TRABAJO DE VALIDACIÓN Y VERIFICACIÓN

Pablo Guerra 6502
Mathew Avilés 6672
[redacted]
José López 6449

Ecuación de segundo grado

La ecuación de segundo grado es una ecuación cuadrática de una variable que tiene la forma de una suma algebraica en términos cuyo máximo grado es dos. Es decir, está representada por un polinomio de segundo grado. Cuenta con una variable, denominada x, y tres constantes, que son a, b y c.

Método de resolución

El método que nosotros utilizamos para resolver La ecuación de segundo grado es utilizando la fórmula general Y tiene la siguiente fórmula.

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Descripción del problema

El problema, el cual queremos resolver Es dar solución A las ecuaciones de segundo grado mediante el Uso de la fórmula general Y que el usuario ingrese Los valores de los coeficientes Y mediante métodos matemáticos Y la utilización de un Lenguaje de programación ¿Dar solución al problema.

¿Qué vamos a hacer?

Realizaremos un software con la capacidad de obtener La respuesta de un polinomio de segundo grado, Ingresando valores por teclado Y realizar Ciertas pruebas unitarias A nuestro programa.

Funcionalidades

1. El sistema sólo permitirá el ingreso de Datos numéricos.
2. El sistema deberá Verificar que el discriminante de la ecuación sea mayor o igual a cero.
3. El sistema verificará que el denominador de la ecuación sea distinto a cero.
4. El sistema nos dará como resultado las raíces reales y complejas de ser el caso.

Casos de prueba

1. El primer caso de prueba va a ser cuando el discriminante sea menor que cero.
   En este caso, como vamos a ver a continuación, nos dice que el resultado de la ecuación de segundo grado no son números reales, sino son números complejos.

Ecuación:

$$1 \cdot x^2 + 2 \cdot x + 3 = 0$$

El discriminante es:

$$\Delta = 2^2 - 4 \cdot 1 \cdot 3$$

$$\Delta = -8 < 0$$

Por tanto, no hay soluciones reales. Hay dos soluciones complejas:

$$x = \frac{-2 \pm \sqrt{2^2 - 4 \cdot 1 \cdot 3}}{2 \cdot 1} =$$

$$= \frac{-2 \pm \sqrt{-8}}{2} =$$

$$= \frac{-2 \pm \sqrt{8} \cdot i}{2} =$$

$$= -1.00000 \pm 1.41421 \cdot i$$

$$x_1 = -1.00000 + 1.41421 \cdot i$$

$$x_2 = -1.00000 - 1.41421 \cdot i$$

```python
import unittest

from raices_polonomio_grado_2 import PolinomioGrado2 as polinomio2


class Test_raices_polinomio_grado_2(unittest.TestCase):

    def test_uno(self):
        polinomio = polinomio2(1, 2, 3)
        # determina si los valores obtenidos con 2 cifras significaticas es acetado
        # en este caso los valores esperados son imaginarios y son correctos
        self.assertEqual(polinomio.calcularRaices(2), [-1+1.41j, -1-1.41j])
```

2. El segundo caso de prueba va a ser Que nosotros podemos controlar el número de decimales que queremos en el resultado de la ecuación cuadrática, Como podemos observar, en este caso nosotros queríamos que la respuesta nos dé con 3 decimales cómo se va a ver a continuación.

```python
    def test_dos(self):
        polinomio = polinomio2(1, 2, 3)
        # determina si los valores obtenidos con 3 (por defecto)
        # cifras significaticas es acetado
        # en este caso los valores esperados son imaginarios y son correctos
        self.assertEqual(polinomio.calcularRaices(), [-1+1.414j, -1-1.414j])
```

3. En el tercer caso de prueba nosotros asignamos los valores de 1, 6 y 2. Para obtener el resultado de la ecuación cuadrática, el cual va a ser un resultado real. Y Su respuesta va a ser con tres decimales.

Ecuación:

$$1 \cdot x^2 + 6 \cdot x + 2 = 0$$

El discriminante es:

$$\Delta = 6^2 - 4 \cdot 1 \cdot 2$$

$$\Delta = 28 > 0$$

Por tanto, hay dos soluciones reales distintas:

$$x = \frac{-6 \pm \sqrt{6^2 - 4 \cdot 1 \cdot 2}}{2 \cdot 1} =$$

$$x = \frac{-6 \pm \sqrt{28}}{2} =$$

$$= -3.00000 \pm 2.64575$$

$$x_1 = -0.35425$$

$$x_2 = -5.64575$$

4. el cuarto caso de prueba es cuando a la variable a le asignamos un valor de 0, como bien sabemos la división para 0 no está definida dentro de los reales y el darle el valor de cero a la variable a, nos indica que no es una ecuación de segundo grado por lo que nos arroja un mensaje de error.

