Trabajo grupal o individual

Identificación del trabajo

a. **Módulo:** 2
b. **Asignatura:** Scripting
c. **RA:** Resultado de aprendizaje del módulo
d. **Docente Online:** Gonzalo Cárdenas
e. **Fecha de entrega**: 04-07-2023

Identificación del/los estudiante/s

Nombre y apellido	**Carrera**
William Huera	Técnico en Data Science

Introducción

Estás trabajando en el Área de Desarrollo de una empresa, como parte del equipo de control de calidad del software desarrollado internamente.

Se le encarga la tarea de crear un script desarrollado en Python, que será considerado como la guía del estándar interno del control de calidad, al revisar el correcto uso del lenguaje en términos de comparación, declaración de variables, y uso de funciones.

Este script debe, además, tener los comentarios que expliquen lo que se hace en cada línea del programa, de tal manera que sirva como guía para otros programadores.

El programa debe ser capaz de resolver una ecuación de segundo grado.

Una ecuación de segundo grado es una ecuación del tipo:

$$ax^2 + bx + c = 0$$

donde

a, b y c son números cualesquiera.

Problemática

Resolver lo siguiente:

Las soluciones que se pide obtener corresponden a las variables x_1 y x_2, con la siguiente fórmula:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Los pasos son los siguientes:

1. El programa debe mostrar por pantalla "RESOLUCIÓN DE ECUACIÓN DE SEGUNDO GRADO" e indicar que el valor de *a* debe ser distinto de cero.
2. Solicitar al usuario ingresar los valores de *a, b* y *c.*
3. $b^2 \quad 4ac$ Calcular el componente y guardarlo en una variable auxiliar llamada *aux.*
4. Calcular el valor de la *raíz* de *aux* y guardarlo en la variable "*aux2*"
5. Calcular el valor de (*-b + aux2*) guardándolo en la variable *temp1.*
6. Calcular el valor de *(temp1 / 2a),* guardándolos en la variable *x1.*
7. Mostrar el valor de *x1* desplegándolo por pantalla.

Desarrollo

Item1:

Código

```
mport math
print("RESOLUCIÓN DE ECUACIÓN DE SEGUNDO GRADO")
# Ingreso de variables
# mostramos en pantalla y realizamos un set a las variables definidas
# seteamos en la variable valor_a, el primer ingreso en consola
# seteamos en la variable valor_b, el segundo ingreso en consola
# seteamos en la variable valor_c, el tercer ingreso en consola

valor_a = int(input("Ingrese el primer valor: "))
valor_b = int(input("Ingrese el segundo valor: "))
valor_c = int(input("Ingrese el tercer valor: "))

# cálculo de componente en la variable aux vamos almacenar el cálculo (b^2 - 4ac)
# utilizamos la función pow para elevar al cuadrado el valor de la variable: valor_b
aux = (pow(valor_b, 2) - (4 * (valor_a * valor_c)))
print("valor de aux: ",aux)

# calulo valor raiz de aux
# para este caso utilizamos el operador sqr que nos devuelve el
# valor de la raiz cuadrada de un número
aux2 = math.sqrt(aux)
print("calulo valor raiz de aux: ",aux2)

# calculo (-b + aux2)
# en la variable temp1 vamos almacenar los valores obtenidos del cálculo de: (-b + aux2)
temp1 = ((valor_b * -1) + aux2)
print("valor temp: ",temp1)

## calcular el valor de (temp1 / 2a)
## para este cason utilizamos el operador división
x1 = (temp1 / (2 * valor_a))
print("valor x1: ",x1)

## tipos de datos:
print(f"tipo de dato variable valor_a: {type(valor_a)}")
```

```
print(f"tipo de dato variable valor_b: {type(valor_b)}")
print(f"tipo de dato variable valor_c: {type(valor_c)}")
print(f"tipo de dato variable aux: {type(aux)}")
print(f"tipo de dato variable aux2: {type(aux2)}")
print(f"tipo de dato variable temp1: {type(temp1)}")
print(f"tipo de dato variable x1: {type(x1)}")

# Operadores matemáticos usados:
# división, suma, resta, multiplicación, elevación al cuadrado
# raiz cuadrada
```

Git: https://github.com/whuera/scripting/blob/master/modulo2/m2-t1.py

Item2:

Determina el procedimiento de resolución de la ecuación de segundo grado, en una función que reciba como parámetros a, b y c, y que retorne un float con el valor de x1. Para esto, debes cumplir las siguientes tareas

