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 3
Parte 1

PECIER
Comité Nacional Peruano de la CIER

Un vistazo a la sesión anterior

En la sesión anterior se revisaron las estructuras de datos, para almacenar y manipular grandes cantidades de información en una sola variable.

Tipos:
Listas []
Tuplas ()
Conjuntos {}
Diccionarios {"clave": valor}

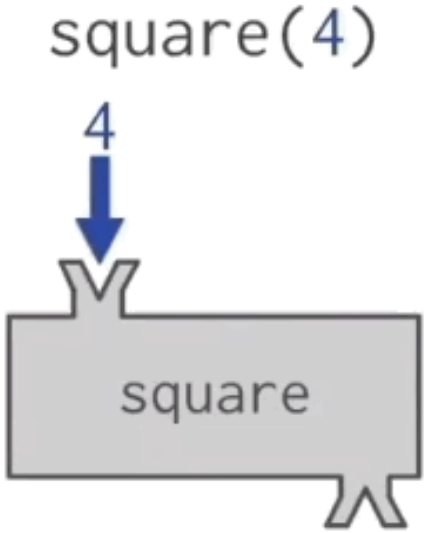
PECIER
Comité Nacional Peruano de la CIER

Definición y Uso de Funciones en Python:

Las **funciones** son bloques de código reutilizables que realizan una tarea específica. En Python, las funciones se definen utilizando la palabra clave **`def`**, seguida del nombre de la función y un par de paréntesis.

PECIER
Comité Nacional Peruano de la CIER

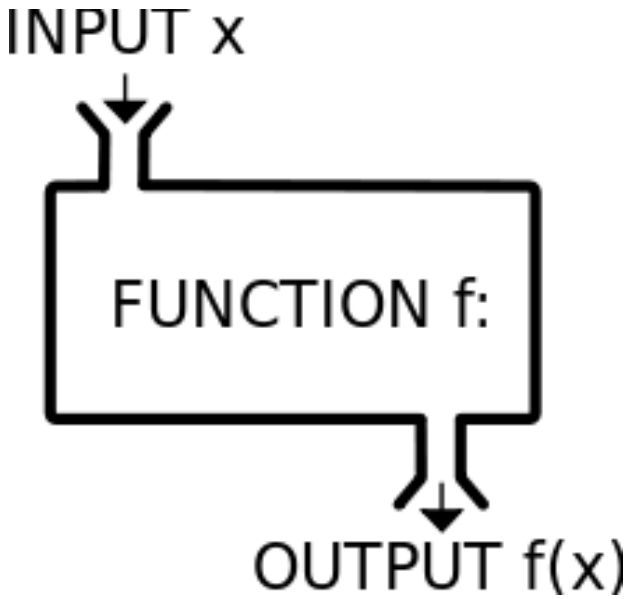
Funciones



PECIER
Comité Nacional Peruano de la CIER

Definición y Uso de Funciones en Python

Introducción a las Funciones



¿Por qué usar funciones?

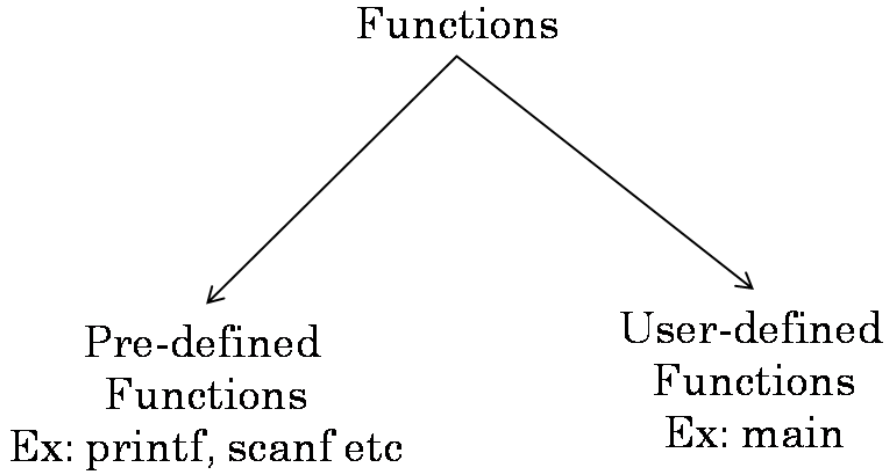
- Reutilizar código sin repetirlo
- Organizar el programa en módulos lógicos
- Facilitar el mantenimiento

