PECIER

Comité Nacional Peruano de la CIER

Curso Virtual

Python para el Análisis de Datos y la Automatización en el Sector Eléctrico

27, 29 y 31 de Octubre, 03, 05, 07, 10 y 12 de Noviembre 2025.

MARVIN COTO – FACILITADOR
E-mail: mcotoj@gmail.com

PECIER

Comité Nacional Peruano de la CIER

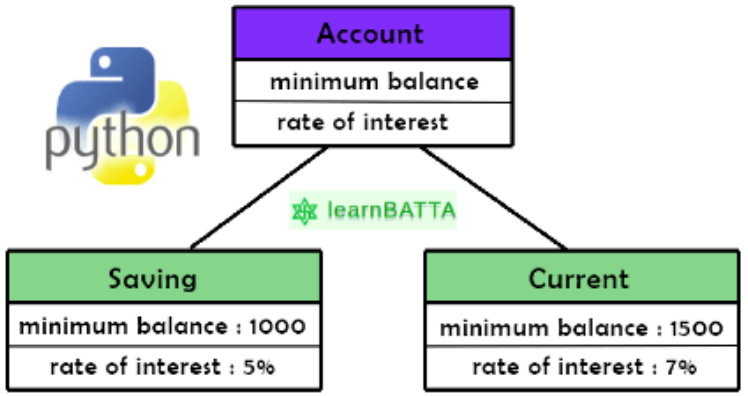
SESIÓN 6
Parte 1

PECIER
Comité Nacional Peruano de la CIER

Introducción al uso de clases en Python para análisis de datos con Pandas

Objetivo:

Aprender a crear y usar **clases** en Python.



understanding of class in python

PECIER
Comité Nacional Peruano de la CIER

Introducción

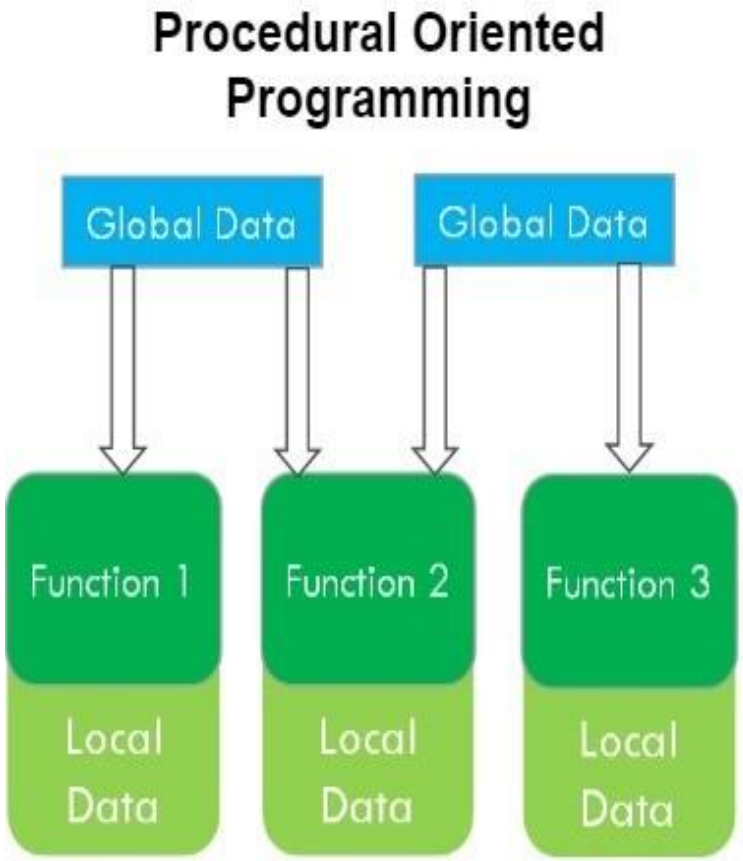
En análisis de datos, a menudo tenemos:

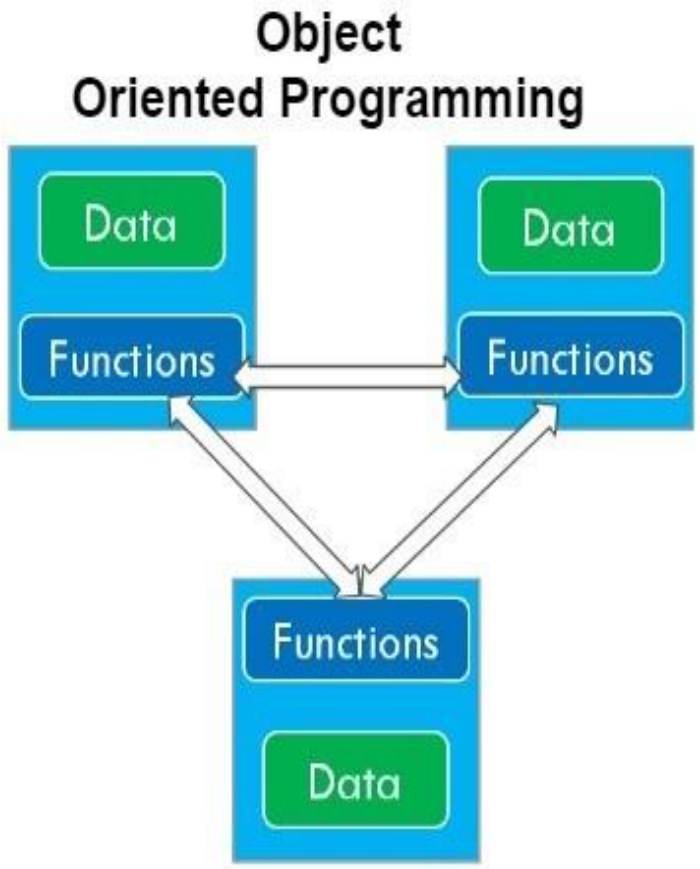
- Varias hojas de datos (por semana, por zona, por mes).
- Procesos repetitivos (calcular promedios, detectar valores fuera de rango, etc.).

Las clases nos permiten organizar el código, reutilizar análisis y almacenar el estado del DataFrame, facilitando el trabajo con múltiples conjuntos de datos.
Son la manera en que se realiza la **programación orientada a objetos.**

PECIER
Comité Nacional Peruano de la CIER

Introducción





PECIER
Comité Nacional Peruano de la CIER

Introducción breve

La programación orientada a objetos es un paradigma de programación, que se basa en:

- Centrar el programa en los datos, antes que en la lógica que usa esos datos.
- Agrupar datos y funciones (lógica) en objetos que pertenecen a clases (tipos definidos por el programador)
- Los objetos tienden a modeliar entidades del mundo real.

PECIER
Comité Nacional Peruano de la CIER

Preparemos los datos:

```
python import pandas as pd
```

Datos simulados

```
df_semana1 = pd.DataFrame({ 'Hora': ['06:00', '07:00', '08:00'], 'Fase R (V)': [220.5, 221.3, 219.8],
'Fase S (V)': [220.2, 219.8, 220.5], 'Fase T (V)': [219.1, 220.0, 218.9], }).set_index('Hora')

df_semana2 = pd.DataFrame({ 'Hora': ['06:00', '07:00', '08:00'], 'Fase R (V)': [222.0, 221.7, 223.2],
'Fase S (V)': [222.1, 221.6, 223.3], 'Fase T (V)': [221.9, 221.8, 223.0], }).set_index('Hora')
```

PECIER
Comité Nacional Peruano de la CIER

Datos simulados:

Las clases se definen con la palabra class, y dentro de ella se escriben las funciones.

`class :` instrucción instrucción ...

Lo más esencial de una clase son las funciones miembro.

Puede aparecer la función **init** especial; es el constructor, que se llama cuando se crea una nueva instancia de la clase.

