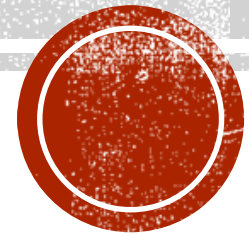


PYTHON

Christian Camilo Urcuqui López, MSc



PRESENTACIÓN

Christian Camilo Urcuqui López

Ing. Sistemas, Magister en Informática y Telecomunicaciones

Big Data Professional

Big Data Scientist

Deep Learning Specialization

Grupo de investigación i2t

Líder de investigación y desarrollo

Ciberseguridad y ciencia de datos aplicada

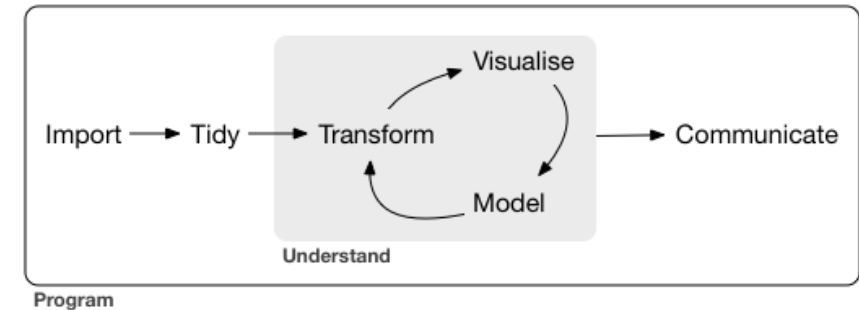
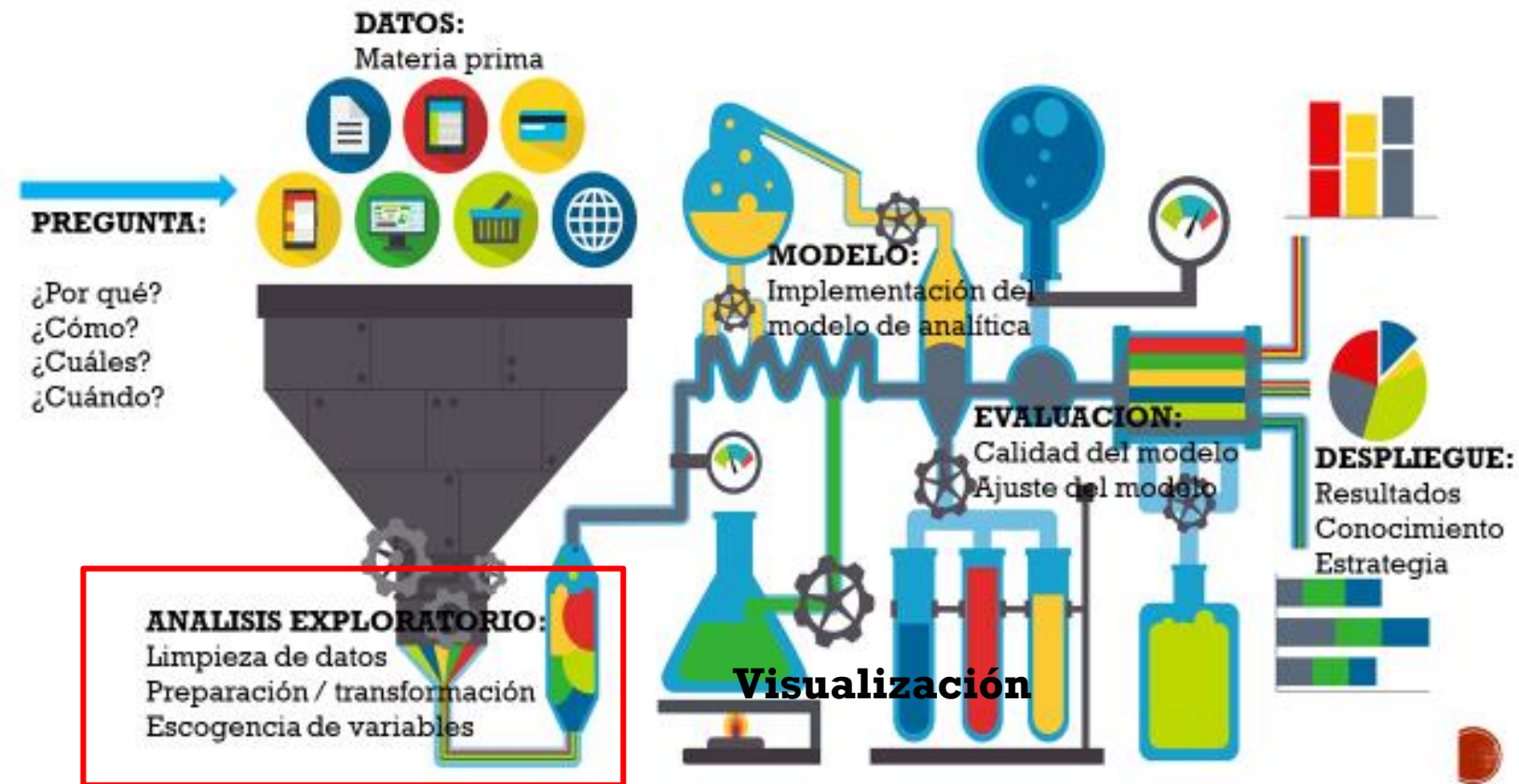
ccurcuqui@icesi.edu.co

