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 2
Parte 1

PECIER
Comité Nacional Peruano de la CIER

Recordatorio de la Sesión 1

En la primera sesión se comentaron aspectos muy importante sobre Python:

- Su origen.
- Su filosofía: simplicidad, fácil lectura.
- Sus características destacadas: Manejo simple de variables, indentación.

PECIER
Comité Nacional Peruano de la CIER

Recordatorio de la Sesión 1

- Existen varias formas de ejecutar el código (Jupyter/Colab, línea de comandos, en línea).

PECIER
Comité Nacional Peruano de la CIER

Estructuras de datos

Imaginemos que debemos trabajar con todas las lecturas de un sensor que lee cada 15 minutos, durante los últimos 5 años.

¿Usamos una variable por lectura?

```
Lectura_1_enero_2021_00_00am = 125.2
Lectura_1_enero_2021_00_15am = 122.2
Lectura_1_enero_2021_00_30am = 122.6
...
```

PECIER
Comité Nacional Peruano de la CIER

Estructuras de datos

O trabajar con la lista de personas que laboran en una empresa.

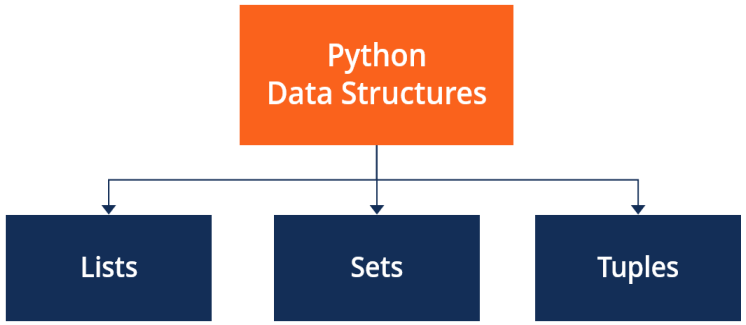
¿Usamos una variable por presona?

```
persona1 = José Rodríguez
edad_persona1=35
departamento_persona1=RRHH
...
```

PECIER
Comité Nacional Peruano de la CIER

Estructuras de datos

Las estructuras de datos almacenan muchos valores en una sola variable.



PECIER
Comité Nacional Peruano de la CIER

Estructuras de datos

En Python, las estructuras de datos permiten almacenar y organizar múltiples valores dentro de un solo objeto.

Las principales son:

- **Listas**: Son colecciones ordenadas y modificables.
- **Tuplas**: Colecciones ordenadas e inmutables.
- **Diccionarios**: Colecciones no ordenadas de pares clave-valor.
- **Conjuntos**: Colecciones no ordenadas de elementos únicos.

PECIER
Comité Nacional Peruano de la CIER

Estructuras de datos

Veremos cada una:

Listas
Las listas son colecciones ordenadas y modificables de elementos. Se definen con paréntesis cuadrados []. Aptas para manejar listas de valores, como series de tiempo.

PECIER
Comité Nacional Peruano de la CIER

Listas

Crear y Acceder a Listas

Para crear una lista se le debe poner nombre, y establecer los valores iniciales con paréntesis cuadrados.

Ejemplo:

```
# Crear una lista de valores de corriente
corrientes = [5, 10, 15, 20]

# Acceder a elementos
print("Primera corriente:", corrientes[0])
print("Última corriente:", corrientes[-1])
```

Salida esperada:

```
Primera corriente: 5
Última corriente: 20
```

PECIER
Comité Nacional Peruano de la CIER

Listas

```
# Crear una lista de valores de corriente
corrientes = [5, 10, 15, 20]

# Acceder a elementos
print("Primera corriente:", corrientes[0])
print("Última corriente:", corrientes[-1])
```

Importante: Observemos la forma en que se indexan los elementos de la lista.

Listas

PECIER
Comité Nacional Peruano de la CIER

Modificar Listas

Las listas son mutables, por lo que se pueden cambiar los elementos, aumentarlas o disminuirlas.

