Nombre de la práctica	Pandas			No.	2
Asignatura:	Simulación	Carrera:	Ingeniería en Sistemas Computacionales	Duración de la práctica (Hrs)	

NOMBRE DEL ALUMNO: Vanesa Hernández Martínez
GRUPO: 3501

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):
Actividades en aula de clases y en equipo personal

III. Material empleado:
- Laptop
- Anaconda

Pandas

Pandas es una librería de Python que se utiliza para la manipulación y análisis de datos, especialmente para manejar grandes cantidades de información de manera eficiente. Es muy popular en el análisis de datos y la ciencia de datos debido a su facilidad para trabajar con datos estructurados, como tablas (dataframes), y para realizar tareas de limpieza, filtrado, agregación y visualización de datos.

Dentro de la siguiente practica abordamos los siguientes conceptos:

Series: Es una estructura unidimensional similar a una lista o un array de Numpy, pero con etiquetas o índices.
DataFrame: Es una estructura bidimensional (tabla) que contiene columnas etiquetadas, cada una de las cuales puede ser de un tipo de dato diferente (numérico, cadenas, booleanos, etc.).

Pandas permite leer y escribir datos desde varios formatos como CSV, Excel, bases de datos SQL, JSON, entre otros.
Facilita la manipulación de datos con técnicas como filtrado de filas/columnas, ordenamiento, y agrupamiento de datos.

Introducción a Pandas

Pandas es una biblioteca que proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y faciles de usar.

- La estructura de datos principal es el DataFrame, que puede considerarse como una tabla 2D en memoria (como una hoja de cálculo, nombres de columna y etiquetas de fila).
- Muchas funciones disponbles en excel estan disponibles mediante programacion, como crear las talas dinámicas, calcular columnas basadas en otras columnas, trazar gráficos, etc.
- Proporciona un alto rendimiento para manipular(unir, dividir, modificar,etc.) grandes volumenes de datos.

Import

```
import pandas as pd
```

Estructuras de datos en Pandas

La biblioteca Pandas, de manera genérica, contiene las siguientes estructuras de datos:

- **Series**: Array de una dimensión.
- **DataFrame**: Se corresponde con una tabla de dos dimensiones.
- **Panel**: Similar aun diccionario de DataFrames.

Creación del Objeto Series

```
# Creación del objeto Series.

s = pd.Series([2, 4, 6, 8, 10]) print(s)

0    2

1    4

2    6

3    8

4   10

dtype: int64

# Creación de un objeto series e inicializarlo con un diccionario de Python.

Emilio    169
Anel      145
Chucho    170
9, "Anel": 145, "Chucho": 170, "Jocelin": 170} s = pd.Series(Altura)
```

```
Jocelin    170
dtype: int64
```

```
# Creación de un Objeto Series e inicializarlo con algunos elemntos de un diccionario de Python.

Altura = {"Emilio": 169, "Anel": 145, "Chucho": 170, "Jocelin": 170} s = pd.Series(Altura, index
= ["Jocelin", "Emilio"])
print(s)
```

```
Jocelin    170
Emilio     169
dtype: int64
```

```
# Creación de un objeto Series e inicializarlo con un escalar.

s = pd.Series(34, ["Num1", "Num2", "Num3", "Num4"]) print(s)
```

```
Num1    34
Num2    34
Num3    34
Num4    34
dtype: int64
```

Acceso a los elementos de un array

Cada elemento es un objeto Series tiene un identificador que se denomina **index label** .

```
# Crear un Objeto Series

s = pd.Series([2, 4, 6, 8], index=["Num1", "Num2", "Num3", "Num4"]) print(s)
```

```
Num1    2

Num2    4

Num3    6

Num4    8
dtype: int64
```

```
## Acceder al tercer elemento del objeto

s["Num3"]
```

```
/tmp/ipykernel_5692/2186343123.py:2: FutureWarning: Series.__getitem__
treating keys as positions is deprecated. In a future version, integer
```

```
# Tambien se puede acceder por posición.

s[2]
```

```
np.int64(6)

# loc es la forma estandar de acceder a un elemento de un Objeto Series por atributo.

s.loc["Num3"]

np.int64(6)

# iloc es la forma estandar de acceder a un elemento de un Objeto Series por posición

s.iloc[2] np.int64(6)

# Accedendo al tercer elemento por posición.

s.iloc[2:4]

Num3      6
Num4      8
dtype: int64
```