Las instancias de una clase son los objetos, creados al llamar la clase.

Se pueden definir múltiples instancias para una misma clase (esa es la idea :-))

PECIER
Comité Nacional Peruano de la CIER

```
import matplotlib.pyplot as plt # Las bibliotecas se llaman fuera de la clase
(normal)

class AnalizadorDeVoltaje:
    def __init__(self, df): # Esto se usa tal cual en las clases
        self.df = df        # "Guarda" el objeto df para todas las funciones.

    def voltaje_promedio(self):
        return self.df[['Fase R (V)', 'Fase S (V)', 'Fase T (V)']].mean(axis=1)

    def detectar_anomalias(self, umbral):
        promedio = self.voltaje_promedio()
        return self.df[promedio > umbral]

    def maximos_por_fase(self):
        return self.df.max()
```

PECIER
Comité Nacional Peruano de la CIER

```
def resumen_estadistico(self):
    return self.df.describe()

def horas_normales(self, umbral_bajo, umbral_alto):
    promedio = self.voltaje_promedio()
    return self.df[promedio.between(umbral_bajo, umbral_alto)]

def graficar_voltaje(self):
    promedio = self.voltaje_promedio()
    ax = self.df.plot(marker='o')
    promedio.plot(ax=ax, style='--', color='black', label='Promedio')
    plt.title('Voltajes por hora')
    plt.ylabel('Voltaje (V)')
    plt.grid(True)
    plt.legend()
    plt.show()
```

PECIER
Comité Nacional Peruano de la CIER

¿Qué es self?

self es una referencia al propio objeto que se está creando o utilizando.

- Cuando escribimos self.df = df, estámos guardando el DataFrame dentro del objeto, para que lo puedas usar después en otros métodos.
- Todos los métodos dentro de la clase deben recibir self como primer argumento, aunque al llamarlos noa lo veamos (lo agrega Python automáticamente).
- Analogía: Si pensamos en una hoja de cálculo con varias pestañas, self.df sería la pestaña activa.

PECIER
Comité Nacional Peruano de la CIER

¿Cómo las usaremos?

Las llamadas son semejantes a las funciones:

```
analizador1 = AnalizadorDeVoltaje(df_semana1)
analizador2 = AnalizadorDeVoltaje(df_semana2)

print("Semana 1 - Promedios")
print(analizador1.voltaje_promedio())

print("\nSemana 2 - Anomalías (> 221.0 V):")
print(analizador2.detectar_anomalias(221.0))

print("\nSemana 2 - Máximos por fase:")
print(analizador2.maximos_por_fase())
```

PECIER
Comité Nacional Peruano de la CIER

¿Cómo las usaremos?

Las llamadas son semejantes a las funciones:

```
Semana – Promedios
Hora
06:00    219.933333
07:00    220.366667
08:00    219.733333
Dtype: float 64

Semana 2 – Anomalías (> 221.0 V):
       Fase R (V) Fase S (V) Fase T (V)
Hora
06:00    222.0    222.1    221.9
07:00    221.7    221.6    221.8
08:00    223.2   223.3    223.0

Semana 2 – Máximos por fase:M
Fase R (V)    223.2
Fase S (V)    223.3
Fase T (V)    223.0
Dtype: float64
```

PECIER
Comité Nacional Peruano de la CIER

Para finalizar esta breve introducción:

- Las clases permiten organizar el código en torno a una estructura lógica, como un conjunto de mediciones.
- Son útiles cuando se tienen múltiples datasets similares (por semana, zona, etc.).
- Permiten mantener un "estado" interno (en este caso, el DataFrame), y construir métodos que lo manipulen sin repetir argumentos.
- Sin embargo, todo esto podría realizarse sin clases:

