

Debugging

Week 5, day 4

But first ...

- Neat programs vs working programs
- Multiple levels of referencing
 - e.g. `theData._dict['PRO'].coord[5]`
 - exposes too much implementation

Bugs

- Are inevitable
- Are ubiquitous
- Are inescapable
- Are part of programming

Syntax errors

- Grammar & punctuation mistakes
- Translation failure
- Not really a problem ...

```
def fibo (n:  
    """Calculates fibonacci value  
    for numbers > 0"  
    if (n in [1, 2])  
        return 1  
    else:  
        return fibo(n-1) + fibo(n-2)
```

Semantic errors

- Errors of meaning
- Programmer errors
- Major problem

```
# list squares from 1 to 10
for i in range (1, 10)
    print i**2.0,
```

Runtime errors

- Subclass of semantic errors
- Sometimes called “exceptions”
- Exceptional or unexpected

```
x = range (5)
print x, len (x)
print x[10]
```

Infinite loops

- Failure to terminate loop
- Esp. floats
- Goes forever ...

```
# print all the odd
# numbers less than 10
i = 1
while (i != 10)
    print i
    i += 2
```

Infinite recursion

- Failure to appropriately terminate recursion / specify base case
- “Maximum recursion depth exceeded”

```
def fibo (n):  
    """  
    When n is 2, this  
    function calls  
    ``fibo (0)`` which  
    recurses infinitely.  
    """  
    if (n == 1):  
        return 1  
    else:  
        return fibo (n-1) + \  
            fibo (n-2)
```

Fencepost / off-by-one error

- Measure a fence by posts or the number of cross-beams?
- Mistake bounds for length
- Under- or overshoot
- `range()`

Math errors

- Mostly result of limited representation of numbers
- DivideByZero
- Underflow, overflow
- Solutions:
 - test for zero
 - `sys.maxint`

Pythonic errors

```
# extract names from search results
theNames = []
for entry in theSearchResults:
    theNames.append (entry.title)
    print "There were", len (theNames), \
"matches found:" theNames
```

- Inconsistent indentation
- tabs & spaces
- ImportError

Steps to debugging

- Is there an error?
- What is the error?
- When / where does the error occur?
- Why does the error occur?
- Correct the error

The diagnostic print

```
class ReactionNetwork (object):  
    [...]  
  
    def addNewActor (a):  
        print "I'm in addNewActor!"  
        [...]  
  
    def step (nsteps):  
        print "I'm in step with %s" % nsteps  
        [...]
```

Dump

```
class Graph:
    "Adjacency list based graph"
    def __init__(self):
        self._nodeDict = {}

    [...]

    def dump (self):
        print "dumping graph object"
        for k,v in self._nodeDict.iteritems():
            print "Node %s:list %s" % (k, v)
```

Program defensively

- Catch or trigger errors at earliest opportunity
- Preconditions & postconditions
- Asserts
- Test & throw

Assorted strategies

- Always be in a valid state
- Examine objects via prompt
- Logging

Validate

```
class Graph:
    "Adjacency list based graph"
    def __init__(self):
        self._nodeDict = {}
    [...]

    def validate (self):
        for k,v in self._nodeDict.iteritems():
            for neighbour in v:
                nlist = self._nodeDict[neighbour]
                assert (k in nlist), """"%s does
                    not have %s in its list"""" % (
                    neighbour, k)
```

Logging

- Standard module
- Ways of sending messages of different levels to different logfiles
- Better than rolling your own
- Can subclass/customise
- Crapulous documentation

Logging example

```
import logging

logging.basicConfig (filename='mylog.txt',
                    format='%(levelname)s: %(asctime)s, %(message)',
                    level=logging.DEBUG,
                    )
logging.debug ("I'm debugging")
logging.critical ("This is %s", "critical")
```

Logging API

- Initialise with `logging.basicConfig()` & args:
 - filename
 - filemode (default 'a')
 - format
 - datefmt
 - level

CRITICAL
ERROR
WARNING
INFO
DEBUG

Logging API 2

- Calls for every severity level (logging.critical(), logging.error() etc.)
- Calls have the form logging.call (fmtstr, *args)
- Put in log file in given format

CRITICAL
ERROR
WARNING
INFO
DEBUG

Resources

- A real Python logging example at <http://antonym.org/node/76>
- Simple use of Python's logging module at http://www.mechanicalcat.net/richard/log/Python/Simple_usage_of_Python_s_logging_module
- Debugging in the Python cookbook <http://aspn.activestate.com/ASPN/Cookbook/Python?kwd=Debuggin>