PECIER
Comité Nacional Peruano de la CIER

Definición y Uso de Funciones en Python

Sintaxis Básica:

```
def nombre_funcion(parametros):
    """Comentarios (documentación)"""
    #Cuerpo de la función
    Return resultado
```



PECIER
Comité Nacional Peruano de la CIER

Definición y Uso de Funciones en Python

Ejemplo:

```
def mostrar_menu():
    print("\n1. Calcular potencia")
    print("2. Analizar circuito")
    print("3. Salir")

mostrar_menu()  # Llamada a la función
```

PECIER
Comité Nacional Peruano de la CIER

Definición y Uso de Funciones en Python

Sintaxis Básica:

- **`def:`** La palabra clave def se utiliza para definir una función.
- **`mostrar_menu`**: Es el nombre de la función. En este caso, el nombre describe claramente lo que hace la función: **mostrar un menú**.
- **`()`**: Los paréntesis después del nombre de la función indican que esta función no recibe ningún parámetro. Esto significa que no necesita ninguna entrada externa para ejecutarse.
- **`print`**: Dentro de la función, usamos la instrucción print() para mostrar el menú con tres opciones en la pantalla.

Lo podemos probar en https://www.online-python.com/

PECIER
Comité Nacional Peruano de la CIER

Definición y Uso de Funciones en Python

Flujo de Ejecución:

1. El programa comienza y ve que hay una función llamada mostrar_menu(), pero aún no se ejecuta nada, ya que una función solo se ejecuta cuando la **llamamos** explícitamente.

2. Cuando se llega a la línea mostrar_menu(), el programa invoca esa función. Entonces, Python ejecuta las instrucciones dentro de la función, que son las tres llamadas a print(), mostrando las opciones del menú.

PECIER
Comité Nacional Peruano de la CIER

Definición y Uso de Funciones en Python

Salida esperada:

```
1.Calcular potencia
2.Analizar circuito
3.Salir
```

Las funciones sin parámetros son útiles cuando queremos encapsular una tarea o conjunto de instrucciones que no requieren entrada externa para ejecutarse.
En este caso, la función `mostrar_menu()` siempre mostrará el mismo menú, independientemente de las condiciones o datos proporcionados por el usuario.

PECIER
Comité Nacional Peruano de la CIER

Uso de la Función con Parámetros Posicionales

```
def calcular_potencia(voltaje, corriente):
    """Calcula P = V * I"""
    return voltaje * corriente

# Uso con parámetros posicionales
potencia = calcular_potencia(220, 10)
print(f"Potencia: {potencia}W")

# Uso con parámetros nombrados
print(f"Potencia: {calcular_potencia(corriente=15, voltaje=110)}W")
```

PECIER
Comité Nacional Peruano de la CIER

Uso de la Función con Parámetros Posicionales

- **`calcular_potencia(220, 10):`** Aquí estamos pasando los valores 220 (para el voltaje) y 10 (para la corriente) como **parámetros posicionales**. El valor 220 se asignará al parámetro voltaje y el valor 10 al parámetro corriente.

- El resultado de la multiplicación (220 * 10 = 2200) es asignado a la variable potencia, y luego se imprime en la consola.

PECIER
Comité Nacional Peruano de la CIER

Uso de la Función con Parámetros Nombrados

```
print(f"Potencia: {calcular_potencia(corriente=15, voltaje=110)}W")
```

- **`calcular_potencia(corriente=15, voltaje=110)`:** En este caso, estamos usando **parámetros nombrados**. Esto significa que podemos pasar los parámetros en cualquier orden, siempre y cuando los nombres de los parámetros se especifiquen correctamente.
- Aquí, `corriente=15` se asigna a corriente y voltaje=110 se asigna a voltaje, y la función calcula (`P = 110 X 15 = 1650`).

