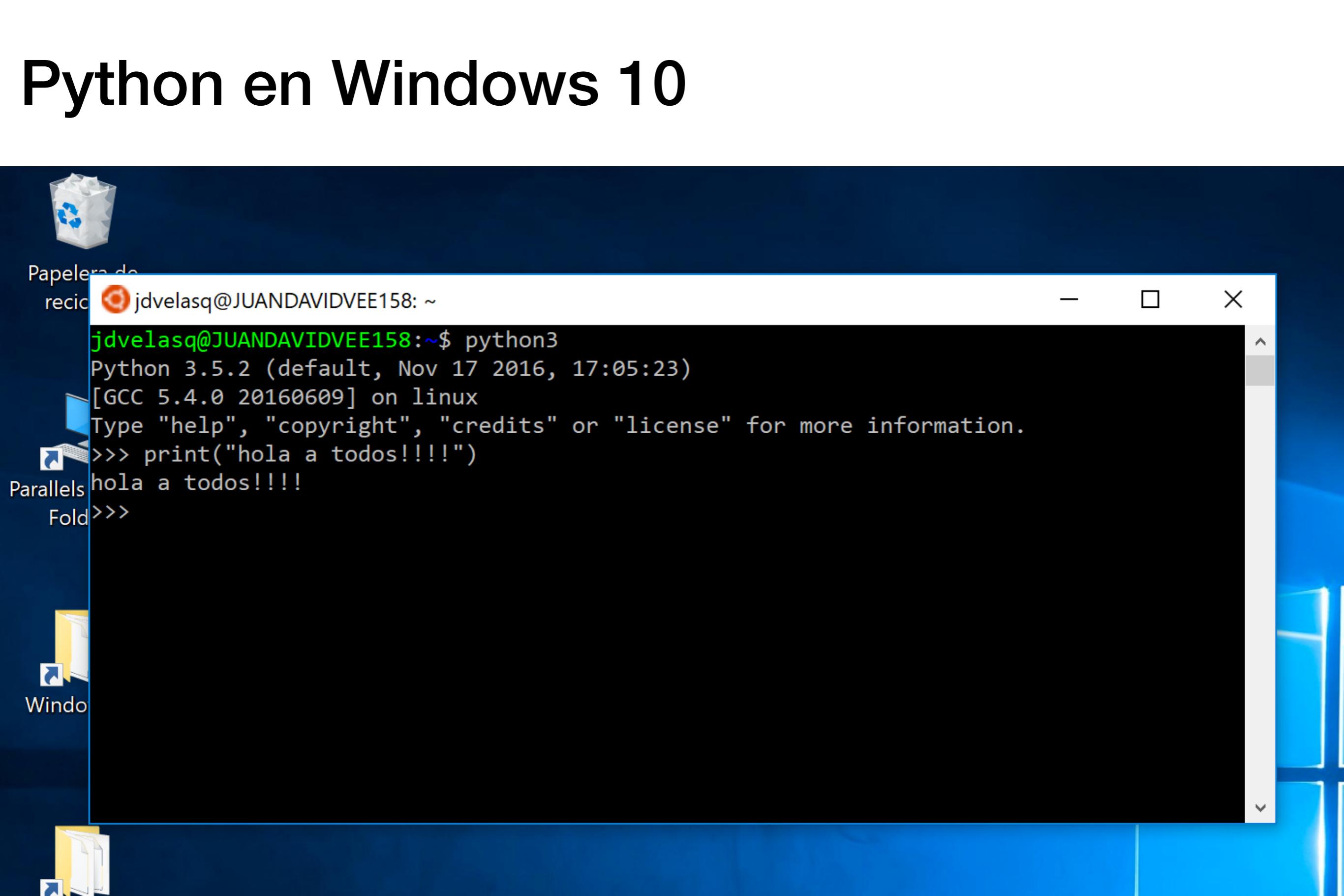
Lenguajes usados en Analytics/Data Science



Distribuciones de Python

- Preinstalado en los sistemas operativos Linux, mac OS y Bash on Ubuntu on Windows
- Python Software Foundations – www.python.org
- Anaconda – www.continuum.io
 - Blaze
 - Bokeh
 - conda
 - Dask
 - llvmlite
 - PhosphorJS
 - Numba
- Enthought Canopy – www.enthought.com/products/canopy/
 - Python for Excel (PyXLL)

Python en Windows 10



A screenshot of a Windows 10 desktop environment. In the center is a terminal window titled 'Papelero de reciclaje' showing a Python session. The desktop background is blue with a grid pattern. Icons for 'Reciclaje', 'Papelero de reciclaje', 'jdvelasq@JUANDAVIDVEE158', 'Parallels', 'Fold', 'Windows', and 'Windows Help和支持中心' are visible on the left.

```
jvelasq@JUANDAVIDVEE158:~$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hola a todos!!!!")
hola a todos!!!!
>>>
```

IPython en Windows 10

```
$ sudo apt install ipython3
```

The screenshot shows a Windows terminal window with two tabs. The top tab is a standard command-line interface where the user has just run the command to install IPython 3. The bottom tab is a Jupyter notebook interface where the user is interacting with the IPython environment.

Top Tab (Command Line):

```
jdvelasq@JUANDAVIDVEE158: ~
jdvelasq@JUANDAVIDVEE158:~$ ipython3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
Type "copyright", "credits" or "license" for more information.

IPython 2.4.1 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref -> Quick reference.      jdvelasq@JUANDAVIDVEE158: ~
help       -> Python's own help sysHelp on built-in function print in module builtins:
object?   -> Details about 'object'
                  print(...)
```

Bottom Tab (Jupyter Notebook):

```
In [1]: print("hola a todos")
hola a todos

In [2]: help(print)
Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file: a file-like object (stream); defaults to the current sys.stdout.
sep: string inserted between values, default a space.
end: string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.

In [3]: (END)
```

Alternativas para ejecutar un programa en Python

En el prompt de comandos

```
$ nano hola.py
```

```
jdvelasq@JUANDAVIDVEE158: ~
```

```
GNU nano 2.5.3
```

```
File: hola.py
```

```
print("hola mundo cruel")
```

```
jdvelasq@JUANDAVIDVEE158: ~
```

```
jdvelasq@JUANDAVIDVEE158:~$ nano hola.py
```

```
jdvelasq@JUANDAVIDVEE158:~$ python3 hola.py
```

```
hola mundo cruel
```

```
jdvelasq@JUANDAVIDVEE158:~$
```

Alternativas para ejecutar un programa en Python

En el prompt de Python de forma interactiva

```
jdvelasq@JUANDAVIDVEE158: ~
jdvelasq@JUANDAVIDVEE158:~$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hola mundo cruel")
hola mundo cruel
>>>
```

Alternativas para ejecutar un programa en Python

En el prompt de IPython de forma interactiva

```
jdvelasq@JUANDAVIDVEE158: ~
jdvelasq@JUANDAVIDVEE158:~$ ipython3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
Type "copyright", "credits" or "license" for more information.

IPython 2.4.1 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: print("hola mundo cruel")
hola mundo cruel

In [2]: quit()
jdvelasq@JUANDAVIDVEE158:~$
```

Alternativas para ejecutar un programa en Python

Desde un entorno de desarrollo

- PyCharm – www.jetbrains.com/pycharm/
- Wingware – <http://wingware.com>
- Spyder – <https://pythonhosted.org/spyder/>
- IDLE (**nativo, viene con la instalación del lenguaje**).
- IDLEX (**extensiones que amplían la capacidad de IDLE**).

Alternativas para ejecutar un programa en Python

Desde un entorno interactivo

Jupyter Notebook

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter Untitled1 (unsaved changes)
- Top Bar:** File, Edit, View, Insert, Cell, Kernel, Help, Trusted, Logout, Python 3
- Toolbar:** Includes icons for file operations (Save, New, Cut, Copy, Paste), cell navigation (Up, Down, Previous, Next), and code editor settings.
- Code Cell 1:** In [1]: `print("hola mundo cruel")`
Output: hola mundo cruel
- Code Cell 2:** In []: [Empty cell]

Alternativas para ejecutar un programa en Python

Desde un entorno interactivo

Apache Zepellin

Multi-purpose Notebook

The Notebook is the place for all your needs

- >Data Ingestion
- Data Discovery
- Data Analytics
- Data Visualization & Collaboration

The screenshot shows the Apache Zeppelin web interface with three open notebooks:

- maxAge**: A pie chart showing the distribution of ages. The legend lists ages from 19 to 34. The chart segments are labeled with their respective values: 34, 26, 27, 28, 29, 30, 31, 32, 33, and 34.
- Under age < 35**: A bar chart showing the count of individuals under 35. The x-axis represents age groups (22, 26) and the y-axis represents the count (0 to 103). The chart is grouped by age.
- marital**: A line chart showing the value of marital status over time. The x-axis ranges from 19 to 69, and the y-axis ranges from 0 to 105. The line starts at (19, 1), peaks at approximately (38, 105), and then generally declines.

Each notebook has a "FINISHED" status indicator and various visualization options like Grouped or Stacked bars.

