«Talento Tech»

Data Analytics con Python

Clase 05

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 5 | Conceptos básicos

Temario:

- ¿Qué es Pandas? ¿Para qué sirve?
- Uso y configuración de Pandas.
- Creación y manipulación de DataFrames.
- Importación y exportación de datos (CSV, Excel).

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Pandas

¿Qué es Pandas?

Pandas es una biblioteca de código abierto para el lenguaje de programación Python, utilizada principalmente para el análisis y la manipulación de datos. Pandas proporciona estructuras de datos y funciones de manipulación de datos de alto rendimiento y fáciles de usar, particularmente útiles en tareas de limpieza y preparación de datos.

Se basa en las estructuras de datos o arrays de la librería NumPy por lo que al momento de instalar Pandas también se instalará numpy si no lo tenemos instalado.

Las estructuras que están disponibles en esta librería son: **Dataframe** y **Series**.

¿Para qué sirve?

Pandas se utiliza para una variedad de tareas en el análisis de datos, incluyendo:

- **Manipulación y limpieza de datos**: Eliminar duplicados, manejar valores faltantes, y transformar datos.
- **Análisis exploratorio de datos**: Estadísticas descriptivas, agrupaciones y agregaciones.
- **Transformación de datos**: Filtrar, ordenar, y modificar conjuntos de datos.

- **Almacenamiento y recuperación de datos**: Leer y escribir datos de y hacia varios formatos de archivos (CSV, Excel, SQL, etc.).

Uso y configuración de Pandas

Para comenzar a usar Pandas, primero debes instalarlo (si aún no lo tienes instalado) y luego importarlo en tu script de Python.

Instalación
Puedes instalar Pandas usando **pip**:

Código
```
##
pip install pandas
##
```

*Si lo realizas desde Colab:

Código
```
##
!pip install pandas
##
```

Importación
Una vez instalado, puedes importar Pandas en tu script:
Código
```
##
import pandas as pd
##
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Series

¿Qué es una Serie?

Una serie es una estructura de datos unidimensional en Pandas, similar a un array o lista, pero con etiquetas en cada elemento.

Creación de Series

Para crear una Serie primero debemos importar la librería Pandas

Código

```
import panda as pd
```

Una vez importada, definimos la serie usando la función Series de la libería. Luego a nuestra función Series le pasamos una lista.

Código

```
series = pd.Series([1, 2, 3, 4, 5])
print(series)
```

Lo que veremos en pantalla será:

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

En la primera columna podemos ver los índices de los elementos en la serie, en la segunda columna cada elemento. También observamos que nos devuelve el tipo de dato que

contiene nuestra serie (**dtype**). Como todos los valores de la serie son enteros el tipo de dato es **int64**, un dato int de 64 bits.

Veamos algunas funciones que podemos aplicar a las series.

Código

```
cantidad_elementos = serie.count()
suma_elementos = serie.sum()
promedio_elementos = serie.mean()
maximo = serie.max()
minimo = serie.min()
```

- count(): La función count() nos devuelve la cantidad de elementos que contiene la serie.

Si los datos de nuestra serie son numéricos (float o int) podremos aplicar funciones que realicen cálculos.

- sum(): La función sum() realiza una suma de los elementos de la serie.
- mean(): La función mean() nos devuelve el promedio de la serie.
- max(): Nos devuelve el valor máximo de la serie.
- min(): Nos devuelve el valor mínimo de la serie.

DataFrames

¿Qué es un DataFrame?

Un DataFrame es una estructura de datos bidimensional con ejes etiquetados (filas y columnas), similar a una tabla en una base de datos o una hoja de cálculo de Excel. Cada columna será una serie.

Creación de DataFrames

Puedes crear un DataFrame de varias maneras, incluyendo a partir de un diccionario, una lista de listas, o leyendo un archivo de datos.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Desde un diccionario:

Código

