

Nombre de la práctica	PANDAS			No.	
Asignatura:	Simulación	Carrera :	Inf. Sistemas Computacionales	Duración de la práctica (Hrs)	

Nombre: Ana Edith Hernández Hernández

I. Competencia(s) específica(s):

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro): Aula

III. Material empleado:

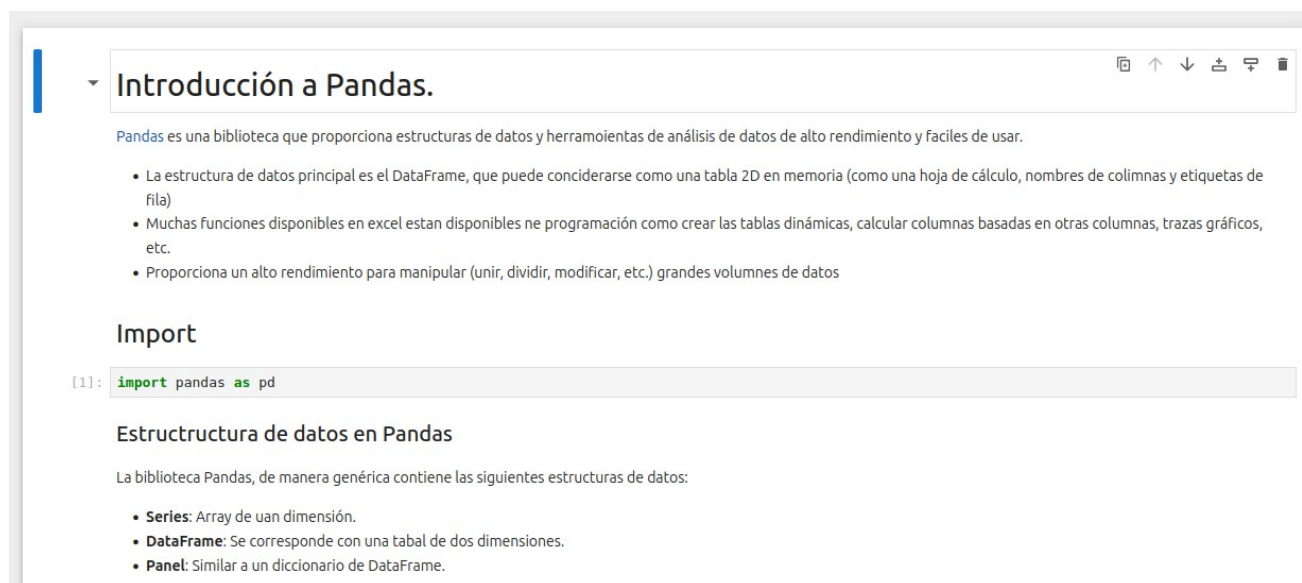
Laptop

Anaconda

IV. Desarrollo de la práctica:

Pandas

Se



Introducción a Pandas.

Pandas es una biblioteca que proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles 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 columnas y etiquetas de fila)
- Muchas funciones disponibles en excel están disponibles en programación como crear las tablas dinámicas, calcular columnas basadas en otras columnas, trazas gráficos, etc.
- Proporciona un alto rendimiento para manipular (unir, dividir, modificar, etc.) grandes volúmenes de datos

Import

```
[1]: import pandas as pd
```

Estructura 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 a un diccionario de DataFrame.

crea una introducción y un acceso directo para la documentación de Pandas.

En esta describe la estructura de datos.

Comenzamos con la creación del objeto series.

Creacion del objeto series.

```
[2]: # Creacion 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

[3]: # Creacion de un objeto series e inicializarlo con un diccionario de python
Altura = {"Emilio": 169, "Anel": 145, "Chucho": 170, "Jocelin": 170}
s = pd.Series(Altura)
print(s)

Emilio    169
Anel      145
Chucho    170
Jocelin    170
dtype: int64

[4]: # Creacion de un objeto series e inicializarlo con algunos elementos 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

[5]: # Creacion de un objeto series e inicializarlo con un ecalar.
s = pd.Series(34, ["Num1", "Num2", "Num3", "Num4"])
print(s)

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

En la creacción del objeto series se crea un diccionario de datos

Acceso de los elementos del array

Acceso a los elementos de un Array

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

```
[6]: #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

[7]: # Acceder al tercer elemento del objeto.
s["Num3"]