PECIER
Comité Nacional Peruano de la CIER

Para finalizar esta breve introducción:

```
def voltaje_promedio(df):
    return df[['Fase R (V)', 'Fase S (V)', 'Fase T (V)']].mean(axis=1)

def detectar_anomalias(df, umbral):
    return df[voltaje_promedio(df) > umbral]

def horas_normales(df, umbral_bajo, umbral_alto):
    promedio = voltaje_promedio(df)
    return df[promedio.between(umbral_bajo, umbral_alto)]

def maximos_por_fase(df):
    return df.max()

def graficar_voltaje(df):
    promedio = voltaje_promedio(df)
    ax = df.plot(marker='o')
    promedio.plot(ax=ax, style='--', color='black', label='Promedio')
    plt.title('Voltajes por hora')
    plt.ylabel('Voltaje (V)')
    plt.grid(True)
    plt.legend()
    plt.show()
```

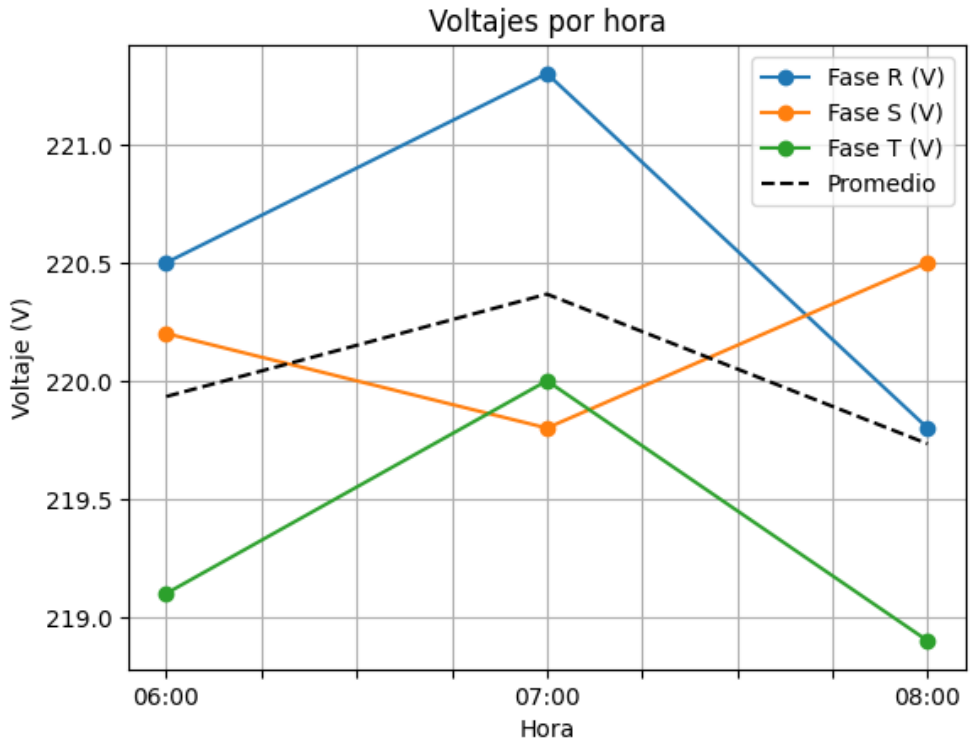
PECIER
Comité Nacional Peruano de la CIER

Para finalizar esta breve introducción:

Y cada vez:

```
print(voltaje_promedio(df_semana1))
print(detectar_anomalias(df_semana2, 221.0))
graficar_voltaje(df_semana1)
```

```
Hora
06:00    219.933333
07:00    220.366667
08:00    219.733333
dtype: float64
       Fase R (V)  Fase S (V)  Fase T (V)
Hora
06:00       222.0       222.1       221.9
07:00       221.7       221.6       221.8
08:00       223.2       223.3       223.0
```



PECIER
Comité Nacional Peruano de la CIER

Ejemplos y ejercicios

PECIER_dia6_clases

colab.research.google.com

PECIER

Comité Nacional Peruano de la CIER

E-mail: pecier@cier.org