

# Pythonによるデータアクセス



ストーリー:

①PCからPythonでIoTデバイスの代わりに  
任意のデータを送る

②IoTHubに送られてくるデータをPythonで  
モニタリングする。

内容に応じて、Cloud to Deviceメッセージを  
IoTデバイスに送る

③CosmosDBをPythonで読み込み、  
グラフ表示やCSV出力を行う。



改訂記録:

2022/06/02 初版 作成 陣内

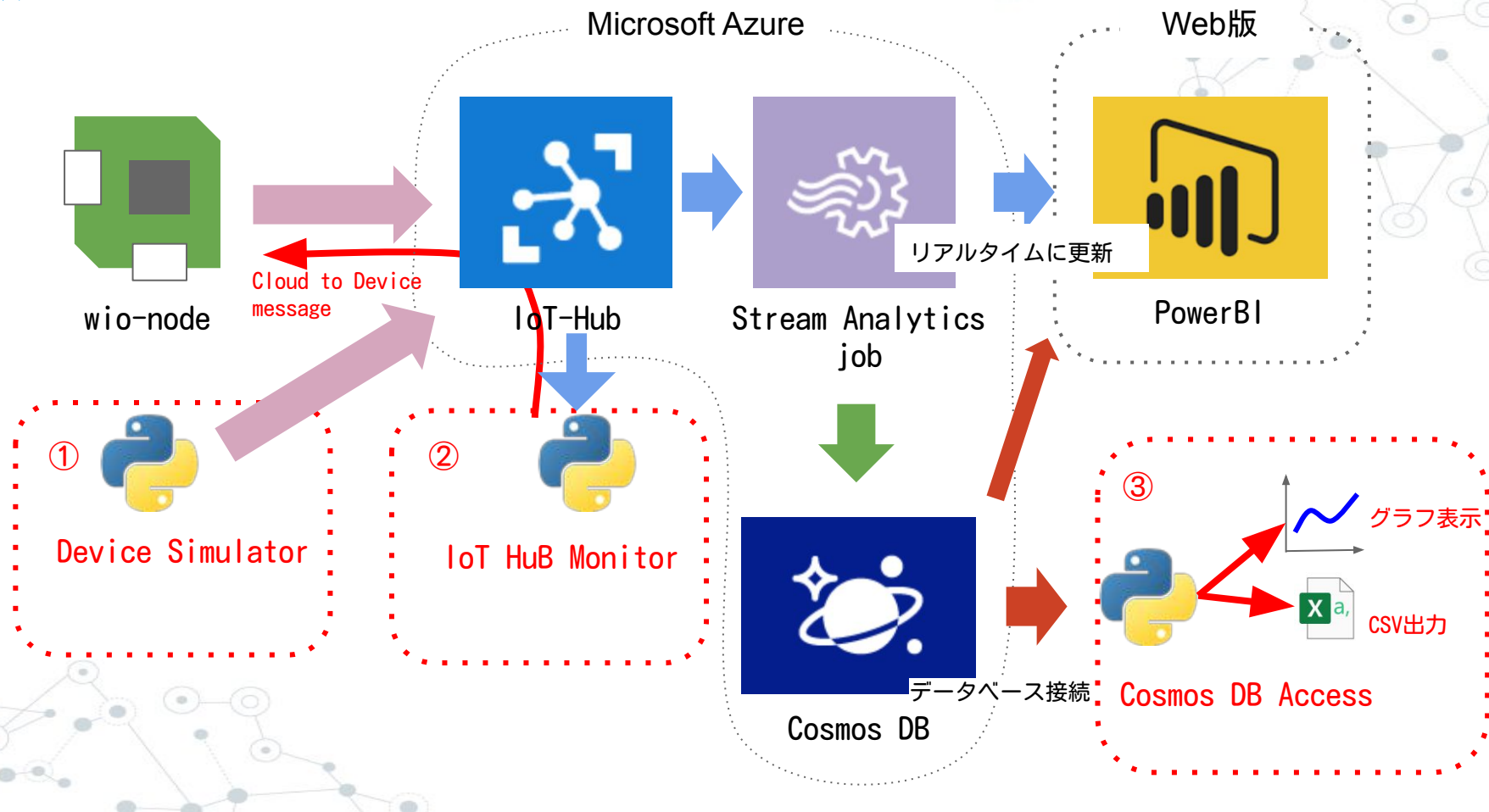




# 1. 概要

Python で Azureの IoT Hub、Cosmos DBに  
アクセスする

# 構成図



## 概要: ①Device Simulator

課題： サーバサイドのクエリやレポートの開発時に、テストのために必要なデータをほしい場合、実デバイス、センサーでは、任意のデータを送れない。

解決策： PythonでIoTデバイスの代わりに任意のデータを送ることができるとサーバサイドのテストがしやすくなる。




## 概要: ②IoT Hub Monitor

課題： Stream Analyticsのクエリでは、複数のデバイス間の数値を比べる処理など、複雑で記述が困難になる。

解決策： Pythonを使用すると、送られてくるデータを必要なだけ貯めて、任意の比較処理を記述できる。

さらにCloud to Deviceメッセージで、アラートをデバイスに送信することもできる。






## 概要: ③Cosmos DB Access

課題： CosmosDBのデータを自由に検索、データ処理したい。

解決策： PythonからCosmosDBのデータを、任意のSQLクエリで取得し分析できる。

一旦取り込んでしまえば、グラフ化やCSV出力も行える。





## 2. インストール

Pythonや、必要なモジュールのインストール方法について





## Pythonのインストール

本稿で使用するPythonのバージョンは、  
**3.10.4** です。

Windows版Pythonのインストール方法は、  
こちらのページを参照してください。

<https://www.python.jp/install/windows/install.html>

インストール時に以下のチェックを入れるのをお勧めします。  
"Add Python 3.x to PATH"

## 必要モジュールのインストール

コマンドプロンプトで以下のコマンドを順に実行します。

コマンド	モジュール内容
<code>python -m pip install azure-cosmos</code>	Cosmos DB用モジュール
