GOBIERNO DEL ESTADO DE MÉXICO

MANUAL DE PRÁCTICAS

Nombre de la práctica	Manual de Practicas Pandas en Anaconda			**No.**	**1**
Asignatura:	Simulación 3501	**Carrera:**	Ingeniería en Sistemas Computacionales	**Duración de la práctica (Hrs)**	**8**

I. Competencia(s) específica(s):

Alumno: Raúl Ciriaco Castillo 3501

Github:

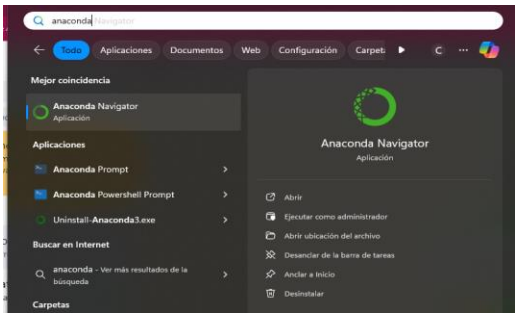
Comentado [T1]: https://github.com/RaulCiriaco

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

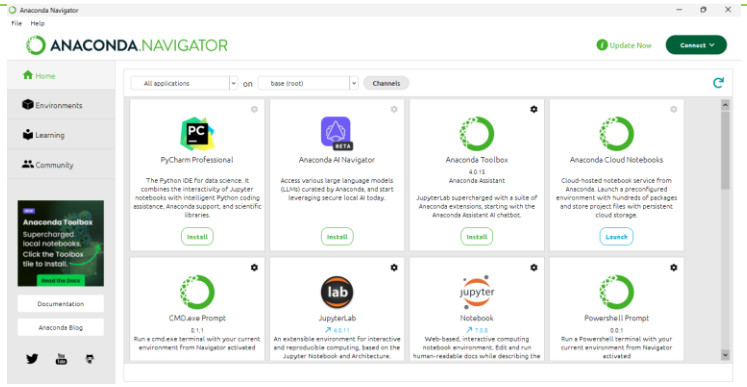
III. Material empleado:

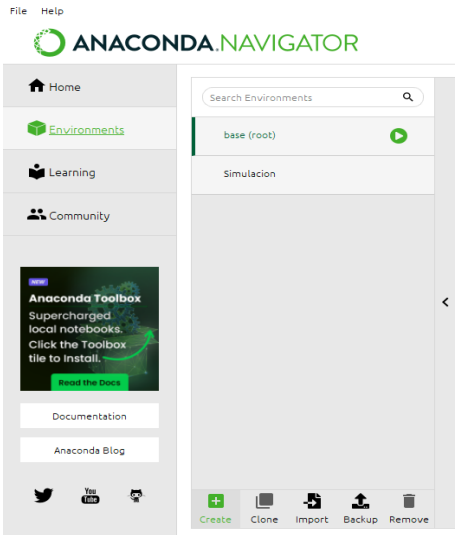
Desarrollo de la práctica:

Como primeros pasos debemos instalar Anaconda Studio en nuestro Equipo de Computo que vamos a utilizar



Una vez que instalamos las paqueterías y todo lo requerido abrimos Anaconda y procedemos a crear nuestro ambiente para trabajar.

