

Proyecto: “Una mejor calculadora”

¿Qué tienen que hacer?

Crear una calculadora en Python que funcione desde la consola. La calculadora debe tener estas 4 funciones: `addmultiplenumbers` (lista), `multiplymultiplenumbers` (lista), `isiteven` (numero), `isitaninteger` (numero).

1.- Instalé las herramientas - Librerías: `pytest`, `pytest-cov`, `black`, `flake8`, `isort`, `mypy` con `pip`

Abrir VS Code y el terminal

- Abrir tu proyecto o carpeta creada “mejor calculadora” en VS Code.
- Abre la terminal integrada: Menú: Ver → Termina

Activar tu entorno virtual (opcional pero recomendado)

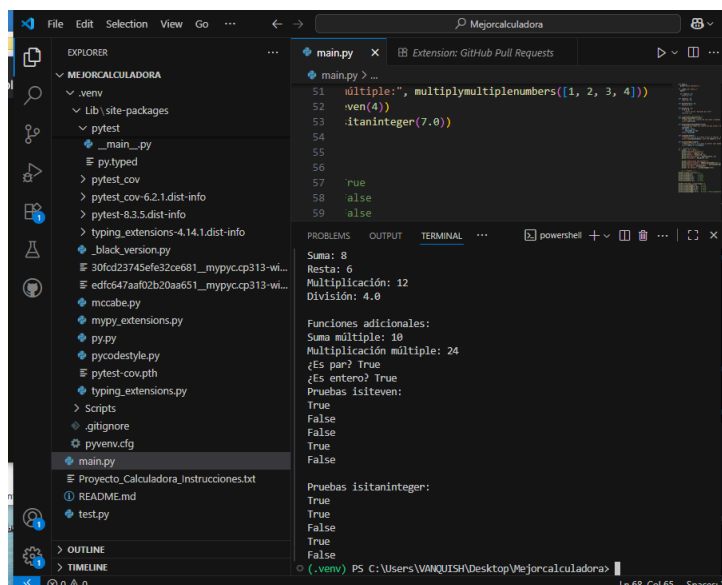
- `python -m venv venv`

Instalar las librerías con pip

- `pip install pytest pytest-cov black flake8 isort mypy`

Verificar la instalación

- `pytest --version`
- `black --version`
- `flake8 --version`
- `isort --version`
- `mypy --version`



2.- Instalar las extensiones VS Code: Python, Pylance, Black Formatter, isort, GitHub Pull Requests and Issues.

Abrir el panel de extensiones

- Menú: Ver → Extensiones

Buscar e instalar cada extensión

- En la barra de búsqueda escribe el nombre de cada extensión y haz clic en **"Instalar"**

3.- Crear una carpeta llamada "mejor calculadora" en la PC.

4.- Abrir la carpeta llamada mejor calculadora en visual studio code y abrir el archivo [main.py](#) que es en donde haremos la calculadora básica con 4 funciones adicionales.

5.- Se creó el código para la calculadora:

```
def main():
```

```
    print("Hello learners!")
```

```
if __name__=="__main__":
```

```
    main()
```

```
    def sumar(a, b):
```

```
        return a + b
```

```
    def restar(a, b):
```

```
        return a - b
```

```
    def multiplicar(a, b):
```

```
        return a * b
```

```
    def dividir(a, b):
```

```
        if b == 0:
```

```
            return "Error: División por cero"
```

```
        return a / b
```

```
# --- Funciones adicionales ---
```

```
def addmultiplenumbers(lista):
```

```

    """Suma todos los números de una lista y regresa el resultado."""
    return sum(lista)

def multiplymultiplenumbers(lista):
    """Multiplica todos los números de una lista y regresa el resultado."""
    resultado = 1
    for num in lista:
        resultado *= num
    return resultado

def isiteven(numero):
    """Regresa True si el número es par (y entero), o False si no."""
    return isinstance(numero, int) and numero % 2 == 0

def isitaninteger(numero):
    """Regresa True si el número es entero (por ejemplo 7 o 7.0), o False si no."""
    return numero == int(numero)

# --- Ejemplo de uso ---

if __name__ == "__main__":
    print("Funciones básicas:")
    print("Suma:", sumar(5, 3))
    print("Resta:", restar(10, 4))
    print("Multiplicación:", multiplicar(2, 6))
    print("División:", dividir(8, 2))
    print("\nFunciones adicionales:")
    print("Suma múltiple:", addmultiplenumbers([1, 2, 3, 4]))
    print("Multiplicación múltiple:", multiplymultiplenumbers([1, 2, 3, 4]))
    print("¿Es par?", isiteven(4))
    print("¿Es entero?", isitaninteger(7.0))

