# Top RGB

def top\_rgb(filename):

bit = Bit(filename)

bit.paint('red')

while bit.front\_clear():

bit.move()

bit.paint('green')

bit.paint('blue')

# Top Bot

def top\_bot(filename):

bit = Bit(filename)

bit.paint('blue')

while bit.front\_clear():

bit.move()

bit.paint('blue')

bit.right()

while bit.front\_clear():

bit.move()

bit.right()

bit.paint('green')

while bit.front\_clear():

bit.move()

bit.paint('green')

# Top Logic 1

def top\_logic1(filename):

bit = Bit(filename)

while bit.front\_clear():

if bit.get\_color() != 'red':

bit.move()

if bit.get\_color() == 'red':

bit.paint('green')

# Top Logic 2

def top\_logic2(filename):

bit = Bit(filename)

bit.move()

while bit.front\_clear():

if bit.get\_color() != 'blue':

bit.paint('green')

bit.move()

if bit.get\_color() == 'blue':

bit.move()

if bit.get\_color() != 'blue':

bit.paint('green')

# Top Blue Green

def top\_bluegreen(filename):

bit = Bit(filename)

bit.paint('blue')

while bit.front\_clear():

bit.move()

bit.paint('green')

if bit.front\_clear():

bit.move()

bit.paint('blue')

# Hidey Hole

def go\_hide(filename):

bit = Bit(filename)

while bit.front\_clear():

if not bit.right\_clear():

bit.move()

if bit.right\_clear():

bit.right()

bit.move()

# Big Red S

def move\_until\_blocked(bit):

"""Move bit forward until front is blocked."""

while bit.front\_clear():

bit.move()

pass

def red\_until\_blocked(bit):

"""

Move bit forward until front is blocked,

painting every square red.

"""

bit.paint('red')

while bit.front\_clear():

bit.move()

bit.paint('red')

pass

def half\_s(bit):

"""Implement half-s as described above."""

while bit.front\_clear():

red\_until\_blocked(bit)

bit.right()

while bit.front\_clear():

red\_until\_blocked(bit)

bit.left()

bit.left()

move\_until\_blocked(bit)

bit.left()

move\_until\_blocked(bit)

bit.left()

bit.left()

pass

def big\_red(filename):

"""Implemement big-red, as described above."""

bit = Bit(filename)

while bit.front\_clear():

bit.move()

if bit.get\_color() == 'green':

bit.left()

half\_s(bit)

bit.right()

bit.move()

bit.right()

half\_s(bit)

bit.left()

pass

# Triples

def do\_triple(filename):

"""

Bit starts facing along the first side.

Solves the triple problem, moving through

3 rectangles.

"""

bit = Bit(filename)

one\_side(bit)

one\_side(bit)

one\_side(bit)

reposition(bit)

one\_side(bit)

one\_side(bit)

one\_side(bit)

reposition(bit)

one\_side(bit)

one\_side(bit)

one\_side(bit)

reposition(bit)

bit.left()

def one\_side(bit):

bit.paint('green')

while not bit.left\_clear():

bit.paint('green')

bit.move()

bit.left()

bit.move()

pass

def reposition(bit):

bit.right()

bit.right()

bit.move()

pass

# Checkerboard

def do\_checkerboard(filename):

"""

Starts at the north west corner facing south. Moves south

completeing the checkerboard rows. Ends in the bottom

row facing south.

"""

bit = Bit(filename)

blue\_white\_row(bit)

while bit.front\_clear():

if bit.get\_color() != 'blue':

white\_blue\_row(bit)

blue\_white\_row(bit)

if bit.get\_color() == 'blue':

blue\_white\_row(bit)

if bit.get\_color() != 'blue':

white\_blue\_row(bit)

def blue\_white\_row(bit):

bit.paint('blue')

bit.left()

while bit.front\_clear():

bit.move()

if bit.front\_clear():

bit.move()

bit.paint('blue')

bit.right()

bit.right()

while bit.front\_clear():

bit.move()

bit.left()

if bit.front\_clear():

bit.move()

def white\_blue\_row(bit):

bit.left()

bit.move()

bit.paint('blue')

while bit.front\_clear():

bit.move()

if bit.front\_clear():

bit.move()

bit.paint('blue')

bit.right()

bit.right()

while bit.front\_clear():

bit.move()

bit.left()

if bit.front\_clear():

bit.move()