PECIER
Comité Nacional Peruano de la CIER

Entonces:Entonces:

- **Parámetros posicionales**: Los valores se asignan según el orden en que se pasan. En el caso de calcular_potencia(220, 10), el primer valor (220) se asigna a voltaje y el segundo valor (10) a corriente.
- **Parámetros nombrados**: Permiten pasar los valores en cualquier orden, siempre que se especifiquen los nombres de los parámetros. En el caso de calcular_potencia(corriente=15, voltaje=110), no importa el orden de los parámetros, siempre y cuando se especifiquen correctamente sus nombres.

PECIER
Comité Nacional Peruano de la CIER

Funciones que Retornan Múltiples Valores

En Python, una función no solo puede devolver un único valor, sino que también puede retornar varios valores al mismo tiempo.

Esto se hace generalmente retornando una tupla (recordatorio: estructura de datos ordenada).

PECIER
Comité Nacional Peruano de la CIER

Funciones que Retornan Múltiples Valores

¿Por qué retornar múltiples valores?

- Es útil cuando una sola función puede analizar o calcular varias cosas a partir de los mismos datos.
- Permite mantener el código más limpio, evitando múltiples llamadas o funciones separadas.
- Es ideal para modelos en ingeniería, donde muchas veces se derivan múltiples propiedades físicas a partir de una misma entrada.

PECIER
Comité Nacional Peruano de la CIER

Funciones que Retornan Múltiples Valores

El uso de funciones que retornan múltiples valores permite encapsular lógica compleja y devolver toda la información necesaria en una sola operación.

Ámbito de Variables (Scope)

PECIER
Comité Nacional Peruano de la CIER

En programación, el **ámbito** (o *scope*) se refiere al lugar del programa donde una variable es válida y accesible. En Python, existen principalmente dos tipos de ámbito:

- **Variables globales**: Son las que se definen fuera de cualquier función y están disponibles en todo el programa.
- **Variables locales**: Son las que se definen dentro de una función y solo existen mientras esa función se está ejecutando.

PECIER
Comité Nacional Peruano de la CIER

Funciones Avanzadas

En esta sección vamos a explorar funciones con capacidades más flexibles, especialmente útiles en aplicaciones eléctricas o técnicas donde los datos pueden variar según el contexto.

PECIER
Comité Nacional Peruano de la CIER

Funciones Avanzadas

Parámetros por defecto

Los parámetros por defecto permiten definir valores que serán utilizados si no se proporciona un argumento al llamar la función. Esto hace que nuestras funciones sean más versátiles y fáciles de usar.

PECIER
Comité Nacional Peruano de la CIER

Funciones Avanzadas

Argumentos arbitrarios (*args y **kwargs)

En algunos casos, no sabemos cuántos argumentos exactos necesita una función. Python permite definir funciones que acepten:

- Un **número variable de argumentos posicionales** (`*args`)
- Un **número variable de argumentos con nombre** (`**kwargs`)

PECIER
Comité Nacional Peruano de la CIER

Ejemplos y ejercicios

PECIER_dia3_parte1

colab.research.google.com

PECIER

Comité Nacional Peruano de la CIER

SESIÓN 3
Parte 2

PECIER
Comité Nacional Peruano de la CIER

Parámetros y Valores de Retorno en Python

En esta sección se hará énfasis en la presentación de los parámetros a las funciones, y las prácticas recomendadas para mantener el código claro.

PECIER
Comité Nacional Peruano de la CIER

Parámetros en Funciones

Los **parámetros posicionales** son aquellos cuyo valor se asigna según el orden en que se pasan al llamar la función.

```
def calcular_potencia(voltaje, corriente):
    """
    Calcula potencia eléctrica (P = V × I)
    """
    return voltaje * corriente

# Llamada correcta
print(calcular_potencia(220, 10))  # 2200 W

# Llamada incorrecta (valores invertidos)
print(calcular_potencia(10, 220)) # Resultado correcto, pero mal interpretado
```

Parámetros en Funciones

PECIER
Comité Nacional Peruano de la CIER

```
def calcular_potencia(voltaje, corriente):
    """
    Calcula potencia eléctrica (P = V × I)
    """
    return voltaje * corriente

# Llamada correcta
print(calcular_potencia(220, 10))  # 2200 W

# Llamada incorrecta (valores invertidos)
print(calcular_potencia(10, 220)) # Resultado correcto, pero mal interpretado
```

Nota: El código es sintácticamente válido en ambos casos, pero en el segundo hay un error lógico por el orden de los parámetros. Observe el uso de """TEXTO""" para comentarios amplios, lo cual es una buena práctica en funciones.

