"""Question 1: get\_grade"""

def get\_grade(grade):

  return

"""Testing Question 1 implementation"""

def test\_get\_grade():

  print("Testing get\_grade()...", end="")

  assert(get\_grade(0) == 'F')

  assert(get\_grade(59) == 'F')

  assert(get\_grade(63) == 'D')

  assert(get\_grade(71) == 'C')

  assert(get\_grade(80) == 'B')

  assert(get\_grade(100) == 'A')

  print("...done!")

"""Question 2: pay\_bill"""

def pay\_bill(total, method):

    return

"""Testing Question 2 implementation"""

def test\_pay\_bill():

  print("Testing pay\_bill()...", end="")

  assert(pay\_bill(50, "cash") == 60.0)

  assert(pay\_bill(15, "cash") == 18.0)

  assert(pay\_bill(15, "card") == 18.3)

  assert(pay\_bill(33, "card") == 39.6)

  assert(pay\_bill(50, "card") == 60.0)

  print("... done!")

"""Question 3: split\_bill"""

def split\_bill(total, num\_people):

  return

"""Testing Question 3 implementation"""

def test\_split\_bill():

  print("Testing split\_bill()...", end="")

  assert(split\_bill(10, 3) == None)  *# prints "Each person pays 4.0 dollars."*

  assert(split\_bill(55, 4) == None)  *# prints "Each person pays 16.5 dollars."*

  assert(split\_bill(0, 9) == None)   *# prints "Each person pays 0.0 dollars."*

  assert(split\_bill(73, 5) == None)  *# prints "Each person pays 17.52 dollars."*

  assert(split\_bill(109, 8) == None) *# prints "Each person pays 16.35 dollars."*

  print("... done!")

"""Question 4: circles\_intersect"""

def circles\_intersect(x1, y1, r1, x2, y2, r2):

  return

"""Testing Question 4 implementation"""

def test\_circles\_intersect():

  print("Testing circles\_intersect()...", end="")

  assert(circles\_intersect(0, 0, 2, 3, 0, 2) == True)

  assert(circles\_intersect(0, 0, 2, 4, 0, 2) == True)

  assert(circles\_intersect(0, 0, 2, 5, 0, 2) == False)

  assert(circles\_intersect(3, 3, 3, 3, -3, 3) == True)

  assert(circles\_intersect(3, 3, 3, 3,- 3, 2.99) == False)

  print("... done!")

"""Question 5: valid\_RGB"""

def valid\_RGB(): *# fill in the parameters!*

  return

"""Testing Question 5 implementation"""

def test\_valid\_RGB():

  print("Testing valid\_RGB()...", end="")

  assert(valid\_RGB() == True)

  assert(valid\_RGB(150, 200, 15) == True)

  assert(valid\_RGB(100, 300, 100) == False)

  assert(valid\_RGB(green=255) == True)

  assert(valid\_RGB(red=0, green=45) == True)

  assert(valid\_RGB(red=256) == False)

  print("... done!")

"""Question 6: Fix the Code"""

*# Remove the triple quotes from the following block of code to run this!*

'''

def buggy\_function(x)

  is\_even = (x % 2 == 0)

  if type(x) != int:

    return False

  if x <= 10:

    if is\_even:

      return True

  if x > 10:

    if not is\_even:

      return True

  else:

    return False

def test\_buggy\_function():

  print("Testing buggy\_function()...", end="")

  assert(buggy\_function(4) == True)

  assert(buggy\_function(10) == True)

  assert(buggy\_function(11) == True)

  assert(buggy\_function(7) == False)

  assert(buggy\_function(0) == True)

  assert(buggy\_function(-5) == False)

  assert(buggy\_function(-8) == True)

  assert(buggy\_function(14) == False)

  assert(buggy\_function(13.0) == False)

  assert(buggy\_function("uh oh") == False)

  print("...done!")

'''

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

  test\_get\_grade()

  test\_pay\_bill()

  test\_split\_bill()

  test\_circles\_intersect()

  test\_valid\_RGB()

*# test\_buggy\_function()    # uncomment for Q6*