Operaciones aritméticas con Series

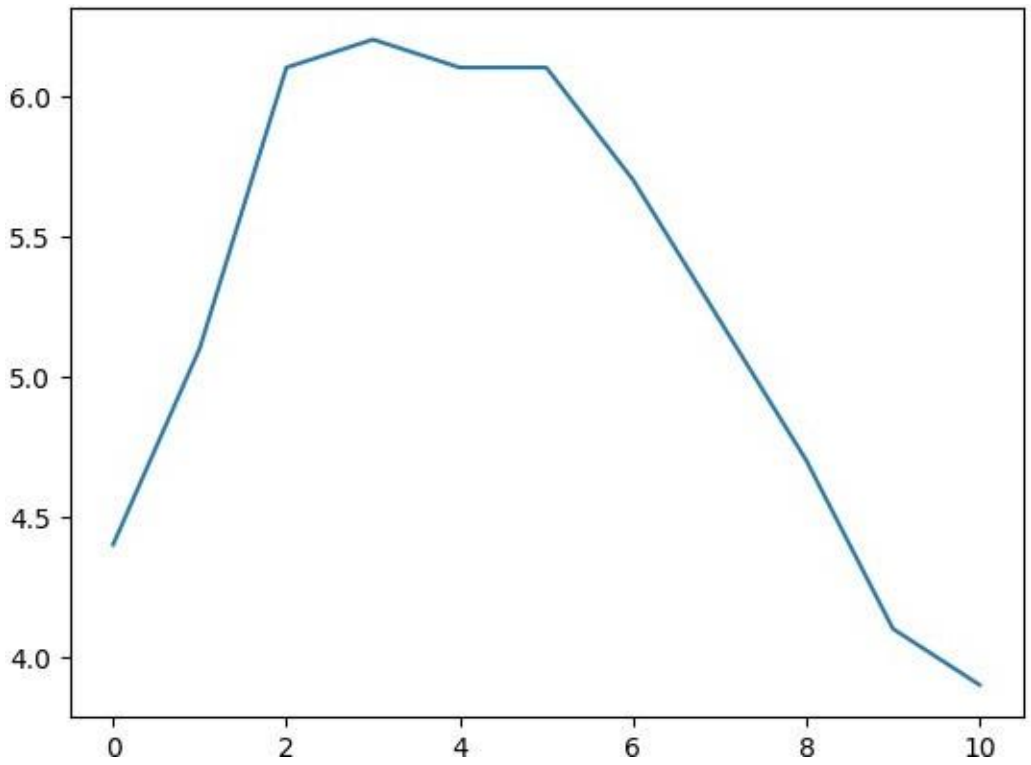
```
# Crear un objeto Series

s = pd.Series([2, 4, 6, 8, 10]) print(s)

0         2
1         4
2         6
3         8
4        10
dtype: int64

# Los obejtos series son similares y compatibles con los Arrays de Numpy.

import numpy as np
# ufun de Numpy para sumar los elementos.

np.sum(s)

np.int64(30) s *2

0         4
1         8
2        12
3        16
4        20
dtype: int64
```

Representación gráfica de un objeto Series

```
# Crear un objeto Series denominado Temperaturas.

Temperaturas = [4.4, 5.1, 6.1, 6.2, 6.1, 6.1, 5.7, 5.2, 4.7, 4.1, 3.9]
s = pd.Series(Temperaturas, name="Temperaturas") s
```

```
0     4.4
1     5.1
2     6.1
3     6.2
4     6.1
5     6.1
6     5.7
7     5.2
8     4.7
9     4.1
10    3.9
Name: Temperaturas, dtype: float64
```

```
# Representación gráfica del objeto Series.

%matplotlib inline
import matplotlib.pyplot as plt

s.plot() plt.show()
```



