

# ACOM1900 Intro. to Programming:

## Exception Handling

Roy Ruddle

with thanks to Tony Jenkins & Nick Efford  
(CR11 and previous ACOM1900 material)

1

University of Leeds

Roy Ruddle

## Objectives

- How to make our programs more *robust*
  - Consider the nature of an "error"
  - Look at handling errors using conditional statements
  - Extend this to consider "Exceptions"

2

University of Leeds

Roy Ruddle

## The idea

- Programs must be error-tolerant
  - Unexpected situations
  - or user may not provide expected input
  - Often requires a lot of code!
- How?
  - Check function return codes
  - Handle exceptions

3

University of Leeds

Roy Ruddle

## Example

if everything is fine:  
do the operation

```
if bottom != 0:  
    print top / bottom  
else:  
    print 'Error!'
```

try:  
do this

```
try:  
    print top / bottom  
except:  
    print 'Error!'
```

if it all goes wrong:  
deal with it

4

University of Leeds

Roy Ruddle

## What can go wrong ...

```
# A program to compute a square-root  
# NDE, 2007-09-29  
import math  
# Read floating-point number from the keyboard  
number = float(raw_input('Enter a number: '))  
# Compute and display result  
result = math.sqrt(number)  
print 'The square-root of', number, 'is', result
```

5

University of Leeds

Roy Ruddle

## Exception

- Signal that exceptional condition has occurred
- Interrupt normal flow of control in program
  - Will halt program if not intercepted

```
try:  
    # Code that may generate exception  
except:  
    # Handle exception  
else:  
    # If exception wasn't thrown  
finally:  
    # Executed no matter what
```

6

University of Leeds

Roy Ruddle

## Types of exception

- Telles pp. 227-228
  - ImportError
  - KeyboardInterrupt
  - ZeroDivisionError
  - and many more ...

7

University of Leeds

Roy Ruddle

## Multiple exceptions

```
while True:
    try:
        v = int( raw_input("Enter a value: ") )
        x = 100 / v
        break
    except KeyboardInterrupt:
        print "<ctrl> C pressed"
        break
    except ZeroDivisionError:
        print "You can't divide by zero"
    except:
        print "Some other error occurred"
```

8

University of Leeds

Roy Ruddle

## Quiz: Find the two errors in this code

```
if (number > 1000 or number < 0):
    print 'Number out of range'
elseif number != 7
    print 'Not a lucky number'
```

9

University of Leeds

Roy Ruddle

## Raising your own exceptions

```
# Function that calculates factorial
def calcFactorial( value )
    if value < 0:
        raise ValueError
    # etc.

# Main program
try:
    fact( -1 )
except ValueError:
    print "Can't take factorial of -ve number"
```

10

University of Leeds

Roy Ruddle

## Quiz: What is wrong here?

```
name = raw_input('Who are you?')
if name = 'Nick':
    print 'Hello, Nick!'
else:
    print "I don't know you..."
```

11

University of Leeds

Roy Ruddle

## Summary

- Identifying exceptions (program design)
- Handling exceptions (building robust programs)
- Assignment
  - Complete **Exception Handling** worksheet

12

University of Leeds

Roy Ruddle