<code>python -m pip install azure-iot-hub</code>	IoT HuB
<code>python -m pip install setuptools</code>	パッケージ管理
<code>python -m pip install azure-iot-device</code>	IoT デバイスシミュレーション
<code>python -m pip install azure-eventhub</code>	IoT Hub読み取り
<code>python -m pip install matplotlib</code>	グラフ描画
<code>python -m pip install pandas</code>	表形式データ操作



サンプルプログラムのダウンロード

FSPython フォルダをダウンロードして適当なフォルダに配置



# 3. Device Simulator



## プログラムの修正箇所

FSDeviceSimulator.py の以下のCONNECTION\_STRING  
の箇所に、 IoT Hub デバイスの接続文字列を必要なデ  
バイス数だけ設定する

```
23 #CONNECTION_STRING = os.getenv("IOTHUB_DEVICE_CONNECTION_STRING")
24 CONNECTION_STRING = [
25     "HostName=jinnouchiIoTHuB.azure-devices.net;DeviceId=jinnouchi07sim01;SharedAccessKey=YwvAS+alrUaseFZliteEWaIC86VCzePAVMxZt/LJIU=",
26     "HostName=jinnouchiIoTHuB.azure-devices.net;DeviceId=jinnouchi07sim02;SharedAccessKey=0e0MNFTZgt2Pgqd9uRBVnxisxWofoCXlt+tsJRRJ4uo=",
27     "HostName=jinnouchiIoTHuB.azure-devices.net;DeviceId=jinnouchi07sim03;SharedAccessKey=z1sNPqhz6CbMLTvUnfhAGamq+Lq1bVU0W04WRPAX9d4=",
28 ]
29
```

この例では3デバイス分をシミュレートする。

## プログラムの解説

- 先のページで設定した接続文字列の数だけ、シミュレータ(`run_telemetry_sample`関数)をマルチスレッドで起動する。
- `run_telemetry_sample`関数は、2つの振幅、周期、位相の異なるsin波と定数を足し合わせた値をIoT Hubへ送信する。sin波や定数のパラメータは、呼び出し時に乱数で決定する。