COMPETENCIAS

- Utilizar las librerías de Python para proyectos de analítica de datos.
- Numpy
- Pandas
- Matplotlib



CICLO DE VIDA



Marco de trabajo típico de un proyecto de ciencia de datos.
R for Data Science

explainability, fairness,
bias, ethics

AIF360

deon

Aequitas

Skater

visualization

Seaborn

Altair

Bokeh

Shapely

Rasterio

Pydot

Matplotlib

plotnine

Plotly

Cartopy

Geopandas

analysis, modeling

Airflow

Rasa

AllenNLP

scikit-learn

Keras

PyTorch

StatsModels

Theano

TensorFlow

PyMC3

SciPy

Gensim

NetworkX

data representation

Pandas

NumPy

datasketch

Modin

spaCy

NLTK

RDFlib

data access

SQLAlchemy

Pillow

BeautifulSoup

network resources

PyArrow

Scrapy

Requests

Flasgger

Istio

application frameworks

JupyterLab

Dask

PyWren

Ray

Flask

PySpark

Project Jupyter

Jupyter Enterprise
Gateway

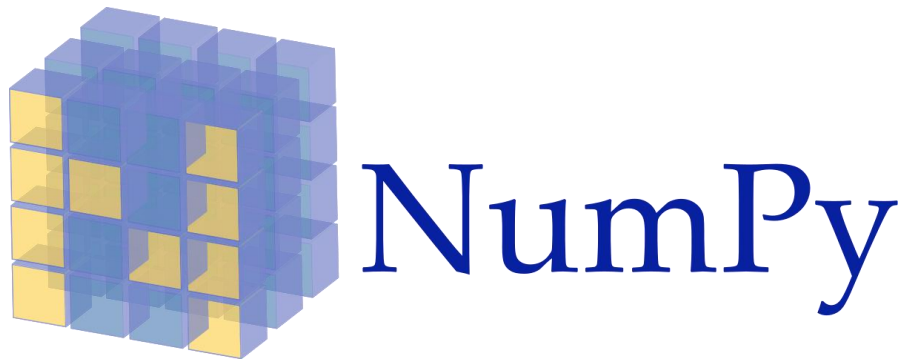
Gunicorn

package management

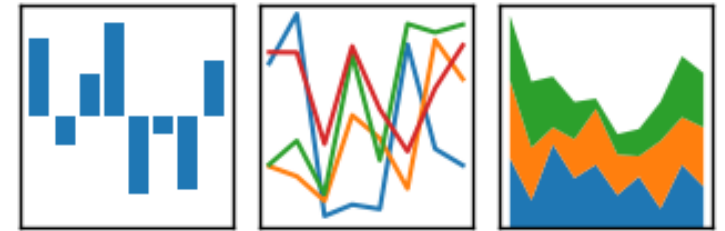
Pip

Conda

PAQUETES A REVISAR



pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$


PAQUETES A REVISAR



SciPy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:



NumPy
Base N-dimensional
array package



SciPy library
Fundamental library for
scientific computing



Matplotlib
Comprehensive 2D
Plotting



IPython
Enhanced Interactive
Console



Sympy
Symbolic mathematics

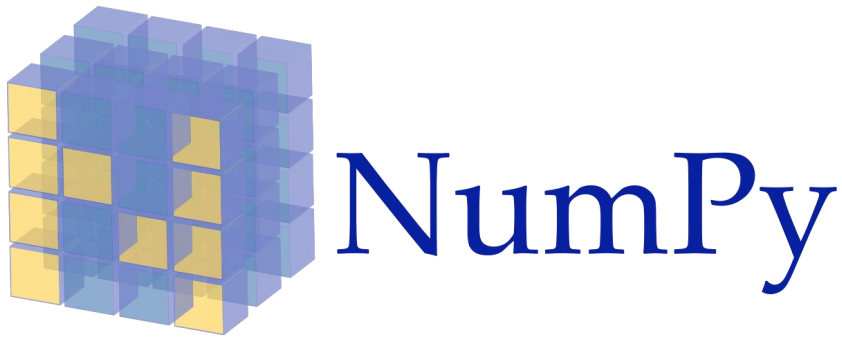


pandas
Data structures &
analysis



Large parts of the SciPy ecosystem (including all six projects above) are fiscally sponsored by
NumFOCUS.

