Programming for Science
Possible answers for workshop 8

Here are possible answers to the exercises in the Workshop 8. Usually these are not the only way of solving the problem, but they are relatively straightforward and clean ways.

1. Here's code to build a list of numbers and find the average. Note that the list should be reset to empty each time the average is calculated.

```
def average(L):
    """
    Return the average of the numbers in the list.
    """
    S = 0.0
    for x in L:
        S += x
    return S/len(L)


L = []                                  # List of numbers so far
while True:
    try:
        s = raw_input('Enter a number or "average" or "q": ')
        print s
        if s[0] == 'q' or s[0] == 'Q':
            break
        elif s == 'average':
            if len(L) == 0:
                print 'No numbers to average yet'
                continue
            av = average(L)
            print 'Average is', av
            L = []                      # Start again
        else:
            x = float(s)
            L.append(x)

    except ValueError:
        print 'Cannot convert', s
    except EOFError:
        print 'Exiting on end of file.  Bye!'
        break
    except:
        print 'Caught an exception! Continuing'
```

2. No code for this yet.

3. Here's code to run and catch the exceptions raised by the bomb.

```
from time import sleep
from dice import roll, choose

def bomb():
    """
    Raise an exception after a waiting a few seconds
    """
    wait = roll()
    for i in range(wait, 0, -1):
        print i
        sleep(1)

    if roll() == 1:
        print "Bomb didn't explode"
        return
    message = choose(['jelly', 'gelignite', 'fruit', 'TNT', 'atom',
                      'bath']) + ' bomb'
    raise RuntimeError, message

try:
    print 'Oops! Dropped the bomb'
    bomb()
except  RuntimeError, details:
    print 'The', details, 'exploded'
```

4. Here is `factorial` equipped to raise exceptions and some test code.

```
def factorial(n):
    """
    Compute the factorial of n (for integer n >= 0).
    """
    if not isinstance(n, int):
        raise ValueError, "factorial takes integer arguments"
    if n < 0:
        raise ValueError, "negative argument to factorial"

    if n == 0:
        return 1
    else:
        return n*factorial(n-1)

def test_factorial():
    """
    Tests for factorial
    """
    assert factorial(5) == 120
```

```
    # Edge cases
    assert factorial(0) == 1
    assert factorial(1) == 1
    print 'Factorial passed non-negative integer tests'

    try:
        factorial(3.141)
    except ValueError, detail:
        print 'Caught ValueError for floating point argument:', detail

    try:
        factorial(-2)
    except ValueError, detail:
        print 'Caught ValueError for negative argument:', detail

if __name__ == "__main__": # when run as a script
    test_factorial()
```

5. Here's a program to get information about postcodes. It's not terribly robust, but illustrates the idea.

```
import urllib

def postcode_info(pc):
    """
    Given the major and minor parts of a postcode (eg ['EX4', '4QF']) prin
    the postcode's latitude, longitude and district by looking it up on
    http://www.doogal.co.uk/UKPostcodesCSV.php?Search=EX20
    """
    try:
        url = "http://www.doogal.co.uk/UKPostcodesCSV.php?Search="+pc[0]
        connection = urllib.urlopen(url)
    except:
        print 'Could not open', url
        return None

    for i in range(len(pc)):
        pc[i] = pc[i].upper()

    for line in connection.fp:
        field = line.split(',')
        postcode = field[0].split()
        if len(postcode) > 1 and postcode[1] == pc[1]:
            print '-'*10, pc[0], pc[1], '-'*10
            print 'Latitude and longitude', field[1], field[2]
            print 'District:', field[6]
            return
```

```
    print 'Could not find', pc[0], pc[1], 'in', url
    return


while True:
    try:
        s = raw_input('Enter a postcode with a space, eg EX4 4QF: ')
        if len(s) == 0:
            continue
        if len(s) == 1 and (s[0] == 'q' or s[0] == 'Q'):
            break
        pc = s.split()
        if len(pc) != 2:
            print 'Could not split postcode into major and minor parts', s
            continue

        postcode_info(pc)

    except ValueError:
        print 'Cannot convert', s
    except EOFError:
        print 'Exiting on end of file.  Bye!'
        break
```