Alternativas para ejecutar un programa en Python

Desde un entorno interactivo

Beaker / BeakerX

BeakerX: Beaker extensions for Jupyter

[build passing](#) [chat on gitter](#) [JitPack 0.1.1](#) [npm package 0.0.6](#) [pypi](#)

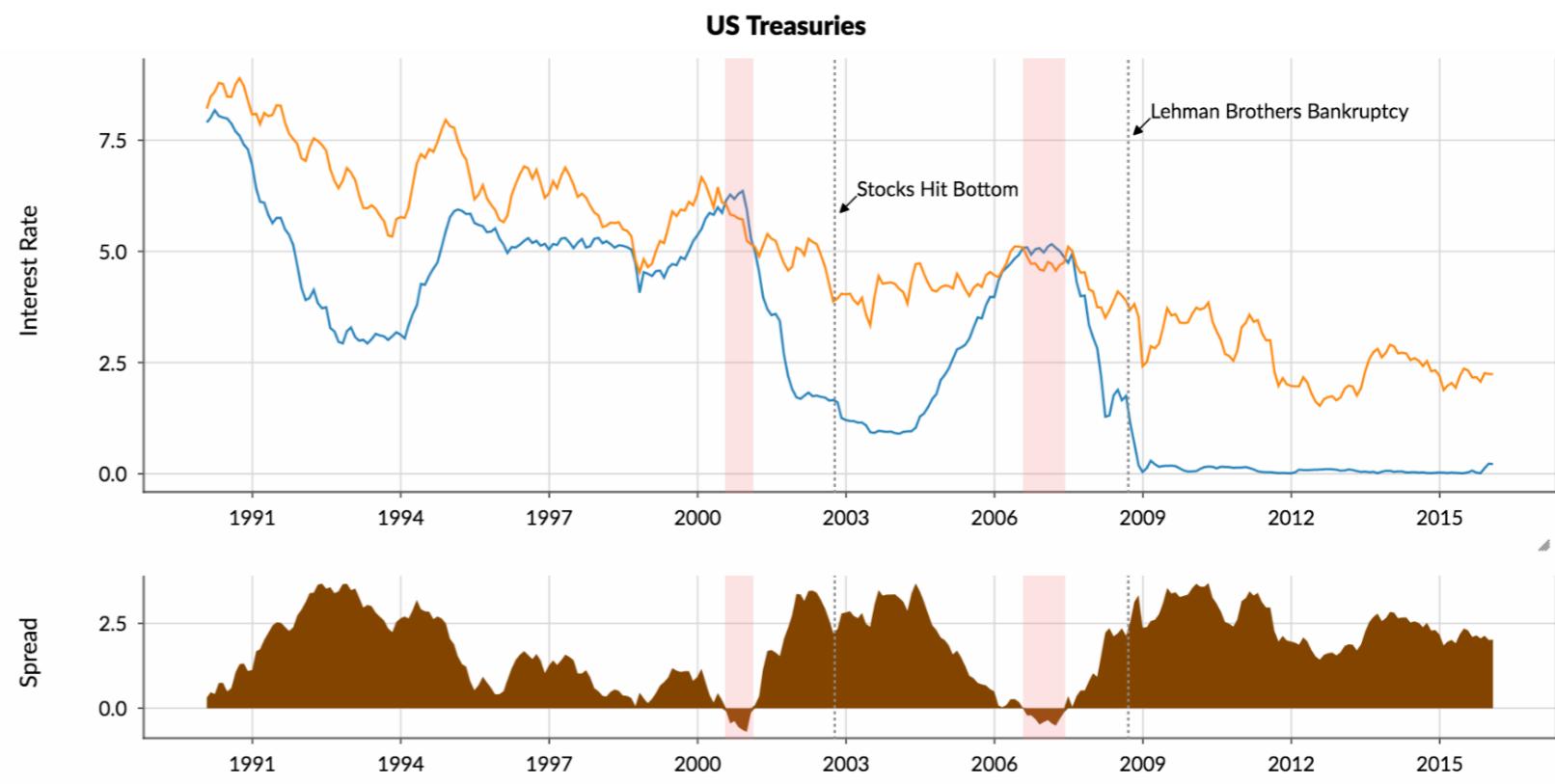
BeakerX is a collection of JVM kernels with widgets, plotting, etc. for Notebook and Jupyter Lab. BeakerX is in alpha, with major feature changes without notice.

The [documentation](#) consists of tutorial notebooks on GitHub.

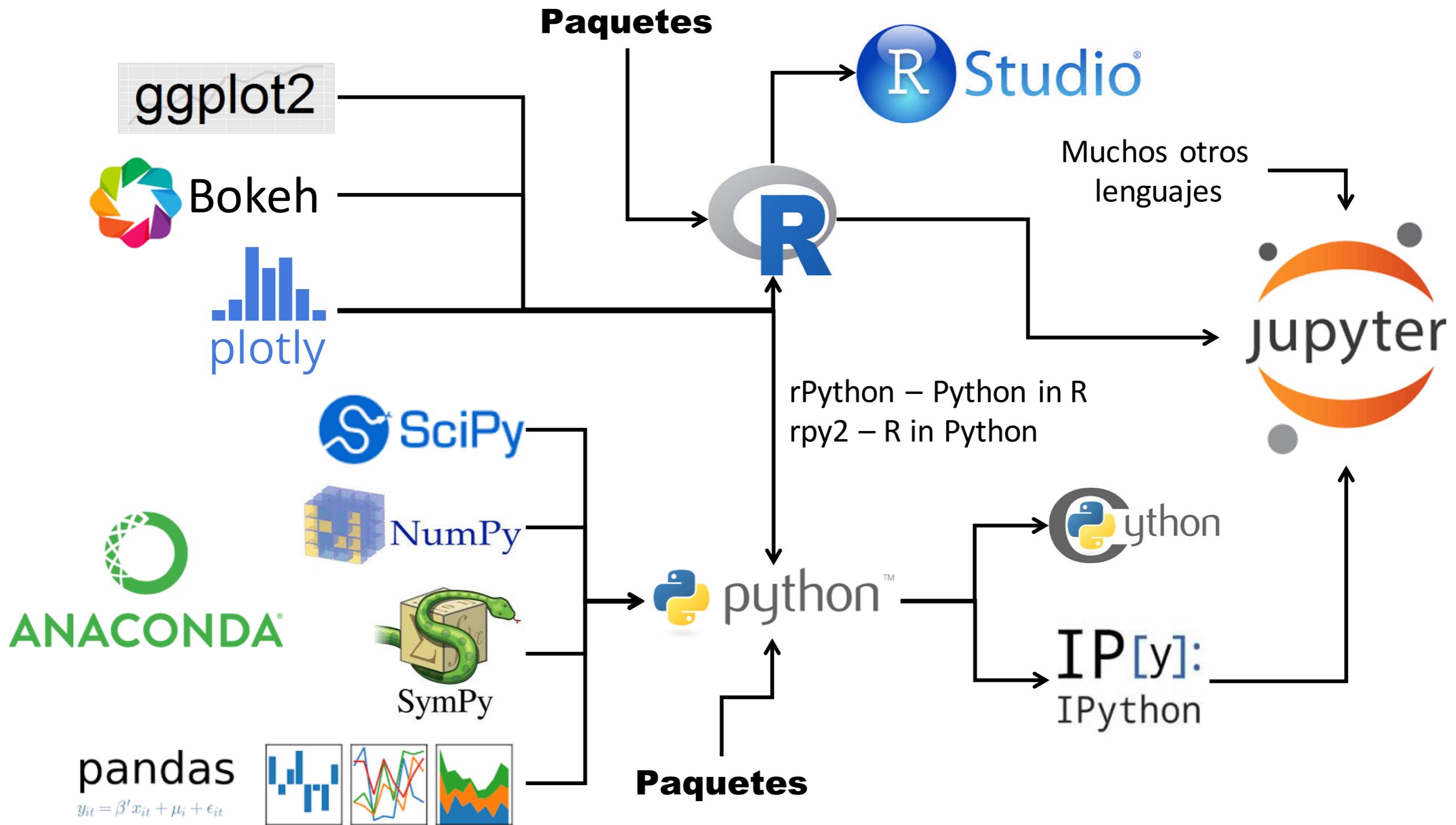
BeakerX is the successor to the [Beaker Notebook](#) ([source code](#)) and we are [hiring](#).

Groovy with Interactive Plotting and Tables:

```
// Then use a CombinedPlot to get stacked plots with linked X axis.  
def c = new CombinedPlot(title: "US Treasuries", initWidth: 1000)  
  
// add both plots to the combined plot, and including their relative heights.  
c.add(p1, 3)  
c.add(p2, 1)
```



Ecosistema de computación científica



Anaconda Python

Instalación de Anaconda Python

<https://www.continuum.io/downloads>



Anaconda Cloud Documentation Blog Contact Q

DOWNLOAD

What Is Anaconda? PRODUCTS SUPPORT & SOLUTIONS COMMUNITY ABOUT RESOURCES

DOWNLOAD ANACONDA DISTRIBUTION

Version: 4.4.0 | Release Date: May 31, 2017

Download for:



[Home](#)[Environments](#)[Projects \(beta\)](#)[Learning](#)[Community](#)[Documentation](#)[Developer Blog](#)[Feedback](#)

Applications on

root

Channels

Refresh



notebook

5.0.0

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

[Launch](#)

qtconsole

4.3.0

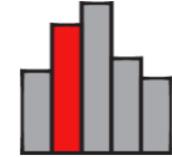
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

[Launch](#)

spyder

3.1.4

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

[Launch](#)

glueviz

0.10.4

Multidimensional data visualization across files. Explore relationships within and among related datasets.

[Install](#)

orange3

3.4.1

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

[Install](#)

rstudio

1.0.153

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

[Install](#)

[Home](#)[Environments](#)[Projects \(beta\)](#)[Learning](#)[Community](#)[Documentation](#)[Developer Blog](#)[Feedback](#)[Documentation \(11\)](#)[Training \(2\)](#)[Video \(20\)](#)[Webinar \(20\)](#)

Anaconda Skills Accelerator Program

[Explore](#)

The Next Generation of Data Products | Hilary Mason | AnacondaCON 2017

[View](#)

Dask: A Pythonic Distributed Data Science Framework PyCon 2017

[View](#)

Open Data Science Education | Jonathan Cornelissen | AnacondaCON 2017

[View](#)

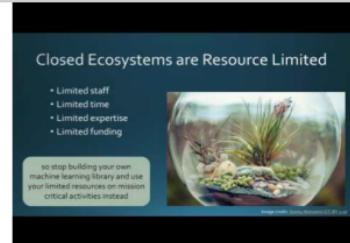
Parallel Data Analysis PyCon 2017

[View](#)

Developing & Deploying Credit Risk Models with Anaconda | Hussain Sultan | AnacondaCON 2017

[View](#)

Future of Data Science in the City of Boston | Andrew Therriault | AnacondaCON 2017

[View](#)

Leveraging Open Technologies in Closed Ecosystems | Dharhas Pothina | AnacondaCON 2017

[View](#)

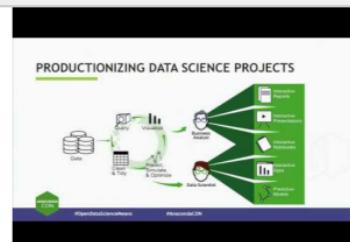
Keynote: The Anaconda Roadmap | Michele Chambers | AnacondaCON 2017

[View](#)

Intel Partner Showcase | AnacondaCON 2017

[View](#)

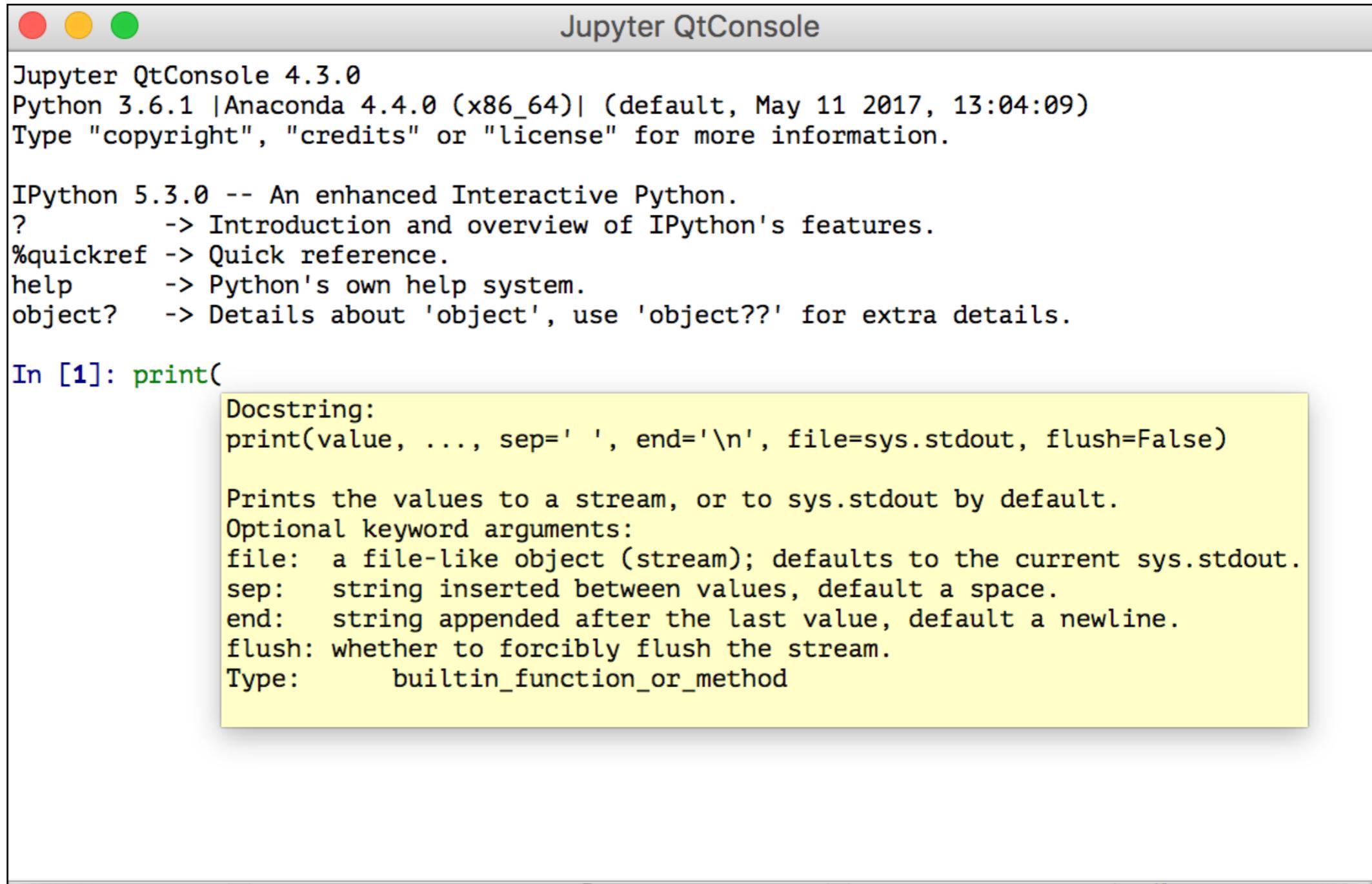
AnacondaCREW Panel | AnacondaCON 2017

[View](#)

Scalable & Deployable Data Science with Anaconda | Kris Overholt | AnacondaCON 2017

[View](#)

Jupyter QtConsole



Jupyter QtConsole 4.3.0
Python 3.6.1 |Anaconda 4.4.0 (x86_64)| (default, May 11 2017, 13:04:09)
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
? → Introduction and overview of IPython's features.
%quickref → Quick reference.
help → Python's own help system.
object? → Details about 'object', use 'object??' for extra details.

In [1]: `print(`

Docstring:
`print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)`

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file: a file-like object (stream); defaults to the current sys.stdout.
sep: string inserted between values, default a space.
end: string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type: builtin_function_or_method

Spyder

Spyder (Python 3.6)

Editor

temp.py*

```
1 print("hola")
```

Arguments

```
print(value, .., sep=' ', end='\n',
      file=sys.stdout, flush=False)
```

Help

Source Console Object

Usage

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Variable explorer File explorer Help

IPython console

Console 1/A

```
Python 3.6.1 |Anaconda 4.4.0 (x86_64)| (default, May 11
2017, 13:04:09)
Type "copyright", "credits" or "license" for more
information.

IPython 5.3.0 -- An enhanced Interactive Python.
?          --> Introduction and overview of IPython's
features.
%quickref --> Quick reference.
help      --> Python's own help system.
object?   --> Details about 'object', use 'object??' for
extra details.

In [1]:
```

Python console History log IPython console

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 1 Column: 13 Memory: 69 %

Jupyter Notebook

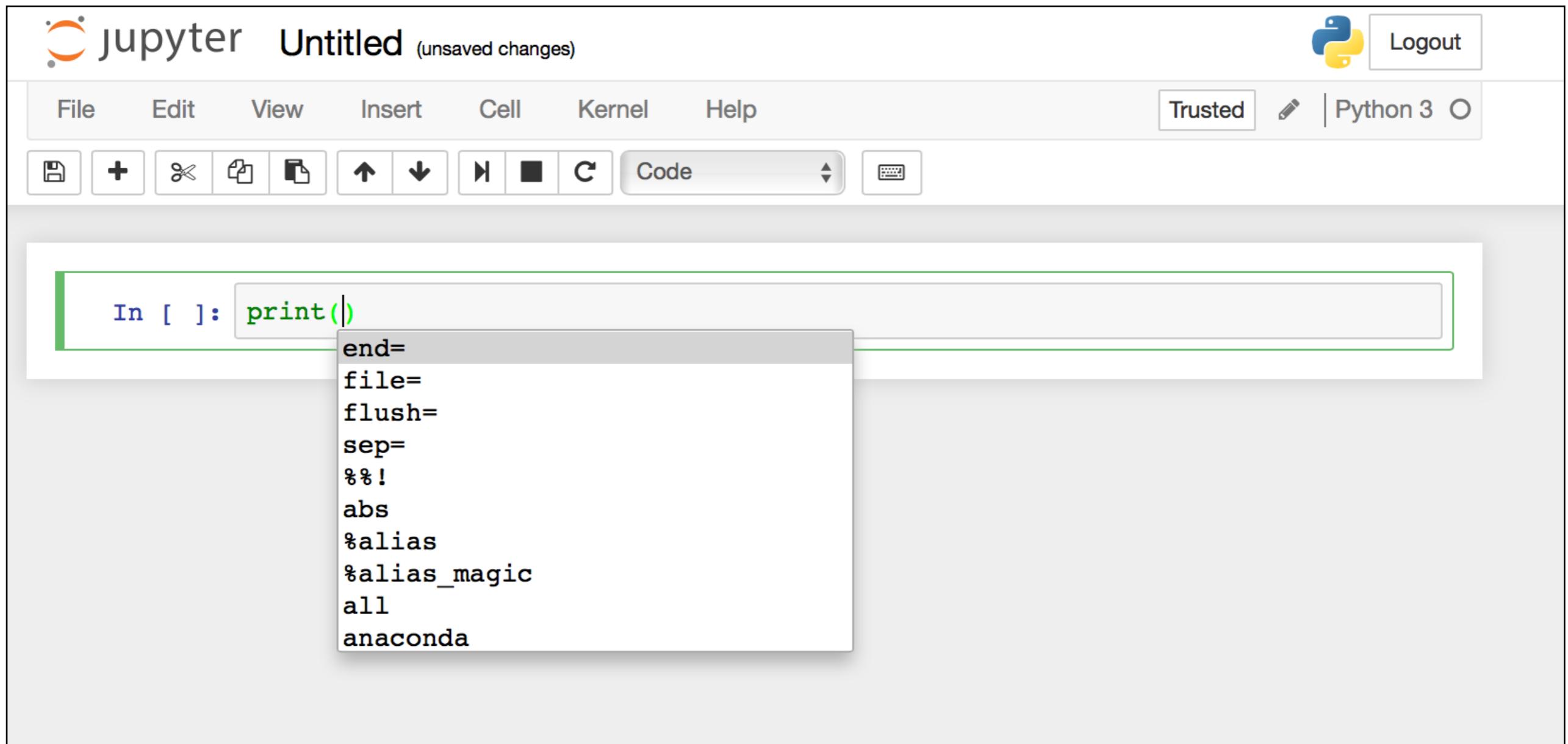
The screenshot shows the Jupyter Notebook interface. At the top left is the "jupyter" logo. On the right are "Logout", "Files" (selected), "Running", and "Clusters" tabs. Below them is a message: "Select items to perform actions on them." To the right is a toolbar with "Upload", "New ▾", and a refresh icon. A context menu is open on the right side, containing:

- Notebook:
 - Bash
 - Python 3
 - R
- Other:
 - Text File
 - Folder
 - Terminal

Two red arrows point to the "New ▾" button and the "Python 3" option in the context menu.