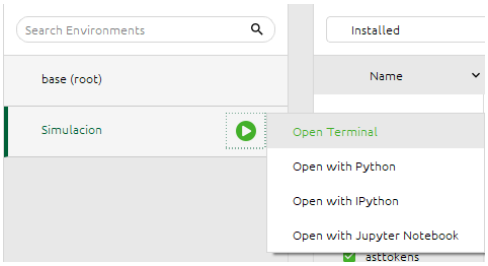




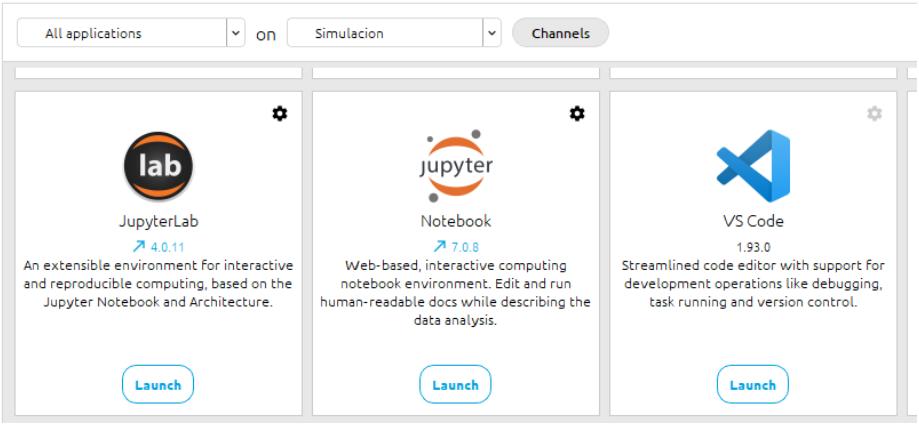
Debemos dar click en el apartado “Environments” y abajo pulsamos el “Create”

Después pulsamos en Open Terminal y vamos a proceder a instalar las paqueterías:

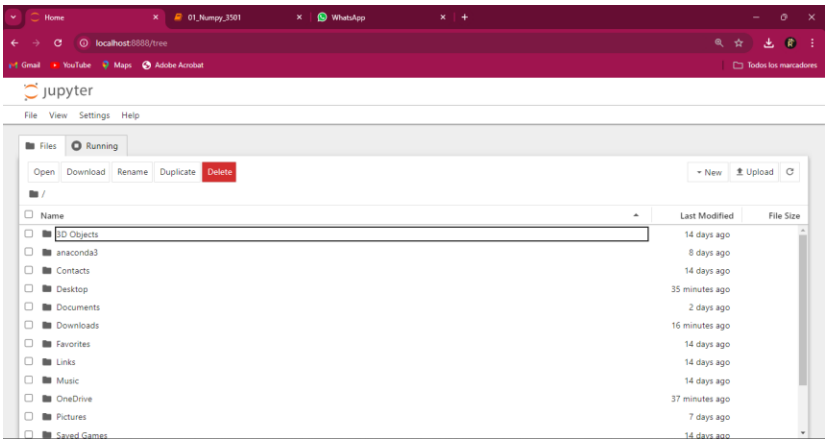
- Numpy
- Pandas
- Matplotlib
- SkLearn



Vamos a descargar el Jupyter Notebook y pulsamos en Launch.

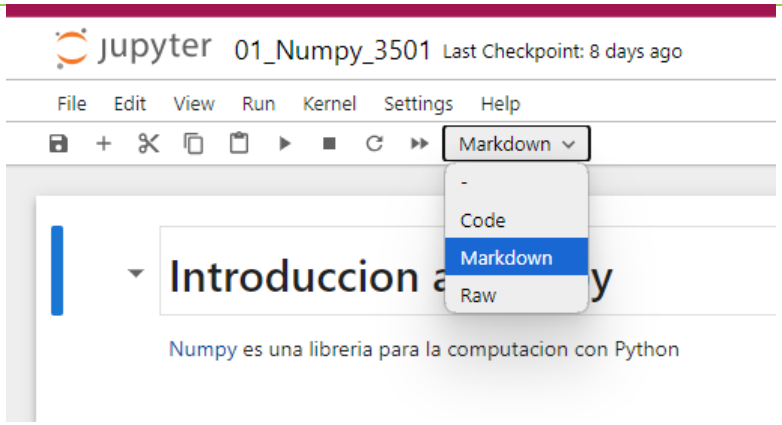


Nos abrirá esta pantalla donde podremos navegar libremente, crear archivos, modificar y eliminarlos.