```

```
# --- Pruebas ---
```

```
print("Pruebas isiteven:")
```

```
print(isiteven(4))    # True
```

```
print(isiteven(7))    # False
```

```
print(isiteven(4.0))  # False
```

```
print(isiteven(-8))   # True
```

```
print(isiteven(3.5))  # False
```

```
print("\nPruebas isitaninteger:")
```

```
print(isitaninteger(7))  # True
```

```
print(isitaninteger(7.0)) # True
```

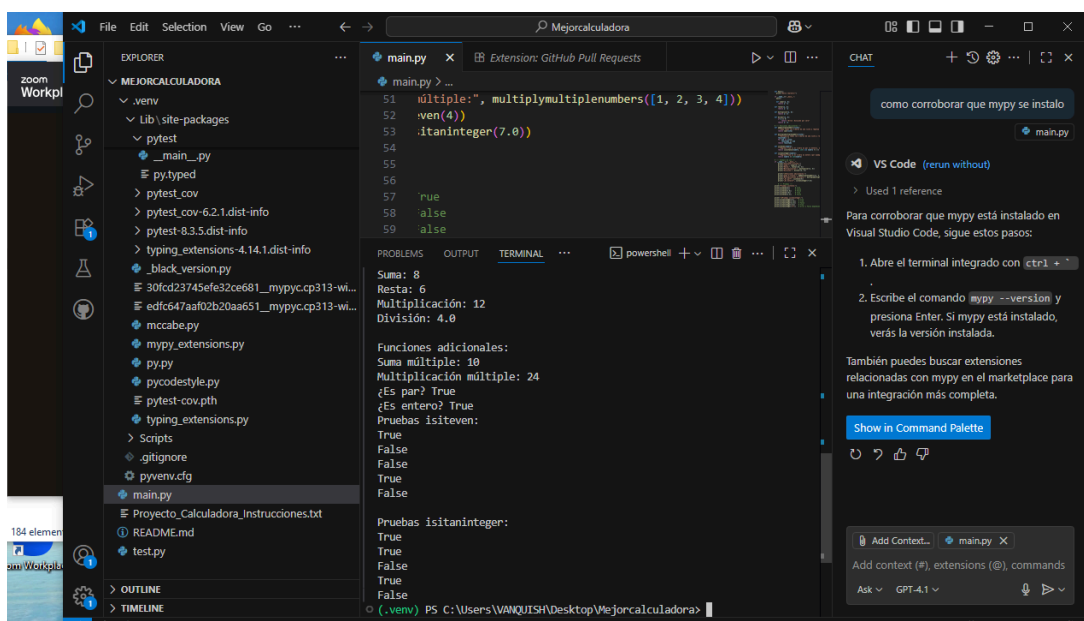
```
print(isitaninteger(3.5)) # False
```

```
print(isitaninteger(-2)) # True
```

```
print(isitaninteger("5")) # Error o False dependiendo del tipo
```