Cell In[7], line 2
s["Num3"]
^
IndentationError: unexpected indent

[ ]: s[2]

[ ]: # loc es la forma estandar de acceder a un elemento de un Objeto Series
s.loc["Num3"]

[ ]: # iloc es la forma estandar de acceder a un elemento de un objeto por posicion
s.iloc[2]

[ ]: # Accediendo al segundo y tercer elemento por posicion
s.iloc[2:4]
```

Se accede a los elementos del diccionario dependiendo su posición.

OPERACIONES ARITMÉTICAS SERIES

Operaciones Aritméticas con Series

```
[ ]: #Crear un objeto Series
s = pd.Series([2, 4, 6, 8, 10])
print(s)

[ ]: # Los objetos series son similares y compatibles con los Array de numpy.
import numpy as np
# ufunc de Numpy para sumar los elementos.
np.sum(s)

[ ]: s * 2
```

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

[ ]: # Representacion gráfica del objeto Series
%matplotlib inline
import matplotlib.pyplot as plt

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

Hacemos la creación de DataFrame.



Creación de un objeto DataFrame .

```
[ ]: # Creacion de 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": 165, "Chucho": 170, "Jocelin": 170}),
    "Mascotas": pd.Series([2, 9], ["Anel", "Jocelin"])
}

df = pd.DataFrame(Personas)
df
```

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

```
[ ]: # Creacion de 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": 165, "Chucho": 170, "Jocelin": 170}),
    "Mascotas": pd.Series([2, 9], ["Anel", "Jocelin"])
}

df = pd.DataFrame(
    Personas,
    columns = ["Altura", "Peso"],
    index = ["Chucho", "Emilio"])
df
```

```
[ ]: # Creacion de un DataFrame e inicializar con una lista de listas de python
# NOTA: Deben especificarse las columnas e indices por separado
Valores = [
    [169, 3, 72],
    [145, 2, 60],
    [170, 1, 74],
]

df = pd.DataFrame(
    Valores,
    columns = ["Altura", "Mascotas", "Peso"],
    index = ["Jocelin", "Emilio", "Anel"]
)
df
```



```
[9]: # Creacion de un DataFrame e inicializarlo con un diccionario en Python
Personas = {
    "Peso": {"Emilio": 72, "Anel": 60, "Chucho": 74, "Jocelin": 73},
    "Altura": {"Emilio": 169, "Anel": 165, "Chucho": 170, "Jocelin": 170}}

df = pd.DataFrame(Personas)
df
```

```
[9]:
```

	Peso	Altura
Emilio	72	169
Anel	60	165
Chucho	74	170
Jocelin	73	170

Acceso a los elementos de un DataFrame

```
[12]: # Creacion 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": 165, "Chucho": 170, "Jocelin": 170}),
    "Mascotas": pd.Series([2, 9], ["Anel", "Jocelin"])
}
df = pd.DataFrame(Personas)
df
```

```
[12]:
```

	Peso	Altura	Mascotas
Anel	60	165	2.0
Chucho	74	170	NaN
Emilio	72	169	NaN
Jocelin	73	170	9.0

Accediento a los elemnto de las filas del DataFrame

```
[21]: # Mostrar el DataFrame
df
```

```
[21]:
```

	Peso	Altura	Mascotas
Anel	60	165	2.0
Chucho	74	170	NaN
Emilio	72	169	NaN
Jocelin	73	170	9.0

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

```
[22]:
```

Peso	72.0
Altura	169.0
Mascotas	NaN
Name: Emilio, dtype: float64	

```
[23]: df.iloc[1:3]
```

```
[23]:
```

	Peso	Altura	Mascotas
Chucho	74	170	NaN
Emilio	72	169	NaN

Consulta Avanzada de los elementos de un DataFrame

```
[24]: # Mostrar el df
df
```

```
[24]:
```

	Peso	Altura	Mascotas
Anel	60	165	2.0
Chucho	74	170	NaN
Emilio	72	169	NaN
Jocelin	73	170	9.0

```
[25]: df.query("Altura >= 170 and Peso >= 73")
```

```
[25]:
```

	Peso	Altura	Mascotas
Chucho	74	170	NaN
Jocelin	73	170	9.0

Copiar un DataFrame.

