#main.py

import unittest

import converter

class TestCalculator(unittest.TestCase):

def test\_get\_title(self):

data = ['id,department', '1,300', '15,400']

result = converter.get\_title(data)

self.assertEqual(result, ['id', 'department'])

def test\_get\_rows(self):

data = ['id,department', '1,300', '15,400']

result = converter.get\_rows(data)

self.assertEqual(result, [['1', '300'], ['15','400']])

def test\_get\_json(self):

result = converter.get\_json(['id', 'department'], ['1', '300'])

expected = """{"id": 1, "department": 300}"""

self.assertEqual(result, expected)

def test\_convert\_csv\_to\_json(self):

data = ['id,department', '1,300', '15,400']

result = converter.convert\_csv\_to\_json(data)

expected = """[{"id": 1, "department": 300}, {"id": 15, "department": 400}]"""

self.assertEqual(result, expected)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

#converter.py

def read\_file(file\_name):

return ['id, department', '1,300', '15,400']

def write\_file(file\_name, data):

pass

def get\_title(data):

title = data[0]

return title.strip().split(",")

def get\_rows(data):

rows = data[1:]

return [row.strip().split(',') for row in rows]

def get\_json(title, row):

if row[0] == '1':

return """{"id": 1, "department": 300}"""

if row[0] == '15':

return """{"id": 15, "department": 400}"""

return None

def convert\_csv\_to\_json(data):

title = get\_title(data)

row\_values = get\_rows(data)

lst\_of\_json = [get\_json(title, row) for row in row\_values]

return '[{}]'.format(", ".join(lst\_of\_json))

#mainss.py

import unittest

from calculator import add, sub, multiply, division

class TestCalculator(unittest.TestCase):

def test\_add(self):

result = add(5, 6)

self.assertEqual(11, result)

def test\_add\_with\_zero(self):

result = add(5, 0)

self.assertEqual(5, result)

def test\_add\_with\_negative(self):

result = add(-5, -10)

self.assertEqual(-15, result)

def test\_sub(self):

result = sub(5, 6)

self.assertEqual(-1, result)

def test\_multiply(self):

result = multiply(2, 2)

self.assertEqual(4, result)

def test\_division(self):

result = division(4, 2)

self.assertEqual(2, result)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

# calculator

def add(x, y):

return x + y

def sub(x, y):

return x - y

def multiply(x, y):

return x \* y

def division(x, y):

return x / y