
PUCP

Curso: Python aplicado a Data Science

Sesión 2: Preprocesamiento de datos con Numpy y Pandas

Break!! Regresamos 10:30 AM

Contenido del curso

Sesión 1: Introducción a Ciencia de Datos y Python

Sesión 2: Preprocesamiento de datos con **Numpy y Pandas**

Sesión 3: Visualización de datos con **Seaborn, Matplotlib y Plotly**

Sesión 4: Visualización y Transformación de datos (PCA)

Sesión 5: Análisis de Asociaciones: **Market Basket Analysis**

Sesión 6: Análisis de Agrupamientos: **K-Means, DBSCAN**

Sesión 7: Analítica predictiva: Modelos de **regresión**

Sesión 8: Analítica predictiva: Modelos de **clasificación**

Contenido

- Librerías para Ciencia de Datos
- **Numpy**: La librería por excelencia para el manejo de datos numéricos
- Vectores y matrices
- Laboratorio guiado: Numpy

- **Pandas** para el manejo de datos tabulares
- Funciones más usadas
- Laboratorio guiado: Pandas

- Explicación de aplicaciones prácticas: Reto Semanal

¿Qué es una librería?

Las **funciones** y **métodos** son muy poderosos..

… pero

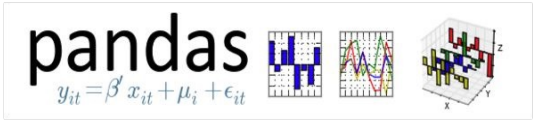
- Mucho código
- Desorden
- Código que no se usa
- Problemas de mantenimiento