Crearemos un nuevo archivo para comenzar con la librería **"Pandas".**

En este apartado principal tendremos las herramientas que podemos utilizar en nuestro archivo, tenemos las opciones que se muestran en la imagen



- **Code:** Este apartado nos sirve para poder realizar instrucciones que queramos ejecutarlas después, que cumplan con alguna condición o no, cualquier actividad que tengamos que codificar se verá reflejada en este espacio.
- **Markdown:** Este nos permite la escritura de cualquier información, ya sean formulas o códigos para ejemplificar, no funciona si queremos mostrar o imprimir algún dato o valor.
- **Raw:**

Comenzaremos con una línea que nos permitirá enlazar con la página principal de la librería, en la cual podemos acceder para conocer información acerca de la misma, cualquier duda que tengamos podemos encontrarla en la página oficial.

Introduccion a Pandas

Pandas es una biblioteca que propociona estructuras de datos y herramientas de analisis de alto rendimiento y faciles de usar.

- La estructura de datos proncipal es el DataFrame que puede considerarse comouna tabla 2D en memoria (Comouna hoja de calculo , nombres de columnas y etiquetas de fila).
- Muchas funciones disponibles en excel estan disponibles mediante programacion como crear las tablas dinamicas, clcular colunmas basadas en otras columnas , trazar gradicos, etc.
- Proporciona un alto rendimiento para manipular (unir, dividir,modificar,etc.) grandes volumenes de datos

Para poder trabajar con esta herramienta, recordemos que debemos importar la librería de Pandas

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

A continuación, mostraremos los temas que se abordaran

Estructuras de datos en pandas

La biblioteca pandas, de manera generica, contiene las siguientes estructuras de datos:

- **Series**:Array de una dimencion.
- **DataFrame**: Se corresponde con una tabla de dos dimenciones.
- **Panel**: Simular a un de DataFrames.

Creacion del ojeto 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, "Jesus":170,"jefa":170}
s= pd.Series(altura)
print(s)
```

```
Emilio    169
Anel      145
Jesus     170
jefa      170
dtype: int64
```

[4]:
```
# Creacion de un objeto Series e inicializarlo con algunos elementos de un diccionario de python.
altura = {"Emilio": 169, "Anel":145, "Jesus":170,"jefa":170}
s = pd.Series(altura, index=["jefa","Emilio"])
print(s)
```

```
jefa      170
Emilio    169
dtype: int64
```

[5]:
```
# Creacion de un objeto Series he 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 lo elementos de un Array

Cada elemento en un objeto Series tiene un identificador quese denomina **Index label**.

[6]:
```
# Crear un objeti 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]:
```
# Aceder al tercer elemento del objeto
s["num3"]
```

[7]: np.int64(6)

[8]:
```
# Tambien se puede acceder por posicion.
s[2]
```

```
C:\Users\THINKL13\AppData\Local\Temp\ipykernel_11428\1191963456.py:2: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a f
uture version, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  s[2]
```

[8]: np.int64(6)

```
# loc es la forma estandar de acceder a un elemento de un objeto Series
s.loc["num3"]
```

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

```
# iloc est la forma estandar de acceder a un elemento de un objeto series por posicion
s.iloc[2]
```

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

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

