

# SEMINARIO

## Análisis de Serie de Tiempo con Pandas

Grupo: 10

Chelin, Martin  
Velazquez, Sabrina

# Introducción

- MANEJAR Y COMPRENDER GRANDES VOLÚMENES DE DATOS
- HERRAMIENTA PARA ANÁLISIS Y MANIPULACIÓN DE DATOS
- FÁCIL DE USAR PARA TRANSFORMAR, LIMPIAR Y VISUALIZAR DATOS
- TAREAS COMPLEJAS, POCAS LÍNEAS DE CÓDIGO
- TECNICAS DE MANIPULACION DE DATOS
- COMBINACIÓN CON MATPLOTLIB

# Que es Pandas?

Es una poderosa biblioteca de código abierto para Python utilizada principalmente para análisis de datos y manipulación de datos estructurados.

- CIENCIAS DE DATOS
- ANÁLISIS FINANCIERO
- INGENIERÍA
- INVESTIGACIÓN ACADÉMICA
- ALGÚN DOMINIO QUE NECESITE TRABAJO CON DATOS ESTRUCTURADOS

# Características y funcionalidades

- DATAFRAMES
- SERIES
- LECTURA Y ESCRITURA DE DATOS
- SELECCIÓN Y FILTRADO DE DATOS
- AGRUPACIÓN DE DATOS
- MANIPULACIÓN DE DATOS
- LIMPIEZA DE DATOS
- OPERACIONES ARITMÉTICAS Y ESTADÍSTICAS

# Ventajas en el Análisis de Datos

- FACILIDAD DE USO
- MANEJO DE DATOS ESTRUCTURADOS
- EFICIENCIA EN EL RENDIMIENTO
- VISUALIZACIÓN DE DATOS
- INTEGRACIÓN CON OTRAS BIBLIOTECAS DE PYTHON
- COMUNIDAD ACTIVA

# Estructuras de Datos - Series

Son estructuras unidimensionales conteniendo un array de datos y un array de etiquetas asociados a los datos llamados índice.

- DATOS HOMOGENEOS O HETEROGENEOS
- ETIQUETAS (INDICE)
- OPERACIONES VECTORIZADAS
- ETIQUETADO Y ALINEACIÓN DE DATOS

Index	Data
0	Mark
1	Justin
2	John
3	Vicky

# Estructuras de Datos - DataFrame

Es una estructura bidimensional compuesta por filas y columnas, es una agrupación de series unidas bajo los mismos índices dando como resultado estructuras similares a una tabla.

- TABLA BIDIMENSIONAL
- DATOS HETEROGÉNEOS
- ETIQUETAS DE ÍNDICE Y COLUMNAS
- FLEXIBILIDAD
- OPERACIONES DE SQL

Series			Series			DataFrame			
	apples			oranges			apples	oranges	
0	3	+	0	0	=	0	3	0	
1	2		1	3		1	2	3	
2	0		2	7		2	0	7	
3	1		3	2		3	1	2	

# Lectura y Estructura de Datos

- Lectura de datos:
  - Leer datos desde un archivo CSV: `pd.read_csv()`
  - Leer datos desde un archivo Excel: `pd.read_excel()`
  - Leer datos desde un archivo JSON: `pd.read_json()`
  - Leer datos desde una base de datos: `pd.read_sql()`
- Escritura de datos:
  - Guardar los datos en un archivo CSV: `pd.to_csv()`
  - Guardar los datos en un archivo Excel: `pd.to_excel()`
  - Guardar los datos en un archivo JSON: `pd.to_json()`



# Manipulación y Limpieza de Datos

- FILTRADO Y SELECCIÓN DE DATOS
  - Consultar una columna mediante su nombre:  
`"Nombre_DataFrame["Nombre_Columna", "..."]"`
  - Seleccionar una fila mediante su etiqueta o índice:  
`"Nombre_DataFrame.loc["Nombre_Fila"]"`  
`"Nombre_DataFrame.iloc[índice]"`
  - Seleccionar una celda en particular:  
`"Nombre_DataFrame["Nombre_Fila", "Nombre_Columna"]"`
  - Selección condicionada:  
`"Nombre_DataFrame[Condición]"`

# Manipulación y Limpieza de Datos

- TRATAMIENTO DE VALORES FALTANTES
  - Comprobar registros nulos:  
`"Nombre_DataFrame.isnull()"`
  - Descartar filas o columnas con registros nulos:  
`"Nombre_DataFrame.dropna()"`  
`"Nombre_DataFrame.dropna(axis=1)"`
  - Rellenar los registros de las filas vacías con un valor:  
`"Nombre_DataFrame.fillna(value=0)"`