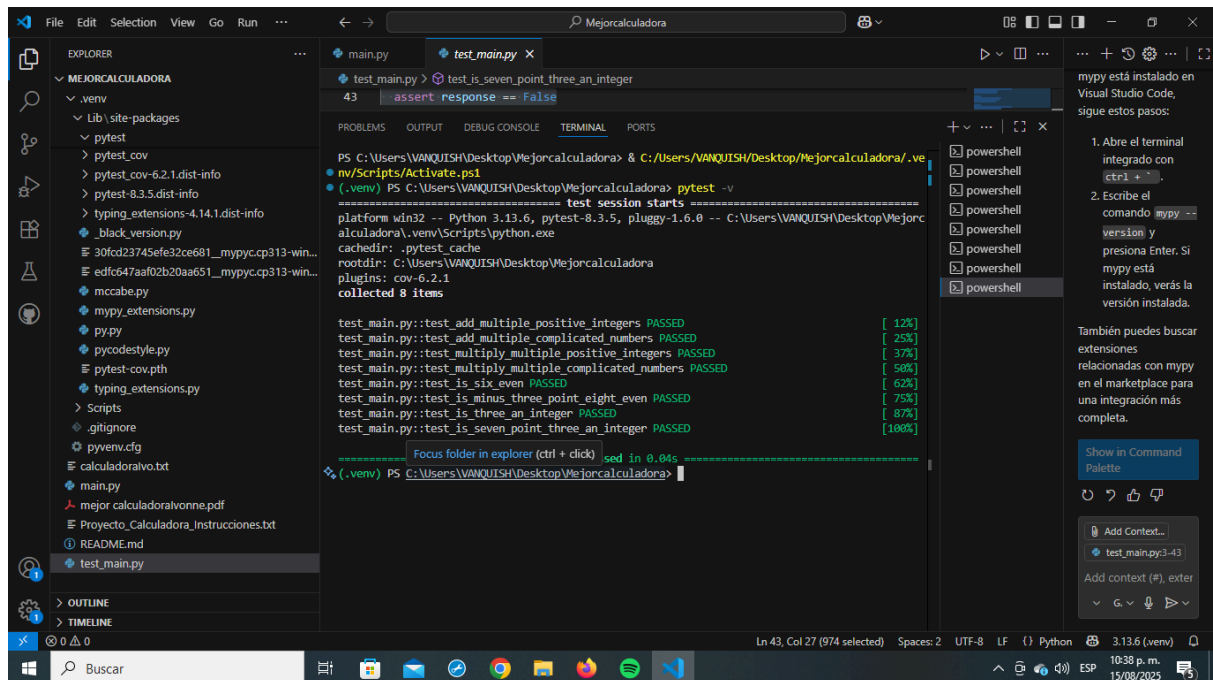
6.- Se corrió el código en la terminal, dándole click derecho al archivo [main.py](#) → open in integrated terminal.

7.- En la terminal se escribió `py main.py` → enter



8.- Se corrió el test con 8 pruebas que pasaron exitosamente.

- Archivo test_main.py → click derecho → open in integrated terminal
- Escribir el comando pytest -v



The screenshot shows the Visual Studio Code interface with the file explorer on the left, the editor in the center, and the integrated terminal at the bottom. The file explorer shows a project named 'MEJORCALCULADORA' with a subdirectory '.venv'. The editor displays the file 'test_main.py' with a line number 43 and the code 'assert response == False'. The integrated terminal shows the command 'pytest -v' being executed, resulting in 8 tests passing. The output of the tests is as follows:

```
platform win32 -- Python 3.13.6, pytest-8.3.5, pluggy-1.6.0 -- C:\Users\VANQUISH\Desktop\Mejorcalculadora\.venv\Scripts\python.exe
rootdir: C:\Users\VANQUISH\Desktop\Mejorcalculadora
plugins: cov-6.2.1
collected 8 items

test_main.py::test_add_multiple_positive_integers PASSED [ 12%]
test_main.py::test_add_multiple_complicated_numbers PASSED [ 25%]
test_main.py::test_multiply_multiple_positive_integers PASSED [ 37%]
test_main.py::test_multiply_multiple_complicated_numbers PASSED [ 50%]
test_main.py::test_is_six_even PASSED [ 62%]
test_main.py::test_is_minus_three_point_eight_even PASSED [ 75%]
test_main.py::test_is_three_an_integer PASSED [ 87%]
test_main.py::test_is_seven_point_three_an_integer PASSED [100%]
```

The terminal also shows the command 'pytest -v' being executed and the output of the tests. The status bar at the bottom indicates the file is at line 43, column 27, with 974 characters selected, in UTF-8 encoding, using LF line endings, and the Python interpreter is set to '3.13.6 (.venv)'.