```
import pandas as pd

data = {
    'Nombre': ['Alice', 'Bob', 'Charlie'],
    'Edad': [25, 30, 35],
    'Ciudad': ['Nueva York', 'Los Ángeles', 'Chicago']
}

df = pd.DataFrame(data)
print(df)
```

Salida:

Código

```
      Nombre  Edad       Ciudad
  0    Alice    25     Nueva York
  1      Bob    30   Los Ángeles
  2  Charlie    35       Chicago
```

A partir de Series:

Podemos crear un dataframe vacío e incorporar las Series como columnas.
Para esto creamos por un lado el dataframe vacío usando pd.DataFrame() y luego cada serie que luego agregaremos al dataframe.
Código

```
import pandas as pd

df = pd.DataFrame()  # creamos el dataframe vacío
nombres = pd.Series(['Alice', 'Bob', 'Charlie']) # serie nombres
edades = pd.Series([25, 30, 35])  # serie edades
```

```
ciudades = pd.Series(['Nueva York', 'Los Ángeles', 'Chicago'])  # serie ciudades

# agregamos las columnas al dataframe como df[nombre_columna] = serie
df['Nombre'] = nombres # creamos la columna Nombre en el dataframe
df['Edad'] = edades # creamos la columna Edad en el dataframe
df['Ciudad'] = ciudades # creamos la columna Ciudad en el dataframe
print(df)
```

Salida:

Código

```
     Nombre  Edad       Ciudad
0     Alice    25   Nueva York
1       Bob    30  Los Ángeles
2   Charlie    35      Chicago
```

Manipulación de DataFrames

Obtener y renombrar columnas

Sin necesidad de conocer el contenido del dataset de antemano, podemos utilizar el atributo columns para identificar los nombres de las columnas. Esto nos permite realizar consultas específicas y, si es necesario, modificar los nombres de las mismas.

Código:

```
df.columns
```

Accediendo al atributo columns del DataFrame, podemos obtener un array que contiene los nombres de todas las columnas.

```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

Para modificar estos nombres tenemos que igualar el atributo columns con un nuevo array que contenga los nuevos nombres de las columnas.

Código:

```
df.columns = ['id_film', 'tipo', 'titulo', 'director', 'cast', 'pais',
              'fecha_agregacion', 'anio_lanzamiento', 'rating','duracion',
              'genero', 'descripcion']
```

Para confirmar si el cambio se realizó correctamente, utilizaremos el método head(), que nos permite ver las primeras 5 filas del DataFrame.

Código:

```
df.head()
```

Consola:

	id_film	tipo	titulo	director	cast	pais	fecha_agregacion	anio_lanzamiento	rating	duracion	genero	descripcion
0	s1	Movie	The Grand Seduction	Don McKellar	Brendan Gleeson, Taylor Kitsch, Gordon Pinsent	Canada	March 30, 2021	2014	NaN	113 min	Comedy, Drama	A small fishing village must procure a local d...
1	s2	Movie	Take Care Good Night	Girish Joshi	Mahesh Manjrekar, Abhay Mahajan, Sachin Khedekar	India	March 30, 2021	2018	13+	110 min	Drama, International	A Metro Family decides to fight a Cyber Crimin...
2	s3	Movie	Secrets of Deception	Josh Webber	Tom Sizemore, Lorenzo Lamas, Robert	United States	March 30, 2021	2017	NaN	74 min	Action, Drama, Suspense	After a man discovers his wife is

Selección de columnas:

Código

```
print(df['Nombre'])
```

Explicación:
La selección de columnas en un DataFrame de Pandas permite acceder a los datos de una columna específica utilizando el nombre de la columna como una clave. Esto es útil para analizar o manipular los datos de una columna en particular.
Este código imprime todos los valores de la columna `Nombre` del DataFrame `df`. Aquí se accede directamente a la columna utilizando su nombre dentro de corchetes. Esto devolverá una Serie de Pandas que contiene todos los valores de la columna `Nombre`.

Selección de filas:

Código

```
print(df.iloc[0])  # Selecciona la primera fila
print(df.loc[1])   # Selecciona la fila con índice 1
```

Explicación:
La selección de filas en Pandas se puede hacer utilizando las funciones **`iloc`** y **`loc`**, que permiten acceder a filas por su posición o por su índice, respectivamente.
`df.iloc[0]`: Utiliza el índice entero (basado en la posición) para seleccionar la primera fila del DataFrame.
`df.loc[1]`: Selecciona la fila que tiene el índice igual a 1. Es útil cuando los índices no son enteros o cuando han sido definidos manualmente.