Creación de un objeto DataFrame.

```
# Creación de un DataFrame e inicializarlo objetos Series.          con un diccionario de

Personas = {

        "Peso": pd.Series([72, 60, 74, 73], ["Emilio", "Anel", "Chucho", "Jocelin"]),

        "Altura": pd.Series({"Emilio": 169, "Anel": 145, "Chucho": 170,

"Jocelin": 170}),

        "Mascotas": pd.Series([2, 9], ["Anel", "Jocelin"])

`
```

```
         Peso   Altura   Mascotas
Anel       60      145        2.0
Chucho     74      170        NaN
Emilio     72      169        NaN
Jocelin    73      170        9.0
```

Es posible forzar el DataFrame a que presente determinadas columnas y en orden determinado.

```
# Creación de un DataFrame e inicializarlo con un diccionario de objeto de series.

Personas = {

        "Peso": pd.Series([72, 60, 74, 73], ["Emilio", "Anel", "Chucho", "Jocelin"]),

        "Altura": pd.Series({"Emilio": 169, "Anel": 145, "Chucho": 170,

"Jocelin": 170}),

        "Mascotas": pd.Series([2, 9], ["Anel", "Jocelin"])

}

df = pd.DataFrame( Personas,

        columns = ["Altura", "Peso"],

        index = ["Chucho", "Emilio"])

df
```

```
            Altura   Peso
Chucho         170     74
```

```
]

df = pd.DataFrame( Valores,

        columns = ["Altura", "Mascotas", "Peso"], index = ["Jocelin",
        "Emilio", "Anel"]

)
```

```
         Altura   Mascotas   Peso
Jocelin    169         3       72
Emilio     145         2       60
Anel       170         1       74
```

```
# Creación de un DataFrame e inicializarlo con un diccionario de Python.

Personas = {

        "Peso": {"Emilio": 72, "Anel": 60, "Chucho": 74, "Jocelin":73},

        "Altura": {"Emilio": 169, "Anel": 145, "Chucho": 170, "Jocelin":

170}

`
```

```
         Peso   Altura
Emilio    72      169
Anel      60      145
Chucho    74      170
Jocelin   73      170
```

Acceso a los elementos de un DataFrame

```
# Creación de un DataFrame e inicializarlo con un diccionario de Python.

Personas = {

        "Peso": pd.Series([72, 60, 74, 73], ["Emilio", "Anel", "Chucho", "Jocelin"]),

        "Altura": pd.Series({"Emilio": 169, "Anel": 145, "Chucho": 170,

"Jocelin": 170}),

        "Mascotas": pd.Series([2, 9], ["Anel", "Jocelin"])

}
```

```
         Peso   Altura   Mascotas
Anel      60      145       2.0
```

```
Chucho     74      170       NaN
```

```
Emilio      72      169       NaN
Jocelin     73      170       9.0
```

Acceso a los elementos de las columnas del DataFrame.

```
df["Peso"]
```

```
Anel        60
Chucho
Emilio      74
Jocelin
Name: Peso, dtype: int64
```

```
df[["Peso". "Altura"]]
```

```
          Peso   Altura
Anel       60      145
Chucho     74      170
Emilio     72      169
Jocelin    73      170
```

```
# Pueden convinarse los elementos anteriores con expresiones booleanas.

df["Peso"] > 73
```

```
Anel        False
Chucho       True
Emilio      False
Jocelin     False
Name: Peso, dtype: bool
```

```
# Pueden convinarse los métodos anteriores con expresiones boleanas y mostrar el DataFrame

df[df["Peso"] > 72]
```

```
            Peso  Altura  Mascotas Chucho
              74      170            NaN
```

Accediendo a los elementos de las filas del DataFRame

```
# Mostrar el DataFrame
```

```
          Peso   Altura   Mascotas
Anel       60      145       2.0
Chucho     74      170       NaN
Emilio     72      169       NaN
Jocelin    73      170       9.0
```

```
df.loc["Emilio"]
```

```
Peso Altura      72.0
Mascotas
                169.0


Name: Emilio, dtype: float64 df.iloc[1:3]
```

	Peso	Altura	Mascotas
Chucho	74	170	NaN
Emilio	72	169	NaN

Consultas avanzada de los elementos de un DataFrame

```
# Mostrar el DataFrame
```

