

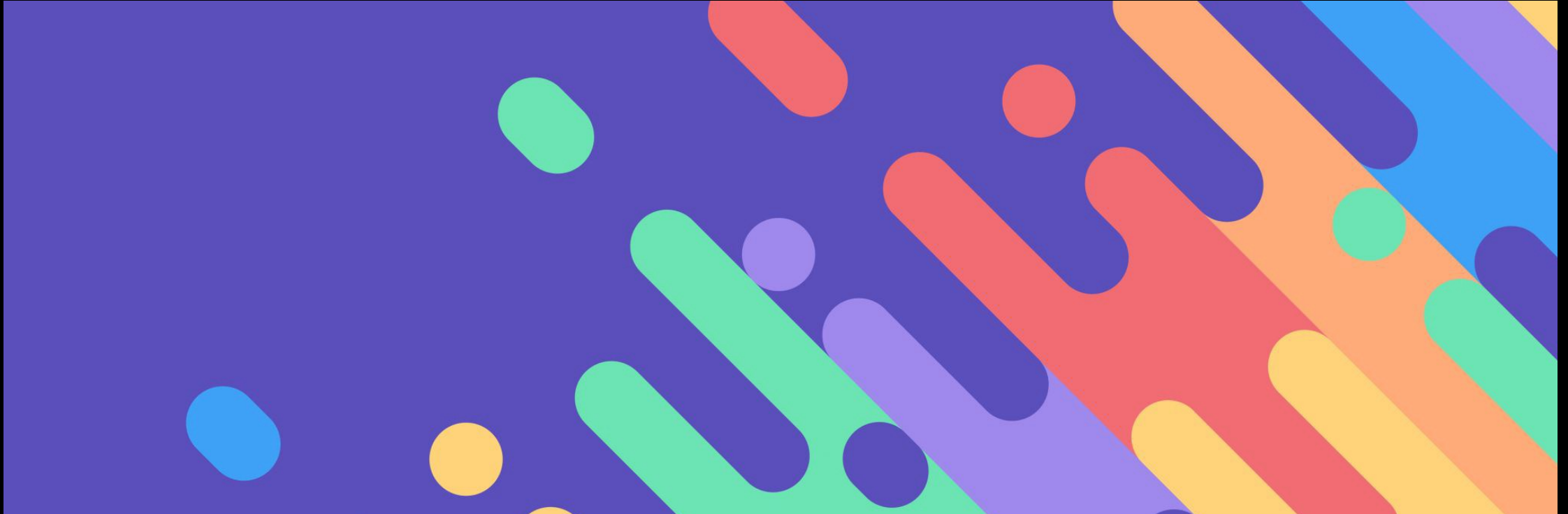
# MANEJO DE DATOS CON NUMPY Y PANDAS



Aprendizaje  
Automático

CEIoT - FIUBA

Dr. Ing. Facundo Adrián  
Lucianna



---

LO QUE VIMOS LA CLASE ANTERIOR...