Ejemplo:

```
# Modificar un elemento
corrientes[1] = 12
print("Lista modificada:", corrientes)

# Agregar un elemento
corrientes.append(25)
print("Lista con nuevo elemento:", corrientes)
```

Salida esperada:

```
Lista modificada: [5, 12, 15, 20]
Lista con nuevo elemento: [5, 12, 15, 20, 25]
```

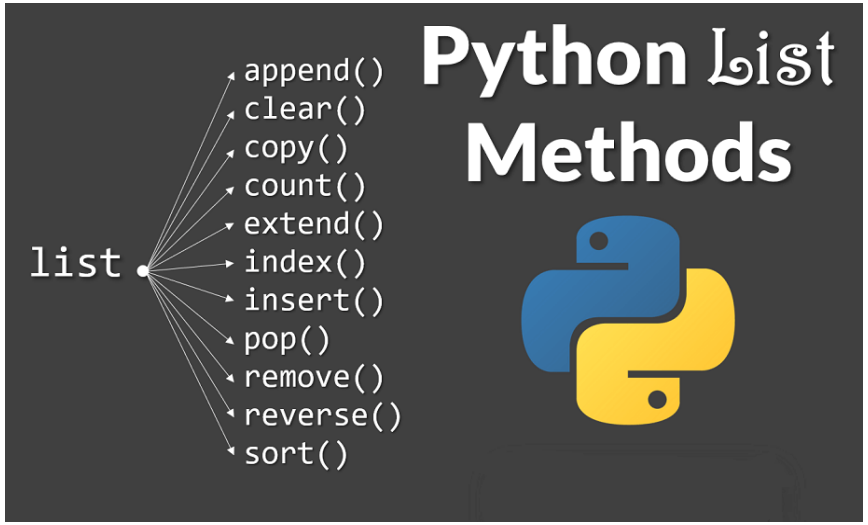
Listas

PECIER
Comité Nacional Peruano de la CIER

Métodos útiles de Listas

Otras opraciones importantes a considerar:

- append(): Agrega un elemento al final.
- remove(): Elimina un elemento específico.
- sort(): Ordena la lista.



Listas

PECIER
Comité Nacional Peruano de la CIER

Ejemplo:

```
# Ordenar la lista de corrientes
# Crear una lista de valores de corriente
corrientes = [5, 10, 1, 20]
corrientes.sort()
print("Lista ordenada:", corrientes)

# También se puede considerar:
corrientes.sort(reverse=True)
print("Lista ordenada en orden inverso:", corrientes)

# Eliminar un elemento
# Eliminar el elemento en el índice 2 (el valor 15)
corrientes.pop(2)
print("Lista después de eliminar el índice 2:", corrientes)
```

Salida esperada:

```
Lista ordenada: [5, 12, 15, 20, 25]
Lista sin el segundo elemento: [5, 12, 20, 25]
```

Listas

PECIER
Comité Nacional Peruano de la CIER

Observe que el uso de .sort() sobreescribe la lista, lo cual no es reversible.

Puede ser mejor idea realizar una copia de la lista antes de ordenarla, o utilizar `.sorted()`, lo cual solo despliega el resultado.

La lista se puede copiar con:

```
corrientes_original = corrientes[:]
```

Tuplas

PECIER
Comité Nacional Peruano de la CIER

Las tuplas son colecciones ordenadas e inmutables (no se pueden modificar). Se definen con paréntesis ( ).

List vs Tuples

[1,2] (1,2)

Tuplas

PECIER
Comité Nacional Peruano de la CIER

Crear y Acceder a Tuplas

Ejemplo:

```
# Crear una tupla de valores de temperatura
temperaturas = (75, 80, 85, 90)

# Acceder a elementos
print("Primera temperatura:", temperaturas[0])
print("Última temperatura:", temperaturas[-1])
```

Salida esperada:

```
Primera temperatura: 75
Última temperatura: 90
```

Tuplas

PECIER
Comité Nacional Peruano de la CIER

Inmutabilidad de las Tuplas

Las tuplas no pueden modificarse después de su creación.

Ejemplo:

```
# Intentar modificar una tupla (generará un error)
temperaturas = (75, 80, 85, 90)

temperaturas[1] = 82
```