	Peso	Altura	Mascotas
Anel	60	145	2.0
Chucho	74	170	NaN
Emilio	72	169	NaN
Jocelin	73	170	9.0

```
df.query("Altura >= 170 and Peso >= 73")
```

	Peso	Altura	Mascotas
Chucho	74	170	NaN
Jocelin	73	170	9.0

Copiar un DataFrame

```
# Crear un DataFrame e inicializarlo con un diccionario de objetos Series

Personas = {

        "Peso": pd.Series([72, 60, 74, 73], ["Emilio", "Anel", "Chucho", "Jocelin"]),

        "Altura": pd.Series({"Emilio": 169, "Anel": 145, "Chucho": 170,

"Jocelin": 170}),

        "Mascotas": pd.Series([2, 9], ["Anel", "Jocelin"])

}
```

	Peso	Altura	Mascotas
Anel	60	145	2.0
Chucho	74	170	NaN
Emilio	72	169	NaN
Jocelin	73	170	9.0

```
# Copia del DataFrame df en df_copy.

# NOTA:Al modificar un elemento del df_copy no se modifica el df.
```

Modificación de un DataFrame

```
# Anadir una nueva columna al DataFrame

df["Anio_Nac"] = [2004, 2004, 2004, 2004]
df
```

	Peso	Altura	Mascotas	Anio_Nac
Anel	60	145	2.0	2004
Chucho	74	170	NaN	2004
Emilio	72	169	NaN	2004
Jocelin	73	170	9.0	2004

```
# Añadir una nueva columna calculada al DataFrame.

df["Edad"] = 2024- df["Anio_Nac"] df
```

	Peso	Altura	Mascotas	Anio_Nac	Edad
Anel	60	145	2.0	2004	20
Chucho	74	170	NaN	2004	20
Emilio	72	169	NaN	2004	20
Jocelin	73	170	9.0	2004	20

```
# Añadir una nueva columna creando un DataFrame nuevo.

df_mod = df.assign(Hijos = [2, 1, 2, 1]) df_mod
```

	Peso	Altura	Mascotas	Anio_Nac	Edad	Hijos
Anel	60	145	2.0	2004	20	2
Chucho	74	170	NaN	2004	20	1
Emilio	72	169	NaN	2004	20	2
Jocelin	73	170	9.0	2004	20	1

```
df
```

	Peso	Altura	Mascotas	Anio_Nac	Edad
Anel	60	145	2.0	2004	20
Chucho	74	170	NaN	2004	20
Emilio	72	169	NaN	2004	20
Jocelin	73	170	9.0	2004	20

```
# Eliminar una columna existente del DataFrame

del df["Peso"] df
```

	Altura	Mascotas	Anio_Nac	Edad

```
Anel      145        2.0        2004      20
```

```
Chucho       170        NaN      2004      20
Emilio       169        NaN      2004      20
Jocelin      170        9.0      2004      20
```

 # Eliminar una columna existente, devlviendo una copia del DataFrame resultante.

```
          Peso   Altura   Mascotas   Anio_Nac  Edad
Anel        60    145          2.0       2004   20
Chucho      74    170          NaN       2004   20
Emilio      72    169          NaN       2004   20
Jocelin     73    170          9.0       2004   20
```

```
df
```

```
          Altura   Mascotas   Anio_Nac  Edad
Anel       145          2.0       2004   20
Chucho     170          NaN       2004   20
Emilio     169          NaN       2004   20
Jocelin    170          9.0       2004   20
```

Evaluación de expresiones sobre un DataFrame

 # Crear un DataFrame e inicializarlo con un diccionario de Objetos Series.

```
Personas = {
        "Peso": pd.Series([72, 60, 74, 73], ["Emilio", "Anel", "Chucho", "Jocelin"]),
        "Altura": pd.Series({"Emilio": 169, "Anel": 145, "Chucho": 170,
"Jocelin": 170}),
        "Mascotas": pd.Series([2, 9], ["Anel", "Jocelin"])
}
```

```
          Peso  Altura  Mascotas
Anel       60     145        2.0
Chucho      74      170      NaN
Emilio      72      169      NaN
Jocelin     73      170      9.0
```

