Py tut §8: Exceptions

7 Nov 2008
CMPT14x
Dr. Sean Ho
Trinity Western University

What's on for today (Py tut 8)

- Exceptions:
  - Handling
  - Raising
  - else
  - finally
  - User-defined exceptions
  - Passing auxiliary data with an exception

Options for error handling

- Use a combination of these:
  - Ask the user to be nice:
    - User manual, precondition comments, prompts
  - Print an error message to screen
  - Set a result flag:
    - e.g., return False upon error
  - Panic and die: sys.exit()
  - Raise an exception: ZeroDivisionError

Exceptions

- Exceptions are a way of terminating execution of the current context
- When an exception is raised (thrown),
  - execution of the current procedure stops, and
  - Control jumps to the nearest exception handler (catches the exception)
- The exception handler can cleanup
- Execution then continues after that block
- If the exception reaches outermost level, an error message is automatically generated

try / except

- If an exception is raised within a try block,
- Execution of the block terminates and control jumps to the except clause:

```
try:
    while True:
        numer = input('Numerator: ')
        denom = input('Denominator: ')
        print '%d / %d = %d' % (numer, denom, numer /
          denom)
except:
    print 'Oops!'
```

Catching specific exceptions

- We can opt to catch only specific exceptions:

```
try:
    while True:
        numer = input('Numerator: ')
        denom = input('Denominator: ')
        print '%d / %d = %d' % (numer, denom, numer /
          denom)
except ZeroDivisionError:
    print 'Oops! Divide by zero!'
```

- Any other exception falls through to the next exception handler

Handling exceptions

- The standard math.sqrt() raises ValueError on a negative argument:
  - from math import sqrt
  - sqrt(-1)   # ValueError
- We can handle this:
  - try:
    - num = input('Find sqrt of: ')
    - result = sqrt(num)
    - print 'The square root is', result
  - except ValueError:
    - print "Can't take square root of", num

Raising exceptions

- We can force exceptions to be raised:
  - try:
    - while True:
      - if input('Guess a number: ') == 5:
        - raise ZeroDivisionError
  - except ZeroDivisionError:
    - print 'You got it!'
- Within a handler, can re-raise the current exception:
  - try:
    - raise ZeroDivisionError
  - except ZeroDivisionError:
    - print 'oops, divided by zero!'
    - raise # raises ZeroDivisionError

'else' clauses for exceptions

- The optional else clause is executed only if the try block completes without throwing any exceptions:
  - try:
    - for tries in range(3):
      - if input('Guess a number: ') == 5:
        - raise ZeroDivisionError
  - except ZeroDivisionError:
    - print 'You got it!'
  - else:
    - print 'Too bad, you ran out of tries!'

'finally' clauses for exceptions

- The optional finally clause is always executed before leaving the section, whether an exception happened or not.
  - try:
    - for tries in range(3):
      - if input('Guess a number: ') == 5:
        - raise ZeroDivisionError
  - except ZeroDivisionError:
    - print 'You got it!'
  - else:
    - print 'Too bad, you ran out of tries!'
  - finally:
    - print 'Bye!'

User-defined exceptions

- Like everything else in an OO language, exceptions are objects: instances of the Exception class.
- You can define your own exceptions by making a subclass of the Exception class:
  - **class MyException(Exception):**
    - **pass**
- Make an instance of your class and raise it:
  - **myEx1 = MyException()**
  - **raise myEx1**
  - **raise MyException()**

Passing data with an exception

- Override __init__ to add an instance variable:
  - class MyException(Exception):
    - def __init__(self, tries=0):
      - self.numtries = tries
- Now we can package auxiliary data with the exception, using the constructor:
  - raise MyException(5)
- Unpack the data in the handler:
  - except MyException, e:
    - print '%d tries' % e.numtries
  - Second param e refers to the exception instance

Example: user-defined exception

- class MyException(Exception):
  - def __init__(self, t=0):
    - self.numtries = t
- try:
  - for tries in range(1, 6):
    - if input('Guess a number: ') == 5:
      - raise MyException(tries)
- except MyException, e:
  - print 'You got it in only %d tries!' % e.numtries
- else:
  - print 'Too bad, you ran out of tries!'

More info on exceptions

- The Python tutorial is a good resource:
- http://docs.python.org/tut/node10.html
-