Nombre de la práctica	Pandas			No.	1
Asignatura:	SIMULACION	Carrera:	INGENIERÍA EN SISTEMAS COMPUTACIONALES	Duración de la práctica (Hrs)	5 horas

NOMBRE DEL ALUMNO: Francisco David Colin Lira
GRUPO: 3502

I. Competencia(s) específica(s):

Analiza, modela, desarrolla y experimenta sistemas productivos y de servicios, reales o hipotéticos, a través de la simulación de eventos discretos, para dar servicio al usuario que necesite tomar decisiones, con el fin de describir con claridad su funcionamiento, aplicando herramientas matemáticas.

Encuadre con CACEI: Registra el (los) atributo(s) de egreso y los criterios de desempeño que se evaluarán en esta práctica.

Encuadre con CACEI

No. atributo	Atributos de egreso del PE que impactan en la asignatura	No.Criterio	Criterios de desempeño	No. Indicador	Indicadores
1	El estudiante identificará los principios de las ciencias básicas para la resolución de problemas prácticos de ingeniería	CD1	Propone alternativas de solucion	I1	diseño algoritmico
				I2	empleo de formulas y funciones
2	el estudiante diseñará esquemas de trabajo y procesos, usando metodologías congruentes en la resolución de problemas de ingeniería en sistemas computacionales	CD1	identifica metodologías y procesos empleados en la resolución de problemas	I1	identificacion y reconocimiento de distintas metodologias para la resolucion de problemas.
				I2	Manejo de procesos especificos en la solucion de problemas y/o deteccion de necesidades.
		CD2	diseña soluciones a problemas, empleando metodologías apropiadas al area	I1	uso de metodologias para el modelado de la solucion de sistemas y aplicaciones
				I2	diseño algoritmico.

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

Aula y equipo de cómputo personal.

III. Material empleado:

- Equipo de cómputo
- Jupyter Notebook
- Anaconda
- Pandas
- Python

IV. Desarrollo de la práctica:

Introduccion a Pandas

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

- La estrucutra de datos pricipal es el DataFrame, que puede considerarse como una tabla 2D en memoria (como una hoja de calculo, con nombres de columna y etiquetas de fila).
- Muchas funciones disponibles en Excel, estan disponibles mediante programacion como crear tablas dinamicas, calcular columnas basadas en otras columnas, trazar graficos, etc.
- Proporciona un alto rendimiento para manipular (unir, dividir, modificar, etc.) grandes conjuntos de datos

Import

```
import pandas as pd
```

Estructuras de Datos de Pandas

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

- **Series:** Array de una dimensión
- **DataFrame:** Corresponde a una tabla de 2 dimensiones
- **Panel:** Similar a un diccionario de DataFrames

Creacion de objeto Series

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

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

```
# Creacion de un objeto Series inicializando con un diccionario de Python
Altura =  {
    "Gabo": 170,
    "Lucero": 160,
    "Braulio": 180,
    "Jessica": 161
} # <-- Este es un diccionario de python
s = pd.Series(Altura)
print(s)
```

```
Gabo       170
Lucero     160
Braulio    180
Jessica    161
dtype: int64
```

```
# Crear un Objeto Series e inicializarlo con algunos de los elementos de un diccionario de python
Altura = {
     "Gabo": 170,
    "Lucero": 160,
    "Braulio": 180,
    "Jessica": 161
}

s = pd.Series(Altura, index = ["Braulio", "Gabo"])
print(s)
```

```
Braulio    180
Gabo       170
dtype: int64
```

```
# Creacion de un Objeto Series e inicializarlo con un escalar.
s = pd.Series(34, ["Test1", "Test2", "Test3"])
print(s)
```

```
Test1    34
Test2    34
Test3    34
dtype: int64
```

Acceso a los elementos de un objeto Series

Cada elemento en un objeto Series tiene un identificador unico 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"]
```

