Nombre de la práctica	Pandas			No.	2
Asignatura:	Simulación	Carrera:	ISIC	Duración de la práctica (Hrs)	8

Nombre del Alumno: MARIA LUCERO RODEA MARTINEZ
Grupo: 3502

I. Competencia(s) específica(s):

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

Aula

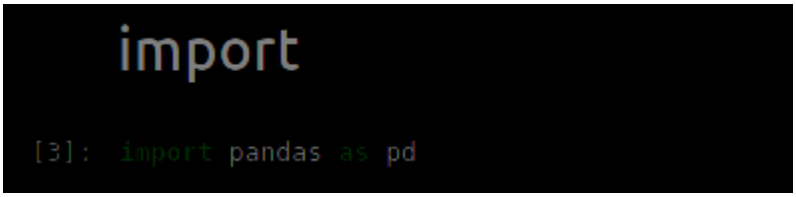
III. Material empleado:
Maquina laptop
phyton

IV. Desarrollo de la práctica:

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 dato principal de datos es el dataframe que puede considerarse como una tabla 2D en memoria como una hoja de calculo con nombres de columnas y etiquetas
- Muchas funciones disponibles en Excel, están disponibles en programación, como crear tablas dinámicas, calcular columnas basadas en otras columnas, trazar gráficos, etc.
- Proporciona un alto rendimiento de manipular (unir, dividir, modificar, etc.) grandes conjuntos de datos,

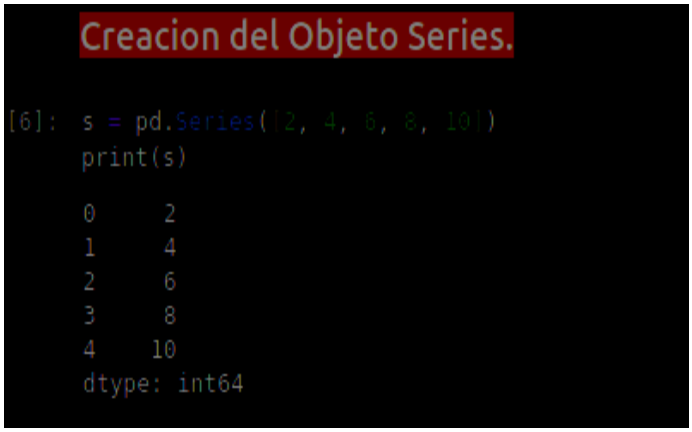


Estructuras de Datos de Pandas.

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

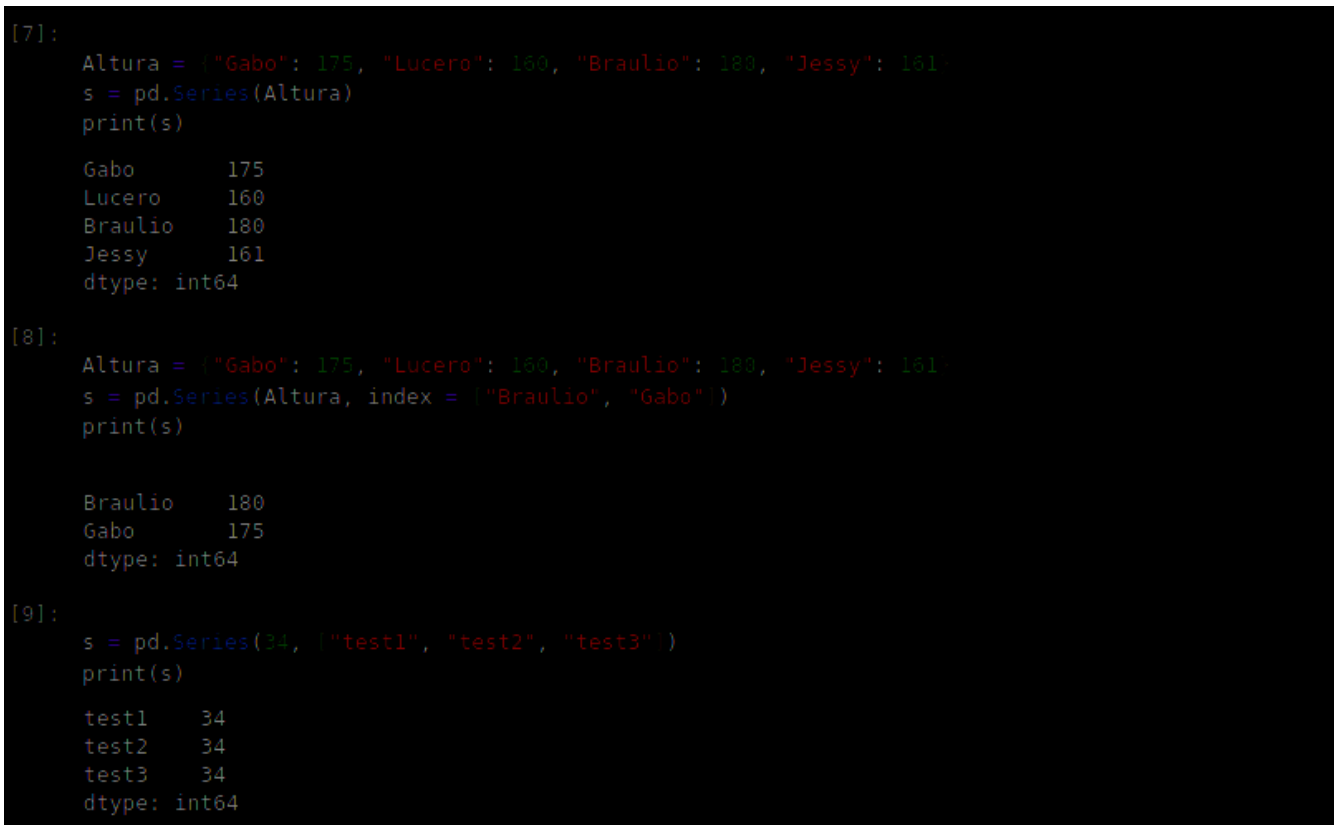
- **Series**: Array de una dimensión.
- **DataFrame**: Corresponde a una tabla de dos dimensiones.
- **Panel**: Similar a un diccionario de DataFrame.

