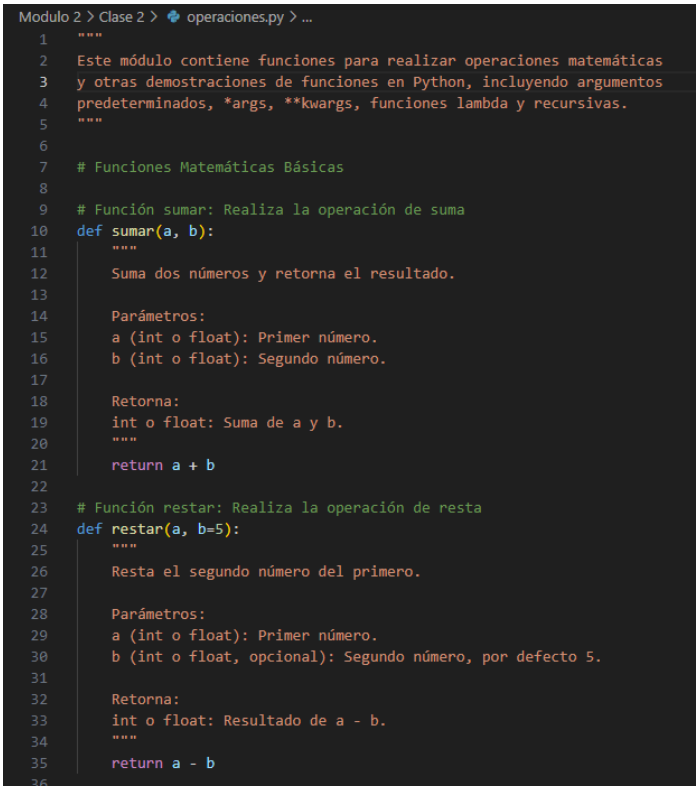
Actividad 2 Módulo 2

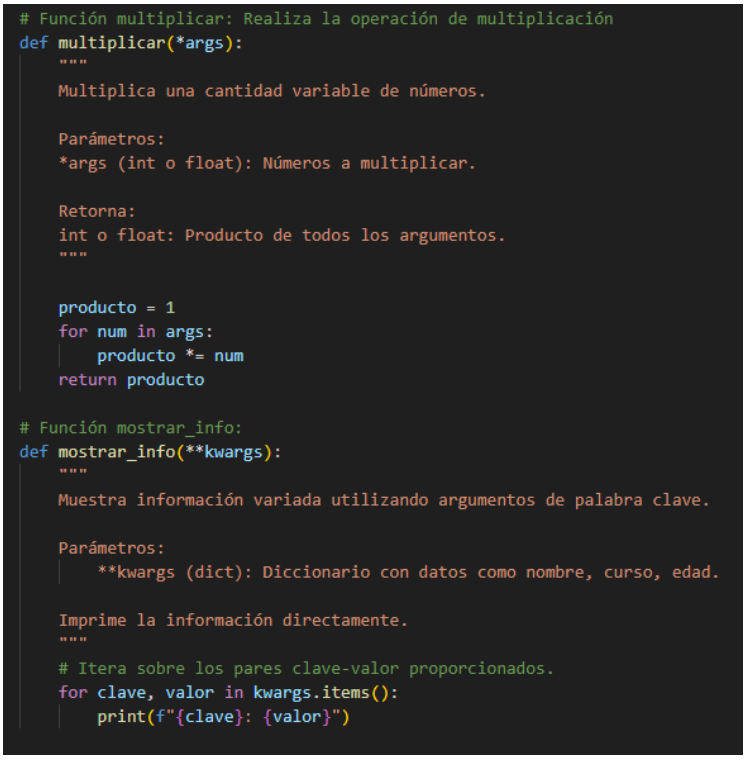
Nombre: Carlos Saldivia Susperreguy

Capturas:

Operaciones.py



```
Modulo 2 > Clase 2 > operaciones.py > ...
"""
Este módulo contiene funciones para realizar operaciones matemáticas
y otras demostraciones de funciones en Python, incluyendo argumentos
predeterminados, *args, **kwargs, funciones lambda y recursivas.
"""

# Funciones Matemáticas Básicas

# Función sumar: Realiza la operación de suma
def sumar(a, b):
    """
    Suma dos números y retorna el resultado.

    Parámetros:
    a (int o float): Primer número.
    b (int o float): Segundo número.

    Retorna:
    int o float: Suma de a y b.
    """
    return a + b

# Función restar: Realiza la operación de resta
def restar(a, b=5):
    """
    Resta el segundo número del primero.

    Parámetros:
    a (int o float): Primer número.
    b (int o float, opcional): Segundo número, por defecto 5.

    Retorna:
    int o float: Resultado de a - b.
    """
    return a - b
```



```
# Función multiplicar: Realiza la operación de multiplicación
def multiplicar(*args):
    """
    Multiplica una cantidad variable de números.

    Parámetros:
    *args (int o float): Números a multiplicar.

    Retorna:
    int o float: Producto de todos los argumentos.
    """

    producto = 1
    for num in args:
        producto *= num
    return producto

# Función mostrar_info:
def mostrar_info(**kwargs):
    """
    Muestra información variada utilizando argumentos de palabra clave.

    Parámetros:
        **kwargs (dict): Diccionario con datos como nombre, curso, edad.

    Imprime la información directamente.
    """
    # Itera sobre los pares clave-valor proporcionados.
    for clave, valor in kwargs.items():
        print(f"{clave}: {valor}")
```

```python
# Función lambda para potencia
potencia = lambda base, exponente: base ** exponente

"""
Función lambda para calcular la potencia de un número.

Parámetros:
base (int o float): Base de la potencia.
exponente (int o float): Exponente de la potencia.

Retorna:
int o float: base elevado a exponente.
"""

# Función factorial:

def factorial(n):
    """
    Calcula el factorial de un número entero no negativo de forma recursiva.

    Args:
        n (int): El número entero (no negativo).

    Returns:
        int: El factorial de 'n'.

    Raises:
        ValueError: Si 'n' es un número negativo.
    """
    if n == 0:
        return 1
    elif n < 0:
        raise ValueError("El factorial solo está definido para números no negativos.")
    else:
        return n * factorial(n - 1)
```