```
num3    6
num4    8
dtype: int64
```

Operaciones aritmeticas con series

[12]:
```
# 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
```

[13]:
```
# los objetos series son similares y compatibles con los arrays de Numpy.
import numpy as np
# ufunc de numpy para sumar los elementos.
np.sum(s)
```

[13]:
```
np.int64(30)
```

[14]:
```
s*2
```

[14]:
```
0     4
1     8
2    12
3    16
4    20
dtype: int64
```

Reprecentacion grafica de un objeto series

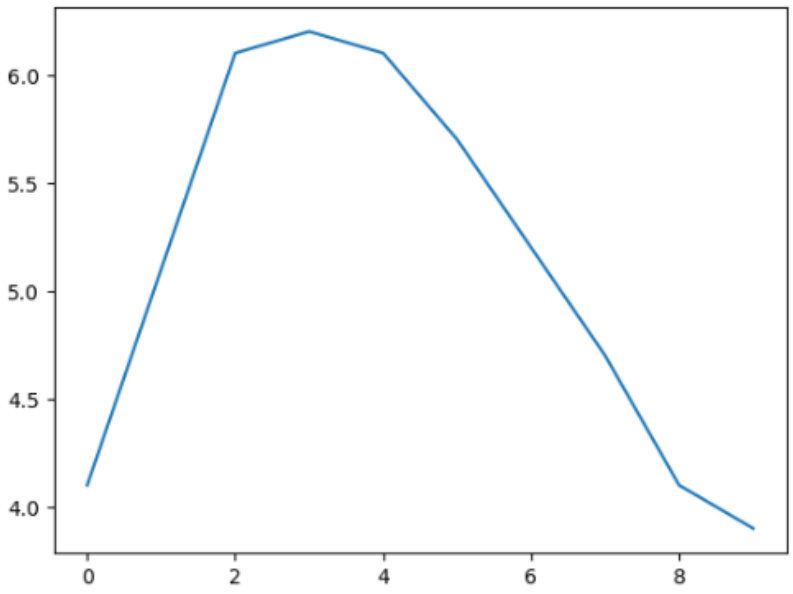
[15]:
```
# Crear un objeto series denominado Temperaturas.
Temperaturas = [4.1 , 5.1 ,6.1,6.2,6.1, 5.7,5.2,4.7,4.1,3.9]
s= pd.Series(Temperaturas, name="Temperaturas")
s
```

[15]:
```
0    4.1
1    5.1
2    6.1
3    6.2
4    6.1
5    5.7
6    5.2
7    4.7
8    4.1
9    3.9
Name: Temperaturas, dtype: float64
```

Recordemos que para poder graficar alguna función debemos de importar también la librería de Matplotlib como se muestra en la siguiente imagen.

```
[16]: # Representacion grafica del objeto series
      %matplotlib inline
      import matplotlib.pyplot as plt

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



Creacion de un objeto DataFrame

[17]:
```
# Creacion de un DataFrame e inicializarlo con un diccionario de objetos series.
Personas = {
    "Peso": pd.Series([72, 60, 74, 73], ["Emilio", "Anel", "Jesus", "jefa"]) ,
    "Altura": pd.Series({"Emilio": 169, "Anel": 145, "Jesus": 170, "jefa": 70}) ,
    "Mascotas": pd.Series([2, 9], ["Anel", "jefa"])
}

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

[17]:

	Peso	Altura	Mascotas
Anel	60	145	2.0
Emilio	72	169	NaN
Jesus	74	170	NaN
jefa	73	70	9.0

Es posible forzar al dataFrame a que presente determinadas columnas en orden determinado

[18]:
```
# Creacion de un DataFrame e inicializarlo con un diccionario de objetos series.
Personas = {
    "Peso": pd.Series([72, 60, 74, 73], ["Emilio", "Anel", "Jesus", "jefa"]) ,
    "Altura": pd.Series({"Emilio": 169, "Anel": 145, "Jesus": 170, "jefa": 70}) ,
    "Mascotas": pd.Series([2, 9], ["Anel", "jefa"])
}

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

[18]:

	Peso	Altura	Mascotas
Anel	60	145	2.0
Emilio	72	169	NaN
Jesus	74	170	NaN
jefa	73	70	9.0

```
[19]: df["Peso"]
```

```
[19]: Anel      60
      Emilio    72
      Jesus     74
      jefa      73
      Name: Peso, dtype: int64
```

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

[20]:

	Peso	Altura
Anel	60	145
Emilio	72	169
Jesus	74	170
jefa	73	70

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

[21]:

	Peso	Altura	Mascotas
Jesus	74	170	NaN
jefa	73	70	9.0

Accediendo a los elementos de la fila de DataFrame.