```
np.int64(6)
```

```
# Tambein se puede acceder al elemento por posicion.
s[2]
```

```
/var/folders/hz/wt361b2n07lddh2hq8l8sy_40000gn/T/ipykernel_59179/2188709470.py:2: FutureWarning: Series.__getitem__ treating keys as position
s is deprecated. In a future version, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value b
y position, use `ser.iloc[pos]`
  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 posicion.
s.iloc[2]
```

```
np.int64(6)
```

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

```
Num3    6
Num4    8
dtype: int64
```

Operaciones aritmeticas con Objetos Series

```
# Creacion de 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 objetos Series son similares y compatibles con los arrays de Numpy
import numpy as np
# ufunc de numpy para sumar los elementos de un array.
np.sum(s)
```

```
np.int64(30)
```

```
# El resto de las operaciones aritmeticas de Numpy sobre Arrays tambien son posibles.
# NOTA: para mas informacion al respecto ir a la introduccion de Numpy ó a su documentacion.
s * 2
```

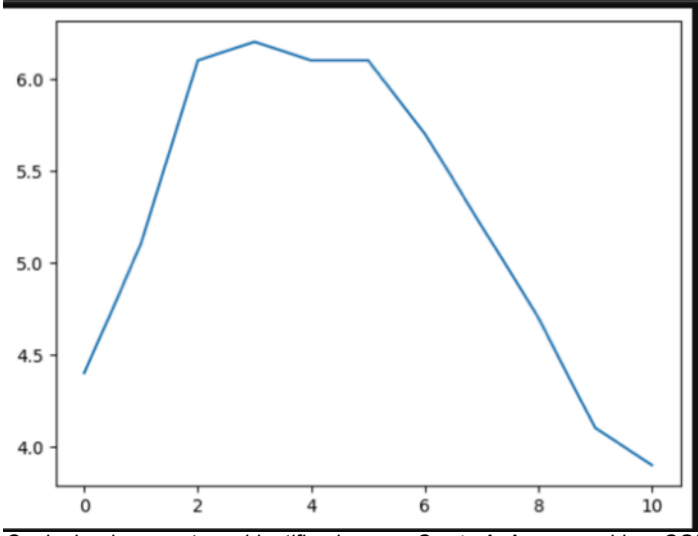
```
0     4
1     8
2    12
3    16
4    20
dtype: int64
```

Representacion Grafica de un Objeto Series

```
Temperaturas = [4.4, 5.1, 6.1, 6.2, 6.1, 6.1, 5.7, 5.2, 4.7, 4.1, 3.9] #<-- Estan dados en grados centigrados
s = pd.Series(Temperaturas, name = "Temperaturas")
print(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
```

```
# Representacion grafica
%matplotlib inline
import matplotlib.pyplot as plt
s.plot()
plt.show()
```



Creacion de un Objeto DataFrame

```
# Creacion de un DataFrame inicializandolo con un diccionario de objeto Series
Personas = {
    "Peso": pd.Series([70, 68, 60, 70], ["Gabo", "Lucero", "Jessica", "Braulio"]),
    "Altura": pd.Series({
                    "Gabo": 175,
                    "Lucero": 160,
                    "Braulio": 180,
                    "Jessica": 161
                }),
    "Mascotas": pd.Series([4, 7], ["Gabo", "Braulio"])
}
df = pd.DataFrame(Personas)
df
```

	Peso	Altura	Mascotas
Braulio	70	180	7.0
Gabo	70	175	4.0
Jessica	60	161	NaN
Lucero	68	160	NaN

Puede forzarce el DataFrame a que presente unas columnas determinadas y en orden orden determinado

```
Personas = {
    "Peso": pd.Series([70, 68, 60, 70], ["Gabo", "Lucero", "Jessica", "Braulio"]),
    "Altura": pd.Series({
                    "Gabo": 175,
                    "Lucero": 160,
                    "Braulio": 180,
                    "Jessica": 161
                }),
    "Mascotas": pd.Series([4, 7], ["Gabo", "Braulio"])
}
df = pd.DataFrame(
    Personas,
    columns = ["Altura", "Peso"],
    index = ["Gabo", "Braulio", "Lucero"]
)
df
```

	Altura	Peso
Gabo	175	70
Braulio	180	70
Lucero	160	68

