Modulo 1

# Introducción a la programación en Python

Centro de e-Learning
UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES


# Unidad 3: Visualización de datos

# Organización del primer módulo

- **Clase 1:** Introducción al curso + Que es Python + Elección del editor de código + Definición de programa e instrucciones + Tipos de datos básicos

- **Clase 2:** Listas + Diccionarios + Tuplas + Estructuras de control condicionales y cíclicas + Funciones + Librería: NumPy

- **Clase 3:** Dataframes + Librería: Pandas + Concatenación y Merge de dataframes + Filtros + Limpieza de datos + Tipos de datos: fechas

- **Clase 4:** Visualización de datos + Principios generales del diseño analítico + Librería: Matplotlib + Gráficos mas populares

# Visualizaciones efectivas

Ahora que ya tenemos conocimiento sobre como manipular y transformar nuestros datos, es momento de buscar la manera de **comunicar** los resultados.

Lo primero que debemos averiguar es que queremos **lograr** con la visualización y para dar respuesta a este interrogante, es bueno empezar por haciéndonos las siguientes preguntas:

- ✔ **¿Quién es mi publico?**
- ✔ **¿Qué preguntas tienen?**
- ✔ **¿Qué respuestas estoy encontrando para ellos?**
- ✔ **¿Qué estoy tratando de decir?**
- ✔ **¿Qué otras preguntas inspirara mi visualización o que conversaciones podría generar?**

# Visualizaciones efectivas - Storytelling

Cuando tengamos dichas respuestas y un objetivo bien claro de lo que queremos lograr, comienza el proceso de **storytelling**.

El **storytelling** es el arte de contar, desarrollar y adaptar historias utilizando elementos específicos para transmitir un mensaje de forma inolvidable al conectarse con el lector a nivel emocional. Es decir, no basta con simplemente crear los gráficos y mostrarlos.

Acá es donde entran en juego los **Principios Generales del Diseño Analítico** (o Principios generales de Tufte):

- **Principio 1: Muestra comparaciones, contrastes, diferencias.**
- **Principio 2: Muestra causalidad, mecanismo, explicación, estructura sistemática.**
- **Principio 3: Muestra datos multivariados, es decir, mas de una o dos variables.**
- **Principio 4: Integra palabras, números, imágenes y diagramas.**
- **Principio 5: Describe la totalidad de la evidencia. Muestra fuentes usadas y problemas relevantes.**
- **Principio 6: Las presentaciones analíticas, se sostienen de la calidad, relevancia e integridad de su contenido.**

# Visualizaciones en Python

**Python** tiene implementado una amplia gama de herramientas para realizar gráficos. Las dos principales son: **matplotlib** y **seaborn**.

Hoy veremos ejemplos utilizando la base de datos ***titanic*** (adjunta en los archivos de la unidad) y la librería **matplotlib**. Lo primero que haremos es cargar la base de datos, podemos hacerlo desde el archivo o directamente desde una url.

In [3]:
```
import pandas as pd
# Cargamos el conjunto de datos del Titanic desde un archivo CSV (o URL)
titanic_data_url = "https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv"
titanic_data = pd.read_csv(titanic_data_url)

# Mostramos las primeras filas del conjunto de datos para explorarlo
titanic_data.head()
```

Out[3]:

| | Survived | Pclass | Name | Sex | Age | Siblings/Spouses Aboard | Parents/Children Aboard | Fare |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | Mr. Owen Harris Braund | male | 22.0 | 1 | 0 | 7.2500 |
| **1** | 1 | 1 | Mrs. John Bradley (Florence Briggs Thayer) Cum... | female | 38.0 | 1 | 0 | 71.2833 |
| **2** | 1 | 3 | Miss. Laina Heikkinen | female | 26.0 | 0 | 0 | 7.9250 |
| **3** | 1 | 1 | Mrs. Jacques Heath (Lily May Peel) Futrelle | female | 35.0 | 1 | 0 | 53.1000 |
| **4** | 0 | 3 | Mr. William Henry Allen | male | 35.0 | 0 | 0 | 8.0500 |

# Gráfico de barras

Comencemos con un grafico de barras que muestre la cantidad de pasajeros que sobrevivieron y los que no. Este tipo de visualización nos brinda una visión clara de la proporción de sobrevivientes en el desafortunado suceso del Titanic.

