

CHEMISTRY PROJECT REPORT

Significant Figures Calculator Using
Python (Tkinter GUI)

By: **GROUP 7**

Slot : **C11+C12+C13**

Subject: **CHEMISTRY**

Teacher: **SAURAV PRASAD**

Group members : **1. KARTIKEY MISHRA
2. AKSHAT SHUKLA
3. ANUSHA SINHA**

1. INTRODUCTION

In scientific measurements, accuracy and precision are extremely important. Chemistry often involves calculations based on measured quantities, which must be represented correctly using **significant figures**. To simplify this process, I created a **Python-based Significant Figures Calculator** with a user-friendly GUI.

This project helps students automatically calculate significant figures, apply correct rounding rules, and perform operations like addition, subtraction, multiplication, and division.

2. AIM OF THE PROJECT

To design and implement a Python program with a graphical interface that:

- Accepts two numbers
- Performs selected arithmetic operations
- Automatically applies significant figure rules
- Displays the rounded result along with an explanation

3. IMPORTANCE OF SIGNIFICANT FIGURES IN CHEMISTRY

- They indicate the **precision** of a measurement.
- Prevents false accuracy in calculations.

- Ensures reliable scientific communication.
- Helps maintain consistency in lab calculations.

Typical uses:

- ✓ Concentration calculations
- ✓ Molar mass
- ✓ Reaction yields
- ✓ pH and dilution equations

4. RULES FOR SIGNIFICANT FIGURES

General Rules

1. Non-zero digits are always significant.
2. Leading zeros are not significant.
3. Captive zeros are significant.
4. Trailing zeros are significant only if the number contains a decimal.
5. Scientific notation does not affect significant figure count.

Rules for Operations

Addition & Subtraction

The result must have the **least number of decimal places** among the inputs.

Number A: 12.35 → 2 decimal places

Number B: 4.2 → 1 decimal place

Least decimals = 1

Raw result → 16.55

Rounded → 16.6

Multiplication & Division

The result must have the **least number of significant figures** among the inputs.

Number A: 12.3 → 3 significant figures

Number B: 4.56 → 3 significant figures

Least sig figs = 3

Raw result → 56.088

Rounded to 3 sig figs → 56.1

5. PROJECT OVERVIEW

This project is a desktop application made using **Python Tkinter**.
The program:

- Takes inputs A and B
- Lets you choose +, −, ×, or ÷
- Calculates the raw mathematical result
- Counts significant figures or decimal places
- Applies correct rounding
- Shows rule explanation and rounded output

6. TECHNOLOGIES USED

Component	Purpose
Python	Programming language
Tkinter	GUI (Graphical User Interface)
math module	Logarithm used for rounding significant figures
ttk widgets	Modern styled interface

7. FLOW OF THE PROGRAM

User enters numbers →

Selects operation →

Program calculates raw value →

Counts sig figs or decimal places →

Applies chemistry rounding rules →

Displays final rounded result with explanation

8. EXPLANATION OF CODE (Insert screenshots here)

Below I'll structure the sections so you can place screenshots easily:

8.1 Importing Libraries

```
import tkinter as tk
from tkinter import ttk, messagebox
from math import log10, floor
```

Explanation:

- `tkinter` and `ttk` are used to build the application's GUI.
- `messagebox` handles error alerts.

- `log10` and `floor` help in rounding to significant figures.

8.2 Function: `count_sig_figs()`

```
def count_sig_figs(num_str: str) -> int:
    """Count significant figures following standard chemistry/physics rules."""
    s = num_str.strip().lower()
    if 'e' in s:
        s = s.split('e')[0]
    if s and s[0] in '+-':
        s = s[1:]
    if not s or s == '.':
        return 0
    sig_count = 0
    if '.' in s:
        for c in s:
            if c.isdigit():
                sig_count += 1
    else:
        started = False
        for c in s:
            if c.isdigit():
                if not started and c == '0':
                    continue
                sig_count += 1
                started = True
    return sig_count
```

Purpose:

- Reads the number as a string.
- Removes signs and scientific notation.
- Counts significant digits according to standard rules.

8.3 Function: `round_sig_figs()`

```
def round_sig_figs(x: float, n: int) -> float:
    """Round to n significant figures."""
    if x == 0 or n <= 0:
        return 0.0
    if n >= 16:
        return x
    power = floor(log10(abs(x)))
    factor = 10 ** (power - n + 1)
    return round(x / factor) * factor
```

Purpose:

- Rounds a given number **x** to **n** significant figures using logarithms.
-

8.4 Function: decimals_after_point()

```
def decimals_after_point(num_str: str) -> int:
    """Count decimal places (least precise position)."""
    s = num_str.strip()
    if 'e' in s.lower():
        s = s.lower().split('e')[0]
    if '.' not in s:
        return 0
    decimal_part = s.split('.')[1]
    return len(decimal_part.rstrip('0'))
```

Purpose:

- For addition/subtraction: counts number of digits after the decimal point.

