*# Testing purposes - please ignore*

def almostEqual(d1, d2, epsilon=10\*\*-7):

  return (abs(d2 - d1) < epsilon)

"""[Spicy] triangle\_area"""

def triangle\_area(x1, y1, x2, y2, x3, y3):

  return

"""Testing triange\_area implementation"""

def test\_triangle\_area():

  print("Testing triangle\_area()...", end="")

  almostEqual(triangle\_area(1, 1, 3, 4, 5, 3), 4.0)

  almostEqual(triangle\_area(0, 0, 4, 0, 3, 3), 6.0)

  assert(triangle\_area(0, 1, 3, 4, 2, 6) == 4.5)

  almostEqual(triangle\_area(-1, 2, 5, 3, 4, -2), 14.5)

  almostEqual(triangle\_area(-3, -3, -2, -1, -6, -8), 0.5)

  print("... done!")

"""[Spicy] is\_legal\_triangle"""

def is\_legal\_triangle(s1, s2, s3):

  return

"""Testing is\_legal\_triangle implementation"""

def test\_is\_legal\_triangle():

  print("Testing is\_legal\_triangle()...", end="")

  assert(is\_legal\_triangle(3, 4, 5) == True)

  assert(is\_legal\_triangle(3, 3, 5) == True)

  assert(is\_legal\_triangle(2, 3, -1) == False)

  assert(is\_legal\_triangle(1, 1, 2) == False)

  assert(is\_legal\_triangle(0, 3, 4) == False)

  assert(is\_legal\_triangle(4, 5, 7) == True)

  print("... done!")

if \_\_name\_\_ == '\_\_main\_\_':

  test\_triangle\_area()

  test\_is\_legal\_triangle()