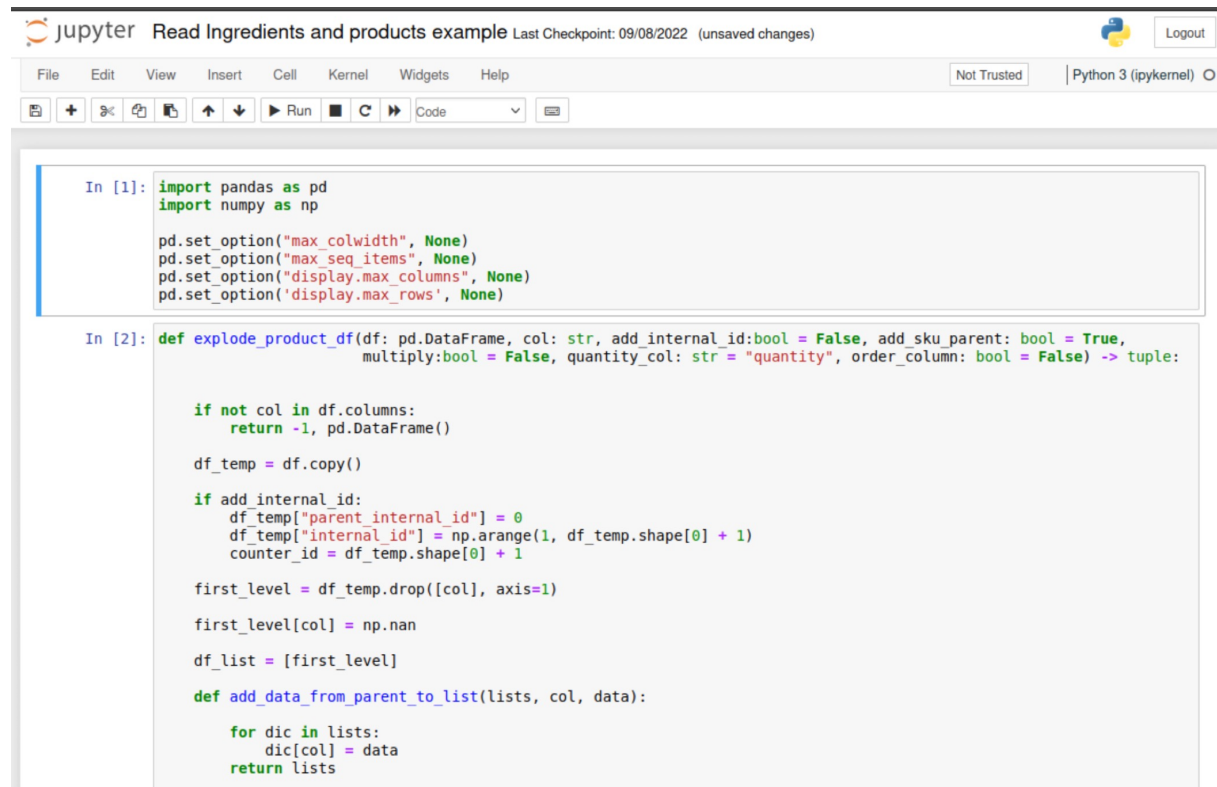
---

# PYTHON

- Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código.
- Python es un lenguaje de programación **multiparadigma**. Permite varios estilos: **programación orientada a objetos**, programación imperativa y programación funcional.
- OBS: En el fondo, Python es un lenguaje orientado a objetos, todo, absolutamente todo es un objeto.
- Python usa tipado dinámico y conteo de referencias para la gestión de memoria.
- Python reemplazó en gran medida a LISP en IA, principalmente por ser multiparadigma.

# PYTHON

## Jupyter Notebook



```
jupyter Read Ingredients and products example Last Checkpoint: 09/08/2022 (unsaved changes) Python 3 (ipykernel) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

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

pd.set_option("max_colwidth", None)
pd.set_option("max_seq_items", None)
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)

In [2]: def explode_product_df(df: pd.DataFrame, col: str, add_internal_id: bool = False, add_sku_parent: bool = True,
multiply: bool = False, quantity_col: str = "quantity", order_column: bool = False) -> tuple:

    if not col in df.columns:
        return -1, pd.DataFrame()

    df_temp = df.copy()

    if add_internal_id:
        df_temp["parent_internal_id"] = 0
        df_temp["internal_id"] = np.arange(1, df_temp.shape[0] + 1)
        counter_id = df_temp.shape[0] + 1

    first_level = df_temp.drop([col], axis=1)

    first_level[col] = np.nan

    df_list = [first_level]

    def add_data_from_parent_to_list(lists, col, data):
        for dic in lists:
            dic[col] = data
        return lists
```





---

# NUMPY

---

# NUMPY

Cuando trabajamos en aplicaciones de Aprendizaje Automático, trabajamos con muchos tipos de datos y en grandes cantidades.