PECIER
Comité Nacional Peruano de la CIER

Parámetros en Funciones

Parámetros Nombrados (Keyword Arguments)

Permiten indicar explícitamente el nombre del parámetro al que corresponde un valor. Esto mejora la legibilidad y reduce errores.

```
def configurar_transformador(voltaje_primario, voltaje_secundario, frecuencia=60):
    relacion = voltaje_primario / voltaje_secundario
    print(f"Transformador {relacion:.2f}:1 configurado a {frecuencia}Hz")

configurar_transformador(
    voltaje_secundario=110,
    voltaje_primario=440,
    frecuencia=50
)
```

Con esta manera de tratar las funciones, se permiten cambios en el orden de los argumentos, se hace más legible el código, y se permiten usar valores por defecto opcionales.

PECIER
Comité Nacional Peruano de la CIER

Parámetros en Funciones

Parámetros Arbitrarios: *args y **kwargs

`*args` (argumentos posicionales variables)

En ocasiones, una función requiere recibir una cantidad arbitraria de parámetros, o al menos que inicialmente no se sabe cuántos parámetros va a recibir. Por ejemplo, calcular un promedio de valores. Con el uso de `*args`, se permite pasar cualquier número de argumentos.

PECIER
Comité Nacional Peruano de la CIER

Parámetros en Funciones

Parámetros Arbitrarios: *args y **kwargs

`*args` (argumentos posicionales variables)
En ocasiones, una función requiere recibir una cantidad arbitraria de parámetros, o al menos que inicialmente no se sabe cuántos parámetros va a recibir. Por ejemplo, calcular un promedio de valores. Con el uso de `*args`, se permite pasar cualquier número de argumentos.

`**kwargs` (argumentos nombrados variables)
Este caso permite recibir cualquier número de argumentos nombrados como un diccionario.

PECIER
Comité Nacional Peruano de la CIER

Valores de Retorno

Retorno Simple vs Múltiple

Una función puede devolver:

- Un valor simple
- Múltiples valores (empaquetados en una tupla)

Los ejemplos previos han mostrado funciones que retornan un única valor. Que retorne más de uno se programa de forma simple con el 'return'.

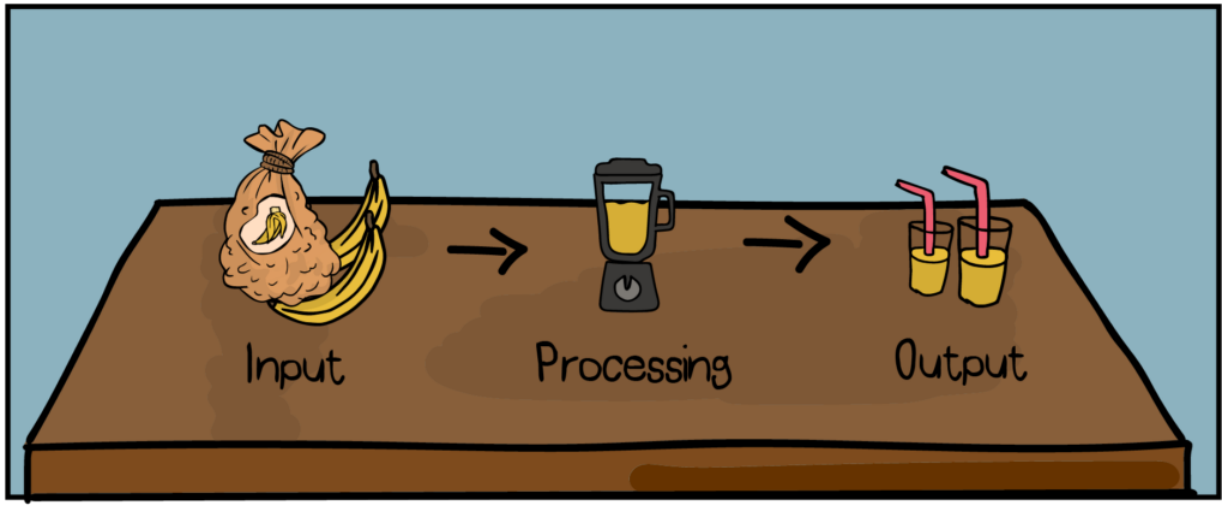
PECIER
Comité Nacional Peruano de la CIER

