Angel Alberto Ravelero Ramirez

Proyecto: “Una mejor calculadora” (versión fácil)

¿Qué tienen que hacer? Crear una calculadora en Python que funcione desde la consola. La calculadora debe tener estas 4 funciones (con estos nombres exactos):

1. addmultiplenumbers(lista) Suma todos los números de una lista y regresa el resultado.
2. multiplymultiplenumbers(lista) Multiplica todos los números de una lista y regresa el resultado.
3. isiteven(numero) Regresa True si el número es par (y entero), o False si no.
4. isitaninteger(numero) Regresa True si el número es entero (por ejemplo 7 o 7.0), o False si no.

Reglas del trabajo:

- Usar una función principal para arrancar el programa.
- Dividir el código en funciones (no todo junto).
- Aceptar números positivos y negativos.
- Pensar en casos especiales (lista vacía, números decimales).
- Mensajes claros para el usuario.

Calidad mínima:

- Código ordenado y con nombres claros.
- Comentarios cortos donde haga falta.
- Archivo README con una breve explicación.

Cómo trabajar con el repositorio (GitHub):

1. Hacer Fork del repositorio original.
2. Clonar su Fork a su computadora.
3. Crear una rama de trabajo.
4. Subir cambios y hacer Pull Request.

Herramientas necesarias:

- Python 3 instalado.
- Librerías: pytest, pytest-cov, black, flake8, isort, mypy (estas se instalan con pip).
- Extensiones VS Code: Python, Pylance, Black Formatter, isort, GitHub Pull Requests and Issues.

Evaluación (8 puntos):

- (2 pts) Suma y multiplicaciones correctas (incluye casos especiales).
- (2 pts) isiteven e isitaninteger correctas.
- (2 pts) Proyecto ordenado y pruebas pasando.
- (1 pt) Mensajes claros para el usuario.
- (1 pt) README breve con explicación.

Sugerencia: Si se traban con una función, avancen con otra y luego regresen.

Realización:

Se utilizaron las mismas herramientas para generar los marcos en los mensajes de salida que el código anterior:

```
# Funciones de diseño
def m_Superior():
  print("╔" + "═" * (ancho - 2) + "╗")

def m_Inferior():
  print("╚" + "═" * (ancho - 2) + "╝")

def linea_texto(texto):
    espacio = ancho - 4 - len(texto)  # Define e
    izq = espacio // 2  # Espacio de margen haci
    der = espacio - izq # Espacio de margen haci
    print(f"║ {' ' * izq}{texto}{' ' * der} ║")

def linea_vacia():
    print("║" + " " * (ancho - 2) + "║")
```

Se tomo la estructura de navegación y diseño del proyecto anterior:

```
# Menú
def menu_inicial():
    marco_superior()
    linea_texto("CALCULADORA")
    linea_vacia()
    linea_texto("1. Operaciones básicas")
    linea_texto("2. Operaciones libres")
    linea_texto("3. Conversión entre sistemas")
    linea_texto("4. Salir")
    linea_vacia()
    marco_inferior()
```

```
╔══════════════════════════════════════╗
║              CALCULADORA             ║
╠══════════════════════════════════════╣
║                                      ║
║          1. Operaciones básicas      ║
║          2. Operaciones libres       ║
║       3. Conversión entre sistemas   ║
║                  4. Salir            ║
╚══════════════════════════════════════╝
```

Después de los cambios:

```python
def menu_incial():
    m_Superior()
    linea_texto("Calculadora Mejorada")
    linea_vacia()
    linea_texto("1. Sumar múltiples números")
    linea_texto("2. Multiplicar múltiples número")
    linea_texto("3. Verificar si el número es par y entero")
    linea_texto("4. Verificar si es número es entero")
    linea_texto("5. Salir")
    linea_vacia()
    m_Inferior()
```

```
Calculadora Mejorada

1. Sumar múltiples números
2. Multiplicar múltiples número
3. Verificar si el número es par y entero
4. Verificar si es número es entero
5. Salir
```

Se crearon las funciones documentando su proceso:

```python
#Funciones Requeridas
def addmultiplenumbers(numeros):
    """
    Suma todos los números en una lista
    Entrada: Lista de números [num1, num2, ...]
    Salida: Suma total (float/int)
    """
    resultado = sum(numeros)
    return round(numeros,2)

def multiplymultiplenumbers(numeros):
    """
    Multiplica todos los número en una lista
    Entrada: numeros de números [num1, num2, ...]
    Salida: Producto total (float/int)
    """
    total = 1
    for num in numeros:
        total *= num
    return round(total,2)
```

```python
def multiplymultiplenumbers(numeros):
    """
    Multiplica todos los número en una lista
    Entrada: numeros de números [num1, num2, ...]
    Salida: Producto total (float/int)
    """
    total = 1
    for num in numeros:
        total *= num
    return round(total,2)

def isiteven(numero):
    """
    Verifica si un número es par y entero
    Entrada: un número
    Salida: True/False
    """
    return isitaninteger(numero) and numero % 2 == 0

def isitaninteger(numero):
    """
    Verifica si un número es entero
    Entrada: un número
    Salida: True/False
    """
    return isinstance(numero, int) or (isinstance(numero, float) and numero.is_integer())
```

El flujo principal del programa se maneja desde “main”, el ingreso de los datos proporcionados por el usuario contiene validaciones para evitar errores durante el proceso del código, así como una contracción en los resultados de las sumas y multiplicaciones para visualizar una estructura más simple.

```
Ingrese número separados por espacios
Ejemplo : 5 10 46 34: 2.3123 3.123 6.123
Resultado: 44.22
```

Conclusión:

El programa funciona de la manera esperada, y las validaciones logran su objetivo, los mensajes dados al usuario son claros y dan la información adecuada para un correcto uso del programa.