Los conjuntos de datos pueden provenir de una amplia gama de fuentes y formatos, como colecciones de documentos, imágenes, clips de sonido, mediciones numéricas, entre otros. A pesar de esta aparente heterogeneidad, nos ayudará pensar en todos los datos fundamentalmente como arrays de números.

El almacenamiento y manipulación eficientes de arrays numéricos son absolutamente fundamentales en el proceso de hacer Aprendizaje Automático.

Esto lo podemos hacer usando NumPy.

NumPy nos ofrece los arrays que puede almacenar y operar datos de manera eficiente.

- Una forma que logra ser eficiente es que los arrays sus elementos tienen que ser del mismo tipo.
- Los arrays son **mutables** por defecto, pero se puede cambiar este comportamiento.

---

# NUMPY

## Tipos de datos

Debido a que NumPy está construido en C, los tipos que usan son familiares para los usuarios de C, Fortran y otros lenguajes relacionados:


bool_	int_	intc	intp
int8	int16	int32	int64
uint8	uint16	uint32	uint64
float_	float16	float32	float64
complex_	complex64	complex128	

---

# NUMPY

## Atributos

Los arrays en NumPy, *dado que todo es un objeto*, tiene atributos que podemos aprovechar como:



```
1 x.ndim # Dimensiones del array
2 x.shape # Tamaño de cada dimensión
3 x.size # Cantidad de elementos en el array
4 x.dtype # Tipo de dato que almacena el array
5
6 x.itemsize # Tamaño en bytes de un elemento
7 x.nbytes # Tamaño en bytes del array
```

[codetolimej.com](https://codetolimej.com)

Si desean profundizar en NumPy, seguir en [capítulo 2](#) de Python Data Science.





---

# ESTRUCTURA DE DATOS DE PANDAS

---

# ESTRUCTURAS DE DATOS DE PANDAS

A un nivel muy básico, los objetos de Pandas son una versión mejorada de las estructuras de NumPy en los cuales las filas y columnas se identifican con etiquetas.

Tiene dos estructuras fundamentales:

- Series: Una serie de Pandas es un array uni-dimensional de datos indexados.
- DataFrame: Es análogo a un array de dos dimensiones con índices de filas y nombres de columnas. Un DataFrame se forma con una Serie para cada columna.



---

# OPERANDO CON PANDAS

---

# OPERANDO CON PANDAS

Una de las partes principales de NumPy es la capacidad de realizar operaciones rápidas elemento por elemento, tanto con aritmética básica como con operaciones más sofisticadas

Pandas hereda gran parte de esta funcionalidad de NumPy.



---

# COMBINANDO DATOS

---

# COMBINANDO DATOS

En el proceso de armar nuestros dataset para entrenar modelos de **Aprendizaje Automático**, muchas veces debemos combinar datos de diferentes fuentes.

Pandas nos da varias herramientas que nos permiten hacer combinación de ellos.



---

# AGRUPANDO DATOS

---

# AGRUPANDO DATOS

Una pieza esencial del análisis de datos es el agregado eficiente: calcular operaciones como **sum**, **mean**, **median**, **min** y **max**.

Pandas nos ofrece herramientas para hacer estas agrupaciones, desde operaciones simples, hasta operaciones más sofisticadas.

Las agregaciones incorporadas de Pandas son:

Agregación	
count()	Número de ítems
first(), last()	Primer y último ítem
mean(), median()	Media y mediana
min(), max()	Mínimo y máximo
std() var()	Desvío estándar y varianza
prod()	Producto de todos los ítems
sum()	Suma de todos los ítems
mad()	Desviación media





---

# GRAFICANDO

---

# GRAFICANDO

En Python tenemos múltiples herramientas para graficar, podemos mencionar como herramientas a:

- Matplotlib
- Pandas
- Seaborn

---

# MAS MATERIAL

Parte del contenido de esta clase la obtuvimos de los capítulos 3 y 4 de [Python Data Science Handbook](#).

Si desean profundizar pueden visitar al libro.