Valores de Retorno

Retorno de Estructuras Complejas

Además de múltiples valores, las funciones también pueden devolver:

- Diccionarios
- Listas
- Objetos personalizados



PECIER
Comité Nacional Peruano de la CIER

Ejemplos y ejercicios

PECIER_dia3_parte2

colab.research.google.com

PECIER

Comité Nacional Peruano de la CIER

SESIÓN 3
Parte 3

PECIER
Comité Nacional Peruano de la CIER

Importación de Módulos y Bibliotecas en Python

Fundamentos de Importación

¿Qué son los módulos en Python?

Un **módulo** es simplemente un archivo con extensión .py que contiene funciones, clases o variables que pueden ser reutilizadas en otros programas. Es una forma de organizar y reutilizar el código.

PECIER
Comité Nacional Peruano de la CIER

Importación de Módulos y Bibliotecas en Python

Fundamentos de Importación

Ventajas de usar módulos:

- **Evitan repetir código** (principio DRY: Don't Repeat Yourself).
- **Facilitan el mantenimiento del programa** (diferentes personas pueden dar mantenimiento o desarrollar partes separadas).
- **Permiten dividir un proyecto en partes lógicas y manejables**.
- **Fomentan la reutilización y colaboración** entre personas y proyectos.

PECIER
Comité Nacional Peruano de la CIER

Importación de Módulos y Bibliotecas en Python

La palabra clave de la importación es **import** seguida del nombre del módulo.

Existe una enorme variedad de módulos, para diferentes fines (análisis de datos, gráficas, IA, etc., etc., etc.).

En muchos contextos, conocer y utilizar módulos y bibliotecas es gran parte del trabajo de desarrollo.

PECIER
Comité Nacional Peruano de la CIER

Importación de Módulos y Bibliotecas en Python

Diferentes formas de importar

Existen diferentes formas de importar funciones de bibliotecas. Usualmente no importamos más de lo que necesitamos, por un asunto de eficiencia de recursos (memoria, velocidad).

PECIER
Comité Nacional Peruano de la CIER

Importación de Módulos y Bibliotecas en Python

a. Importación estándar

```
import math

print(math.sin(math.pi / 2))
```

PECIER
Comité Nacional Peruano de la CIER

Importación de Módulos y Bibliotecas en Python

b. Importar con alias (usado para abreviar nombres largos)

```
import numpy as np

array = np.array([1, 2, 3])
print(array * 2)
```

c. Importar elementos específicos

```
from math import sqrt, pi

print(sqrt(49))  # No necesitas usar math.sqrt
print(pi)
```

PECIER
Comité Nacional Peruano de la CIER

Importación de Módulos y Bibliotecas en Python

d. Importar todo el contenido (NO recomendado)

```
from math import *

print(sin(pi))  # Puede generar conflictos si hay nombres duplicados
```

Bibliotecas Esenciales para el Sector Eléctrico

PECIER
Comité Nacional Peruano de la CIER

NumPy para Cálculos Numéricos

NumPy permite realizar cálculos con arreglos (vectores y matrices), lo que es esencial para análisis de sistemas eléctricos, cálculos de potencia, corriente, etc.

```
import numpy as np

voltajes = np.array([220, 215, 210])
corrientes = np.array([5.2, 5.0, 4.8])
potencias = voltajes * corrientes

print(f"Potencias: {potencias}")
print(f"Potencia promedio:
{np.mean(potencias):.2f} W")
```

Ventaja: Las operaciones son mucho más rápidas que usar listas tradicionales.