```
[26]: # Crear un DataFrame e inicializarlo con un diccionario de objetos Series
# Creacion 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": 165, "Chucho": 170, "Jocelin": 170}),
    "Mascotas": pd.Series([2, 9], ["Anel", "Jocelin"])
}
df = pd.DataFrame(Personas)
df
```

```
[26]:
```

	Peso	Altura	Mascotas
Anel	60	165	2.0
Chucho	74	170	NaN
Emilio	72	169	NaN
Jocelin	73	170	9.0

```
[28]: # Copias del DataFrame df en df_copy.
#Nota: Al modificar un elemnto del df_copy no se modifica el df.
df_copy = df.copy()
```

Modificacion de DataFrame

```
[31]: # Añadir una nueva columna al DataFrame.
df["Anio_Nac"] = [2004, 2004, 2004, 2004]
df
```

```
[31]:
```

	Peso	Altura	Mascotas	Anio_Nac
Anel	60	165	2.0	2004
Chucho	74	170	NaN	2004
Emilio	72	169	NaN	2004
Jocelin	73	170	9.0	2004



```
[33]: # Añadir una nueva columna creando un DataFrame nuevo.  
df_mod = df.assign(Hijos = [2, 1, 2, 1])
```

```
[35]: df_mod
```

```
[35]:
```

	Peso	Altura	Mascotas	Anio_Nac	Edad	Hijos
Anel	60	165	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

```
[36]: df
```

```
[36]:
```

	Peso	Altura	Mascotas	Anio_Nac	Edad
Anel	60	165	2.0	2004	20
Chucho	74	170	NaN	2004	20
Emilio	72	169	NaN	2004	20
Jocelin	73	170	9.0	2004	20

```
[37]: # Eliminar una columna existente del DataFrame.  
del df["Peso"]  
df
```

```
[37]:
```

	Altura	Mascotas	Anio_Nac	Edad
Anel	165	2.0	2004	20
Chucho	170	NaN	2004	20
Emilio	169	NaN	2004	20
Jocelin	170	9.0	2004	20

```
[39]: # Eliminar una columna existente devolviendo una copia del DataFrame resultante.  
df_mod = df_mod.drop(["Hijos"], axis=1)  
df_mod
```

```
[39]:
```

	Peso	Altura	Mascotas	Anio_Nac	Edad
Anel	60	165	2.0	2004	20
Chucho	74	170	NaN	2004	20
Emilio	72	169	NaN	2004	20
Jocelin	73	170	9.0	2004	20

```
[40]: df
```

```
[40]:
```

	Altura	Mascotas	Anio_Nac	Edad
Anel	165	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

```
[42]: # 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"])  
}  
df = pd.DataFrame(Personas)  
df
```

```
[42]:
```

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

```
[43]: # Evaluar una función sobre una columna del data frame.  
df.eval("Altura / 2 ")
```

```
[43]: Anel      72.5  
Chucho    85.0  
Emilio    84.5  
Jocelin    85.0  
Name: Altura, dtype: float64
```

```
[46]: # Evaluar una función utilizando una variable local  
max_altura = 165  
df.eval("Altura > @max_altura")
```

```
[46]: Anel      False  
Chucho     True  
Emilio     True  
Jocelin     True  
Name: Altura, dtype: bool
```

```
[47]: #Aplicar una función a una columna del DataFrame.  
def func(x):  
    return x + 2  
  
df["Peso"].apply(func)
```

```
[47]: Anel      62  
Chucho    76  
Emilio    74
```




Guardar y Cargar el DataFrame

```
[48]: 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)
df
```

```
[48]:
```

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

```
[50]: # 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")
```

```
[51]: # Cargar el DataFrame en Jupyter
df2 = pd.read_csv("df_Personas.csv")
```

```
[52]: df2
```

```
[52]:
```

	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

```
[54]: # Cargar el DataFrame con la primera columna correctamente asignada
df2 = pd.read_csv("df_Personas.csv", index_col=0)
df2
```

```
[54]:
```

	Peso	Altura	Mascotas
Anel	60	145	2.0
Chucho	74	170	NaN

V. Conclusiones:

Pandas es una biblioteca esencial en Python para el análisis y manipulación de datos. Su capacidad para trabajar con estructuras de datos flexibles y potentes, como los DataFrames y Series, facilita el manejo, limpieza, y análisis de grandes volúmenes de datos de manera intuitiva y eficiente. Ofrece herramientas avanzadas para la integración de datos, la transformación y el análisis estadístico, lo que permite a los usuarios explorar y obtener información valiosa de sus datos con facilidad.