```python
# Creacionn de un DataFrame e inicializarlo con una lista de listas de Python
# NOTA: Dedebn inidicarse los indices y columnas por separado
valores = [
    [167, 2, 70],
    [175, 4, 70],
    [160, 2, 68]
]
df = pd.DataFrame(
    valores,
    columns = ["Altura", "Mascotas", "Peso"],
    index = ["Gabo", "Braulio", "Paco"]
)
df
```

	Altura	Mascotas	Peso
Gabo	167	2	70
Braulio	175	4	70
Paco	160	2	68

```python
# 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}}

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

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

Acceso a los elementos de un DataFrame

```python
# Creación de un DataFrame e inicializarlo con 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"])
}

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

	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 de el DataFrame

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

```
Anel       60
Chucho     74
Emilio     72
Jocelin    73
Name: Peso, dtype: int64
```

```
df[["Peso", "Altura"]]
```

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

```
# Pueden combinarse los elementos anteriores con expresiones booleanas.
df["Peso"] > 73
```

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

```
# Pueden combnarse los metodos anteriores con expresiones booleanas y mostrar el DataFrame
df[df["Peso"] > 72]
```

	Peso	Altura	Mascotas
Chucho	74	170	NaN
Jocelin	73	170	9.0

Accediendo a los elementos de las filas del DataFrame

```
df
```

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

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

	Peso	Altura	Mascotas
Chucho	74	170	NaN
Emilio	72	169	NaN

Consulta avanzada de los elementos de un DataFrame

```
df
```

	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

Copiar un DataFrame

```
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
```

	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 el elemento del df_copy no se modifica df.
df_copy = df.copy()
df_copy
```

	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

Modificacion 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 del 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 colulmna 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, devolviendo una copia del DataFrame resultante
df_mod = df_mod.drop(["Hijos"], axis = 1)
df_mod
```

	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

Evaluacion 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"])
}

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

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

```
# Evaluar una funcion sobre una columna de un DataFrame
df.eval("Altura / 2")
```

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

```
max_altura = 165
df.eval("Altura > @max_altura")
```

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

```
# Aplicar una funcion a una columna del DataFrame
def func(x):
    return x + 2

df["Peso"].apply(func)
```

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

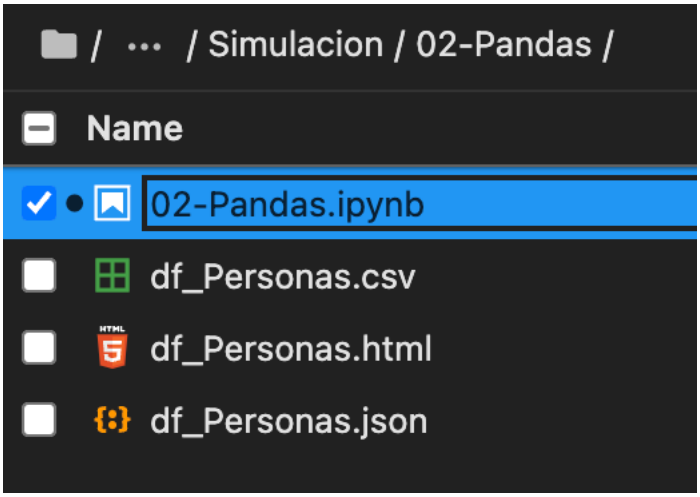
Guardar y cargar el DataFrame

```
# 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"])
}

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

	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")
```




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

	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

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

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

V. Conclusiones:

Con el uso de pandas comprendí el como cargar conjuntos de datos en mi zona de trabajo o libreta para con ellos realojar operaciones en conjunto con Numpy y pues ya juntando estas dos bibliotecas se puede lograr bastantes cosas como un análisis de datos para mostrar lo que decidamos ya que los datos al final de cuenta se pueden mostrar de forma grafica gracias a otra biblioteca llamada matplotlib que hasta el momento solo hemos usado muy poco para graficar cosas sencillas para también es una herramienta muy potente y bastante útil.