PECIER
Comité Nacional Peruano de la CIER

Bibliotecas Esenciales para el Sector Eléctrico

Matplotlib para Visualización

- En el sector eléctrico, los datos deben analizarse no solo con cálculos, sino también visualmente.
- Matplotlib es la biblioteca más utilizada en Python para crear gráficos: de líneas, barras, pastel, dispersión y más.
- Nos permite visualizar series de tiempo, comparaciones de consumo entre equipos, entre otros.
- Visualizar los datos eléctricos facilita detectar patrones, anomalías o tendencias, son elementos fundamentales de análisis e informes.

PECIER
Comité Nacional Peruano de la CIER

Bibliotecas Esenciales para el Sector Eléctrico

Matplotlib para Visualización

Ejemplo: Consumo eléctrico por hora

```
import matplotlib.pyplot as plt

horas = range(24)
consumo = [50 + 10*h + 5*(h % 3) for h in horas]  # simulación simple

plt.figure(figsize=(10, 5))
plt.plot(horas, consumo, 'r-o', label='Consumo (kW)')
plt.title("Perfil de Carga Diario")
plt.xlabel("Hora del día")
plt.ylabel("Potencia (kW)")
plt.grid(True)
plt.legend()
plt.show()
```

PECIER
Comité Nacional Peruano de la CIER

Bibliotecas Esenciales para el Sector Eléctrico

Matplotlib para Visualización

Conviene revisar la documentación:
https://matplotlib.org/cheatsheets/

¡Su propia página web!

PECIER
Comité Nacional Peruano de la CIER

Creación y Uso de Módulos Personalizados

¿Cómo crear nuestros propios módulos?

1.Crear un archivo Python con funciones.

Por ejemplo: **electricidad.py**

Creación y Uso de Módulos Personalizados

PECIER
Comité Nacional Peruano de la CIER

```
# electricidad.py

def ley_ohm(V, I):
    """Calcula resistencia R = V / I"""
    return V / I if I != 0 else float('inf')

def potencia(V, I):
    """Calcula potencia eléctrica P = V × I"""
    return V * I
```

2. Crear otro archivo para usarlo. Por ejemplo: **main.py**

```
# main.py

from electricidad import ley_ohm, potencia

print(ley_ohm(220, 5))        # 44.0
print(potencia(220, 5))       # 1100
```

Ambos archivos deben estar en la misma carpeta.

PECIER
Comité Nacional Peruano de la CIER

Creación y Uso de Módulos Personalizados

¿Qué es un paquete?

Un **paquete** es una carpeta que contiene varios módulos organizados jerárquicamente. Cada carpeta debe incluir un archivo especial __init__.py (aunque puede estar vacío), lo que indica a Python que esa carpeta debe tratarse como un paquete.

Estructura de ejemplo:

```
proyecto/
│
├── main.py
├── paquete_electrico/
│   ├── __init__.py
│   ├── circuitos.py
│   └── analisis/
│       ├── __init__.py
│       └── potencia.py
```

PECIER
Comité Nacional Peruano de la CIER

Creación y Uso de Módulos Personalizados

Contenido de circuitos.py:

```
def calcular_impedancia(R, X):
    """Calcula impedancia Z = √(R² + X²)"""
    return (R**2 + X**2)**0.5
```

Contenido de `potencia.py`:

```
def potencia_aparente(V, I):
    """Calcula potencia aparente S = V × I"""
    return V * I
```

PECIER
Comité Nacional Peruano de la CIER

Creación y Uso de Módulos Personalizados

Uso desde `main.py`:

```
from paquete_electrico.circuitos import calcular_impedancia
from paquete_electrico.analisis.potencia import potencia_aparente

print(calcular_impedancia(10, 5))  # 11.18 ohm
print(potencia_aparente(220, 5))   # 1100 VA
```

PECIER
Comité Nacional Peruano de la CIER

Ejemplos y ejercicios

PECIER_dia3_parte3

colab.research.google.com

PECIER

Comité Nacional Peruano de la CIER

E-mail: pecier@cier.org