Salida esperada:

```
Error: 'tuple' object does not support item assignment
```

Tuplas

PECIER
Comité Nacional Peruano de la CIER

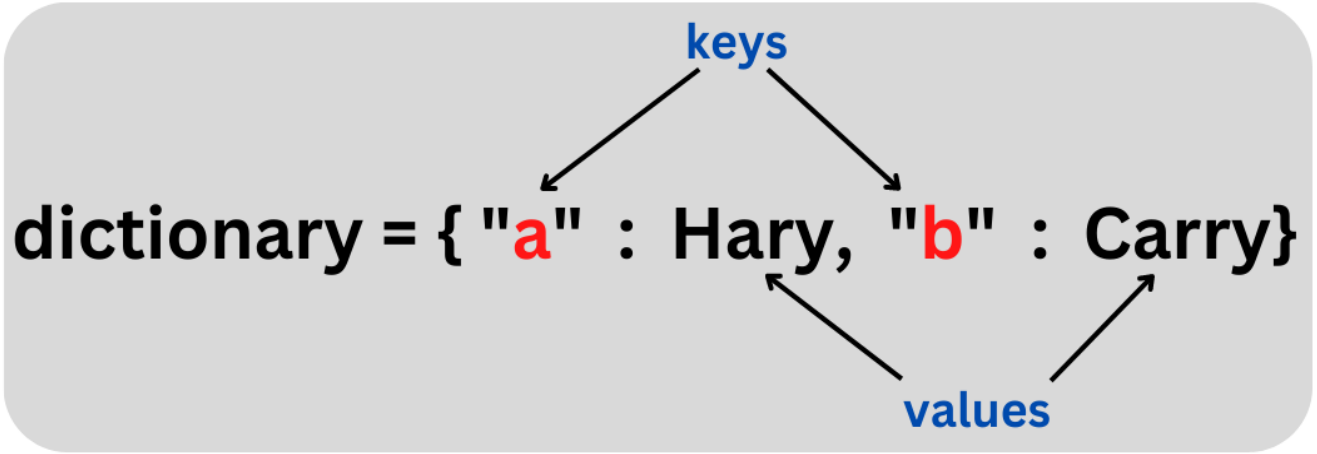
Sin embargo, existen operaciones útiles:

```
len(tupla) – longitud

min(tupla) – mínimo

max(tupla) – máximo

sum(tupla) – suma (si son numéricos)

tuple(lista) – convertir lista a tupla
```

PECIER
Comité Nacional Peruano de la CIER

Diccionarios

Los diccionarios son colecciones no ordenadas de pares clave-valor. Se definen con llaves {}.



PECIER
Comité Nacional Peruano de la CIER

Diccionarios

Crear y Acceder a Diccionarios

Ejemplo:

```
# Crear un diccionario de componentes eléctricos
componentes = {
    "resistencia": "10 Ohm",
    "capacitor": "100 uF",
    "inductor": "50 mH"
}
# Acceder a un valor
print("Valor del capacitor:", componentes["capacitor"])
```

Salida esperada:

```
Valor del capacitor: 100 µF
```

PECIER
Comité Nacional Peruano de la CIER

Diccionarios

Modificar Diccionarios

Los diccionarios son mutables, por lo que puedes agregar, modificar o eliminar pares clave-valor.

Ejemplo:

```
# Crear un diccionario de componentes eléctricos

componentes = { "resistencia": "10 Ω", "capacitor": "100 µF", "inductor":
"50 mH" }

# Agregar un nuevo componente

componentes["diodo"] = "1N4007"
print("Diccionario actualizado:", componentes)
# Modificar un valor
componentes["resistencia"] = "20 Ω"
print("Diccionario modificado:", componentes)
```

PECIER
Comité Nacional Peruano de la CIER

Diccionarios

Salida esperada:

```
Diccionario actualizado: {'resistencia': '10 Ω',
'capacitor': '100 µF', 'inductor': '50 mH', 'diodo':
'1N4007'}

Diccionario modificado: {'resistencia': '20 Ω',
'capacitor': '100 µF', 'inductor': '50 mH', 'diodo':
'1N4007'}
```

PECIER
Comité Nacional Peruano de la CIER

Conjuntos

Los conjuntos son colecciones no ordenadas de elementos únicos. Se definen con llaves {}.

PECIER
Comité Nacional Peruano de la CIER

Conjuntos

Crear y Operar con Conjuntos

Ejemplo:

```
# Crear un conjunto de valores de corriente
corrientes = {5, 10, 15, 20} # Agregar un elemento corrientes.add(25)
print("Conjunto con nuevo elemento:", corrientes)
# Eliminar un elemento
corrientes.remove(10)
print("Conjunto sin el 10:", corrientes)
```

Salida esperada:

```
Conjunto con nuevo elemento: {5, 10, 15, 20, 25} Conjunto sin el 10: {5,
15, 20, 25}
```

PECIER
Comité Nacional Peruano de la CIER

Conjuntos

Operaciones con Conjuntos

- **Unión**: Combina dos conjuntos.
- **Intersección**: Encuentra elementos comunes.