Filtrado de datos:

Código

```
print(df[df['Edad'] > 30])
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Explicación:
El filtrado de datos permite extraer subconjuntos de datos de un DataFrame basándose en condiciones lógicas. Esto es esencial para analizar datos específicos que cumplen ciertos criterios.
Este código filtra el DataFrame para mostrar solo las filas donde la columna Edad es mayor que 30. Se crea una condición booleana **`df['Edad'] > 30`**, que devuelve un DataFrame con solo las filas que cumplen esa condición.

Añadir una nueva columna:

Código

```
df['Salario'] = [50000, 60000, 70000]
print(df)
```

Explicación:
Añadir nuevas columnas a un DataFrame es una operación común para enriquecer los datos con nueva información calculada o importada.
Aquí se añade una nueva columna llamada **`Salario`** al DataFrame **`df`**, asignando a cada fila un valor de salario correspondiente de la lista proporcionada. La longitud de la lista debe coincidir con el número de filas del DataFrame.

Ver índices

df.index -> muestra los índices del dataframe

Código

```
import pandas as pd
# Crear un DataFrame de ejemplo
data = {
     'Nombre': ['Ana', 'Juan', 'Luis', 'Sofía'],
    'Edad': [28, 34, 29, 42],
    'Ciudad': ['Madrid', 'Barcelona', 'Valencia', 'Sevilla']
```

```
}

df = pd.DataFrame(data)

# Mostrar el DataFrame
print("DataFrame:")
print(df)

# Ver los índices del DataFrame
print("\nÍndices del DataFrame:")
print(df.index)
```

Explicación:
El atributo **`index`** de un DataFrame en Pandas devuelve un objeto **`Index`** que contiene los índices de todas las filas del DataFrame. Esto es útil para ver cómo están organizados los datos y para acceder a las filas por su índice.

Importación de datos

Pandas facilita la importación y exportación de datos desde y hacia varios formatos de archivo.

- **Desde un archivo CSV**:

  Código

  ```
  df_csv = pd.read_csv('datos.csv')
  print(df_csv)
  ```

  Explicación:
  Este código utiliza la función **`read_csv`** de Pandas para leer datos de un archivo CSV llamado **`datos.csv`** y crear un DataFrame **`df_csv`**. La función

automáticamente interpreta el archivo CSV y organiza los datos en filas y columnas.

- **Desde un archivo Excel**:

  Código

```
df_excel = pd.read_excel('datos.xlsx', sheet_name='Hoja1')
print(df_excel)
```

  Explicación:

  La función **`read_excel`** permite importar datos desde un archivo Excel (`.xlsx`). Aquí se especifica el nombre del archivo y el nombre de la hoja (`Hoja1`) desde donde se quieren importar los datos. Esto genera un DataFrame **`df_excel`** con los datos de esa hoja.

Exportación de datos

- **A un archivo CSV**:

  Código

```
df.to_csv('datos_exportados.csv', index=False)
```

  Explicación:

  Este código exporta el DataFrame **`df`** a un archivo CSV llamado **`datos_exportados.csv`**. El argumento **`index=False`** se utiliza para no incluir el índice del DataFrame en el archivo CSV exportado.

- **A un archivo Excel**:

  Código

```
df.to_excel('datos_exportados.xlsx', index=False,
sheet_name='Hoja1')
```

  Explicación:

Buenos Aires aprende
Agencia de Habilidades para el Futuro

La función `to_excel` exporta el DataFrame **`df`** a un archivo Excel llamado **`datos_exportados.xlsx`**. El argumento `index=False` evita que el índice del DataFrame se incluya en el archivo, y **`sheet_name='Hoja1'`** especifica el nombre de la hoja en la que se guardarán los datos exportados.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Desafío N° 5:

Utiliza el dataset que contiene información sobre las producciones de Amazon Prime Video. Lo obtienes en este enlace. Crea un dataframe con los datos del dataset. Verifica los nombres de las columnas, sus índices, la cantidad de filas. Reconoce que contiene cada columna.

Colab ejercicio 5


Buenos Aires
aprende
Buenos
Aires
Ciudad