New Code

import json, unittest, datetime

with open("./data-1.json","r") as f:

jsonData1 = json.load(f)

with open("./data-2.json","r") as f:

jsonData2 = json.load(f)

with open("./data-result.json","r") as f:

jsonExpectedResult = json.load(f)

def convertFromFormat1 (jsonObject):

locationParts = jsonObject['location'].split('/')

result = {

'deviceID': jsonObject['deviceID'],

'deviceType': jsonObject['deviceType'],

'timestamp': jsonObject['timestamp'],

'location': {

'country': locationParts[0],

'city': locationParts[1],

'area': locationParts[2],

'factory': locationParts[3],

'section': locationParts[4]

},

'data': {

'status': jsonObject['operationStatus'],

'temperature': jsonObject['temp']

}

}

return result

# IMPLEMENT: Conversion From Type 1

return NotImplemented

def convertFromFormat2 (jsonObject):

date = datetime.datetime.strptime(

jsonObject['timestamp'],

'%Y-%m-%dT%H:%M:%S.%fZ'

)

timestamp = round(

(date - datetime.datetime(1970, 1, 1)).total\_seconds() \* 1000

)

result = {

'deviceID': jsonObject['device']['id'],

'deviceType': jsonObject['device']['type'],

'timestamp': timestamp,

'location': {

'country': jsonObject['country'],

'city': jsonObject['city'],

'area': jsonObject['area'],

'factory': jsonObject['factory'],

'section': jsonObject['section']

},

'data': jsonObject['data']

}

return result

# IMPLEMENT: Conversion From Type 1

return NotImplemented

def main (jsonObject):

result = {}

if (jsonObject.get('device') == None):

result = convertFromFormat1(jsonObject)

else:

result = convertFromFormat2(jsonObject)

return result

class TestSolution(unittest.TestCase):

def test\_sanity(self):

result = json.loads(json.dumps(jsonExpectedResult))

self.assertEqual(

result,

jsonExpectedResult

)

def test\_dataType1(self):

result = main (jsonData1)

self.assertEqual(

result,

jsonExpectedResult,

'Converting from Type 1 failed'

)

def test\_dataType2(self):

result = main (jsonData2)

self.assertEqual(

result,

jsonExpectedResult,

'Converting from Type 2 failed'

)

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

unittest.main()

import json, unittest, datetime

with open("./data-1.json","r") as f:

jsonData1 = json.load(f)

with open("./data-2.json","r") as f:

jsonData2 = json.load(f)

with open("./data-result.json","r") as f:

jsonExpectedResult = json.load(f)

def convertFromFormat1 (jsonObject):

locationParts = jsonObject['location'].split('/')

result = {

'deviceID': jsonObject['deviceID'],

'deviceType': jsonObject['deviceType'],

'timestamp': jsonObject['timestamp'],

'location': {

'country': locationParts[0],

'city': locationParts[1],

'area': locationParts[2],

'factory': locationParts[3],

'section': locationParts[4]

},

'data': {

'status': jsonObject['operationStatus'],

'temperature': jsonObject['temp']

}

}

return result

def convertFromFormat2 (jsonObject):

date = datetime.datetime.strptime(

jsonObject['timestamp'],

'%Y-%m-%dT%H:%M:%S.%fZ'

)

timestamp = round(

(date - datetime.datetime(1970, 1, 1)).total\_seconds() \* 1000

)

result = {

'deviceID': jsonObject['device']['id'],

'deviceType': jsonObject['device']['type'],

'timestamp': timestamp,

'location': {

'country': jsonObject['country'],

'city': jsonObject['city'],

'area': jsonObject['area'],

'factory': jsonObject['factory'],

'section': jsonObject['section']

},

'data': jsonObject['data']

}

return result

def main (jsonObject):

result = {}

if (jsonObject.get('device') == None):

result = convertFromFormat1(jsonObject)

else:

result = convertFromFormat2(jsonObject)

return result

class TestSolution(unittest.TestCase):

def test\_sanity(self):

result = json.loads(json.dumps(jsonExpectedResult))

self.assertEqual(

result,

jsonExpectedResult

)

def test\_dataType1(self):

result = main (jsonData1)

self.assertEqual(

result,

jsonExpectedResult,

'Converting from Type 1 failed'

)

def test\_dataType2(self):

result = main (jsonData2)

self.assertEqual(

result,

jsonExpectedResult,

'Converting from Type 2 failed'

)

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

unittest.main()