Scopes & Namespaces continued

Modules, Classes, Exceptions

Recall ...

Local namespace: the current executing function

Global namespace: the module in which this function was defined

Named namespaces: Referenced from global or local namespace

import grid
def fill(cave, row, cell):
...
fill(cave, row+1, col)

dict is global (in the game_dict module) so that we can read it once and use it each time we call search()

# FIXME: Search the dictionary
Classes and objects: A peek at boggle_board.py

class BoggleBoard(object):
    def __init__(self, tiles):
        ...
        self.content = []
        self.in_use = []
        grid.make(4, 4, 500, 500)
        ...
    def get_char(self, row, col):
        ...
        return self.content[row][col]
    def mark_taken(self, self, row, col):
        ...
        self.in_use[row][col] = True
        ...

Aside ... exceptions

What if the user gives a file name that doesn’t exist?

What if the user types letters when a number is expected?

We could check for everything, everywhere, but our code would get awfully messy.

Exceptions help us organize the extra code and make it easier to read.

Coming next week ... classes

Namespaces are great. I want more! More!

(And I don’t want to write a different module file for each one.
I want to have a bunch of objects, each with their own namespace.)

A class is a lot like a module.

From one class, we make as many objects as we want. Each one has its own namespace
(“attributes”, including functions and variables)

Rules of thumb:

Use exceptions only for bad things, never for normal control flow

Use exceptions to separate standard processing from “oops”
processing

Their purpose is to make code cleaner,

easier to read and to get right.

Never handle exceptions silently (terrible for debugging)