# Actividad relacionada con la lección 2:

**En esta actividad debe implementar un conjunto de test unitarios para validar el conjunto de funciones que implementó en la actividad de la Lección 1. Para desarrollar este conjunto de tests puede hacer uso de la herramienta unittest o pytest.**

**Condiciones de entrega:**

- **Entregue un documento en formato PDF en el que describa la utilidad de cada test unitario desarrollado.**
- **Entregue el código fuente.**

```
import unittest

import actividad


'''

Un programa que testea el conunto de valores que implementé en la Lección 1 con unos valores aleatorios

'''


class PruebasActividad(unittest.TestCase):


    def test_gastos(self): # Aquí testea los gastos de cada mes según los datos adjuntados

        x = [[1, 2, -8, 7], [3, -2, -1, 3], [2, 1, 1, 7], [-9, -2, -5, -7]]

        y = ['Enero', 'Feb', 'Marzo', 'Abril']

        resultado = {'Enero':8, 'Feb':3, 'Marzo':0, 'Abril':23}

        self.assertEqual(actividad.gastos(x, y), resultado)


    def test_ingresos(self): # Aquí testea los ingresos de cada mes según los datos adjuntados

        x = [[1, 2, -8, 7], [3, -2, -1, 3], [2, 1, 1, 7], [-9, -2, -5, -7]]

        y = ['Enero', 'Feb', 'Marzo', 'Abril']

        resultado = {'Enero':10, 'Feb':6, 'Marzo':11, 'Abril':0}

        self.assertEqual(actividad.ingresos(x, y), resultado)
```

```
    def test_ahorros(self): # Aquí testea los ahorros de cada mes según los datos adjuntados
        x = [[1, 2, -8, 7], [3, -2, -1, 3], [2, 1, 1, 7], [-9, -2, -5, -7]]
        y = ['Enero', 'Feb', 'Marzo', 'Abril']
        resultado = {'Enero':2, 'Feb':3, 'Marzo':11}
        self.assertEqual(actividad.ahorros(x, y), resultado)

    def test_mesMasGastos(self): # Aquí testea el mes con más gastos según los datos adjuntados
        x = [[1, 2, -8, 7], [3, -2, -1, 3], [2, 1, 1, 7], [-9, -2, -5, -7]]
        y = ['Enero', 'Feb', 'Marzo', 'Abril']
        resultado = ('Abril',23)
        self.assertEqual(actividad.mesMasGastos(x, y), resultado)

    def test_mesMasAhorros(self): # Aquí testea el mes con más ahorros según los datos adjuntados
        x = [[1, 2, -8, 7], [3, -2, -1, 3], [2, 1, 1, 7], [-9, -2, -5, -7]]
        y = ['Enero', 'Feb', 'Marzo', 'Abril']
        resultado = ('Marzo',11)
        self.assertEqual(actividad.mesMasAhorros(x, y), resultado)

    def test_GastoTotalAnio(self): # Aquí testea los gastos de todo el año según los datos adjuntados
        x = [[1, 2, -8, 7], [3, -2, -1, 3], [2, 1, 1, 7], [-9, -2, -5, -7]]
        y = ['Enero', 'Feb', 'Marzo', 'Abril']
        resultado = (34)
        self.assertEqual(actividad.GastoTotalAnio(x, y), resultado)

    def test_mediaGastosAnio(self): # Aquí testea la media de los gastos de todo el año según los datos adjuntados
        x = [[1, 2, -8, 7], [3, -2, -1, 3], [2, 1, 1, 7], [-9, -2, -5, -7]]
        y = ['Enero', 'Feb', 'Marzo', 'Abril']
```

```
        resultado = (2.83333333333333335)
        self.assertEqual(actividad.mediaGastosAnio(x, y), resultado)


    def test_IngresoTotalAnio(self): # Aquí testea los ingresos totales del año según los datos adjuntados
        x = [[1, 2, -8, 7], [3, -2, -1, 3], [2, 1, 1, 7], [-9, -2, -5, -7]]
        y = ['Enero', 'Feb', 'Marzo', 'Abril']
        resultado = (27)
        self.assertEqual(actividad.IngresoTotalAnio(x, y), resultado)


if __name__ == '__main__':
    unittest.main()
```