Main.py

Modulo 2 > Clase 2 > main.py > ...

```python
# Exploración de Funciones y Módulos:

# ¿Qué es una función en Python?
# Una función en Python es un bloque de código reutilizable diseñado para realizar una tarea específica.
# Ayuda a organizar el código, hacerlo más legible y evitar la repetición.
# Se define con la palabra clave def.

# ¿Qué es un módulo y para qué sirve?
# Un módulo en Python es un archivo (.py) que contiene definiciones de Python (funciones, clases, variables).
# Sirve para organizar el código en unidades lógicas y reutilizables.
# Permite agrupar código relacionado y facilita su importación y uso en otros scripts o módulos.

# ¿Qué ventajas tiene modularizar el código?
# Las ventajas de modularizar el código incluyen:
# 1- Reutilización de código sin repetirlo
# 2- Mantenimiento más sencillo y localizado
# 3- Facilita la colaboración en equipo
# 4- Mejora la legibilidad y organización
# 5- Permite mayor escalabilidad del proyecto
# 6- Facilita las pruebas unitarias

# ¿Qué es un docstring y cómo se usa?
# Un docstring (del inglés "documentation string") es una cadena literal que aparece como la primera declaración en un módulo, clase, función o método.
# Se usa para documentar el propósito y el uso del bloque de código al que pertenece.
# Se encierra entre comillas triples (simples o dobles).

```

```python
import operaciones

# Pedir al usuario ingresar dos números
num1 = float(input("Ingrese el primer número: "))
num2 = float(input("Ingrese el segundo número: "))

# Llamar a las funciones y mostrar resultados

# Suma
resultado_suma = operaciones.sumar(num1, num2)
print(f"\nResultado de la suma ({num1} + {num2}): {resultado_suma}")

# Resta (demostrando argumento predeterminado)
# Llamada completa con dos argumentos
resultado_resta_completa = operaciones.restar(num1, num2)
print(f"\nResultado de la resta ({num1} - {num2}): {resultado_resta_completa}")
# Llamada utilizando el argumento predeterminado para 'b' (b=5)
resultado_resta_default = operaciones.restar(num1)
print(f"\nResultado de la resta (Valor predeterminado b=5) ({num1} - 5): {resultado_resta_default}")

# Multiplicación (*args)
# Demostrando flexibilidad con diferentes cantidades de argumentos
resultado_mult_2 = operaciones.multiplicar(num1, num2)
print(f"\nResultado de la multiplicación ({num1} * {num2}): {resultado_mult_2}")

# Potencia (lambda)
resultado_potencia = operaciones.potencia(num1, num2)
print(f"\nResultado de la potencia ({num1} elevado a la {num2}): {resultado_potencia}")

# Factorial (recursiva)
# Aseguramos que el número sea entero para el factorial
num_factorial = int(num1) # Usamos el primer número para el factorial
if num_factorial >= 0:
    resultado_factorial = operaciones.factorial(num_factorial)
    print(f"\nResultado del factorial de {num_factorial}: {resultado_factorial}")
else:
    print(f"\nNo se puede calcular el factorial de {num_factorial} (número negativo).")

# Mostrar información (**kwargs)
operaciones.mostrar_info(
    nombre="Carlos Saldivia",
    curso="Ingeniería de Datos",
    edad=30,
    ciudad="Santiago"
)
```

```
Ingrese el primer número: 10
Ingrese el segundo número: 2

Resultado de la suma (10.0 + 2.0): 12.0

Resultado de la resta (10.0 - 2.0): 8.0

Resultado de la resta (Valor predeterminado b=5) (10.0 - 5): 5.0

Resultado de la multiplicación (10.0 * 2.0): 20.0

Resultado de la potencia (10.0 elevado a la 2.0): 100.0

Resultado del factorial de 10: 3628800
nombre: Carlos Saldivia
curso: Ingeniería de Datos
edad: 30
ciudad: Santiago
```