Podemos observar rápidamente que la proporción de sobrevivientes es menor.

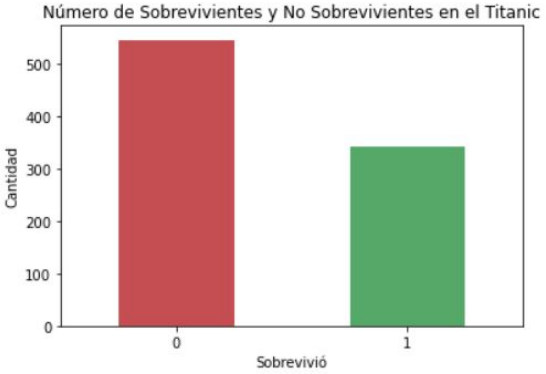
**¿Qué sucedería si decidimos invertir los colores?**

Daria lugar a interpretaciones erróneas si no se analiza detalladamente el grafico, donde uno podría pensar que la proporción de sobrevivientes es mayor, cuando no lo es.

```
In [8]: import matplotlib.pyplot as plt  #Cargamos la libreria

survived_count = titanic_data['Survived'].value_counts()
survived_count.plot(kind='bar', color=['#C44E52', '#55A868'])

plt.title('Número de Sobrevivientes y No Sobrevivientes en el Titanic')
plt.xlabel('Sobrevivió')
plt.ylabel('Cantidad')
plt.xticks(rotation=0)
plt.show()
```



# Gráfico de torta

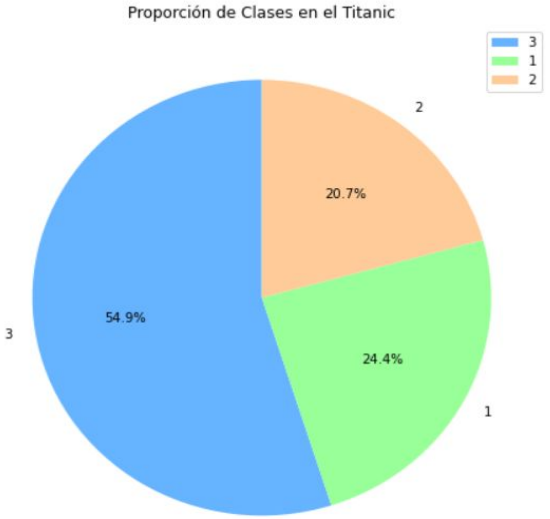
Exploremos la distribución de clases entre los pasajeros. Un **grafico de torta** nos permitirá visualizar fácilmente la proporción de pasajeros en cada clase.

A partir del grafico, podemos observar que más de la mitad de los pasajeros pertenecen a la clase numero 3.

Algo a destacar es que no se recomienda utilizar los gráficos de torta cuando se tienen más de 5 o 6 categorías, o donde unas pocas tienen una proporción muy chica. También es importante no utilizar gráficos 3D ya que se pierde un poco la percepción, dando lugar a interpretaciones erróneas.

```
In [11]: plt.figure(figsize=(8, 8))
         class_counts = titanic_data['Pclass'].value_counts()
         colors = ['#66b3ff', '#99ff99', '#ffcc99']

         plt.pie(class_counts, labels=class_counts.index, autopct='%1.1f%%',
                 startangle=90, colors=colors)
         plt.title('Proporción de Clases en el Titanic')
         plt.legend(labels=class_counts.index, loc="best")
         plt.show()
```