File/Folder	Last Modified
anaconda	5 days ago
Applications	5 days ago
Applications (Parallels)	5 days ago
Archivo de imagen OS X.hdd	5 days ago
Desktop	5 days ago
Documents	4 days ago
Downloads	4 days ago
Library	4 days ago

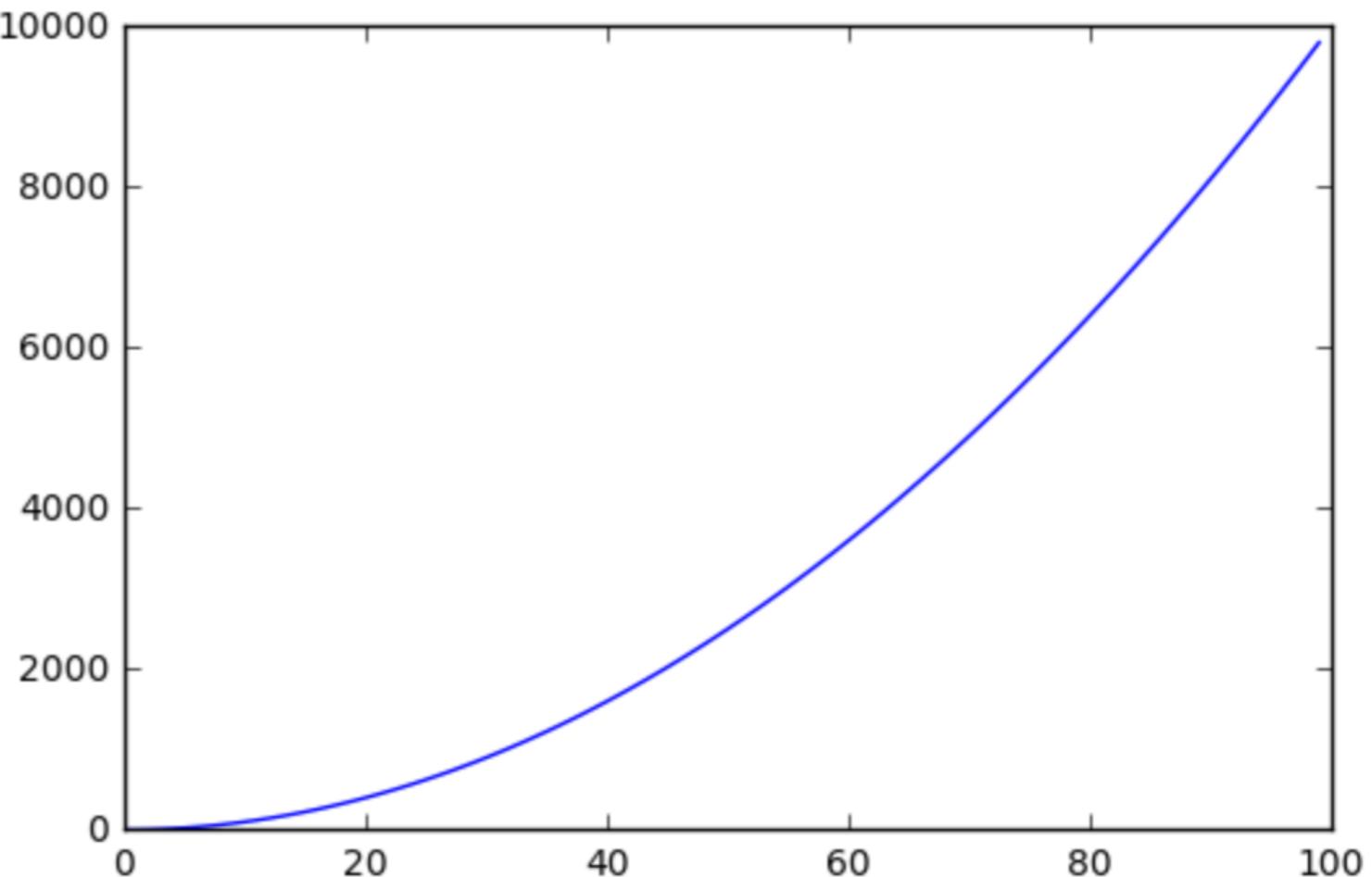
Jupyter Notebook



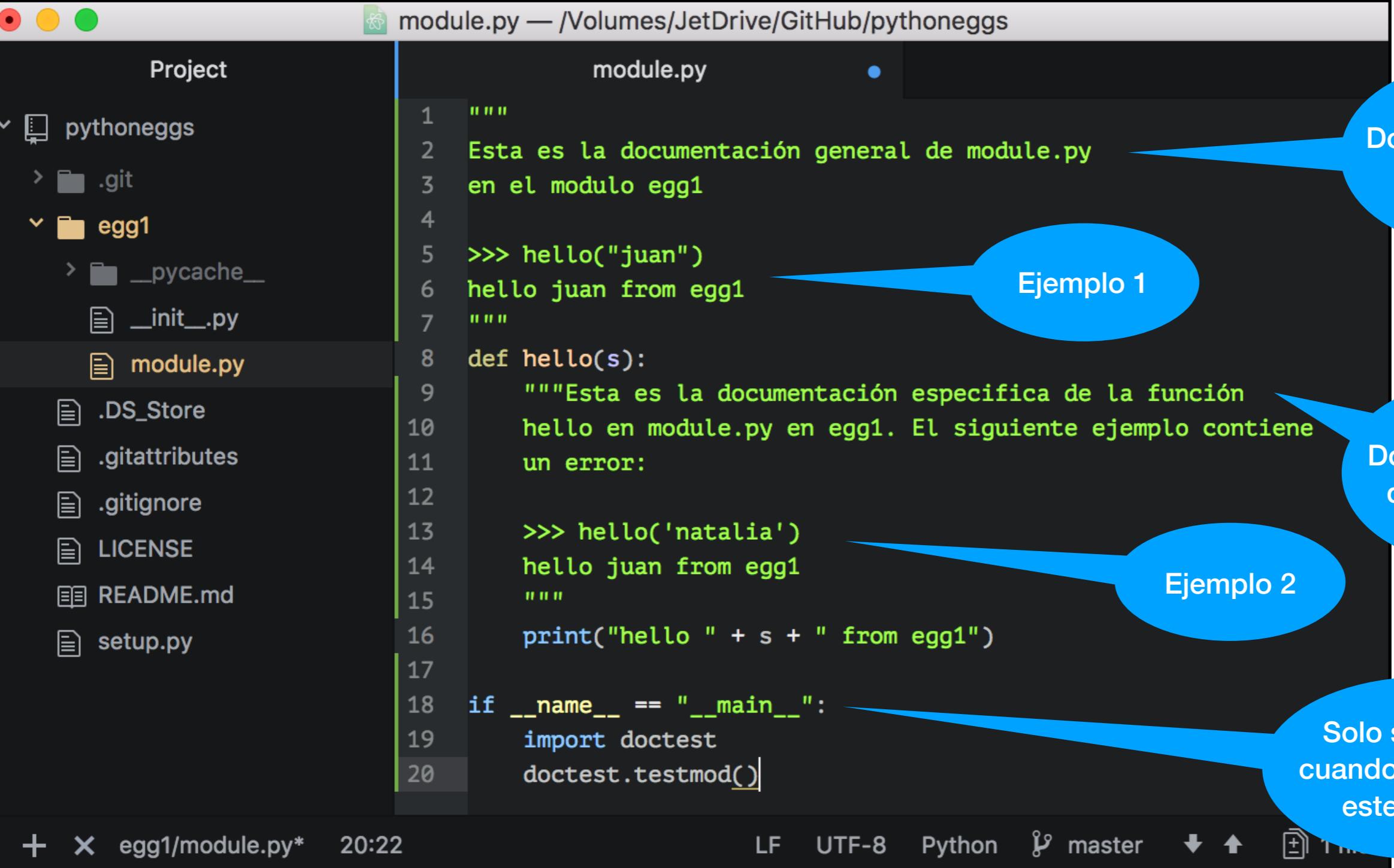
Jupyter Notebook

```
In [1]: import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib as mpl  
%matplotlib inline
```

```
In [2]: x = range(100)  
y = [value ** 2 for value in x]  
plt.plot(x, y)  
plt.show()
```



Programación



```
module.py — /Volumes/JetDrive/GitHub/pythoneggs
Project
pythoneggs
  .git
egg1
  __pycache__
  __init__.py
  module.py
.DS_Store
.gitattributes
.gitignore
LICENSE
README.md
setup.py

module.py
1 """
2 Esta es la documentación general de module.py
3 en el modulo egg1
4
5 >>> hello("juan")
6 hello juan from egg1
7 """
8 def hello(s):
9     """Esta es la documentación específica de la función
10    hello en module.py en egg1. El siguiente ejemplo contiene
11    un error:
12
13 >>> hello('natalia')
14 hello juan from egg1
15 """
16     print("hello " + s + " from egg1")
17
18 if __name__ == "__main__":
19     import doctest
20     doctest.testmod()

Ejemplo 1
Documentación del módulo
Documentación de la función
Ejemplo 2
Solo se ejecuta cuando se compila este módulo
```

The screenshot shows a terminal window with the following details:

- Project:** pythoneggs
- Module:** module.py
- Content:**

```
"""
Esta es la documentación general de module.py
en el modulo egg1

>>> hello("juan")
hello juan from egg1

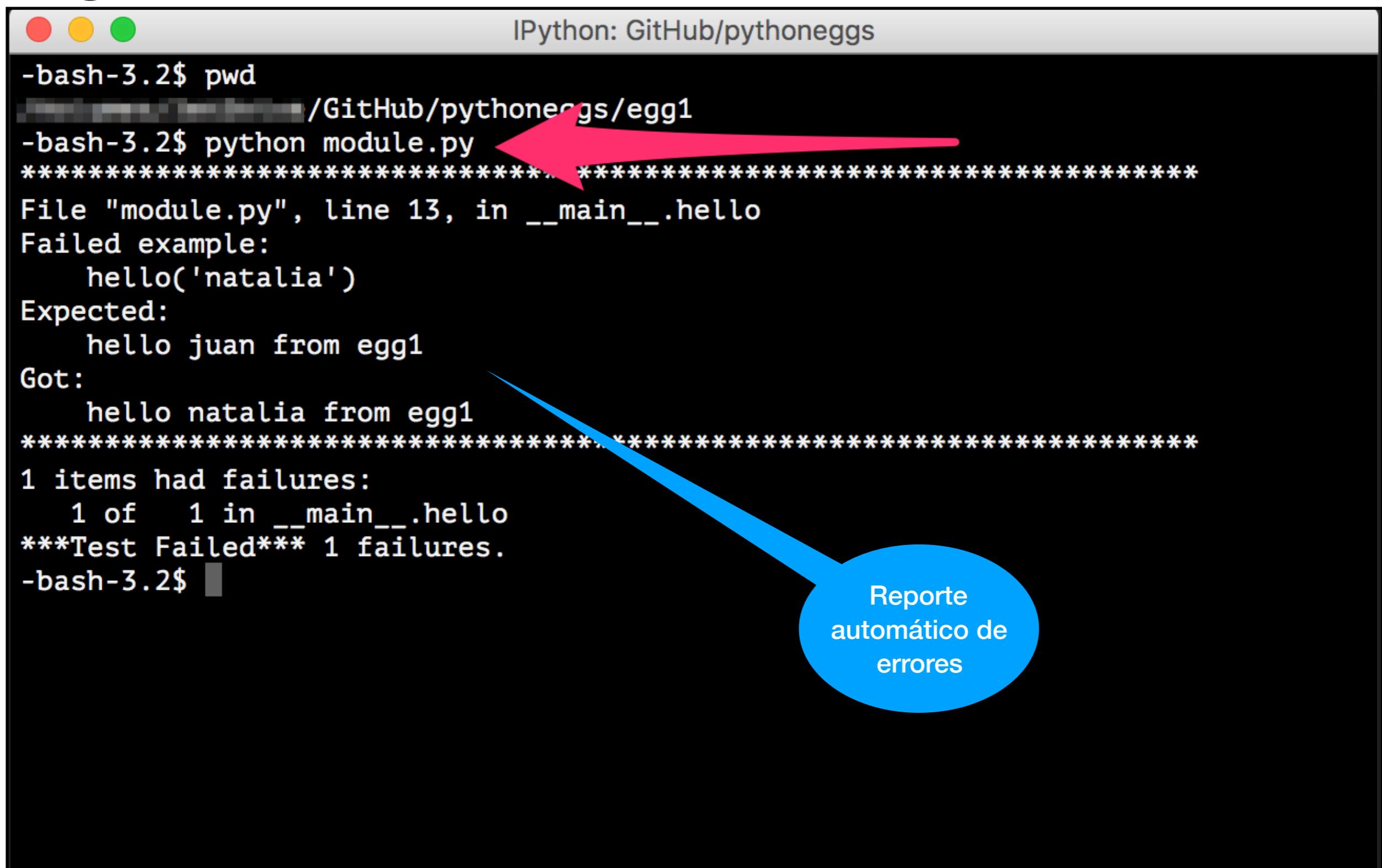
def hello(s):
    """Esta es la documentación específica de la función
    hello en module.py en egg1. El siguiente ejemplo contiene
    un error:

>>> hello('natalia')
hello juan from egg1

print("hello " + s + " from egg1")

if __name__ == "__main__":
    import doctest
    doctest.testmod()
```
- Annotations:**
 - A blue callout points to the first multi-line comment with the text "Documentación del módulo".
 - A blue callout points to the function's docstring with the text "Documentación de la función".
 - A blue callout points to the conditional block at the bottom with the text "Solo se ejecuta cuando se compila este módulo".
 - A blue callout points to the code block above the conditional with the text "Ejemplo 1".
 - A blue callout points to the code block below the conditional with the text "Ejemplo 2".
- Terminal Status:**
 - File: egg1/module.py*
 - Time: 20:22
 - Encoding: LF
 - Format: UTF-8
 - Language: Python
 - Branch: master
 - Icons: download, upload, file, terminal

Programación



-bash-3.2\$ pwd
/GitHub/pythoneggs/egg1

-bash-3.2\$ python module.py

File "module.py", line 13, in __main__.hello
Failed example:
 hello('natalia')
Expected:
 hello juan from egg1
Got:
 hello natalia from egg1

1 items had failures:
 1 of 1 in __main__.hello
Test Failed 1 failures.

-bash-3.2\$

Reporte automático de errores

Programación

The screenshot shows a GitHub repository page for 'IPython-for-data-science' by jdvelasq. The repository has 1 commit, 0 issues, 0 pull requests, 0 projects, and 1 contributor. It is licensed under MIT. The README file contains a section titled 'Cálculos numéricos' with code examples demonstrating basic arithmetic operations in IPython.

jdvelasq / IPython-for-data-science

This repository Search Pull requests Issues Marketplace Gist

Unwatch 1 Star 0 Fork 19

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Notas de clase - Extracción, transformación, visualización y carga de datos usando IPython

Add topics

3 commits 1 branch 0 releases 1 contributor MIT

Branch: master New pull request

jdvelasq readme

Anaconda white papers

files

images

precios-energia

.gitignore

IPy-01-uso-interactivo.ipynb

IPy-02-programacion.ipynb

Cálculos numéricos

Contenido

IPython puede ser usado de forma interactiva como una calculadora. Esto permite que el análisis de datos pueda ser realizado de forma interactiva, de forma similar a como pueden usarse otras herramientas como el lenguaje R o Matlab. A continuación se ejemplifican los cálculos aritméticos básicos.

```
In [1]: 2 + 2 + 1
Out[1]: 5
```

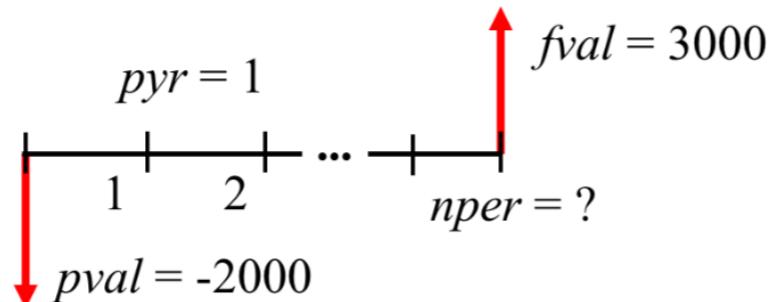
```
In [1]: 50 - 5 * 6 + 8
Out[1]: 28
```

```
In [3]: (50 - 5 * 6) / 4
Out[3]: 5.0
```

<https://github.com/jdvelasq/IPython-for-data-science>

Programación

Ejemplo.-- [3, pág. 88] Se depositan \$ 2000 en una cuenta de ahorros que paga un interés anual del 7.2% (capitalizado anualmente). Si no se hacen otros depósitos en la cuenta, ¿cuánto tiempo se requiere para que la cuenta tenga \$ 3000? R/ 5.83



```
In [9]: cf.pvfv(nrate = 7.2, #  
           pval = -2000, #  
           fval = +3000) #
```

Out[9]: 5.8318433820838607

```
In [10]: # Ya que nper es un valor ent.  
# para tener un balance de al  
# El balance al final de los  
cf.pvfv(nrate = 7.2, # ta  
        pval = -2000, # va  
        nper = 6) # nu
```

Out[10]: 3035.2796326007801

cashflows

Financial investment modeling and advanced engineering economics using Python.

The library can be installed as usual using pip :

```
pip install cashflows
```

Click [here](#) to access the last documentation of the package.

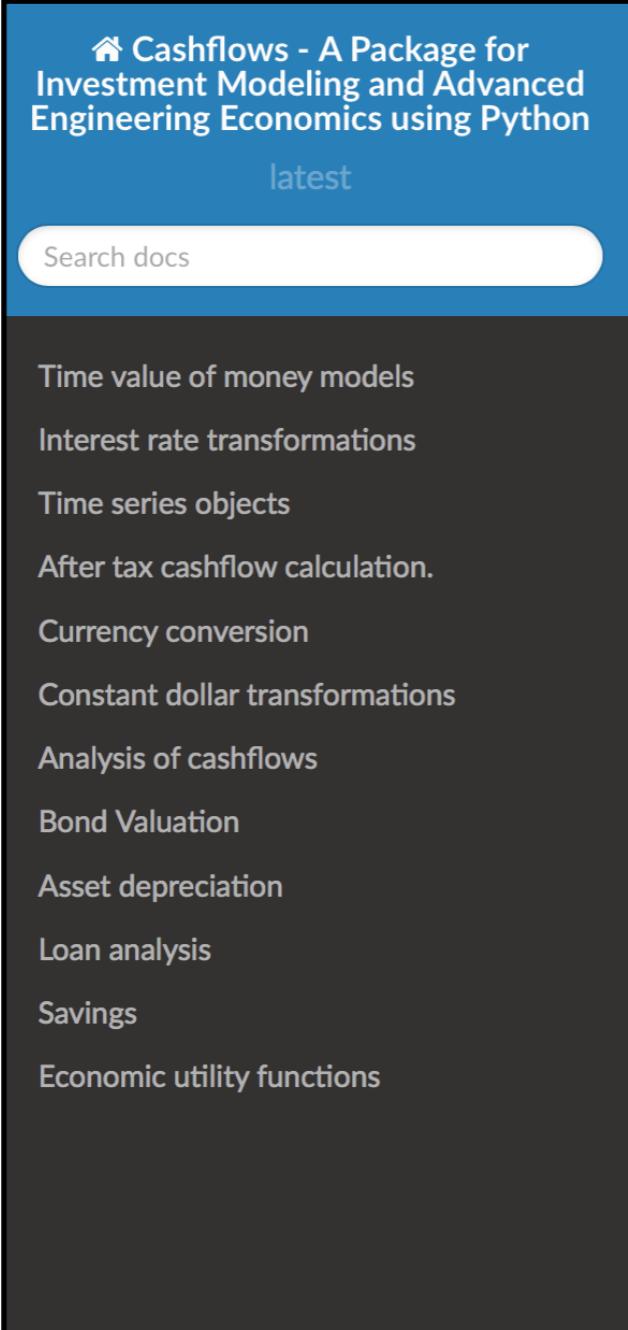
Programación

Programación en la máquina local

Testing automático usando unittests y doctests

Control de versiones en GitHub

Documentación automática con Sphinx



The screenshot shows the documentation for the `Cashflows` Python package. The top navigation bar includes the package name, a "latest" link, and a search bar. The main content area lists various financial models and calculations:

- Time value of money models
- Interest rate transformations
- Time series objects
- After tax cashflow calculation.
- Currency conversion
- Constant dollar transformations
- Analysis of cashflows
- Bond Valuation
- Asset depreciation
- Loan analysis
- Savings
- Economic utility functions

On the right side of the page, there is a sidebar with a "Docs »" link, the package name, and a "Edit on GitHub" button. Below the sidebar, the title of the package is displayed in large, bold, dark font:

Cashflows: A Package for Investment Modeling and Advanced Engineering Economics using Python

Under the title, there is a "Contents:" section followed by a bulleted list of the same items listed in the sidebar:

- Time value of money models
- Interest rate transformations
- Time series objects
- After tax cashflow calculation.
- Currency conversion
- Constant dollar transformations
- Analysis of cashflows
- Bond Valuation
- Asset depreciation
- Loan analysis
- Savings
- Economic utility functions

Programación

Computación
reproducible con
conda

Conda user cheat sheet

CONTINUUM[®]
ANALYTICS

Take a conda test drive at bit.ly/tryconda

For full documentation of any command, type the command followed by **--help**.
conda create --help

TIP: Many options after two dashes (--) have shortcuts.

conda create --help or **conda create -h**

Managing conda and anaconda

conda info

Verify conda is installed, check version #

conda update conda

Update conda package and environment manager to current version

conda update anaconda

Update the anaconda meta package (the library of packages ready to install with **conda** command)

Managing environments

conda info --envs or **conda info -e**

Get a list of all my environments, active environment shown with *

conda create --name snowflakes biopython Create an environment and install program(s)

or

conda create -n snowflakes biopython

TIP: To avoid dependency conflicts, install all programs in the environment (snowflakes) at the same time.