Ecuación:

$$0 \cdot x^2 + 99 \cdot x + 99 = 0$$

No es una ecuación de segundo grado, ya que el coeficiente de a es cero.

```
    def test_excepcion(self):
        # determina si los al momento de ingresar 0 en el primer parametro
        # al declarar la clase se lanza una excepcionde tipo valueError
        with self.assertRaises(ValueError): polinomio2(0, 99, 99)

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

# ESPOCH 🦙 2022
# 6502, 6449, 6672, 6706
```

5. Caso de prueba en el que se ingrese una letra como variable al momento de ingresar los datos

```
def test_excepcion_2(self):
    with self.assertRaises(ValueError): polinomio2('d', 1, 3)
```

Al momento de ejecuta el programa nos salta esta excepción

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

Traceback (most recent call last):
  File "c:\Users\USUARIO\Documents\VyV\P1\Ejercicios-1\test_raices_polinomio.py", line 25, in test_excepcion_2
    with self.assertRaises(ValueError): polinomio2('d', 1, 3)
  File "c:\Users\USUARIO\Documents\VyV\P1\Ejercicios-1\raices_polonomio_grado_2.py", line 16, in __init__
    if isinf(a) or isinf(b) or isinf(c):
TypeError: must be real number, not str

----------------------------------------------------------------------
Ran 6 tests in 0.002s

FAILED (errors=1)
PS C:\Users\USUARIO\Documents\VyV\P1\Ejercicios-1>
```

Ejecución de las pruebas unitarias

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  GITLENS  JUPYTER  COMENTARIOS

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\lives> & C:/laragon/bin/python/python-3.10/python.exe c:/Users/lives/OneDrive/Desktop/Ejercicios/test_raices_polinomio.py
....
----------------------------------------------------------------------
Ran 4 tests in 0.005s

OK
PS C:\Users\lives>
```

MATRIZ DE TRAZABILIDAD

F1: Comprueba que los datos sean enteros

F2: Comprueba que el valor de A sea diferente de 0

F3: Comprueba que el resultado de la raíz sea positivo

F4: Comprueba que los datos de ingreso sean solo datos numéricos

Funcionalidades/casos de prueba	F1	F2	F3	F4
CP1: A=1, B=2, C=3	X		X	X
CP2: A=1, B=2, C=3	X		X	X
CP3: A=1, B=6, C=2	X	X	X	X
CP4:A=0, B=99, C=99	X	X		
CP5: A=D, B=1, C=3				X
CP6:A=∞,B=1,C=2	X			X

Código comentado del programa realizado en Python para resolver ecuaciones de segundo grado:

```python
import math

# clase para la excepcion presonalizada
class Excepcion(ValueError):
    def __init__(self, message, * args):
        super(Excepcion, self).__init__(message, * args)

class PolinomioGrado2:

    # propiedades que represetan los coeficientes de la ecuacion
    # estas propiedades son privadas
    __a: int
    __b: int
    __c: int

    # constructor que admite 3 parametros
    # ax^2 + bx + c
    def __init__(self, a: float, b: float, c: float) -> None:
        self.__asignarCoeficientes([a, b, c])

    # verifica y asigna los coeficientes en las propiedades privadas del objeto
    def __asignarCoeficientes(self, coeficientes: list[float]) -> None:
        # en caso de qu el primer coeficientes (a) se negativo se lanza una excepcion
        if coeficientes[0] == 0:
            raise Excepcion(
                f'El cofeiciente "a" debe ser diferente de 0.')
        self.__a = coeficientes[0]
        self.__b = coeficientes[1]
        self.__c = coeficientes[2]

    # permite calcular el discrimiante en base a los coheficientes
    # determinante = b^2 -4ac
    def calcularDiscriminate(self) -> float:
        return math.pow(self.__b, 2)-4*self.__a*self.__c

    # permite calcular las raices mediante la formula general
    def calcularRaices(self, decimales: int = 3) -> list[float]:
        discriminate = self.calcularDiscriminate()
        complejo = False
        # en caso que el discriminante sea 0 se trata las raices como complejas
        if (discriminate < 0):
            # se obtine la raiz de la parte positiva
            # y se asiga verdadero a la bandera complejo
            raiz = math.sqrt(abs(discriminate))
            complejo = True
        else:
            raiz = math.sqrt(discriminate)
        # se define dos terminos t1 y t2
        # t1 = -b/2*a
        # t2 = discriminante/2*a
        t1 = -self.__b/(2*self.__a)
        t2 = raiz/(2*self.__a)
        # en caso de que la bandera complejo sea verdadero mediante la funcion Complex
        # se crea un numero complejo donde t1 es la parte real t2 es la parte
imaginaria
        # ademas se redondean ambos terminos, por defecto el numero de decimales es 3
        if complejo:
            x1 = (complex(round(t1, decimales), round(t2, decimales)))
            x2 = (complex(round(t1, decimales), -round(t2, decimales)))
        # en caso contrario se suma los terminos t1 y t2 y se redondea
        else:
            x1 = (round(t1 + t2, decimales))
            x2 = (round(t1 - t2, decimales))
        # se retorna x1 y x2 en una lista
        return [x1, x2]

  }

# ESPOCH 🦙 2022
# 6502, 6449, 6672, 6706
```