Código

```
import math

# Definición de una función para calcular una ecuación de segundo grado
# variables de entrada: a, b, c
print("RESOLUCIÓN DE ECUACIÓN DE SEGUNDO GRADO")
def calcular_ecuacion_segundo_grado(a, b, c):
    aux = b**2 - 4*a*c

    if aux > 0:
        x1 = (-b + math.sqrt(aux)) / (2*a)
        x2 = (-b - math.sqrt(aux)) / (2*a)
        return x1, x2
    elif aux == 0:
        x = -b / (2*a)
        return x
    else:
        return "La ecuación no tiene respuesta reales."
```

```
# Ejemplo de uso
respuesta = calcular_ecuacion_segundo_grado(int(input("Ingrese el primer valor: ")),
                                            int(input("Ingrese el segundo valor: ")),
                                            int(input("Ingrese el tercer valor: ")))
print(respuesta)
```

Git: https://github.com/whuera/scripting/blob/master/modulo2/m2-t1-function.py

Item3:

Emplea las variables, operadores y función creadas en los pasos anteriores, para el cálculo de la ecuación, permitiendo al usuario ingresar las variables a, b, c y recibir el resultado de la ecuación.

- Solicitar al usuario ingresar 5 valores enteros y almacenarlos en un arreglo.
- Ordenar el arreglo de menor a mayor.
- Llamar a la función creada anteriormente, entregando los siguientes 3 valores: a es el menor número del arreglo, b es el mayor número del arreglo, c es el segundo menor número del arreglo.
- Escribir el resultado de la ecuación por pantalla

Código

```
# Definición de una función para calcular una ecuación de segundo grado
# variables de entrada: a, b, c
def calcular_ecuacion_segundo_grado(a, b, c):
    aux = b**2 - 4*a*c

    if aux > 0:
        x1 = (-b + math.sqrt(aux)) / (2*a)
        x2 = (-b - math.sqrt(aux)) / (2*a)
        return x1, x2
    elif aux == 0:
        x = -b / (2*a)
        return x
    else:
        return "La ecuación no tiene respuesta reales."
```

```python
# Crear un arreglo vacío
valores = []

# Pedir al usuario que ingrese los valores
for i in range(5):
    valor = input(f"Ingrese un valor {i}: ")
    valores.append(valor)  # Agregar el valor al arreglo
    array_order = sorted(valores)
    valor_a = array_order[0]
    valor_b = array_order.pop()

print("Los valores ingresados son:", array_order)
valor_c = array_order[1]
print("valor_c ", valor_c)
respuesta = calcular_ecuacion_segundo_grado(int(valor_a),
                                            int(valor_b),
                                            int(valor_c))
print(f"respuesta ecuación: {respuesta}")
```

Git: https://github.com/whuera/scripting/blob/master/modulo2/m2-t1-function-array.py

Conclusión

Trabajar con operadores matemáticos en Python es fundamental para realizar cálculos y operaciones numéricas. Algunas conclusiones clave de trabajar con operadores matemáticos en Python son las siguientes:

1. Claridad y expresividad: Python ofrece una sintaxis clara y legible para realizar operaciones matemáticas. Los operadores matemáticos como suma (+), resta (-), multiplicación (*), división (/) y exponente (**), entre otros, son intuitivos y fáciles de entender.

2. Precisión y flexibilidad: Python es capaz de realizar cálculos con diferentes tipos de datos, como enteros, flotantes y complejos, sin perder precisión. Además, ofrece operadores adicionales para trabajar con números enteros y flotantes, como la división entera (//) y el módulo (%).

3. Precedencia y paréntesis: Al igual que en las matemáticas, Python sigue reglas de precedencia de operadores para evaluar expresiones. Es importante tener en cuenta el orden de las operaciones y utilizar paréntesis cuando sea necesario para asegurar el cálculo correcto.

4. Biblioteca math: Python también cuenta con la biblioteca `math`, que proporciona una amplia gama de funciones matemáticas avanzadas, como trigonometría, exponenciales, logaritmos y más. Esta biblioteca amplía las capacidades matemáticas básicas de Python.

5. Manejo de errores: Al trabajar con operadores matemáticos, es importante tener en cuenta la posibilidad de errores, como divisiones entre cero o operaciones no válidas. Python maneja estos errores mediante excepciones, lo que permite identificar y manejar problemas en el cálculo de manera adecuada.

En resumen, trabajar con operadores matemáticos en Python ofrece claridad, precisión y flexibilidad en los cálculos numéricos. Python proporciona una sintaxis intuitiva y una amplia biblioteca matemática para realizar operaciones matemáticas avanzadas. Al utilizar los operadores matemáticos correctamente y tener en cuenta las reglas y precauciones necesarias, Python es una herramienta poderosa para trabajar con matemáticas y cálculos numéricos.

Bibliografía

https://www.w3schools.com/python/

Documentación guía, material de apoyo del módulo 2