# Operaciones en DataFrames

- Agregación:

- Agrupar filas en función de una columna:

`“Nombre_DataFrame.groupby(“Nombre_Columna”)”`

- Calcular media y desviación estándar:

`“Nombre_DataFrame.groupby(“Nombre_Columna”).mean()”`

`“Nombre_DataFrame.groupby(“Nombre_Columna”).std()”`

- Contar registros de cada columna:

`“Nombre_DataFrame.groupby(“Nombre_Columna”).count()”`

# Operaciones en DataFrames

- Agregación:
  - Transpuesta:  
`"Nombre_DataFrame.transpose()"`
  - Reporte de analíticas descriptivas por empresa:  
`"Nombre_DataFrame.groupby("Nombre_Columna").describe()"`
  - Calcular mínimo y máximo:  
`"Nombre_DataFrame.loc[Nombre_DataFrame.groupby["Columna"].idxmin()]"`  
`"Nombre_DataFrame.loc[Nombre_DataFrame.groupby["Columna"].idxmax()]"`

# Operaciones en DataFrames

- Operación de Transformación y cálculos:
  - Primeras y últimas filas:  
`"Nombre_DataFrame.head()"`  
`"Nombre_DataFrame.tail()"`
  - Array de valores únicos de una columna:  
`"Nombre_DataFrame["Nombre_Columna"].unique()"`
  - Contador de valores únicos de una columna:  
`"Nombre_DataFrame["Nombre_Columna"].nunique()"`

# Operaciones en DataFrames

- Operación de Transformación y cálculos:
  - Dataframe con los valores únicos y su contador de una columna:  
`"Nombre_DataFrame["Nombre_Columna"].value_counts()"`
  - Aplicar una función definida:  
`"Nombre_DataFrame["Nombre_Columna"].apply(funcion)"`
  - Ordenar por columna (inplace=False por defecto):  
`"Nombre_DataFrame.sort_values(by="Columna",ascending=Boolean]"`

# Operaciones en DataFrames

- Operación de Transformación y cálculos:
  - Agregar una columna (inplace=False por defecto):  
“Nombre\_DataFrame[Nombre\_Columna\_Nueva] = Valor”
  - Borrar una fila o columna (inplace=False por defecto):  
“Nombre\_DataFrame.drop(“Nombre\_Columna, axis=1”)”

# Operaciones en DataFrames

- Combinaciones:
  - Concatenación (Junta filas, requiere que las dimensiones sean iguales):  
`"pd.concat([Nombre_DataFrame_1, Nombre_DataFrame_2, ....])"`
  - Fusión (Une tablas a partir de una columna común):  
`"pd.merge(Nombre_DataFrame_1, Nombre_DataFrame_2, on="Columna")"`
  - Unión (Une Columnas mediante los índices):  
`"Nombre_DataFrame_1.join(Nombre_DataFrame_2)"`



# Visualizaciones de Datos

Es una herramienta para comprender y comunicar patrones, tendencias y relaciones dentro de nuestros conjuntos de datos.

Biblioteca utilizada: Matplotlib

Gráficas realizadas:

Evolución Mensual de Pedidos por año

Cantidad de Pedidos según Prioridad

# Comparativa con otras alternativas

	SciPy	Dask
<b>Enfoque</b>	Computación científica y matemática	Computación paralela y distribuida
<b>Proporciona</b>	Funciones avanzadas para tareas como álgebra lineal, optimización, interpolación, entre otras.	Estructura de datos y API similares a Pandas y Numpy. Capaz de trabajar con datos más grandes.
<b>Características Principales</b>	<ul style="list-style-type: none"><li>• Módulos especializados</li><li>• Integración con Numpy</li></ul>	<ul style="list-style-type: none"><li>• DataFrames y Array distribuidos</li><li>• Escalabilidad</li><li>• Integración con Pandas y Numpy</li></ul>

# Conclusión

Pandas se ha revelado como una herramienta poderosa y versátil para el análisis de datos en Python. Su facilidad de uso, su capacidad para manejar datos estructurados y su integración con otras bibliotecas hacen de Pandas una opción inestimable para realizar un análisis de forma efectiva y eficiente en trabajar con datos en Python.

**FIN**