Ejemplo:

```
# Operaciones con conjuntos
conjunto1 = {5, 10, 15}
conjunto2 = {10, 15, 20}

print("Unión:", conjunto1.union(conjunto2))
print("Intersección:",
conjunto1.intersection(conjunto2))
```

Salida esperada:

```
Unión: {5, 10, 15, 20} Intersección: {10, 15}
```

PECIER
Comité Nacional Peruano de la CIER

Conjuntos

Conversión de tipos

Los conjuntos pueden transformarse en listas:

Ejemplo:

```
# Conjunto de valores de corriente corrientes = {5, 10,
15, 20} # Convertir a lista lista_corrientes =
list(corrientes) print("Tipo:", type(lista_corrientes))
# <class 'list'> print("Lista:", lista_corrientes)
```

PECIER
Comité Nacional Peruano de la CIER

Preguntas de repaso

1.¿Cuál es la salida del siguiente código?

```
corrientes = [5, 10, 15, 20]
corrientes[1] = 12
print("Lista modificada:", corrientes)
```

PECIER
Comité Nacional Peruano de la CIER

Conjuntos

Preguntas de repaso

2.¿Cuál es la salida del siguiente código?

```
corrientes = [5, 10, 15, 20]

corrientes.append(25)

print("Lista modificada:", corrientes)
```

PECIER
Comité Nacional Peruano de la CIER

Conjuntos

Preguntas de repaso

3.¿Cuál inconveniente tiene la ejecución del siguiente código?

```
corrientes = [5, 10, 1, 20] corrientes.sort()
```

PECIER
Comité Nacional Peruano de la CIER

Conjuntos

Preguntas de repaso

4.¿Cuál es el valor de salida del siguiente código?

```
# Crear una lista de valores de corriente

corrientes = [0, 10, 15, 5, 3] # Acceder a
elementos

print("Corriente:", corrientes[1])
```

PECIER
Comité Nacional Peruano de la CIER

Ejemplos y ejercicios

Google colab

PECIER_dia2_parte1

colab.research.google.com

PECIER

Comité Nacional Peruano de la CIER

SESIÓN 2
Parte 2

PECIER
Comité Nacional Peruano de la CIER

Bucles Anidados y Control de Flujo Avanzado

- Los bucles (for, while) nos permiten realizar acciones repetitivas sobre elementos como listas o conjuntos de valores, lo cual hace mucho más eficiente el código.
- Dado que existen otros elementos tipo bidimensional (como matrices), conviene recorrerarlas como un doble bucle: Recorrer cada columna, y luego cada elemento de la columna (o vicecersa).

Introducción Teórica

PECIER
Comité Nacional Peruano de la CIER

Conceptos Clave

Los bucles anidados permiten:

- Procesar datos multidimensionales (matrices)
- Generar combinaciones complejas (ej: configuraciones de circuitos)
- Simular sistemas iterativos (ej: evolución de temperaturas)

Introducción Teórica

PECIER
Comité Nacional Peruano de la CIER

Ejemplo:

```
# Matriz de covarianzas
cov = [
[0.75, 0.78, 0.80],
[0.77, 0.82, 0.79],
[0.85, 0.88, 0.90]
]
for i, fila in enumerate(cov):
    for j, co in enumerate(fila):
        if co > 0.85:
            print(f" Alerta en posición ({i},{j}): {co}")
        else:
            print(f"Posición ({i},{j}): {co} normal")
```

Importante: La función enumerate devuelve dos valores: el índice de la fila (i) y la fila misma (fila). Por esto es que se utilizan dos variables en el for.

PECIER
Comité Nacional Peruano de la CIER

Bucles Anidados en Profundidad

Ejemplo: Análisis de Circuitos Paralelos

Este código está diseñado para calcular todas las combinaciones posibles de tres resistencias conectadas en paralelo y su resistencia equivalente.
Las combinaciones se extraen de una lista de valores predefinidos de resistencias y se evita la repetición de combinaciones mediante la condición r1 < r2 < r3.

