**Deloitte Australia**

**Looking at task1 and task2 json format and having them retuned in the format present in the final doc.**

1. Here is the background information on the task

We have signed a new account: Daikibo Industrials, a global leader in the manufacturing of heavy machinery, founded and headquartered in Tokyo, Japan. They needed assistance with a variety of problems and were impressed to find out Deloitte could help in all verticals.

Daikibo is in the process of integrating IIoT (industrial internet of things) devices to monitor, measure and analyse their manufacturing processes. Half of their infrastructure uses devices streaming telemetry data in one format, and the other half in another. They need your help to combine the two.

1. Task

* Take the following steps to complete the task:
* Create an account at [**repl.it**](http://repl.it/).
* Fork [**this project**](https://replit.com/@NikolayTsenkov/Task1) into your workspace.
* After navigating to the project, find the Fork button on the page and press it.
* Get acquainted with the 2 data formats by exploring the files (**hint**: it's the same message, represented in 2 different formats):
* *data-1.json*
* *data-2.json*
* Explore the target unified format by looking at the file data-result.json. This is the format you should aim to output in your solution.
* Complete the solution located in the file [*main.py*](http://main.py/)*.*
* The project is set up in a way to test your solution automatically as you run it.
* Find the 2 functions in need of implementation and finish them (look for comment lines starting with "IMPLEMENT:").

1. 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

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()

**Presenting a Proposal to the Client**

Here is the background information on your task

Analysing offline data is great, but having a real-time overview of processes and smart alerts when things break is even better. Our client would like us to build a real-time manufacturing status dashboard. The first task of this process is drafting a development proposal.

Thus, we document how we would like Daikibo to use our services.