Creación del Objeto Series.



Acceso a los elementos de un objeto Series.

Cada elemento en un objeto Series tiene un identificador único que se denomina **índex label**.



Acceso a los elementos de un objeto Series.

Cada elemento en un objeto Series tiene un identificador único que se denomina **índex label**.

```
[11]:
      s = pd.Series( 2, 4, 6, 8 , index = ["Num1", "Num2", "Num3", "Num4" )
      print(s)

      Num1    2
      Num2    4
      Num3    6
      Num4    8
      dtype: int64

[12]:
      s["Num3"]

[12]: 6

[13]:
      s[2]

      /tmp/ipykernel_5413/2315648266.py:2: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future 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]
[13]: 6

[14]:
      s.loc["Num3"]

[14]: 6
```

```
[15]:
       s.iloc[2]

[15]: 6

[16]:
       s.iloc[2:4]

[16]: Num3    6
       Num4    8
       dtype: int64
```

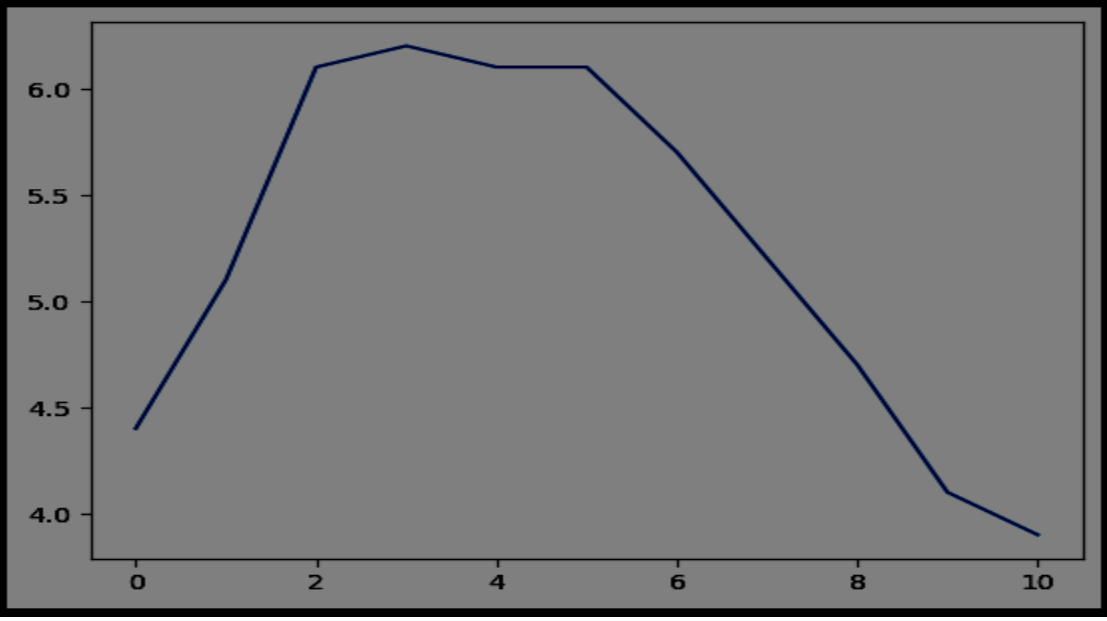
Operaciones Aritméticas con Objetos Series.

```
[18]:
      s = pd.Series([2, 4, 6, 8, 10])
      print(s)

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

[19]:
      import numpy as np

      np.sum(s)

[19]: 30

[20]:

      s * 2

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

Representación grafica de un Objeto Series

