The *try* and *catch* Blocks

To handle and recover from errors, you need a combination of `try` and `catch` blocks. A `try` block is simply a block of code where runtime errors might occur; write the keyword `try`, and then surround the appropriate code in a pair of curly braces, like this:

```
string str;
cin >> str;
try {
    int a = stoi(str);
    cout << "a = " << a << endl;  // skipped if exception thrown
}
catch (const invalid_argument& e) {
    cerr << e.what() << endl;     // if exception thrown
}
```

There are three things to note here.

1. If the user enters `"one"` then `stoi()` will throw an exception, and control **immediately** breaks out of the `try` block and **jumps** to the `catch` block, skipping the line that prints `a`. We're guaranteed that **the rest of the code in the `try` block will not execute**, preventing error cascades.
2. If an exception is thrown and caught, control **does not return** to the `try` block. Instead, control resumes directly **following** the `try`/`catch` pair.
3. Catch exception classes from the standard library **by const reference**. This avoids making copies and **enables polymorphism**. You will get a compiler warning if you don't do this.

If no error occurs, then all of the code inside the `try` block executes as normal, and the subsequent `catch` block is ignored. The function `what()` will return the `string` used to construct the exception object. Here, its used to print the error message in the `catch` block.