Bucles Anidados en Profundidad

PECIER
Comité Nacional Peruano de la CIER

```
# Definir una lista con los valores de resistencia en ohmios (Ohm)
resistencia_valores = [10, 20, 30, 40, 50]

# Lista vacía para almacenar las combinaciones de resistencias y sus respectivas resistencias equivalentes
combinaciones = []

# Iniciar un bucle anidado para iterar sobre las 3 resistencias (r1, r2, r3)
# Esto para probar todas las combinaciones posibles de resistencias
# utilizando los valores disponibles en la lista 'resistencia_valores'
for r1 in resistencia_valores:          # Primer bucle: itera sobre r1
    for r2 in resistencia_valores:      # Segundo bucle: itera sobre r2
        for r3 in resistencia_valores:  # Tercer bucle: itera sobre r3
            # Asegurar que las combinaciones no se repitan, es decir, r1 debe ser menor que r2, y r2 menor que r3
            if r1 < r2 < r3:
                # Fórmula para la resistencia equivalente en paralelo:
                # 1/R_eq = 1/R1 + 1/R2 + 1/R3
                r_eq = 1 / (1/r1 + 1/r2 + 1/r3)  # Calcular la resistencia equivalente
                # Almacenar las resistencias y la resistencia equivalente en la lista 'combinaciones'
                combinaciones.append([r1, r2, r3, round(r_eq, 2)])

# Imprimir las combinaciones de resistencias y su resistencia equivalente
# Mostrar solo las primeras 5 combinaciones (para evitar imprimir demasiado texto)
for comb in combinaciones[:5]:
    print(f"Resistencias: R1 = {comb[0]} Ohm, R2 = {comb[1]} Ohm, R3 = {comb[2]} Ohm -> R_Equiv = {comb[3]} Ohm")

# Imprimir todas las combinaciones
# for comb in combinaciones:
#     print(f"Resistencias: R1 = {comb[0]} Ohm, R2 = {comb[1]} Ohm, R3 = {comb[2]} Ohm -> R_Equiv = {comb[3]}
Ohm")
```

Visualización de Datos con Bucles Anidados

PECIER
Comité Nacional Peruano de la CIER

Un mapa de calor es una representación visual que usa colores para representar valores de una matriz, como en este caso, una matriz de covarianzas. Los colores permiten ver rápidamente patrones y tendencias en los datos, especialmente cuando se trata de grandes matrices o datos complejos.

```
# Matriz de covarianzas
cov = [
    [0.75, 0.78, 0.80],
    [0.77, 0.82, 0.79],
    [0.85, 0.88, 0.90]
]

# Generar mapa de calor de covarianzas
plt.figure(figsize=(8,6))
plt.imshow(cov, cmap='hot', interpolation='nearest')
plt.colorbar(label='Covarianzas')
plt.title("Mapa de Calor")
plt.show()
```

PECIER
Comité Nacional Peruano de la CIER

Visualización de Datos con Bucles Anidados

Nota: imshow es la función de Matplotlib que se usa para mostrar imágenes o matrices en forma de mapas de calor. Analizaremos posteriormente el tema de gráficas.

PECIER
Comité Nacional Peruano de la CIER

Control de Flujo Avanzado

- El control de flujo se refiere a la forma en que se organiza el flujo de ejecución del código.
- Esto incluye el uso de estructuras como bucles (for, while), condicionales (if, else), y elementos de control como break, continue, y return que permiten alterar el comportamiento del flujo.
- En particular, break es una herramienta para interrumpir la ejecución de un bucle de forma anticipada cuando se cumple una determinada condición.

Control de Flujo Avanzado

PECIER
Comité Nacional Peruano de la CIER

Uso de break

En este ejemplo se usa la instrucción break en un sistema de protección que monitorea corrientes eléctricas en tiempo real. Al superar un valor crítico de corriente, el sistema actúa de inmediato para evitar posibles daños.

```
# Sistema de detección de fallas en tiempo real
lecturas = [5, 8, 15, 18, 12, 25, 9]
# Corrientes en A
umbral_critico = 20
for i, corriente in enumerate(lecturas, 1):
        print(f"Lectura {i}: {corriente}A")
        if corriente > umbral_critico:
                print(f"¡Falla detectada! Corte de emergencia en lectura {i}")
                break
        else:
                print("Sistema operando normalmente")
```

El break hace que el bucle se detenga de inmediato, interrumpiendo cualquier procesamiento adicional, ya que ya se ha detectado una situación crítica.