```
[22]: 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

[22]: 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
```

```
[23]: %matplotlib inline
      import matplotlib.pyplot as plt
      s.plot()
      plt.show()
```



Creación de un Objeto DataFrame

```
[25]:
Personas = {
    "Peso": pd.Series( 70, 60, 60, 70], "Gabo", "Lucero", "Jessy", "Braulio"]),
    "Altura": pd.Series( "Gabo": 175, "Lucero": 160, "Jessy": 161, "Braulio": 180}),
    "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
Jessy	60	161	NaN
Lucero	60	160	NaN

Puede forzarse al DataFrame que presente que presente unas columnas determinadas y en un orden determinado.

```
[27]:
Personas = {
    "Peso": pd.Series( 70, 60, 60, 70], "Gabo", "Lucero", "Jessy", "Braulio"]),
    "Altura": pd.Series( "Gabo": 175, "Lucero": 160, "Jessy": 161, "Braulio": 180}),
    "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	60

```
[28]:
valores = [
    [167, 2, 70],
    [180, 4, 68],
    [175, 6, 70]
]
df = pd.DataFrame(
    valores,
    columns = ["Altura", "Mascotas", "Peso"],
    index = ["Braulio", "Jessy", "Lucero"])
df
```

[28]:

	Altura	Mascotas	Peso
Braulio	167	2	70
Jessy	180	4	68
Lucero	175	6	70

```
[29]:
Personas = {
    "Peso": {"Gabo": 70, "Jessy": 60, "Braulio": 70, "Lucero": 70},
    "Altura": {"Gabo": 175, "Jessy": 167, "Braulio": 180, "Lucero": 175}}

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

[29]:

	Peso	Altura
Gabo	70	175
Jessy	60	167
Braulio	70	180
Lucero	70	175

Acceso a los elementos de un DataFrame

```
[33]: df["Peso"]

[33]: Anel       60
      Chucho     74
      Emilio     72
      Jocelin    73
      Name: Peso, dtype: int64

[34]: df[["Peso", "Altura"]]

[34]:          Peso  Altura
         Anel    60     145
       Chucho    74     170
        Emilio   72     169
       Jocelin   73     170

[35]:
      df["Peso"] > 73

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

```
[36]:
      df[df["Peso"] > 72]

[36]:          Peso  Altura  Mascotas
       Chucho    74     170       NaN
       Jocelin   73     170       9.0
```

Accediendo a los elemtos de las filas del DataFrame.

```
[38]:
      df

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

[39]: df.loc["Emilio"]

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

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

[40]:          Peso  Altura  Mascotas
       Chucho    74     170       NaN
        Emilio   72     169       NaN
```

Consulta avanzada de los elementos de un DataFrame.

```
[42]:
      df

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

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

[43]:          Peso  Altura  Mascotas
       Chucho    74     170       NaN
```

Copear un DataFrame.

```
[45]:
      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

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

[80]:

      df_copy = df.copy()
```

Modificación de un DataFrame.

```
[82]:
df["Anio_Nac"] = [2004, 2004, 2004, 2004]
df

[83]:
          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

[85]:
df["Edad"] = 2024- df["Anio_Nac"]
df

[85]:
          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
```

```
[85]:
      df["Edad"] = 2024- df["Anio_Nac"]
      df

[85]:          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


[87]:
      df_mod = df.assign(Hijos = [2, 1, 2, 1])

[89]: df_mod

[89]:          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
```

```
[91]: df

[91]:          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

[93]:
      del df["Peso"]

[95]: df

[95]:          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
```

```
[97]:
      df_mod = df_mod.drop( "Hijos", axis=1)
      df_mod

[97]:          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


[99]:  df

[99]:          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.

```
[102]:
        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
```

[102]:

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

```
[104]:
        df.eval("Altura / 2")

[104]:  Anel       72.5
        Chucho     85.0
        Emilio     84.5
        Jocelin    85.0
        dtype: float64
```

```
[106]:
        max_altura = 165
        df.eval("Altura > @max_altura")

[106]:  Anel       False
        Chucho      True
        Emilio      True
        Jocelin     True
        dtype: bool

[108]:
        def func(x):
            return x + 2

        df["Peso"].apply(func)

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

Guardar y argar el DataFrame.

```
[111]:
        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
```

[111]:

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

```
[113]:
        df.to_csv("df_Personas.csv")
        df.to_html("df_Personas.html")
        df.to_json("df_Personas.json")

[115]:
        df2 = pd.read_csv("df_Personas.csv")
```

```
[117]: df2

[117]:    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


[119]:
       df2 = pd.read_csv("df_Personas.csv", index_col=0)
       df2

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

V. Conclusiones: