```
################################################################################
#                                                                              #
#   LX.py     Image Manipulation II: Gradients, Conditionals, XY Location      #
#                                                                              #
#   Course:   CS1021C Computer Science I (Dr. Talaga)                          #
#   Author:   Kevin Ernst                                                      #
#   Date:     27 March 2013                                                    #
#                                                                              #
################################################################################

################################################################################
#                           R E F E R E N C E S                                #
#                                                                              #
#  1. "Taipan!" - http://en.m.wikipedia.org/wiki/Taipan!                       #
#                                                                              #
#  2. http://www.taipangame.com/                                               #
#                                                                              #
#  3. 'commafy' function from Peterbe.com:                                     #
#     http://www.peterbe.com/plog/thousands-commafy-large-numbers              #
#                                                                              #
################################################################################
import sys #import (executable, version_info)
import os #import (system, name)
from random import randrange
from string import ljust, join
from time import sleep
DEBUGGING = True
if DEBUGGING:
  RUTHLESS_ADVERSARIES, DEFLATION, EARLY_RETIREMENT = 1,1,1
  SQUID_SEASON, IMPATIENT = 1,1
else:
  RUTHLESS_ADVERSARIES, DEFLATION, EARLY_RETIREMENT = 0,0,0
  SQUID_SEASON, IMPATIENT = 0,0

################################################################################
#                       G L O B A L   C O N S T A N T S                        #
################################################################################
# how many times you can sail around the "world" before it's time to retire
RETIREMENT_AGE          = 10 - 9*EARLY_RETIREMENT
# you're a millionaire. retire to a secluded tropical isle
RICH_ENOUGH_TO_RETIRE   = 1e6 - int(DEFLATION*0.99*1e6)
# how much cash you get for defeating a pirate fleet
PIRATE_BOOTY            = 500
# Starting repair cost for 100% damage
REPAIR_COST             = 1000
SHIP_DAMAGE_SEA_VOYAGE  = 3  + 5*RUTHLESS_ADVERSARIES
SHIP_DAMAGE_PIRATES     = 10 + 20*RUTHLESS_ADVERSARIES
SHIP_DAMAGE_SQUID       = 25 + 25*RUTHLESS_ADVERSARIES
PAUSE_MSG               = 2 - IMPATIENT
PAUSE_MSG_SHORT         = 1 - 0.5*IMPATIENT
PAUSE_EVENT             = 5 - 3*IMPATIENT
DEFAULT_FIRM_NAME       = "Permanent Assurance"
DEFAULT_SHIP_NAME       = "The Crimson Permanent Assurance"
DEFAULT_SHANTY          = """
It's fun to charter an accountant,
And sail the wide accountan-cy.
To find, explore the funds offshore,
And skirt the shoals of bankruptcy.
It can be manly in insurance.
We'll up your premium semi-annually.
It's all tax-deductible,
We're fairly incorruptible.
We're sailing on the wide accountan-cy."""

# I decided on a 1-based "array" because the original Tapian! game assigned
# these numbers to the ports, so I wouldn't get confused.
port_names = { 1:'Hong Kong', 2:'Shanghai',  3:'Nagasaki', 4:'Saigon',
               5:'Manila',    6:'Singapore', 7:'Batavia',
               0:'the open ocean' }
port_descriptions = {
    1:"Hong Kong: All the familiar sights, sounds, and smells of home!",
    2:"Shanghai bustles with traders from east and west.\n" +
      "Be on the lookout for pickpockets!",
```

```
    3:"Taipan, I've heard one can obtain Portuguese tobacco here.\n" +
      "And sponge cakes!",
    4:"We've been at sea for many months, Tapian. Let's treat the men\n" +
      "to some nice hot noodles in Saigon.",
    5:"Tapain, the Peruvian silver in Manila will be quite valuable\n" +
      "for trade on the mainland.",
    6:"If we'll be needing any supplies for repair of the ship,\n" +
      "Singapore will be the place to aquire them, sir.",
    7:"You'll be wanting to stay on board in Batavia, sir. Much\n" +
      "bad blood against Westerners here.",
    0:"'I'm on the sea! I'm on the sea!\n" +
      " I am where I would ever be,\n" +
      " With the blue above and the blue below,\n" +
      " And silence wheresoe'er I go.'\n\n" +
      "Nothing quite like it, eh, sir?" }
# Where can you sail from each port (in order of North, South, East, West):
port_routes = { 1:{ 'n':2, 's':5 },
                2:{ 's':1, 'e':3 },
                3:{ 'e':2 },
                4:{ 'n':1, 's':6, 'e':5 },
                5:{ 'n':1, 'w':4 },
                6:{ 'n':4, 's':7 },
                7:{ 'n':4, 'e':6 } }

characters = ['Mc Henry', 'Li Yuen', 'Elder Brother Wu']
directions = { 'n':'north', 's':'south', 'e':'east', 'w':'west' }
conditions = [ 'Poor', 'Fair', 'Good', 'Very Good' ]
help_msg_1 = """
TAIPAN!
=======
Insipred by the Mega Micro Computers game for the TRS-80 and Apple ][.

You are the head of an up-and-coming trading company based in Hong Kong
in the 19th century and the captain of your own ship.

You begin the game in your home port, and can sail to various ports in
and around the South China Sea. Other ports you may visit are Shanghai,
Nagasaki, Saigon (present day Ho Chi Min City in Vietnam), Manila,
Singapore, and Batavia (the old Dutch name for Jakarta, the capital
city of Indonesia).




NOTE: When gameplay begins, be sure to make the command area in JES at least
large enough to see this text and the TAIPAN! title at the top.
"""
# FIXME: Can't travel by cardinal directions.
# you can reference by cardinal direction (North, South, East, or West).
help_msg_2 = """
Traveling close to the mainland or to nearby ports incurs less risk
(of pirate or giant squid attacks) than traveling across open ocean.
Upon arriving at a port, you are presented with a list of neighboring
ports, which you can safely travel to under the protection of
maritime law enforcement.

You can travel to any other port (potentially crossing open ocean) by
pressing one of the number keys assigned to the desired port.

If your ship incurs damage during your travels on the high seas, you
may request repairs in your home port of Hong Kong.

The game ends when your ship is sunk by marauders, or you make
enough money to retire a millionaire.
"""
# FIXME: ---or--- you sail to each port ten times

# Fix up a few things if we're not running in JES/Jython:
# -------------------------------------------------------
if sys.executable == None: # JES does this
```

```
  def cls():
    printNow("\n"*25)
else:
  # We're running in IPython or 'python' in the console
  def cls():
    os.system(['clear','cls'][os.name == 'nt'])

  def printNow(s):
    """Replacement for JES's 'printNow' function when running in CPython"""
    print(s)

  def requestString(message="Enter a value -->"):
    """Replacement for JES's 'requestString' function when running in
    CPython"""
    #no_input = True
    #while no_input:
    print(message), # no newline
    response = str(raw_input())
    #  if len(response) == 0:
    #    print("That is not an acceptable response. Try again.")
    #  else:
    #    no_input = False
    return response

  def requestInteger(message="Enter a number -->"):
    """Replacement for JES's 'requestInteger' function when running in
    CPython"""
    bad_input = True
    while bad_input:
      print(message + " "), # no newline
      response = raw_input()
      if len(response) == 0 or not response.isdigit():
        print("That is not an acceptable response. Try again.")
      else:
        bad_input = False
    return int(response)

##############################################################################
#                    U T I L I T Y   F U N C T I O N S                       #
##############################################################################
def commafy(n):
  rev_n = enumerate(str(n)[::-1])  # a reversed string, enumerated
  r = []
  for i, c in rev_n:
      if i and (not (i % 3)):
          r.insert(0, ',')
      r.insert(0, c)
  return ''.join(r)

##############################################################################
#                          E X C E P T I O N S                               #
##############################################################################
class CantSailThere(Exception):
  def __init__(self):
    self.message = "You can't sail in that direction."
  def __str__(self):
    return self.message

class ShipSunk(Exception):
  def __init__(self, ship_name="The ship"):
    self.message = "Taipan, we've sustained too much damage. " + \
        "%s is going down!" % ship_name
  def __str__(self):
    return self.message

class BattleDefeat(Exception):
  def __init__(self):
    self.message = "We were bested by the pirate fleet, Taipan."
  def __str__(self):
    return self.message

class BattleVictory(Exception):
  def __init__(self):
```

```
    self.message = "We were victorious against the pirate fleet. Huzzah!"
  def __str__(self):
    return self.message


##############################################################################
#                          C L A S S   O B J E C T S                         #
##############################################################################
class Ship:
  def __init__(self, name=DEFAULT_SHIP_NAME, starting_choice=2,
               shanty=DEFAULT_SHANTY):
    """Initialization for the 'Ship' class"""
    if name: # Not None or empty string
      self.name     = name    # give this ship a name
    else:
      self.name     = DEFAULT_SHIP_NAME
    self.condition  = 100     # ship's condition (0-100%)
    self.sea_shanty = shanty  # the ship's official sea shanty

    # initialCash=0, initialDebt=0, initialGuns=5
    # These are the choices from the original Taipan! game:
    if starting_choice == 1: # cash and debt
      self.guns = 0
      self.cash = 400
      self.debt = -5000
    elif starting_choice == 2: # guns and no cash (but no debt)
      self.guns = 5
      self.cash = 0
      self.debt = 0
    # The ship starts with 60 holds. A gun takes 10 holds.
    self.holds = 60 - 10*self.guns

  def printStatus(self):
    """Print a summary of the ship's status: guns, cash, debt, and the
    condition of the ship."""
    pad = len(str(RICH_ENOUGH_TO_RETIRE))
    s = "The status of " + self.name + " is as follows:"
    printNow(s + '\n' + '-'*len(s))
    printNow("CASH:   " + ljust(commafy(str(self.cash)), pad) +
             "   DEBT:   -" + ljust(commafy(str(self.debt)), pad))
    c = self.condition
    if c == 100:
      s = "   REPAIR:   Perfect! (%i%%)" % c
    elif self.condition > 100:
      s = "   REPAIR:   Magically protected! (%i%%)" % c
    else:
      t = conditions[(c/25)%4]
      s = "   REPAIR:   %s (%i%%)" %(t, c)
    printNow("GUNS:   " + ljust(str(self.guns), pad) + s)
    printNow("")

  def causeDamage(self, damage):
    """Cause the ship to sustain battle damage, subtracting 'damage' from
    self.condition. Raise 'ShipSunk' if this causes condition to go less than
    zero."""
    self.condition -= damage
    if self.condition <=0:
      raise ShipSunk

  def doShipRepairs(self):
    """This method gets invoked when the ship's condition is <90%, and
    allows you to pay McHenry from the Hong Kong shipyard"""
    # Should test for location here, but it's no longer local to this class.
    # :/
    assert(self.condition < 100)  # it's a programming error if we get here
                                  # otherwise
    damage = 100 - self.condition
    printNow("Taipan, Mc Henry from the Hong Kong Shipyards has arrived.")
    sleep(PAUSE_MSG_SHORT)
    printNow('')
    printNow("He says, 'I see ye've a wee bit of damage to yer ship.'")
    resp = requestString("Will ye be wanting repairs? [Y/n]")
    if resp == '' or resp[0].lower() == 'y':  # yes or ENTER
```

```
      printNow("\nOch, 'tis a pity to be %i%% damaged." % damage)
      # Compute cost for repairs based on how much cash you've got. If you're
      # richer, McHenry's going to charge you more. Like any good capitalist,
      # he's capturing consumer surplus.
      # ref: http://www.joelonsoftware.com/articles/CamelsandRubberDuckies.html
      price = int((REPAIR_COST + 0.15*self.cash) * damage/100.0)
      printNow("We can fix yer whole ship for " + str(price) +
               ", or make partial repairs if you wish.")
      printNow('')

      good_response = False
      while not good_response:
        resp = requestInteger("How much will ye spend (0-%i)" % price)
        resp = int(resp) # always ignore fractional currency
        if not resp:  # None or 0
          return
        elif resp < 0:
          printNow("That isn't a figure I understand, sir.\n")
          continue
        elif resp > self.cash:
          printNow("That'll bankrupt us, Taipan!\n")
          continue
        else:
          good_response = True
          self.cash -= resp
          # McHenry will gladly let you pay more than the agreed-upon price,
          # but won't fix your ship any more than all the way fixed.
          if resp > price:
            self.condition = 100
          else:
            self.condition += int((resp / float(price)) * damage)
    else:
      # Ye won't be wanting repairs, then.
      return


class Port:
  """A class that contains all the ports of call for Taipan! along with
  methods to determine whether one can sail between two ports directly, and to
  print the "arriving at..." message for the current port."""
  # Don't put this here. It becomes a class variable which is shared (and
  # modified) by all instances:
  #port_to_the = {}  # keep track of which port lies to n,s,e,w

  def __init__(self, portnum):
    self.port_to_the = {}  # keep track of which port lies to n,s,e,w
    self.name = port_names[portnum]
    self.description = port_descriptions[portnum]
    self.port_number = portnum
    # Populate the "port_to_the[direction]" list:
    #for direction in ['n', 's', 'e', 'w']:
    #  try:
    #    self.port_to_the[direction] = port_routes[portnum][direction]
    #  except KeyError:
    #    self.port_to_the[direction] = 0 # open ocean

  #def neighbors(self):
  #  """Return a list of the neighboring ports in NSEW order"""
  #  #                    actually, no it doesn't  ^^^^
  #  # FIXME: make this a generator, so it actually returns in the proper
  #  # order.
  #  t = []
  #  for d in directions.keys():
  #    t.append(self.port_to_the[d].name)
  #  return t

  # def neighboringPortNumbers(self):
  #   """Prints a list of neighboring port numbers (for use in the 'canSailTo'
  #   method"""

  def setPortToThe(self, direction, portref):
    """Set the value at 'direction' in the port_to_the dictionary to the Port
    reference given as the second argument"""
```

```
    # This function allows the Game class to set the values in the
    # 'port_to_the' dict to references to the other Port class instances.
    # Previously, 'port_to_the' just held integers representing the key in the
    # 'ports' dictionary of the Game class.
    assert(direction in directions.keys())
    self.port_to_the[direction] = portref

  def arrivalMessage(self):
    """Prints the arrival message (and a brief ship's status report) for this
    port"""
    #self.__printDescription()
    s = "Arriving in the port of " + self.name + "...."
    printNow('='*len(s) + '\n' + s + '\n' + '='*len(s))
    printNow(self.description + '\n')
    self.printNeighboringPorts()
    printNow("")

  def canSailTo(self, portnum):
    """Returns False if you can't sail *directly* to that port without putting
    out to sea"""
    #if portnum in self.port_to_the.values(): # are Port objs now, won't work
    plist = []
    for p in self.port_to_the.values():
      plist.append(p.port_number)
    if portnum in plist:
      return True
    else:
      #raise CantSailThere
      return False

  def printDescription(self):
    """Prints a description of this port."""
    printNow("You are in the port of " + self.name)

  def printNeighboringPorts(self):
    """Prints a list of all neighboring ports (to which you can sail directly
    without having to put to sea."""
    printNow("To the North lies " + self.port_to_the['n'].name + ".")
    printNow("To the South lies " + self.port_to_the['s'].name + ".")
    printNow("To the East  lies " + self.port_to_the['e'].name + ".")
    printNow("To the West  lies " + self.port_to_the['w'].name + ".")


class HomePort(Port):
  """Derived Port class that only applies to Hong Kong, where you can get your
  ship repaired, visit the warehouse, and borrow money from Elder Brother
  Wu."""
  # I think you need to call the base class' __init__ function here...
  #def __init__:


class Game:
  """The map structure of the game. Basically a list of Ports, and the actions
  to act upon them. Always passed in a reference to 'ship' so it can
  manipulate data structures inside it as the ship sails from port to port."""
  def __init__(self, firm_name=DEFAULT_FIRM_NAME):
    self.ship = Ship()
    if firm_name:  # not None or empty string
      self.firm_name = firm_name
    else:
      self.firm_name = DEFAULT_FIRM_NAME
    self.current_port = None
    self.en_route_to = None
    self.ports = {}
    self.visit_count = {}
    # Initialize the list of ports as a member. The ports already understand
    # their relative position to each other, and set up this mapping in their
    # own __init__ methods.
    for p in port_names.keys():
      self.ports[p] = Port(p)
      self.visit_count[p] = 0  # initialize all visit counts to zero.
      # TODO: assign shorcuts (aliases) to the ports such as g.HongKong.
      #pname = self.ports[p].name.translate(None, "<> ")
```

```
      #eval("self.%s = self.ports[%s]" % (pname,p))

    # Now initialize each ports 'port_to_the' dictionary with references to
    # the newly created ports:
    for i in self.ports: #.keys() is apparently implicit
      for d in ['n', 's', 'e', 'w']:
        try:
          # port_routes[i][d] is the port in the direction ('d') of the
          # current port (key 'i' in self.ports):
          self.ports[i].setPortToThe(d, self.ports[port_routes[i][d]])
        except KeyError:
          # if the key 'd' doesn't exist in the port_routes dictionary, then
          # set this direction's destination to 0 = open ocean
          self.ports[i].setPortToThe(d, self.ports[0])

    # start in Hong Kong
    self.current_port = self.ports[1]

  def printPortMenu(self):
    """Print a menu of ports to which you can sail"""
    s = ''
    for i in range(1, len(self.ports)):  # skips '0'!
      s = s + "[%i] %s    " %(i, self.ports[i].name)
      if i%4 == 0: s = s + '\n'
    printNow(s)

  def sailTo(self, to_port):
    """Set sail from the current port to the given destination port 'to_port'.
    If the destination port is a neighbor, updates self.current_port and
    returns. Otherwise, calls 'putToSea'."""
    to_port = int(to_port) # for my sanity

    # Validate the input first:
    portname = self.ports[to_port].name
    if self.current_port.canSailTo(to_port):
      # We can sail to the destination port along the coast without crossing
      # the open ocean.
      printNow("Aye, sir. We'll set out for " + portname + " straight away!")
      sleep(PAUSE_MSG)
      cls()
      #return self.ports[to_port]
      self.current_port = self.ports[to_port]
    else:
      # Not a neighboring port. Must deploy to open ocean.
      # TODO: Require purchasing supplies first.
      printNow("The journey to " + portname +
               " will require crossing the open ocean, Taipan.")
      printNow("May the sea goddess look favorably on our journey!")
      sleep(PAUSE_EVENT)
      cls()
      #return self.putToSea(to_port)
      self.putToSea(to_port)

  def seaBattle(self, strength=5):
    printNow("Taipan, there are %i pirate ships on the horizon!\n" % strength)
    sleep(PAUSE_MSG)
    # Run or fight?
    if self.ship.condition > 50:
      printNow("We're seaworthy, there's a good chance we can make it " +
               "if we flee now!")
      resp = requestString("Taipan, should we run? [Y/n]")
      if resp == '' or resp[0].lower() == 'y':  # yes or ENTER
        printNow("\nAye, sir. We'll try to get away...")
        sleep(PAUSE_EVENT)
        printNow("\nWe made it!")
        raise BattleVictory
      else:
        printNow("\nAye, sir. We'll fight.")
    else:
      # If the ship's not seaworthy enough to escape the battle...
      printNow("Our ship's in poor condition, sir, she might not make it!")

    # At the moment, every encounter with pirates yields 10% battle damage,
```

```
    # the defeat of the pirate fleet, and $500.
    sleep(PAUSE_EVENT)
    try:
      self.ship.causeDamage(SHIP_DAMAGE_PIRATES)
    except ShipSunk:
      printNow("\nWe can't hold 'em off, Tapian! We're being boarded!")
      raise
    else:
      printNow("\nWe've taken heavy damage, but the pirates are retreating!")
      printNow("Look at the buggers run, Taipan! Huzzah!\n")
      sleep(PAUSE_MSG)
      printNow("Taipan, we've recovered %i in booty from a captured pirate ship!"
               % PIRATE_BOOTY)
      self.ship.cash += PIRATE_BOOTY
      sleep(PAUSE_EVENT)
      cls()
      raise BattleVictory

  def putToSea(self, destination):
    """Puts the ship to sea, with the final destionation of 'destination'.
    As opposed to 'sailTo', which is a short sail to a neighboring port close
    to the coast, 'putToSea' involves preparation for a significant sea
    voyage, buying supplies, and carries with it the inherent risk of pirate
    attacks, storms, and sea monsters."""
    destination = int(destination) # for my sanity
    self.en_route_to = destination
    # In the current implementation, there is a 50% chance that you'll be
    # attacked by pirates (and sunk) and a 10% chance that you'll be eaten by
    # a giant squid.
    chance = randrange(1, 101)
    if chance < 50:          # attacked by pirates
      try:
        self.seaBattle()
      except ShipSunk:
        self.endGame()
      except BattleVictory:
        pass
    # attacked by a giant squid (between 10 and 30%, depending on whether it's
    # SQUID_SEASON):
    elif 50 <= chance < 60 + SQUID_SEASON*30:
      printNow("Taipan, the seas are rough. We're likely to be attacked " +
               "by squid.")
      sleep(PAUSE_EVENT)
      printNow("\nGIANT SQUID OFF THE PORT BOW!!!")
      sleep(PAUSE_MSG)
      try:
        self.ship.causeDamage(SHIP_DAMAGE_SQUID)
      except ShipSunk:
        printNow("\nTaipan, the squid is overtaking the ship!")
        self.endGame()
      else:
        printNow("\nTaipan, we've managed to stave off the squid...")
        printNow("We've taken a lot of damage, but we'll " +
                 "make it back to port this time.\n")
        sleep(PAUSE_EVENT)

    # Travel across open sea incurs a 5% penalty on ship's status:
    # FIXME: Your crew won't even try to warn you if this last 5% will sink
    # your ship!
    try:
      self.ship.causeDamage(SHIP_DAMAGE_SEA_VOYAGE)
    except ShipSunk:
      printNow("\nTaipan, bad storm brewing ahead!")
      sleep(PAUSE_EVENT)
      printNow("\nI don't think she'll hold together in this weather...")
      sleep(PAUSE_MSG_SHORT)
      cls()
      printNow("\nWe're losing her, Taipan. " + self.ship.name +
               " is sinking!")
      self.endGame()

    # Made it this far...
    self.en_route_to=None
```

```
    #return self.ports[destination]
    self.current_port=self.ports[destination]

  def endGame(self, ):
    printNow("\n\nIt's been a pleasure serving with you, sir.\n")
    #printNow("ALL HANDS ABANDON SHIP!\n\n")
    printNow("~~~~~~~~~~~~~~~~~~~\n" +
             " G A M E    O V E R\n" +
             "~~~~~~~~~~~~~~~~~~~\n")
    exit()

  # def goHome(self, ship):
  #   """Immediately sail home (e.g., in case of severe battle damage)"""


################################################################################
#                        I N I T I A L I Z A T I O N                           #
################################################################################
def printHelp():
  cls()  # clear the screen
  printNow(help_msg_1)
  requestString("Press ENTER for the next page of help.")
  cls()
  printNow(help_msg_2)

def runGame():
  """Run Taipan!"""
  printHelp()
  printNow('~'*70)
  resp = requestString("Let's begin, Taipan. What will you name your firm?")
  g = Game(firm_name=resp)
  printNow("Very well, sir. " + g.firm_name + " will need a ship.\n")
  resp = requestString("What will you name your ship?")
  if resp: # not None or empty string
    g.ship.name = resp
  else:
    g.ship.name = DEFAULT_SHIP_NAME
  cls()

  quit_game = False
  while not quit_game:
    g.current_port.arrivalMessage()
    g.ship.printStatus()
    g.printPortMenu()
    printNow('')

    # Allow ship repairs if ship status is <90% and you're in Hong Kong
    # TODO: fix this to be instanceof(HomePort) once the HomePort class is
    # fleshed out.
    if g.current_port.name == "Hong Kong":
      if g.ship.cash > RICH_ENOUGH_TO_RETIRE:
        printNow("Taipan, you have had a successful career and amassed " +
                 "great wealth.\nI think it's high time you retired to a " +
                 "quiet home in the country!")
        self.endGame()
      if g.ship.condition < 90:
        g.ship.doShipRepairs()
    printNow('')

    resp = None
    # Process input for the 'Where shall we sail to?' prompt, allowing N,S,E,W
    # as well as ports 1-7, [H]elp and [Q]uit.
    while not resp:  # loop while the user keeps giving bad input
      resp = requestString("Where shall we sail to, Taipan?\n" +
                           "(or [H]elp or [Q]uit)")
      if not resp: # empty or None
        continue
      if resp[0].lower() == 'q':
        quit_game = True
        continue
      elif resp[0].lower() == 'h':
        printHelp()
        requestString("Press ENTER to continue.")
```

```
        resp = True  # just to make sure we break out of this while loop in
                     # case the user pressed "Cancel"
      else:
        #g.current_port = g.sailTo(resp)
        g.sailTo(resp)

if __name__ == "__main__":
  # If we're running interactively:
  runGame()


# vim:tw=78 sw=2 ts=2 expandtab
```