<https://www.scipy.org/>



Es una herramienta útil para tareas numéricas, proporciona los mecanismos de almacenamiento y operaciones de datos a medida que las matrices crecen en tamaño. Es uno de los paquetes informáticos más importantes de Python (muchos paquetes científicos lo utilizan).

Numpy tiene algunas herramientas útiles, algunas de ellas lo son:

- `ndarray`, es una eficiente matriz multidimensional que proporciona operaciones aritméticas rápidas
- Funciones matemáticas para operaciones rápidas en conjuntos completos de datos sin tener que escribir bucles
- Herramientas para leer/escribir los datos de la matriz en el disco y trabajar con archivos mapeados en memoria.
- Álgebra lineal, generación de números aleatorios y entre otras funciones.
- A C API para conectar NumPy con bibliotecas escritas en C, C++ o FORTRAN.

IMPORTANDO NUMPY

Es común que algunos desarrolladores por estándar usen el seudónimo *np* al momento de importar el paquete Numpy.

```
import numpy as np  
np.__version__
```

PODER COMPUTACIONAL

```
my_arr = np.arange(1000000)
```

```
my_list = list(range(1000000))
```

```
print(type(my_arr))
```

```
print(type(my_list))
```

PODER COMPUTACIONAL

```
my_arr = np.arange(1000000)
```

```
my_list = list(range(1000000))
```

```
print(type(my_arr))
```

```
print(type(my_list))
```

PODER COMPUTACIONAL

- Escriba en una celda la siguiente línea de código y ejecútela

```
%time for _ in range(10): my_arr2 = my_arr * 2
```

- Escriba en una celda la siguiente línea de código y ejecútela

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

- Observe las diferencias en el tiempo...

EL ARREGLO DE NUMPY - NDARRAY

La función array es una de las formas de crear un array tipo NumPy.

- Ejecute las siguientes líneas de código y escriba un comentario con los resultados

```
alist = [1, 2, 3]
#
print(type(alist))
#
arr = np.array(alist)
#
print(type(arr))
#
print(arr.dtype)
#
arr
```

OBSERVACIÓN

Los datos de la lista deben ser del mismo tipo



- Ejecute las siguientes líneas de código y escriba un comentario con los resultados

```
data1 = [6, 7.5, 8, 0, 1]

arr1 = np.array(data1)

print(arr1.dtype)

print(arr1)

arr1 = np.array(["1", 3.5, 5])

print(arr1.dtype)

print(arr1)
```

ALGUNAS FUNCIONES DE UTILIDAD

- `zeros()`: Return a new array of given shape and type, filled with zeros.
- `ones()`: Return a new array of given shape and type, filled with ones.
- `arange()`: Return evenly spaced values within a given interval.
- `linspace()`: Return evenly spaced numbers over a specified interval.
- `random.randn`: Return a sample (or samples) from the "standard normal" distribution.

CREANDO UN ARRAY

Ejercicios

1. Cree un array de cinco ceros.
2. Cree un array de 100 números desde el 0.
3. Cree un array desde 0 hasta 1000 con saltos de 10.

UTILICE LA AYUDA

OBSERVACIÓN

Por defecto los intervalos de linspace son incluyentes pero se puede cambiar con el parámetro `endpoint` a `False` con el fin de no incluir el último extremo