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

[22]:

	Peso	Altura	Mascotas
Anel	60	145	2.0
Emilio	72	169	NaN
Jesus	74	170	NaN
jefa	73	70	9.0

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

```
[23]: Peso         72.0
      Altura      169.0
      Mascotas      NaN
      Name: Emilio, dtype: float64
```

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

[24]:

	Peso	Altura	Mascotas
Emilio	72	169	NaN
Jesus	74	170	NaN

Consulta avanzada de los elementos de un DataFrame

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

[25]:

	Peso	Altura	Mascotas
Anel	60	145	2.0
Emilio	72	169	NaN
Jesus	74	170	NaN
jefa	73	70	9.0

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

[26]:

	Peso	Altura	Mascotas
Jesus	74	170	NaN

Copiar un DataFrame

[27]:
```
# Crear un DataFrame e inicializarlo con un diccionario de objetos series
# Creacion de un DataFrame e inicializarlo con un diccionario de objetos series.
Personas = {
    "Peso": pd.Series([72, 60, 74, 73], ["Emilio", "Anel", "Jesus", "jefa"]) ,
    "Altura": pd.Series({"Emilio": 169, "Anel": 145, "Jesus": 170, "jefa": 70}) ,
    "Mascotas": pd.Series([2, 9], ["Anel", "jefa"])
}

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

[27]:

	Peso	Altura	Mascotas
Anel	60	145	2.0
Emilio	72	169	NaN
Jesus	74	170	NaN
jefa	73	70	9.0

[29]:
```
## Modificacion de un DataFrame
```

[30]:
```
df["Anio_Nacimiento"]=[2004, 2004, 2004, 2004]
df
```

[30]:

	Peso	Altura	Mascotas	Anio_Nacimiento
Anel	60	145	2.0	2004
Emilio	72	169	NaN	2004
Jesus	74	170	NaN	2004
jefa	73	70	9.0	2004

[31]:
```
# Añadir una nueva columna calculada al DataFrame
df["Edad"] = 2024 - df["Anio_Nacimiento"]
df
```

[31]:

	Peso	Altura	Mascotas	Anio_Nacimiento	Edad
Anel	60	145	2.0	2004	20
Emilio	72	169	NaN	2004	20
Jesus	74	170	NaN	2004	20
jefa	73	70	9.0	2004	20

[32]:
```
# Añadir una nueva columna creando un DataFrame nuevo.
df_mod = df.assign(Hijos = [2, 1, 2, 1])
df_mod
```

[32]:

	Peso	Altura	Mascotas	Anio_Nacimiento	Edad	Hijos

[36]:
```
# Evaluacion de expresiones sobre un DataFrame
Personas = {
    "Peso": pd.Series([72, 60, 74, 73], ["Emilio", "Anel", "Jesus", "jefa"]) ,
    "Altura": pd.Series({"Emilio": 169, "Anel": 145, "Jesus": 170, "jefa": 70}) ,
    "Mascotas": pd.Series([2, 9], ["Anel", "jefa"])
}

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

[36]:

	Peso	Altura	Mascotas
Anel	60	145	2.0
Emilio	72	169	NaN
Jesus	74	170	NaN
jefa	73	70	9.0

[37]:
```
# Evaluar una funcion sobre una columna del DataFrame.
df.eval("Altura / 2")
```

[37]:
```
Anel      72.5
Emilio    84.5
Jesus     85.0
jefa      35.0
Name: Altura, dtype: float64
```

```
copia del DataFrame resultante.
```

[34]:

	Peso	Altura	Mascotas	Anio_Nacimiento	Edad
Anel	60	145	2.0	2004	20
Emilio	72	169	NaN	2004	20
Jesus	74	170	NaN	2004	20
jefa	73	70	9.0	2004	20