Las **librerías o paquetes:**

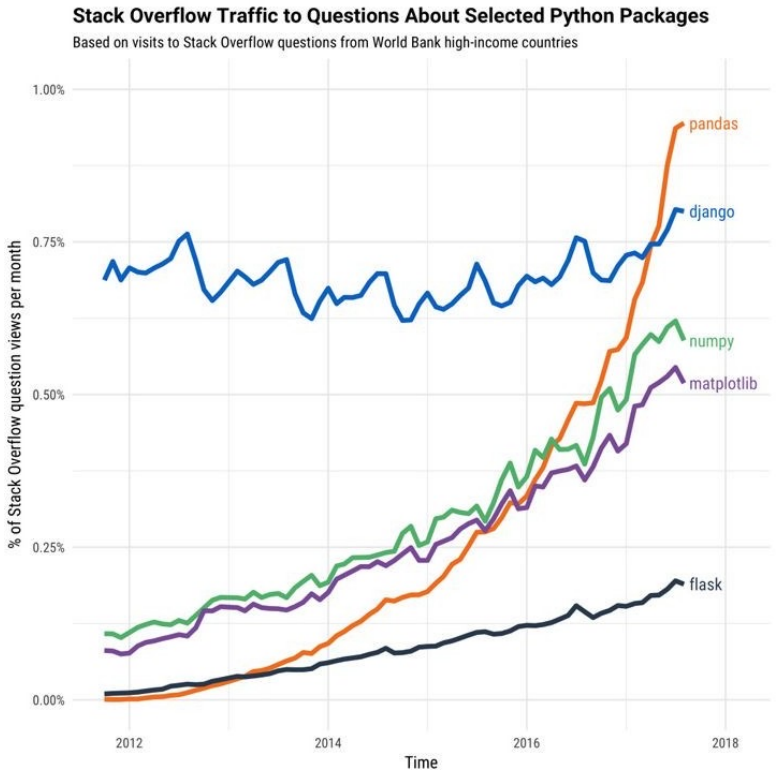
Directorio de Scripts Python

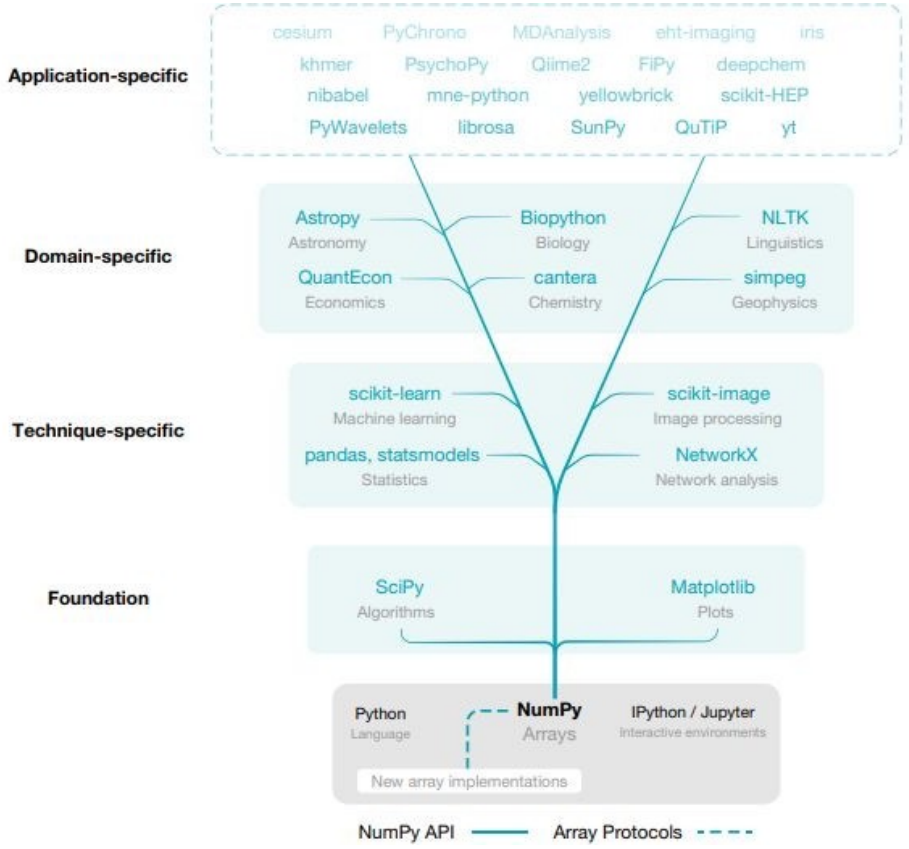
Funciones y métodos específicos

Miles de paquetes disponibles



¿Por qué estudiamos Numpy?





https://doi.org/10.1038/s41586-020-2649-2

Numerical Python

Si bien las listas en Python permiten almacenar secuencias ordenadas de números, Numpy provee objetos más apropiados. Para poder usarlo tenemos que importar la librería


```
import numpy
```

o más convenientemente

```
import numpy as np
```

usaremos esta versión en el curso

Algunas ventajas de Numpy

NumPy es muy útil para realizar cálculos lógicos y matemáticos en arreglos y matrices.

Realiza estas operaciones **mucho más rápido y eficientemente que las listas** de Python.

NumPy utiliza menos memoria y espacio de almacenamiento, lo cual es la principal ventaja. Además, ofrece un mejor rendimiento en cuanto a la velocidad de ejecución.

Numpy es de código abierto y se puede utilizar completamente de forma gratuita.

Vectores en Numpy

- El objeto base para representar vectores en Numpy es el Array (arreglo)

```
x = np.array([-1.1, 0, 3.6, -7.2])
```

```
numpy.ndarray
```

El tipo de dato es ndarray que significa arreglo n-dimensional. En 1 dimensión es vector y en 2 dimensiones será una matriz

Más allá de los vectores

Scalar

(11)

Shape 0

Vector

[1,2,3]

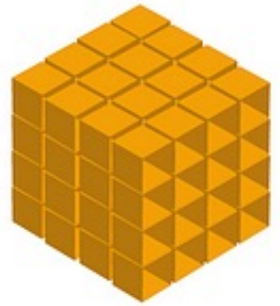
Shape 1

Matrix

[[1,2,3],
[4,5,6],
[7,8,9]]

Shape 2

Tensor



Shape n

https://www.freecodecamp.org/news/tensorflow-basics/

Pandas para el manejo de datos tabulares

Es uno de los paquetes más potentes que tiene Python para el análisis de dataframes.

Panel **Da**ta(**s**)

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

In [ ]:
```

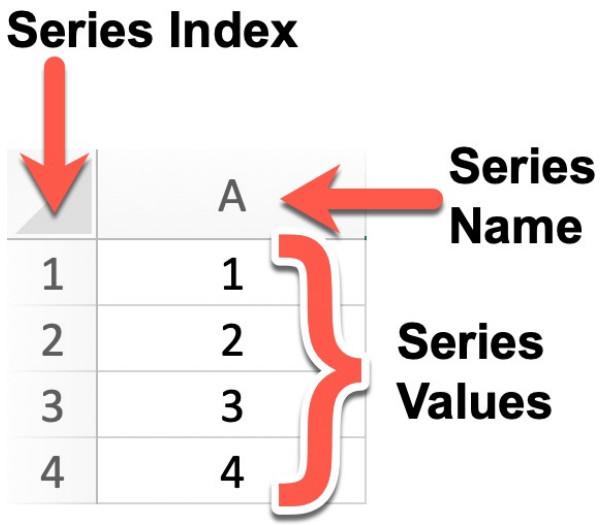
Cargar data tabular desde diferentes fuentes (base de datos, csv, json, etc).

Hacer búsquedas por **filas** o **columnas**

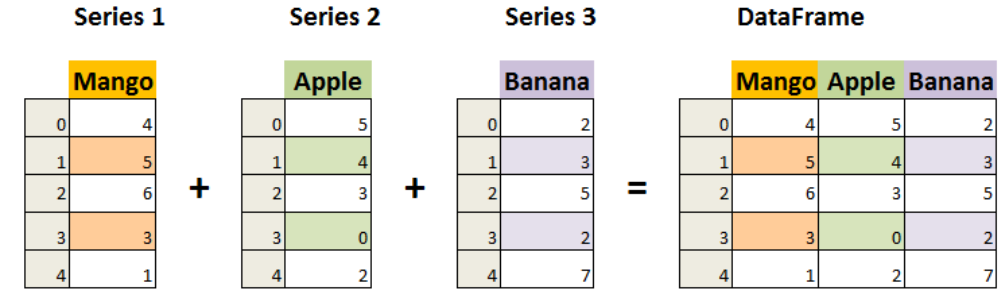
Calcular **estadísticas agregadas**

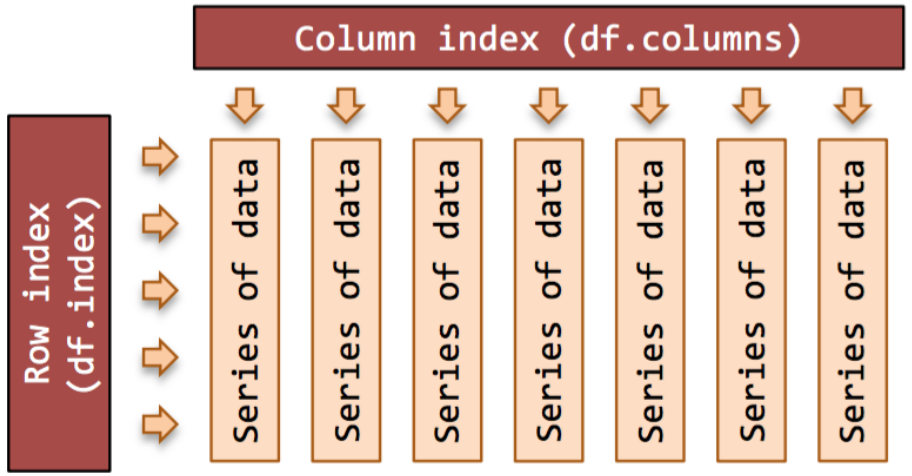
Combinar (unir) datos de distinto origen

Series y Dataframes en Pandas



Una serie es un **ndarray de 1-dimension** con etiquetas (index)





¿Y si concatenamos varias Series?

Trabajando con Dataframes

```
print(df.head())
```

```
                suspect        location          item  price
0         Kirstine Smith  Petroleum Plaza           gas  24.95
1      Fred Frequentist      Burger Mart         fries   1.95
2          Gertrude Cox      Burger Mart         fries   1.95
3   Ronald Aylmer Fisher    Clothing Club         shirt  14.25
4         Kirstine Smith    Clothing Club         dress  20.15
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 3 columns):
letter_index    26 non-null int64
letter          26 non-null object
frequency       26 non-null float64
dtypes: float64(1), int64(1), object(1)
memory usage: 704.0+ bytes
```

Funciones más usadas en Pandas

Importar Data
pd.read_csv
pd.read_excel
pd.read_json
…

Inspeccionar Data
df.head()
df.tail()
df.shape
df.info()
df.describe()
s.value_counts()
s.unique()
s.unique()
…

Seleccionar Data
df[‘col’]
df[[‘col1’,’col2’]]
s.iloc[0]

Limpieza de datos
df.isna().sum()
df.dropna()
df.dropna(axis=1)
df.fillna(x)
s.replace(1,’one’)
s.astype(‘int’)
df.set_index(‘col1’)
…

Filtrado, Ordenado y Agrupación
df[df[‘col1’]>2]
df.sort_values(‘col1’)
df.groupby(‘col1’)
df.groupby(‘col1’).mean()[‘col2’]
df.pivot_table(index=‘col1’,values…)

--- etc.

Referencias

Numpy.org
https://numpy.org/

Pandas Documentation
https://pandas.pydata.org/

Python for Data Analysis
https://wesmckinney.com/book/

Kaggle – Numpy
https://www.kaggle.com/utsav15/100-numpy-exercises

Matrix Algebra for Begineers
https://www.math.hkust.edu.hk/~machas/matrix-algebra-for-engineers.pdf