 # Evaluar una función sobre una columna del DataFrame.

```
   Anel      72.5
   Chucho    85.0
   Emilio    84.5
```

```
# Cargar el DataFrame en Jupyter

df2 = pd.read_csv("df_Personas.csv")
```

```
Anel       False
Chucho      True
Emilio      True
Jocelin     True
```

```
# Cargar el DataFrame con la primera columna correctamnete asignada.

df2 = pd.read_csv("df_Personas.csv", index_col=0) df2
```

```
Anel       62
Chucho     76
Emilio     74
Jocelin    75
Name: Peso, dtype: int64
```

Guardar y cargar el DataFrame

```
# Crear un DataFrame e inicializarlo con un diccionario de objetos Series

Personas = {

        "Peso": pd.Series([72, 60, 74, 73], ["Emilio", "Anel", "Chucho", "Jocelin"]),

        "Altura": pd.Series({"Emilio": 169, "Anel": 145, "Chucho": 170,

"Jocelin": 170}),

        "Mascotas": pd.Series([2, 9], ["Anel", "Jocelin"])

}
```

```
         Peso   Altura   Mascotas
Anel      60      145       2.0
Chucho    74      170       NaN
Emilio    72      169       NaN
Jocelin   73      170       9.0
```

```
# Guardar el DataFrame como CSV, HTML y JSON.
df.to_csv("df_Personas.csv") df.to_html("df_Personas.html")
df.to_json("df_Personas.json")
```

	Unnamed: 0	Peso	Altura	Mascotas
0	Anel	60	145	2.0
1	Chucho	74	170	NaN
2	Emilio	72	169	NaN
3	Jocelin	73	170	9.0

	Peso	Altura	Mascotas
Anel	60	145	2.0
Chucho	74	170	NaN
Emilio	72	169	NaN
Jocelin	73	170	9.0

Resumen del visto en la practica anterior :

Pandas es una biblioteca de Python diseñada para trabajar con datos de forma rápida y eficiente. Permite la manipulación y análisis de datos estructurados como tablas.

DataFrame es la estructura de datos más importante en Pandas, que se asemeja a una tabla 2D (como una hoja de cálculo) con nombres de columnas y etiquetas de filas. Facilita el manejo de datos tabulares.Pandas es muy eficiente para manejar, unir, dividir, y modificar grandes volúmenes de datos de forma rápida.

Estructuras de Datos en Pandas:

- **Series**: Arreglo unidimensional, similar a una columna de una tabla.
- **DataFrame**: Tabla bidimensional que contiene etiquetas de fila y columna.
- **Panel**: Una estructura más compleja, que es un contenedor de DataFrames, aunque no se usa comúnmente en las versiones más recientes de Pandas.

Cada elemento en una **Serie** tiene un **index label** que permite su acceso. En un **DataFrame**, puedes acceder a datos por columnas o por filas usando .loc[] o .iloc[].
Es posible realizar operaciones aritméticas con **Series** y **DataFrames** de forma sencilla.

Los **DataFrames** pueden crearse a partir de diccionarios, listas o archivos externos (como CSV o Excel).
Se pueden modificar, añadir o eliminar columnas y filas en un **DataFrame**. Ejemplo: añadir una nueva columna calculada o una columna con valores fijos.
Pandas permite realizar consultas avanzadas, filtrado de datos y evaluaciones basadas en condiciones lógicas para manipular el contenido del **DataFrame**.
Es posible realizar copias de un **DataFrame** y guardar los datos en archivos como CSV o Excel para su reutilización futura.

Conclusión

Pandas es una herramienta poderosa y fácil de usar que permite trabajar con grandes volúmenes de datos de manera eficiente. Su estructura principal, el **DataFrame**, simplifica la manipulación y análisis de datos como si trabajáramos con una hoja de cálculo, pero con la flexibilidad y velocidad de la programación. Con Pandas, es posible realizar tareas como filtrar, ordenar, modificar y analizar datos, además de crear gráficos y tablas dinámicas. Su capacidad para manejar datos complejos la convierte en una herramienta indispensable para cualquier persona que trabaje con análisis de datos en Python.