Semantics (XII)

Exception Semantics (cont.)
- Python example: If an exception occurs in the try statement, execution moves to the except block if the exception matches one of the exceptions listed.
  - The syntax permits multiple except cases. An except block can also list multiple exceptions inside a tuple, e.g. (RuntimeError, TypeError, NameError, ValueError).
  - An except block with no arguments catches all exceptions not explicitly caught by other except cases.
  - The equivalent of throw in Python is the raise keyword. The raise statement takes the exception class as an argument. [Show trycatch.py]
  - Python also includes a finally clause like Java.

```python
def demo1():
    a = 0
    while a == 0:
        try:
            s = input('enter a number: ')
            val = int(s)
        except ValueError:
            print("not a valid number")
            continue
        if val == 0:
            a = 1
        print("terminating")

def demo2_helper():
    raise 1

def demo2():
    try:
        demo2_helper()
    except:
        print("demo 2 catching the error")
demo1()
#demo2()
```

Comment out demo2() and run the code
$ python3 trycatch.py

enter a number: a
not a valid number
enter a number: 1
enter a number: 0
terminating

Comment out raise inside the except of demo2() function, and run again
$ python3 trycatch.py
demo 2 catching the error
- C++ example: The exceptions can be any type (std::exception, int, etc.). The throw statement essentially calls the catch block. [Show trycatch.cc]

```c++
#include <cstdio> //stdio.h in C
int main ( int argc, char *argv[]) {
    for (;;) {
        int q;
        try {
            printf("Enter a number (0..9): ");
            int k = scanf("%d", &q); // read formatted from stdin
            if (k == 0) {
                scanf("%*s"); //read the value but ignore
                throw "value is not a number";
            }
        //this part is unnecessary, just show different types of exceptions
            if (q < 0) {
                throw 3.0;
            }
            if (q < 0 || q > 9) {
                throw q;
            }
        } catch (const char *s) {
            printf("Error: %s\n", s);
            continue;
        } catch (int v) {
            printf("Error: number %d is out of range (0..9)\n", v);
            continue;
        } catch (...) {
            printf("Unspecified error\n");
            continue;
        }
        if (q == 0)
            break;
    }
    return 0;
}
```

$ ./a.out
Enter a number (0..9): a
Error: value is not a number
Enter a number (0..9): 10
Error: number 10 is out of range (0..9)
Enter a number (0..9): -1
Unspecified error
Enter a number (0..9): 0
$