[35]:
```
df
```

[35]:

	Peso	Altura	Mascotas	Anio_Nacimiento	Edad
Anel	60	145	2.0	2004	20
Emilio	72	169	NaN	2004	20
Jesus	74	170	NaN	2004	20
jefa	73	70	9.0	2004	20

Guardar y Cargar el DataFrame

```
# Crear un DataFrame e inicializarlo con unn diccionario de Objeto Series.
Personas = {
    "Peso": pd.Series([72, 60, 74, 73], ["Emilio", "Anel", "Jesus", "jefa"]) ,
    "Altura": pd.Series({"Emilio": 169, "Anel": 145, "Jesus": 170, "jefa": 70}) ,
    "Mascotas": pd.Series([2, 9], ["Anel", "jefa"])
}

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

	Peso	Altura	Mascotas
Anel	60	145	2.0
Emilio	72	169	NaN
Jesus	74	170	NaN
jefa	73	70	9.0

[38]:
```
# Evaluar una funcion utilizando una variable local.
max_altura = 170
df.eval("Altura > @max_altura")
```

[38]:
```
Anel      False
Emilio    False
Jesus     False
jefa      False
Name: Altura, dtype: bool
```

[39]:
```
# Evaluar una funcion utilizando una variable local.
max_altura = 165
df.eval("Altura > @max_altura")
```

[39]:
```
Anel      False
Emilio     True
Jesus      True
jefa      False
Name: Altura, dtype: bool
```

[40]:
```
# Aplicar una funcion a una columna del DataFrame
def func(x):
    return x + 2
    df["Peso"].apply(func)
```

[42]:
```
# Guardar el DataFrame como CSV, HTML, JSON.
df.to_csv("df_Personas.csv")
df.to_html("df_Personas.html")
df.to_json("df_Personas.json")
```

[43]:
```
# Cargar el DataFrame en Jupyter
df2 = pd.read_csv("df_Personas.csv")
```

[44]:
```
df2
```

[44]:

	Unnamed: 0	Peso	Altura	Mascotas
0	Anel	60	145	2.0
1	Emilio	72	169	NaN
2	Jesus	74	170	NaN
3	jefa	73	70	9.0

[45]:
```
# Cargar el DataFrame con la primera columna correctamente asignada
df2 = pd.read_csv("df_Personas.csv", index_col=0)
df2
```

[45]:

	Peso	Altura	Mascotas
Anel	60	145	2.0
Emilio	72	169	NaN
Jesus	74	170	NaN
jefa	73	70	9.0

IV. Conclusiones:

En esta práctica con Pandas, me di cuenta de lo poderosa que es esta herramienta para trabajar con datos. Aprender a manejar DataFrames, filtrarlos, ordenarlos y transformar la información me ha permitido ver lo eficiente que es para analizar grandes volúmenes de datos. Además, fue interesante ver cómo puedo leer y escribir archivos en formatos como CSV o JSON sin complicaciones.

Al terminar la práctica, me quedó claro que Pandas me facilita muchísimo el análisis de datos, y sé que esta habilidad es esencial si quiero aplicar lo que he aprendido en proyectos más grandes o incluso en el ámbito laboral.

Estas son algunas de las ventajas sobre esta librería y para que nos ayuda en el entorno laboral

- Manejo eficiente de datos: Los estudiantes podrán leer, manipular y analizar grandes cantidades de datos, una habilidad crucial en diversas disciplinas como la ciencia de datos, economía y administración.

- Versatilidad: Al aprender a guardar y cargar datos en diferentes formatos (CSV, HTML, JSON), podrán adaptarse a distintos requerimientos en proyectos o trabajos profesionales.

- Automatización de procesos: El uso de pandas permite automatizar tareas repetitivas relacionadas con el manejo de datos, ahorrando tiempo y minimizando errores.