Control de Flujo Avanzado

PECIER
Comité Nacional Peruano de la CIER

`continue` para Filtrado de Datos

La instrucción continue se utiliza dentro de for o while para saltarse la iteración actual y continuar con la siguiente. Es útil cuando, dentro de un bucle, encontramos una condición que hace que queramos evitar ciertas acciones para algunos elementos pero seguir procesando los demás. Es especialmente valioso cuando se requiere filtrar o descartar elementos que no cumplen ciertos criterios sin interrumpir todo el proceso.

```
# Filtrar mediciones válidas de voltaje
datos_crudos = [218, -1, 225, 999, 230, 0, 235]
# -1 y 999 son errores
datos_validos = []
for voltaje in datos_crudos:
        if voltaje < 200 or voltaje > 250:
                print(f"Descartado: {voltaje}V (fuera de rango)")
                continue
datos_validos.append(voltaje)

print("\nDatos válidos para análisis:", datos_validos)
```

Control de Flujo Avanzado

PECIER
Comité Nacional Peruano de la CIER

`continue` para Filtrado de Datos

- La instrucción continue se utiliza dentro de for o while para saltarse la iteración actual y continuar con la siguiente.
- Es útil cuando, dentro de un bucle, encontramos una condición que hace que queramos evitar ciertas acciones para algunos elementos pero seguir procesando los demás.
- Es especialmente valioso cuando se requiere filtrar o descartar elementos que no cumplen ciertos criterios sin interrumpir todo el proceso.

PECIER
Comité Nacional Peruano de la CIER

Ejemplos y ejercicios

Google colab

PECIER_dia2_parte2

colab.research.google.com

PECIER

Comité Nacional Peruano de la CIER

SESIÓN 2
Parte3

PECIER
Comité Nacional Peruano de la CIER

Ejercicios Prácticos

Listas en Profundidad

Registro histórico de mediciones de voltaje.
Supongamos que debemos realizar un seguimiento continuo de las mediciones de voltaje en diferentes puntos de la red para asegurar que los valores se mantengan dentro de los rangos establecidos.

Las fluctuaciones de voltaje pueden indicar problemas en la distribución de energía, como fallas en los transformadores o elementos sobrecargados.

PECIER
Comité Nacional Peruano de la CIER

Ejercicios Prácticos

Listas en Profundidad

```
# Ejemplo: Procesamiento de mediciones diarias
mediciones = [
    [218, 215, 220],  # Lunes
    [222, 219, 225],  # Martes
    [210, 212, 208]   # Miércoles
]

# Calcular promedio por día
promedios = [sum(dia)/len(dia) for dia in mediciones]
print("Promedios diarios:", promedios)
```

Diccionarios Avanzados

PECIER
Comité Nacional Peruano de la CIER

Caso práctico: Base de datos de componentes.

Supongamos que debemos gestionar el inventario de componentes electrónicos (como baterías, paneles solares, etc.).

Esto es crucial para asegurar el correcto funcionamiento de las instalaciones.

Diccionarios Avanzados

PECIER
Comité Nacional Peruano de la CIER

Estructura de los Diccionarios:

```
# Base de datos de componentes
componentes = {
    "Baterías": {
        "Tipo": "Li-ion",
        "Capacidad": "100Ah",
        "Stock": 35
    },
    "Paneles": {
        "Tipo": "Monocristalino",
        "Watt": 350,
        "Stock": 20
    }
}
```

PECIER
Comité Nacional Peruano de la CIER

Diccionarios Avanzados

Estructura de los Diccionarios:

```
# Base de datos de componentes
componentes = {
    "Baterías": {
        "Tipo": "Li-ion",
        "Capacidad": "100Ah",
        "Stock": 35
    },
    "Paneles": {
        "Tipo": "Monocristalino",
        "Watt": 350,
        "Stock": 20
    }
}

# Acceder al tipo de la batería
tipo_bateria = componentes["Baterías"]["Tipo"]
print("Tipo de Batería:", tipo_bateria)

# Acceder a la capacidad de la batería
capacidad_bateria = componentes["Baterías"]["Capacidad"]
print("Capacidad de la Batería:", capacidad_bateria)
```

PECIER
Comité Nacional Peruano de la CIER

Ejemplos y ejercicios

PECIER_dia2_parte3

colab.research.google.com

PECIER

Comité Nacional Peruano de la CIER

E-mail: pecier@cier.org