# Histograma

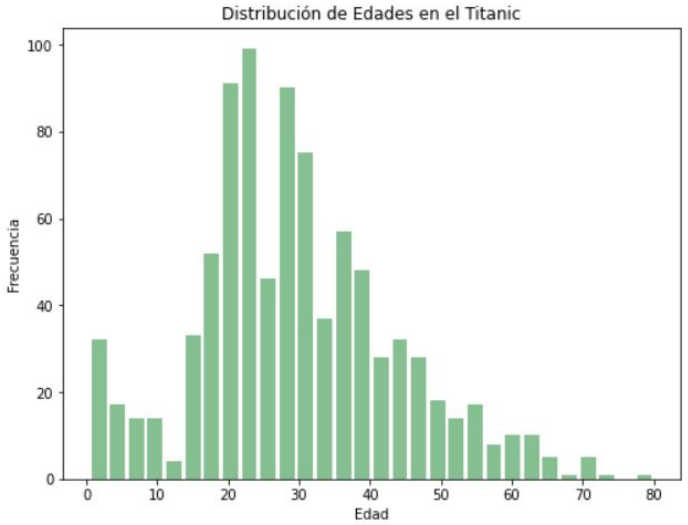
Un **histograma** se utiliza para visualizar la **frecuencia** de ocurrencia de diferentes intervalos o clases en un conjunto de datos continuo.

Podemos utilizar uno para ver como es la **distribución** de las edades de los pasajeros.

Sin entrar en detalles de cuestiones estadísticas, podemos evidenciar que la edad de los pasajeros sigue casi una especie de campana o distribución **normal**. (habría que hacer pruebas para comprobarlo).

Una buena alternativa podría ser combinar este grafico con otras variables, como el género o sobrevivientes.

```
In [2]: plt.figure(figsize=(8, 6))
        plt.hist(titanic_data['Age'].dropna(), bins=30,
                 color='#86bf91', rwidth=0.8)
        plt.title('Distribución de Edades en el Titanic')
        plt.xlabel('Edad')
        plt.ylabel('Frecuencia')
        plt.show()
```



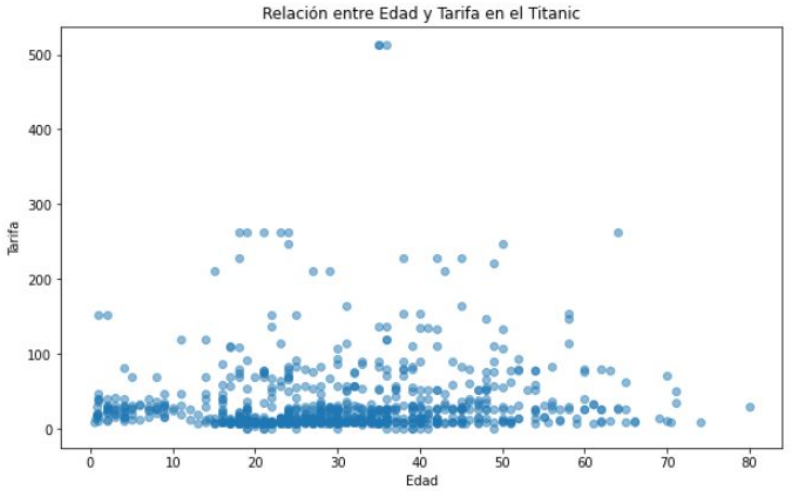
# Gráfico de dispersión

Un **gráfico de dispersión** es una representación visual de la relación entre dos variables donde se utilizan puntos individuales para representar las observaciones. Son muy útiles para encontrar patrones y tendencias, identificación de **outliers** y correlación entre las variables.

Podemos observar que la relación entre la edad y tarifa pagada por los pasajeros no es lineal ni tampoco sigue un patrón en especifico. Solo se evidencia que la mayor proporción pago una tarifa muy baja sin tener en cuenta la edad.

