Slide 1: Title - Calculation Errors

Hello everyone! Today, I'll be talking about calculation errors—mistakes that can disrupt our programs and how to identify them.

Slide 2: Description and Example

Calculation errors arise from mistakes in computations. These can stem from incorrect logic, misuse of operators, or lack of precision. For example, let's look at this code: we try to divide 10 by 0. As you can see, it throws a "Division by zero is not allowed" error, halting execution. Proper error handling, like using try-except, is crucial here.

Slide 3: Types of Calculation Errors

Let's break it down. First, incorrect implementation of computational logic—when the algorithm itself is flawed. Next, mistakes in using operators, like confusing multiplication with addition. Then, lack of precision in calculations, which can lead to rounding errors. We also have data type mismatches, where incompatible types cause failures. Finally, overflow and underflow occur when numbers exceed or fall below a system's limits.

[Pause for emphasis] These errors are common but preventable with careful coding and testing. Always validate inputs, use appropriate data types, and handle exceptions to keep your programs running smoothly.

Thank you!

Presentation Script: CALCULATION ERRORS

Slide 1 – Introduction to Calculation Errors

Hello everyone,
Now I'll be talking about **Calculation Errors**, which are a very common category of software errors.

These errors arise from **mistakes made during calculations** in the program.
They usually stem from things like:

- Misuse of mathematical operators,
- Poor logic implementation,
- Or simply a lack of precision in how values are handled.

In the example shown here, we have a small Python code snippet that attempts to divide 10 by 0.
This results in a **ZeroDivisionError**, which is a classic type of calculation error.

Although this seems simple, in real-world systems, even small calculation mistakes like this can lead to **critical system failures**, **data corruption**, or **financial loss**.

Slide 2 – Types of Calculation Errors

Now let's look at some common causes of calculation errors:

1. **Incorrect implementation of computational logic**

   This refers to writing the wrong formula or algorithm—for example, implementing area = 2 * π * r instead of π * r^2.

2. **Mistakes in using operators**

   A common mistake is using = instead of == in condition checks, or misusing operator precedence like adding before multiplying.

3. **Lack of precision in calculation**

   This happens especially in scientific or financial applications where floating-point rounding errors can accumulate and lead to incorrect results.

4. **Data Type Mismatch**

For example, dividing two integers when you actually need a floating-point result—like `5 / 2` giving `2` instead of `2.5`.

5. **Overflow and Underflow**

   These occur when a number exceeds the data type’s limit. For instance, trying to store a very large number in a small integer type can cause overflow.

All of these issues can be avoided—or at least minimized—through **proper testing**, **code reviews**, and **careful implementation of logic**.

So to summarize, **calculation errors may look small**, but their impact can be significant if they go undetected.
That’s why careful attention to logic, data types, and precision is essential when implementing computational code.

Thank you!