Exceptions

See pages 216 to 219 of the text.

Your mother probably thinks you are an *exceptional* child and this is a good thing.

An *exception* in computer terms is not a good thing -- it represents unexpected behavior of a program, usually something that would cause the program to crash.

Consider this program:

```
def divider(x):
    print( "100/%d = %d."%(x, 100//x))

def main():
    done = False
    while not done:
        number = eval(input( "Enter a positive number, or a
                                                    negative to exit: " ))
        if number < 0:
            done = True
        else:
            divider(number)
main()
```

If you enter 5 it prints "100/5 =20." If you enter -1 it terminates.

But if you enter 0 it does this:

```
Traceback (most recent call last):
  File "C:/Users/bob/Documents/Classes/cs150-spring15/Class Examples and
Notes/February/February 23/crasher.py", line 12, in <module>
    main()
  File "C:/Users/bob/Documents/Classes/cs150-spring15/Class Examples and
Notes/February/February 23/crasher.py", line 11, in main
    divider(number)
  File "C:/Users/bob/Documents/Classes/cs150-spring15/Class Examples and
Notes/February/February 23/crasher.py", line 2, in divider
    print( "100/%d = %d"%(x, 100//x))
ZeroDivisionError: integer division or modulo by zero
```

Obviously we divided by 0 and you can't do that so the program crashed. With this simple program it wouldn't be hard to put in if-statements that tested for the condition and prevented the crash.

With more complex programs it is hard to put in code that checks for every conceivable input some doofus of a user could think of using. On the other hand, **PROGRAMS YOU WRITE FOR SOMEONE ELSE SHOULD NEVER, EVER, EVER CRASH**.

Most programming languages have a feature that allows you to catch exceptions and (hopefully) deal with them instead of allowing the program to crash.

In Python this is the try-except statement.

Here is the form of a try-except statement:

```
try:
        <code>
except <exception name>:
        <what to do in an exception>
```

One fix for our program is to change the *divider* function to

```
def divider(x):
    try:
        print( "100/%d = %d."%(x, 100//x))
    except ZeroDivisionError:
        print( "We can't divide by 0." )
```

Note that we get the name of the exception from the error message that results when we don't catch it. The last line of that message was

ZeroDivisionError: integer division or modulo by zero

Where to put a try-catch statement is more an art than a science. Exceptions are passed up from called function to the caller, and the program only crashes if they aren't caught somewhere. Here is a new version of our program that catches several exceptions:

```
def divider(x):
    print( "100/%d = %d"%(x, 100//x))

def main():
    done = False
    while not done:
        try:
            number = eval(input( "Enter a positive number, or a negative to exit: " ))
            if number < 0:
                done = True
            else:
                divider(number)
        except SyntaxError:
            print( "I didn't understand that input." )
        except ZeroDivisionError:
            print( "We can't divide by 0." )
        except:
            print( "I'm confused." )
main()
```

Note that an except block of the form

```
except:
    <exception handling code>
```

(with no exception name) catches all exceptions. This is too general to be useful in many situations except as a last resort.

What will this program print if we enter x at the prompt?

```
def main():
    x = 23
    x = eval(input( "Enter a number: " ))
    print( x)

main()
```

A: It will crash because you can only enter numbers at an eval(input...) prompt.

B. It will crash because the system can't evaluate **x**.

C. 23

D. 0

Here is a program that crashes if we enter 0:

```
def main():
    done = False
    while not done:
        x = eval(input( "Enter a number: " ))
        if x < 0:
            done = True
        else:
            print( "The reciprocal of that number is %.2f"%(1/x))

main()
```

Where do we put the try-except block to prevent crashes?

A) Before def main()

B) *try* before the while loop, *except* after the loop

C) *try* before print statement *except* after it.

D) *try* before x=eval, *except* at the end of the loop