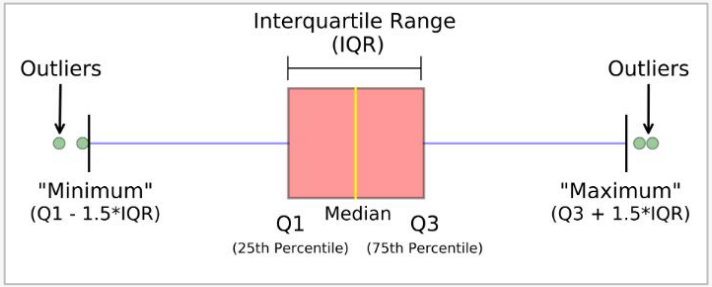
También se detectan 2 casos atípicos cuya tarifa es muy alta y fue pagada por personas de casi 40 años.

```
In [5]: plt.figure(figsize=(10, 6))
        plt.scatter(titanic_data['Age'], titanic_data['Fare'], alpha=0.5)
        plt.title('Relación entre Edad y Tarifa en el Titanic')
        plt.xlabel('Edad')
        plt.ylabel('Tarifa')
        plt.show()
```



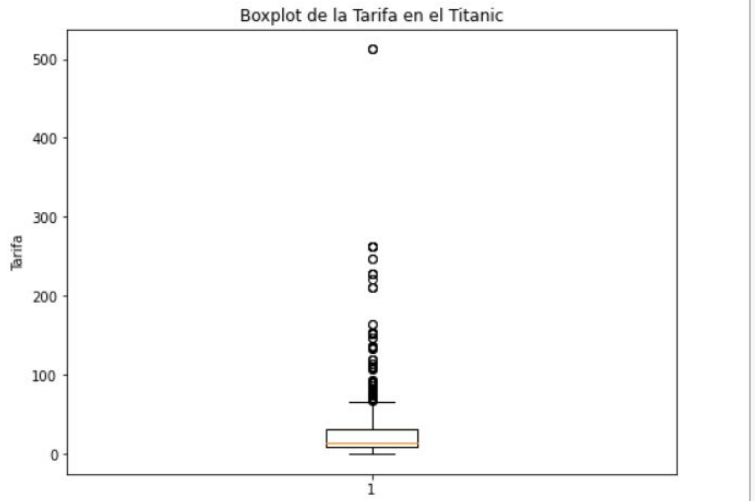
# Boxplot

Un **boxplot** es una representación visual de la distribución estadística de un conjunto de datos. Proporciona información sobre la **mediana**, los **cuartiles** y la presencia de posibles **valores atípicos**.




En el gráfico, podemos observar la presencia de varios valores atípicos por encima del boxplot, donde encontramos nuevamente los puntos detectados en el gráfico de dispersión (Tarifa de 500 aproximadamente).

```
In [7]: plt.figure(figsize=(8, 6))
        plt.boxplot(titanic_data['Fare'])
        plt.title('Boxplot de la Tarifa en el Titanic')
        plt.ylabel('Tarifa')
        plt.show()
```



# Serie de tiempo

Una **serie de tiempo** es una **secuencia** de datos observados o registrados en intervalos regulares a lo largo del tiempo. Estos datos pueden representar cualquier cosa que varíe con el tiempo, como temperaturas diarias, **precios de acciones**, ingresos mensuales, etc.

No tienen ningún parámetro o cuestión especial, solamente que en estos casos una de las variables tiene que ser de tipo **fecha** (y por convención, se posiciona generalmente en el eje X).

Veremos mas adelante ejemplos de estos casos aplicados a los valores de ciertas acciones.

In [15]:
```
# Creamos una serie temporal de temperaturas diarias ficticias
date_rng = pd.date_range(start='2023-01-01', end='2023-02-28', freq='D')
temperature_data = np.random.normal(loc=20, scale=5, size=len(date_rng))
time_series_data = pd.Series(temperature_data, index=date_rng)

plt.figure(figsize=(12, 6))
plt.plot(time_series_data, label='Temperatura Diaria', color='blue')
plt.title('Serie Temporal de Temperaturas Diarias')
plt.xlabel('Fecha')
plt.ylabel('Temperatura (°C)')
plt.legend()
plt.show()
```