TIP: Environments install by default into the envs directory in your conda directory. You can specify a different path; see **conda create --help** for details.

source activate snowflakes (Linux, OS X)

Activate the new environment to use it

activate snowflakes (Windows)

TIP: Activate prepends the path to the snowflakes environment.

conda create -n bunnies python=3.4 astroid Create a new environment, specify Python version

Adquisición de datos

Archivos y servicios Web

- JSON, CSV, XLS
- Formato Científico & GIS formats
- Open source
 - Pandas
 - HDF5 / NetCDF4
 - Blaze

Big Data & Spark

- Blaze ecosystem
- pyspark

NoSQL

- Blaze
- Odo
- pymongo

DW & SQL

- Blaze
- SQLAlchemy
- pyODBC
- mxODBC

Archivos delimitados por caracteres con Pandas

Contenido

[pandas.DataFrame.to_csv](#)
[pandas.DataFrame.from_csv](#)
[pandas.to_csv](#)
[pandas.read_csv](#)

```
In [61]: ## escribe el archivo
df.to_csv('files/data.csv',      # el nombre del archivo
           index = False)       # imprime los nombres de las filas?

## verifica el archivo creado
print(open('files/data.csv', 'r').read())
```

```
index,name,value
1,A,3.03
2,B,5.14
3,C,0.4
4,D,1.13
5,E,8.25
```

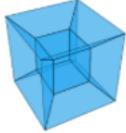
```
In [62]: pandas.read_csv('files/data.csv')
```

```
Out[62]:   index name  value
            0     1    A  3.03
            1     2    B  5.14
            2     3    C  0.40
            3     4    D  1.13
            4     5    E  8.25
```

Adquisición de datos

Blaze

Sponsored by:
CONTINUUM®
ANALYTICS

 The Blaze Ecosystem

The Blaze ecosystem is a set of libraries that help users store, describe, query and process data. It is composed of the following core projects:

- [Blaze](#): An interface to query data on different storage systems
- [Dask](#): Parallel computing through task scheduling and blocked algorithms
- [Datashape](#): A data description language
- [DyND](#): A C++ library for dynamic, multidimensional arrays
- [Libndtypes](#): A C/C++ library for a low-level version of Datashape
- [Ndtypes-python](#): Python bindings for libndtypes
- [Odo](#): Data migration between different storage systems

Adquisición de datos

Blaze

Combining separate, gzipped csv files.

```
>>> from blaze import odo
>>> from pandas import DataFrame
>>> odo(example('accounts_*csv.gz'), DataFrame)
   id      name  amount
0   1      Alice     100
1   2        Bob     200
2   3    Charlie     300
3   4       Dan     400
4   5     Edith     500
```

Adquisición de datos

Blaze

Split-Apply-Combine

```
>>> from blaze import data, by
>>> t = data('sqlite:///iris.db') % example('iris.db')
>>> t.peek()
   sepal_length  sepal_width  petal_length  petal_width      species
0            5.1          3.5           1.4          0.2  Iris-setosa
1            4.9          3.0           1.4          0.2  Iris-setosa
2            4.7          3.2           1.3          0.2  Iris-setosa
3            4.6          3.1           1.5          0.2  Iris-setosa
4            5.0          3.6           1.4          0.2  Iris-setosa
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
...
>>> by(t.species, max=t.petal_length.max(), min=t.petal_length.min())
      species  max  min
0  Iris-setosa  1.9  1.0
1 Iris-versicolor  5.1  3.0
2 Iris-virginica  6.9  4.5
```

Adquisición de datos

Odo

Overview

Odo migrates between many formats. These include in-memory structures like `list`, `pd.DataFrame` and `np.ndarray` and also data outside of Python like CSV/JSON/HDF5 files, SQL databases, data on remote machines, and the Hadoop File System.

The `odo` function

`odo` takes two arguments, a source and a target for a data transfer.

```
>>> from odo import odo  
>>> odo(source, target) # load source into target
```

It efficiently migrates data from the source to the target.

Adquisición de datos

SQLite

```
1 import sqlite3
2 conn = sqlite3.connect('example.db')
3
4 c = conn.cursor()
5 c.execute('''
6     CREATE TABLE person
7         (id INTEGER PRIMARY KEY ASC, name varchar(250) NOT NULL)
8     ''')
9 c.execute('''
10    CREATE TABLE address
11        (id INTEGER PRIMARY KEY ASC, street_name varchar(250), street_number varchar(
12            250),
13             post_code varchar(250) NOT NULL, person_id INTEGER NOT NULL,
14             FOREIGN KEY(person_id) REFERENCES person(id))
15     ''')
16 c.execute('''
17     INSERT INTO person VALUES(1, 'pythoncentral')
18     ''')
19 c.execute('''
20     INSERT INTO address VALUES(1, 'python road', '1', '00000', 1)
21     ''')
22
23 conn.commit()
24 conn.close()
```

```
1 import sqlite3
2 conn = sqlite3.connect('example.db')
3
4 c = conn.cursor()
5 c.execute('SELECT * FROM person')
6 print c.fetchall()
7 c.execute('SELECT * FROM address')
8 print c.fetchall()
9 conn.close()
```

Adquisición de datos

SQLAlchemy

```
1 import os
2 import sys
3 from sqlalchemy import Column, ForeignKey, Integer, String
4 from sqlalchemy.ext.declarative import declarative_base
5 from sqlalchemy.orm import relationship
6 from sqlalchemy import create_engine
7
8 Base = declarative_base()
9
10 class Person(Base):
11     __tablename__ = 'person'
12     # Here we define columns for the table person
13     # Notice that each column is also a normal Python instance attribute.
14     id = Column(Integer, primary_key=True)
15     name = Column(String(250), nullable=False)
16
17 class Address(Base):
18     __tablename__ = 'address'
19     # Here we define columns for the table address.
20     # Notice that each column is also a normal Python instance attribute.
21     id = Column(Integer, primary_key=True)
22     street_name = Column(String(250))
23     street_number = Column(String(250))
24     post_code = Column(String(250), nullable=False)
25     person_id = Column(Integer, ForeignKey('person.id'))
26     person = relationship(Person)
27
28 # Create an engine that stores data in the local directory's
29 # sqlalchemy_example.db file.
30 engine = create_engine('sqlite:///sqlalchemy_example.db')
31
32 # Create all tables in the engine. This is equivalent to "Create Table"
33 # statements in raw SQL.
34 Base.metadata.create_all(engine)
```

Hardware

Numba – <https://numba.pydata.org>

ipyparallel – <https://github.com/ipython/ipyparallel>

mpi4py – <http://pythonhosted.org/mpi4py/>

Theano – <http://deeplearning.net/software/theano/>

pyCUDA – <https://mathematician.de/software/pycuda/>

```
from numba import jit
from numpy import arange

# jit decorator tells Numba to compile this function.
# The argument types will be inferred by Numba when function is called.
@jit
def sum2d(arr):
    M, N = arr.shape
    result = 0.0
    for i in range(M):
        for j in range(N):
            result += arr[i,j]
    return result

a = arange(9).reshape(3,3)
print(sum2d(a))
```

Analytics

Preparación de datos

- Pandas
- Blaze
- GeoPandas
- R plyr, R dplyr, R tidyr, R reshape2, ...

Estadística

- SciPy
- PyMC
- StatsModels

Machine Learning & Deep Learning

- Scikit-learn
- NLTK
- NetworkX
- Theano
- pycaffe
- Pylearn2
- R caret, R glmnet, R randomForest

Simulación y optimización

- SimPy
- PyJMI
- PyFMI
- PyMC
- Pyomo
- CVXOPT
- CVXPY
- tao4py
- pyopt
- Pylpopt
- PyGMO

Analytics

StatsModels

```
In [1]: import numpy as np

In [2]: import statsmodels.api as sm

In [3]: import statsmodels.formula.api as smf

# Load data
In [4]: dat = sm.datasets.get_rdataset("Guerry", "HistData").data

# Fit regression model (using the natural log of one of the regressors)
In [5]: results = smf.ols('Lottery ~ Literacy + np.log(Pop1831)', data=dat).fit()

# Inspect the results
In [6]: print(results.summary())
                OLS Regression Results
=====
Dep. Variable:          Lottery    R-squared:       0.348
Model:                  OLS        Adj. R-squared:   0.333
Method:                 Least Squares   F-statistic:     22.20
Date:      Tue, 28 Feb 2017   Prob (F-statistic):  1.90e-08
Time:           21:38:05    Log-Likelihood:   -379.82
No. Observations:      86        AIC:             765.6
Df Residuals:          83        BIC:             773.0
Df Model:                   2
Covariance Type:    nonrobust
```

Analytics

Pyomo



HOME / ABOUT / DOWNLOAD / DOCUMENTATION / BLOG

Documentation

Online Documentation

[Pyomo Online Documentation \(html, pdf, epub\)](#)
[PySP Online Documentation \(pdf\)](#)
[Pyomo Wikipedia Page \(html\)](#)

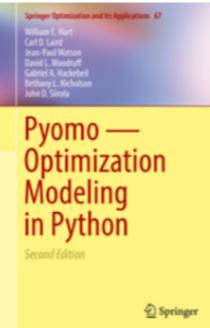
Examples

[Pyomo Gallery \(browse\)](#)
Online examples from the Pyomo software repository: ([browse](#)) ([zipfile](#))

Citation

If you use Pyomo for your work, please cite the Pyomo book ([bibtex](#)) and the Pyomo paper ([bibtex](#)).
If you use PySP for your work, please cite the PySP paper ([bibtex](#)).

The Pyomo Book



Hart, William E., Carl D. Laird, Jean-Paul Watson, David L. Woodruff, Gabriel A. Hackebeil, Bethany L. Nicholson, and John D. Siirola. *Pyomo – Optimization Modeling in Python*. Second Edition. Vol. 67. Springer, 2017.