ARRAY CON DIMENSIONES Y TIPOS

Un objeto ndarray de 5x5

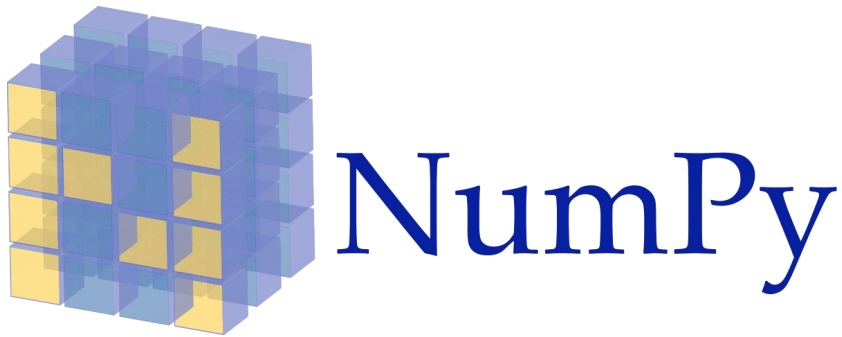
```
np.zeros((5,5))
```

Podemos cambiar el tipo de los elementos del ndarray

```
np.zeros((5, 5, 5)) + 1
```

```
np.zeros((5, 5, 5), dtype=np.int64) + 1
```

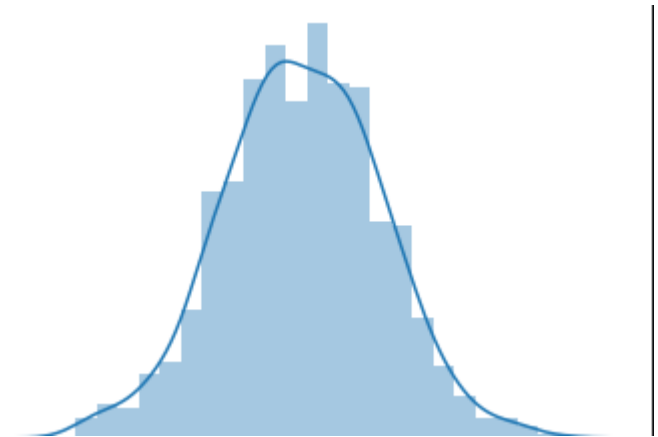
```
np.zeros((5, 5, 5)).astype(int) + 1
```

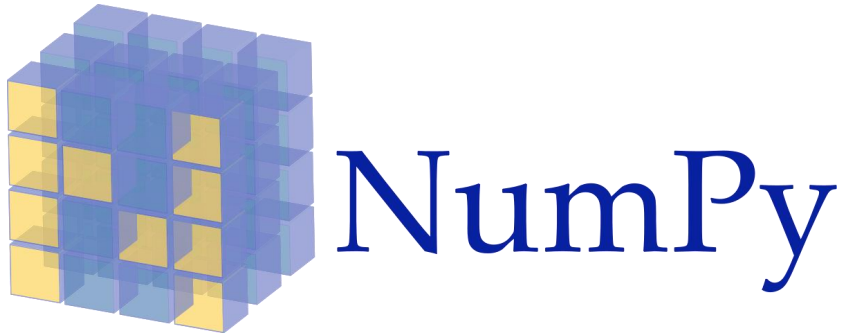


En muchas ocasiones será necesario utilizar array numéricos generados aleatoriamente, veamos algunas alternativas que NumPy nos ofrece

```
# creemos un array de 1x1000 utilizando randn  
data = np.random.randn(1000)
```

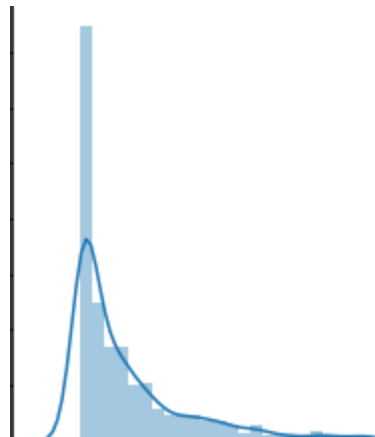
```
data[:10] # slicing
```

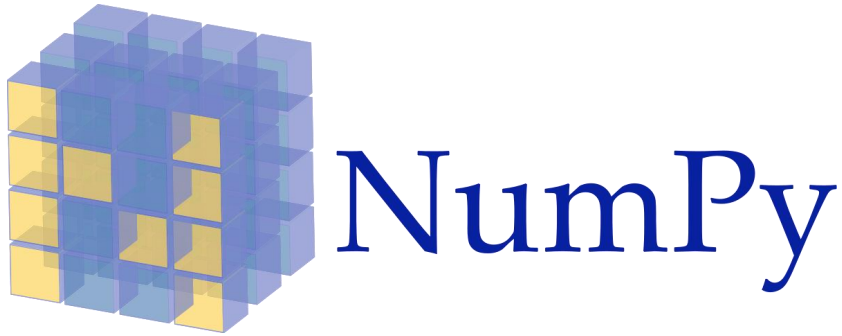




En muchas ocasiones será necesario utilizar array numéricos generados aleatoriamente, veamos algunas alternativas que NumPy nos ofrece

```
np.random.chisquare(1,1000)
```





```
# ndarray de 2x4
data2 = [[1,2,3,4],[5,6,7,8]]

arr2 = np.array(data2)
print("tipo del ndarray: %s" %arr2.dtype)
print("shape: " + str(arr2.shape))
print("dimension: {} ".format(str(arr2.ndim)))
```

