```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-


import sys
import json
import datetime
from web3 import Web3

DEBUG = False #True


# Ethereum にアクセスするライブラリ

class Ethereum:
    # コンストラクタ
    def __init__(self, url, account, password, private_key):
        print("Constructor of Ethereum")
        self.url = url
        self.account = account
        self.password = password
        self.private_key = private_key
        self.contracts = {}

        # Ethereum へ接続するクライアント作成
        self.w3 = Web3(Web3.HTTPProvider(url))



    # デストラクタ
    def __del__(self):
        print("Destructor of Ethereum")


    # 実行するコントラクトの情報
    def set_contract(self, addr, abi):
        print("set_contract")
        self.cnt_addr = addr
        self.cnt_abi = abi
        self.cnt = self.w3.eth.contract(address=self.w3.toChecksumAddress(addr), abi=abi)


    # データ数取り出し
    def get_data_number(self, channel_id,　measurement):
        print("get_data_number")

        res = self.cnt.functions.get_data_num(channel_id,　measurement).call()
        return res

    # データ取り出し
    def get_data(self, channel_id,　measurement, start=0, stop=10, source=None):
        print("get_data")

        res = self.cnt.functions.get_data(channel_id,　measurement, start, stop).call()
        if DEBUG : print("RES = ",res)
        return res

    # データ保存
    def put_data(self, channel_id, measurement, data):
        print("put_data")
        """
        data を timestamp とデータに分ける
        """
        d = json.loads(data)
        if DEBUG:
            print(d)

        timestamp = d['timestamp']
        if timestamp == None:
            timestamp = datetime.datetime.utcnow().strftime('%Y%m%d%H%M%S')

        ch_ac = self.w3.toChecksumAddress(self.account)
        tx = self.cnt.functions.put_data(channel_id, measurement, timestamp, data).buildTransaction({'nonce': self.w3.eth.getTransactionCount(ch_ac)})
        s_tx = self.w3.eth.account.signTransaction(tx, self.private_key)
        tx_hash = self.w3.eth.sendRawTransaction(s_tx.rawTransaction)
        tx_receipt = self.w3.eth.wait_for_transaction_receipt(tx_hash)
        if DEBUG:
            print(tx_hash)
            print(self.w3.eth.get_transaction(tx_hash))

        # トランザクション情報を残しておく
        self.tx = tx
        self.s_tx = s_tx
        self.tx_hash = tx_hash
        self.tx_receipt = tx_receipt


    #　イベント処理コントラクトの登録
    def set_event_contract(self, event, addr, abi):
        if DEBUG : print("set_contract", event, addr,abi)

        self.contracts[event] = {"address":addr, "abi":abi, "contract":self.w3.eth.contract(address=self.w3.toChecksumAddress(addr), abi=abi)}


    #　イベント処理コントラクトの実行
    def put_event_contract(self, event, account, params):
        if DEBUG : print("put_event_contract", event, account, params)
        if event not in self.contracts:
            print("no event:", event)
            return None
        if account == None:
            print("use self.account:", self.account)
            account = self.account
        cnt = self.contracts[event]['contract']
        ch_ac = self.w3.toChecksumAddress(self.account)
        keys = []
        values = []
        for k in params:
            keys.append(k)
            if k == 'account':
                #values.append(self.w3.toChecksumAddress(params[k]))
                values.append(params[k].lower())
                account = params[k].lower()
            else:
                values.append(params[k])
        tx = cnt.functions.put(self.w3.toChecksumAddress(account), keys, values).buildTransaction({'nonce':
self.w3.eth.getTransactionCount(ch_ac)})
        s_tx = self.w3.eth.account.signTransaction(tx, self.private_key)
        tx_hash = self.w3.eth.sendRawTransaction(s_tx.rawTransaction)
        tx_receipt = self.w3.eth.wait_for_transaction_receipt(tx_hash)

    #　イベント処理コントラクトの情報取得
    def get_event_contract(self, event, params):
        if DEBUG : print("get_event_contract", event, params)
        if event not in self.contracts:
            print("no event:", event)
            return None
        cnt = self.contracts[event]['contract']
        keys = []
        values = []
        for k in params:
            keys.append(k)
            if k == 'account':
                #values.append(self.w3.toChecksumAddress(params[k]))
                values.append(params[k].lower())
            else:
                values.append(params[k])
        res = cnt.functions.get(keys, values).call()
        if DEBUG : print("get_event_contract:res:",res)
        return res
```