The Second Edition of the book describes capabilities in the Pyomo 5.x series. The First Edition (2012) describes the capabilities from the Coopr 3.1 release. Some changes beginning in the Pyomo 4.0 release are not backwards compatible with the First Edition.

Storyboards

Notebooks

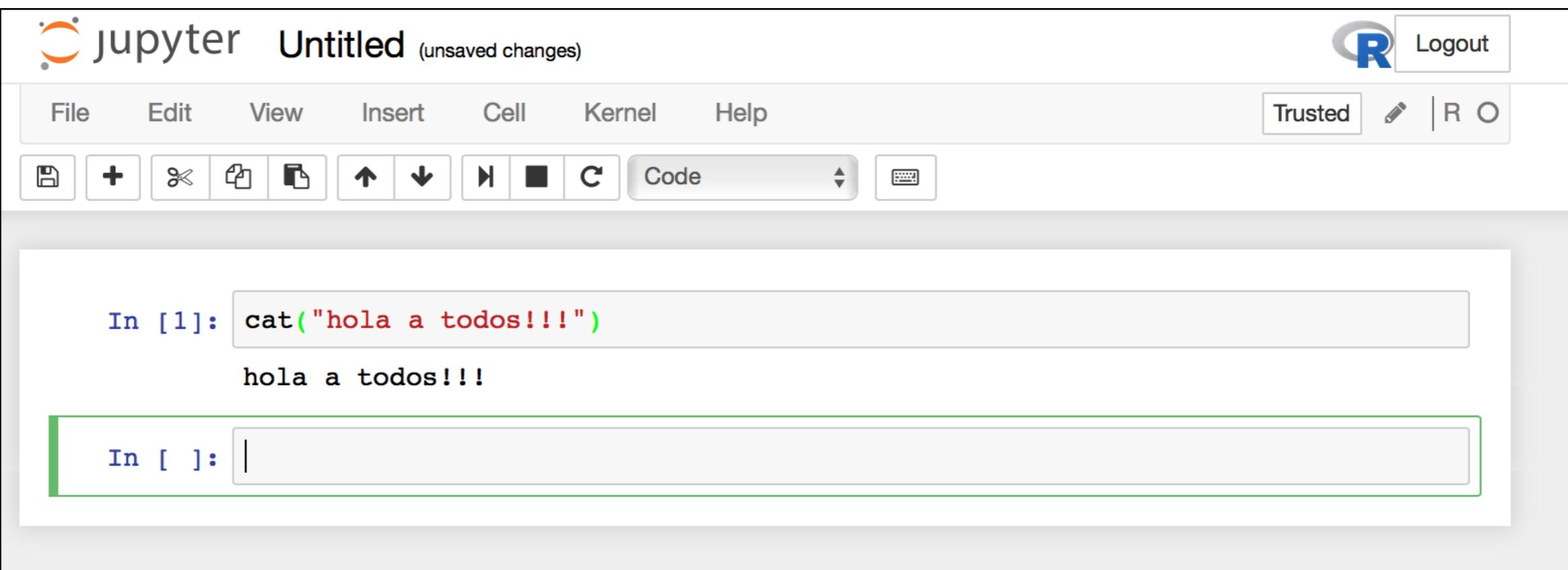
- Jupyter / IPython
- R Jupyter
- R Markdown

Data IDEs

- R Studio
- Spyder

Exploración interactiva

- Bokeh
- Jupyter
- R Shiny



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter Untitled (unsaved changes)
- User Information:** R Logout
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Help, Trusted, R, O
- Tool Buttons:** Save, New, Cut, Copy, Paste, Up, Down, Run, Cell, Code, Keyboard.
- Code Cell:** In [1]: `cat("hola a todos!!!")`
Output: hola a todos!!!
- Input Cell:** In []: |

Storyboards

Bokeh

```
from bokeh.plotting import figure, output_file, show

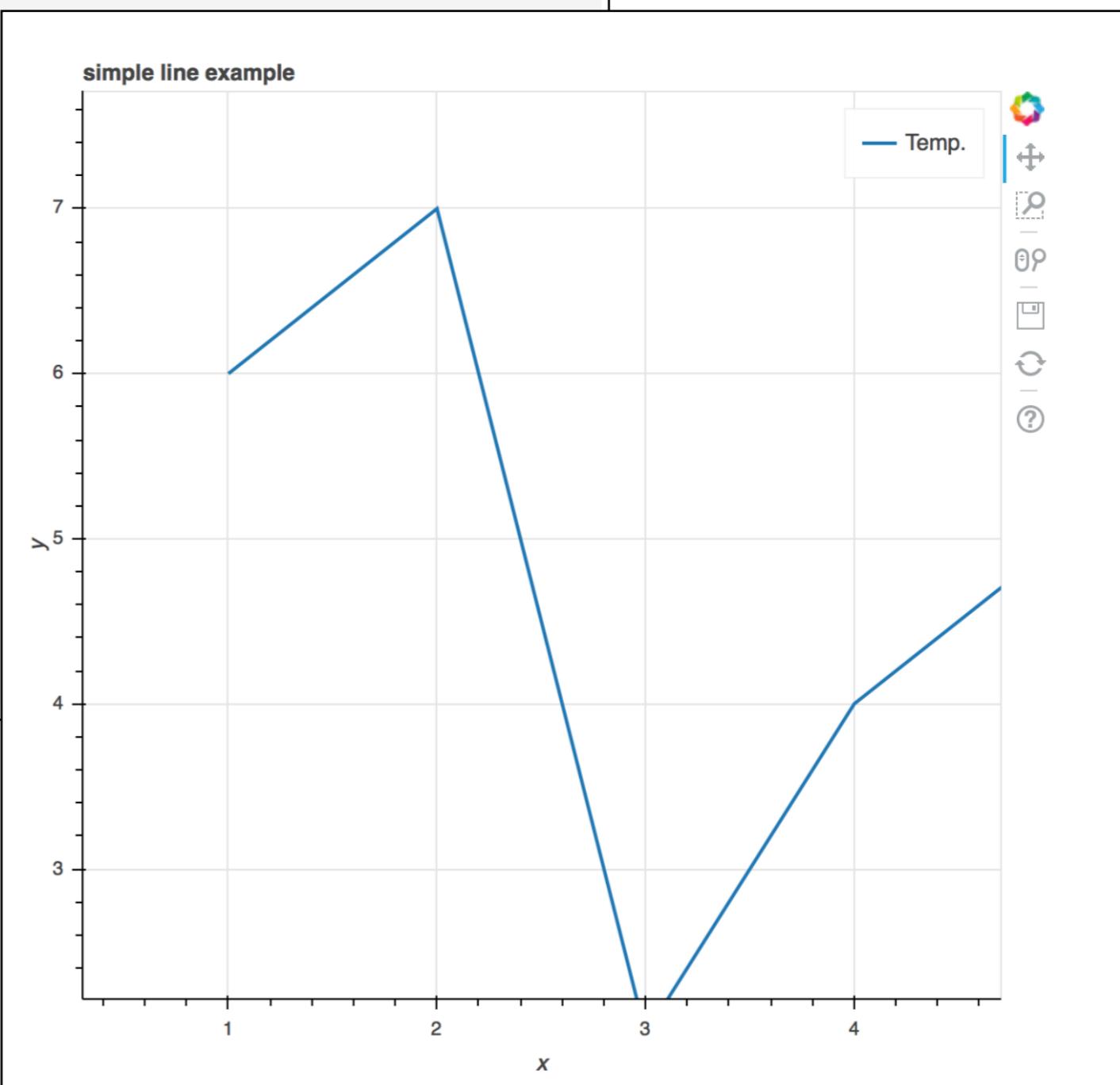
# prepare some data
x = [1, 2, 3, 4, 5]
y = [6, 7, 2, 4, 5]

# output to static HTML file
output_file("lines.html")

# create a new plot with a title and axis labels
p = figure(title="simple line example", x_axis_label='x')

# add a line renderer with legend and line thickness
p.line(x, y, legend="Temp.", line_width=2)

# show the results
show(p)
```