Python For Data Science Cheat Sheet

Python Basics

Learn More Python for Data Science Interactively at www.datacamp.com



Variables and Data Types

Variable Assignment

```
>>> x=5
>>> x
5
```

Calculations With Variables

>>> x+2	Sum of two variables
7	
>>> x-2	Subtraction of two variables
3	
>>> x*2	Multiplication of two variables
10	
>>> x**2	Exponentiation of a variable
25	
>>> x%2	Remainder of a variable
1	
>>> x/float(2)	Division of a variable
2.5	

Types and Type Conversion

str()	'5', '3.45', 'True'	Variables to strings
int()	5, 3, 1	Variables to integers
float()	5.0, 1.0	Variables to floats
bool()	True, True, True	Variables to booleans

Asking For Help

```
>>> help(str)
```

Strings

```
>>> my_string = 'thisStringIsAwesome'
>>> my_string
'thisStringIsAwesome'
```

String Operations

```
>>> my_string * 2
'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
'thisStringIsAwesomeInnit'
>>> 'm' in my_string
True
```

Lists

Also see NumPy Arrays

```
>>> a = 'is'
>>> b = 'nice'
>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Selecting List Elements

Index starts at 0

Subset	
>>> my_list[1]	Select item at index 1
>>> my_list[-3]	Select 3rd last item
Slice	
>>> my_list[1:3]	Select items at index 1 and 2
>>> my_list[1:]	Select items after index 0
>>> my_list[:3]	Select items before index 3
>>> my_list[:]	Copy my_list
Subset Lists of Lists	
>>> my_list2[1][0]	my_list[list][itemOfList]
>>> my_list2[1][:2]	

List Operations

```
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list2 > 4
True
```

List Methods

>>> my_list.index(a)	Get the index of an item
>>> my_list.count(a)	Count an item
>>> my_list.append('!')	Append an item at a time
>>> my_list.remove('!')	Remove an item
>>> del(my_list[0:1])	Remove an item
>>> my_list.reverse()	Reverse the list
>>> my_list.extend('!')	Append an item
>>> my_list.pop(-1)	Remove an item
>>> my_list.insert(0, '!')	Insert an item
>>> my_list.sort()	Sort the list

String Operations

Index starts at 0

```
>>> my_string[3]
>>> my_string[4:9]
```

String Methods

>>> my_string.upper()	String to uppercase
>>> my_string.lower()	String to lowercase
>>> my_string.count('w')	Count String elements
>>> my_string.replace('e', 'i')	Replace String elements
>>> my_string.strip()	Strip whitespace from ends

Libraries

Import libraries

```
>>> import numpy
>>> import numpy as np
Selective import
>>> from math import pi
```



Data analysis



Machine learning



Scientific computing



2D plotting

Install Python



Leading open data science platform powered by Python



Free IDE that is included with Anaconda



Create and share documents with live code, visualizations, text, ...

NumPy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]
>>> my_array = np.array(my_list)
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

Selecting NumPy Array Elements

Index starts at 0

Subset	
>>> my_array[1]	Select item at index 1
2	
Slice	
>>> my_array[0:2]	Select items at index 0 and 1
array([1, 2])	
Subset 2D NumPy arrays	
>>> my_2darray[:,0]	my_2darray[rows, columns]
array([1, 4])	

NumPy Array Operations

```
>>> my_array > 3
array([False, False, False,  True], dtype=bool)
>>> my_array * 2
array([2, 4, 6, 8])
>>> my_array + np.array([5, 6, 7, 8])
array([6, 8, 10, 12])
```

NumPy Array Functions

>>> my_array.shape	Get the dimensions of the array
>>> np.append(other_array)	Append items to an array
>>> np.insert(my_array, 1, 5)	Insert items in an array
>>> np.delete(my_array, [1])	Delete items in an array
>>> np.mean(my_array)	Mean of the array
>>> np.median(my_array)	Median of the array
>>> my_array.corrcoef()	Correlation coefficient
>>> np.std(my_array)	Standard deviation

DataCamp

Learn Python for Data Science Interactively



BIBLIOGRAFÍA

- Lutz, M. (2013). *Learning Python: Powerful Object-Oriented Programming.* " O'Reilly Media, Inc."