8.5 calculate() function

```
def calculate():
    a_str = entry_a.get().strip()
    b_str = entry_b.get().strip()
    op = op_var.get()
    try:
        a = float(a_str)
        b = float(b_str)
    except ValueError:
        messagebox.showerror("Error", "Enter valid numbers for A and B.")
        return
    try:
        if op == "+":
            raw = a + b
        elif op == "-":
            raw = a - b
        elif op == "x":
            raw = a * b
        elif op == "÷":
            raw = a / b
        else:
            messagebox.showerror("Error", "Select an operation.")
            return
    except ZeroDivisionError:
        messagebox.showerror("Error", "Division by zero.")
        return
```

```

label_raw.config(text=f"Raw result: {raw:.12g}")
sig_a = count_sig_figs(a_str)
sig_b = count_sig_figs(b_str)
if op in ["×", "÷"]:
    sig_result = min(sig_a, sig_b)
    if sig_result > 0:
        rounded = round_sig_figs(raw, sig_result)
        rule_text = (f"Rule: multiplication/division → "
                    f"minimum sig figs between inputs = {sig_result}")
        rounded_str = f"{rounded:.12g} (rounded to {sig_result} sig figs)"
    else:
        rounded = raw
        rule_text = "Rule: Invalid sig figs in inputs"
        rounded_str = str(rounded)
else:
    dec_a = decimals_after_point(a_str)
    dec_b = decimals_after_point(b_str)
    min_dec = min(dec_a, dec_b)
    rounded = round(raw, min_dec)
    rule_text = f"Rule: addition/subtraction → min decimals = {min_dec}"
    rounded_str = f"{rounded:.12g} (rounded to {min_dec} decimals)"
label_rule.config(text=rule_text)
label_rounded.config(text=f"Rounded result: {rounded_str}")

```

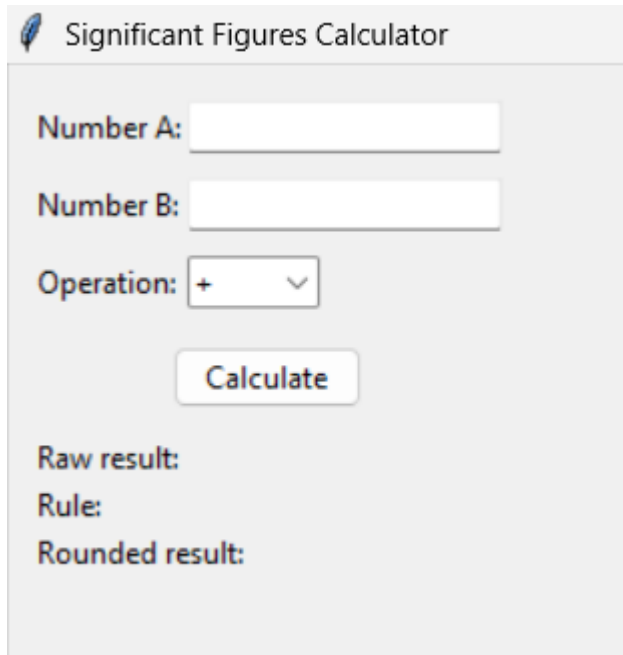
What it does:

- Reads inputs
- Performs operation
- Counts sig figs or decimals
- Produces final rounded value
- Updates GUI labels

9. GUI LAYOUT & WORKING

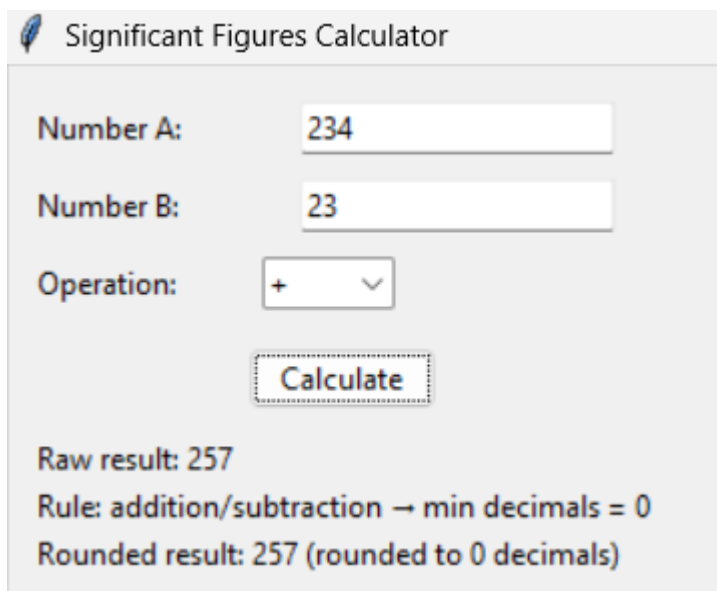
Include:

- App



A screenshot of a software application titled "Significant Figures Calculator". The interface includes two input fields labeled "Number A:" and "Number B:", both currently empty. Below these is a dropdown menu for "Operation:" showing a "+" sign. A "Calculate" button is positioned below the operation menu. At the bottom of the window, there are three labels: "Raw result:", "Rule:", and "Rounded result:", all of which are currently empty.

- Entering values



A screenshot of the same "Significant Figures Calculator" application, now with values entered. "Number A:" contains "234" and "Number B:" contains "23". The "Operation:" dropdown still shows "+". The "Calculate" button is highlighted with a dashed border. The output section at the bottom now displays: "Raw result: 257", "Rule: addition/subtraction → min decimals = 0", and "Rounded result: 257 (rounded to 0 decimals)".

- Rules applied

Rule: addition/subtraction → min decimals = 0
Rounded result: 257 (rounded to 0 decimals)

10. TESTING & SAMPLE CALCULATIONS

Input A	Input B	Operation	Expected Rule	Final Output
12.3	4.56	×	min sig figs = 3	56.1
3.450	2	+	min decimals = 0	5
0.00670	5	×	min sig figs = 2	0.033

11. CONCLUSION

This project successfully automates the otherwise confusing rules of significant figures. It makes chemistry calculations easier, reduces human error, and provides an interactive way to learn the concept.

It demonstrates how computer science can support scientific accuracy and education.