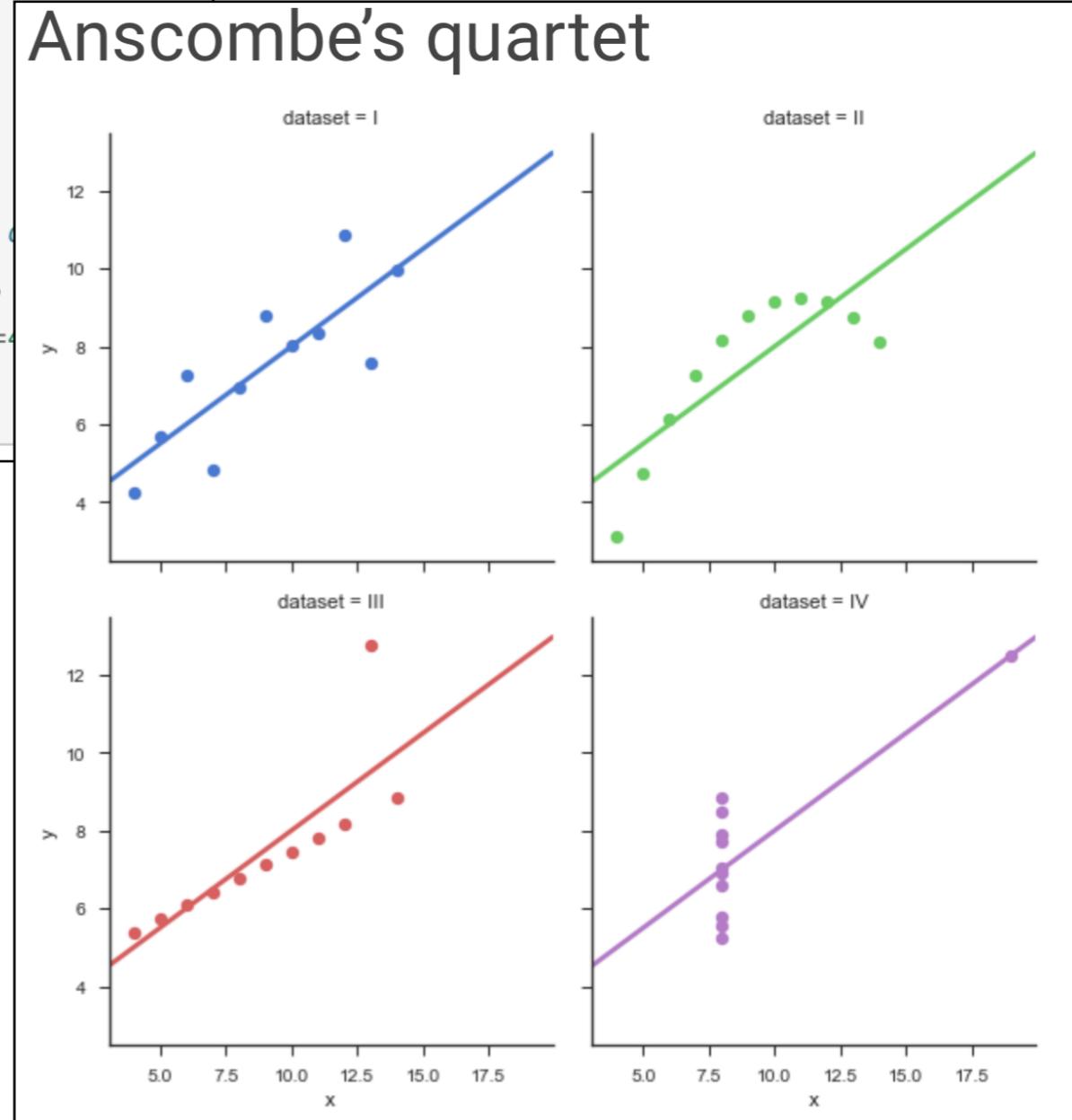
Visualización

- Bokeh
- matplotlib
- seaborn
- R ggplot2
- PyQTgraph
- Veusz
- NetworkX

```
import seaborn as sns
sns.set(style="ticks")

# Load the example dataset for Anscombe's quartet
df = sns.load_dataset("anscombe")

# Show the results of a linear regression within each dataset
sns.lmplot(x="x", y="y", col="dataset", hue="dataset",
            col_wrap=2, ci=None, palette="muted", size=4,
            scatter_kws={"s": 50, "alpha": 1})
```



Interfases de usuario

- Flask
- Pyforms
- Pyramid
- PyQt
- Tkinter
- WxPython



Flask
web development,
one drop at a time

[overview](#) // [docs](#) // [community](#) // [snippets](#) // [extensions](#)

Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions. And before you ask: It's [BSD licensed!](#)!

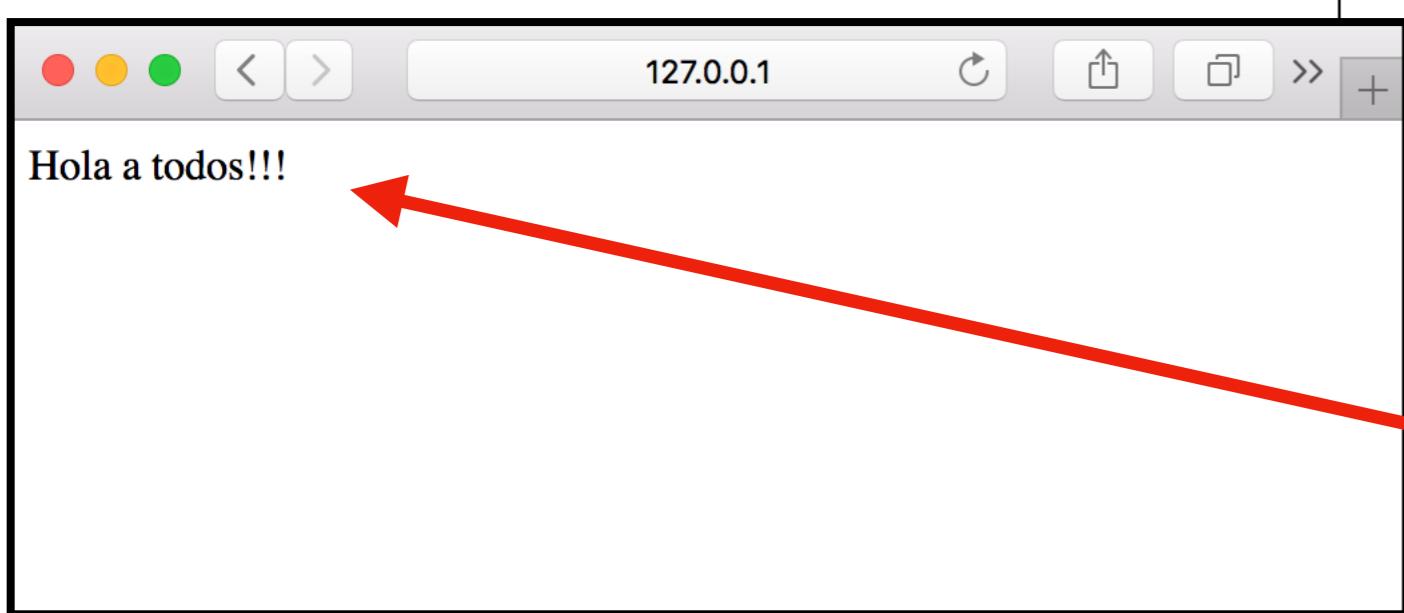
Flask is Fun Latest Version: [0.12.2](#)

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"
```

And Easy to Setup

```
$ pip install Flask
$ FLASK_APP=hello.py flask run
 * Running on http://localhost:5000/
```



Interfases de usuario

Pyforms

Create the Python file that will store your applications.

Example: SimpleExample.py

Import the library.

Import the pyforms library, the BaseWidget and the Controls classes that you will

```
import pyforms
from pyforms import BaseWidget
from pyforms.Controls import ControlText
from pyforms.Controls import ControlButton
```

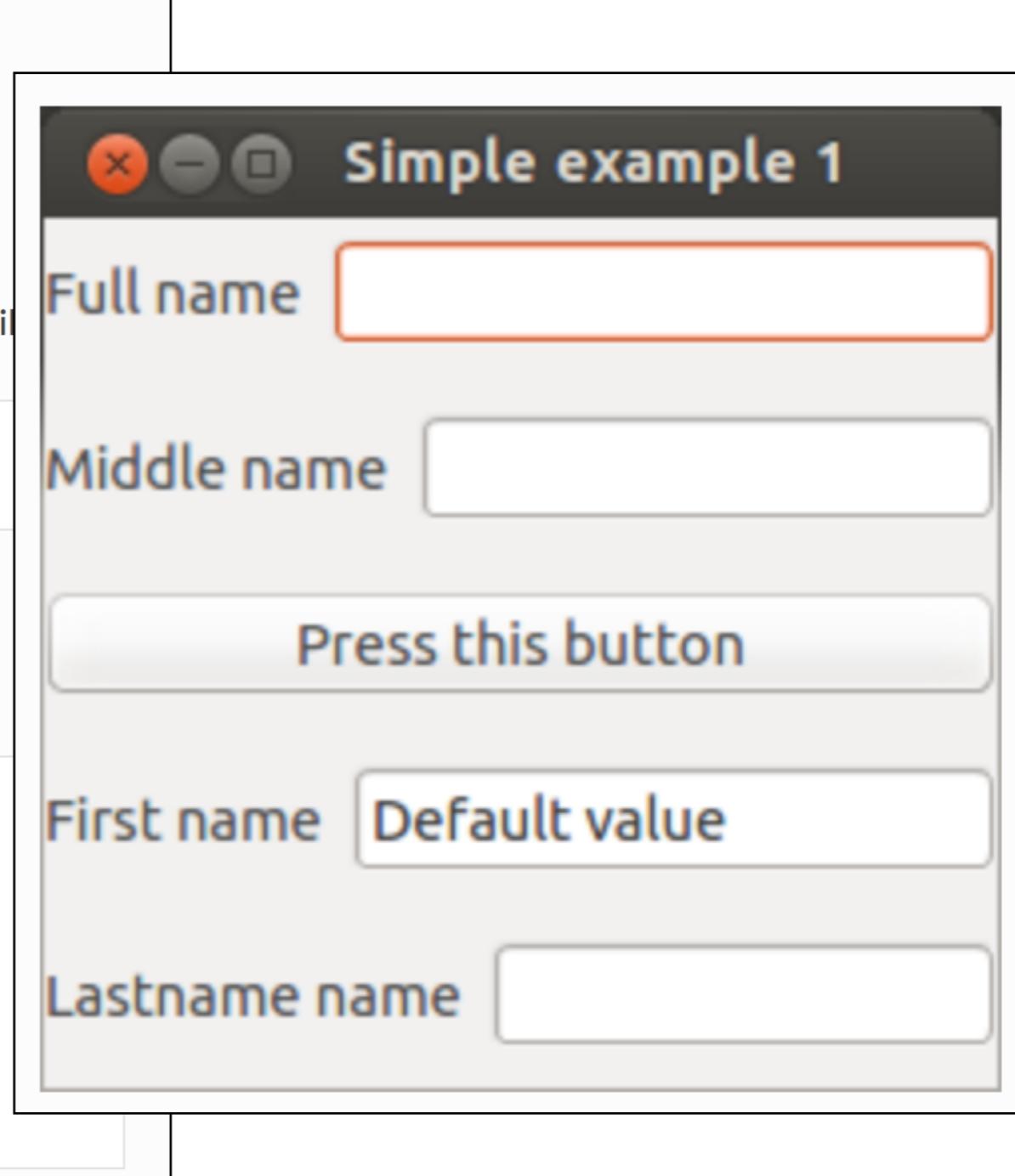
Create your application class.

This class should inherit from the class BaseWidget.

```
class SimpleExample1(BaseWidget):
    def __init__(self):
        super(SimpleExample1, self).__init__('Simple example 1')

        #Definition of the forms fields
        self._firstname = ControlText('First name', 'Default value')
        self._middlename = ControlText('Middle name')
        self._lastname = ControlText('Lastname name')
        self._fullname = ControlText('Full name')
        self._button = ControlButton('Press this button')

    #Execute the application
    if __name__ == "__main__":
        pyforms.start_app( SimpleExample1 )
```



II Congreso Colombiano de
**Investigación
Operativa**

ASOCIO 2017

Una introducción a Analytics con Anaconda Python

JUAN DAVID VELÁSQUEZ HENAO, MSc, PhD

Profesor Titular

Departamento de Ciencias de la Computación y la Decisión
Facultad de Minas
Universidad Nacional de Colombia, Sede Medellín

 jdvelasq@unal.edu.co

 @jdvelasquezh

 <https://github.com/jdvelasq>

 <https://goo.gl/prkjAq>

 <https://goo.gl/vXH8jy>