## 実行

コマンドプロンプトで以下のコマンドを実行する

\$ cd インストールフォルダ

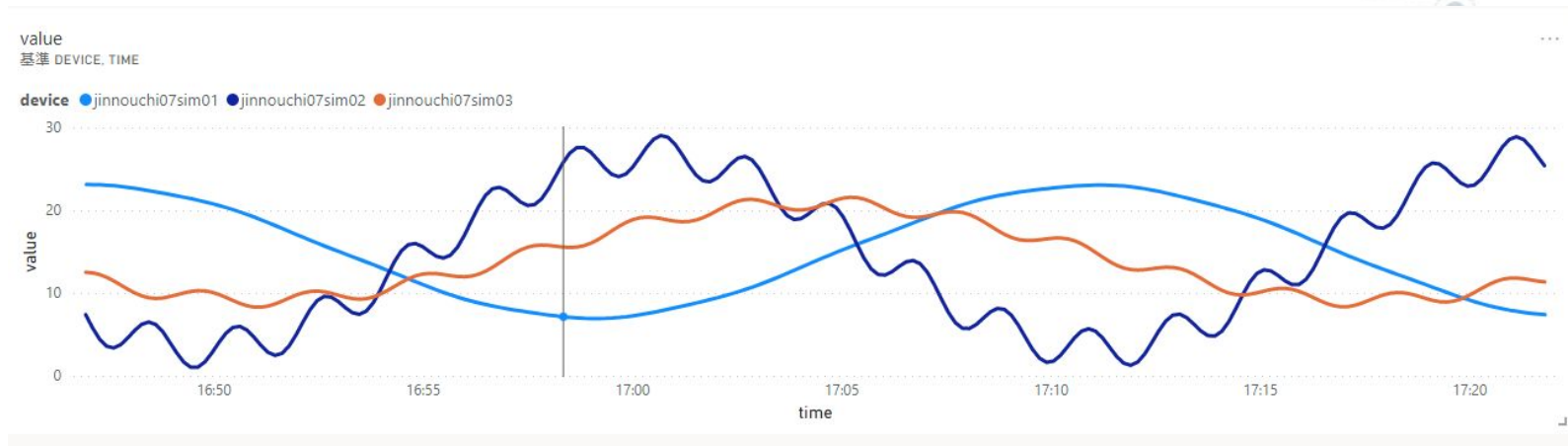
\$ python FSDeviceSimulator.py

```
コマンド プロンプト - python FSDeviceSimulator.py
Microsoft Windows [Version 10.0.19043.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Jinnouchi>cd C:\Users\Jinnouchi\Downloads\MyFSkit\FSPython
C:\Users\Jinnouchi\Downloads\MyFSkit\FSPython>prompt dev$&&S

dev> python FSDeviceSimulator.py
IoT Hub Quickstart #1 - Simulated device
Press Ctrl-C to exit
IoT Hub device sending periodic messages
IoT Hub device sending periodic messages
IoT Hub device sending periodic messages
start jinnouchi07sim03
sending message: {"params":{"sensor":"temp","espvalue": 17.1869470467044},"Dev":"jinnouchi07sim03","Id":0}
message successfully sent
start jinnouchi07sim01
sending message: {"params":{"sensor":"temp","espvalue": 18.485432431612345},"Dev":"jinnouchi07sim01","Id":0}
message successfully sent
start jinnouchi07sim02
sending message: {"params":{"sensor":"temp","espvalue": 19.975983504148378},"Dev":"jinnouchi07sim02","Id":0}
message successfully sent
sending message: {"params":{"sensor":"temp","espvalue": 16.647969237667695},"Dev":"jinnouchi07sim03","Id":1}
message successfully sent
sending message: {"params":{"sensor":"temp","espvalue": 18.84443171098802},"Dev":"jinnouchi07sim01","Id":1}
message successfully sent
sending message: {"params":{"sensor":"temp","espvalue": 18.246857381750477},"Dev":"jinnouchi07sim02","Id":1}
message successfully sent
sending message: {"params":{"sensor":"temp","espvalue": 16.21234058094003},"Dev":"jinnouchi07sim03","Id":2}
message successfully sent
```

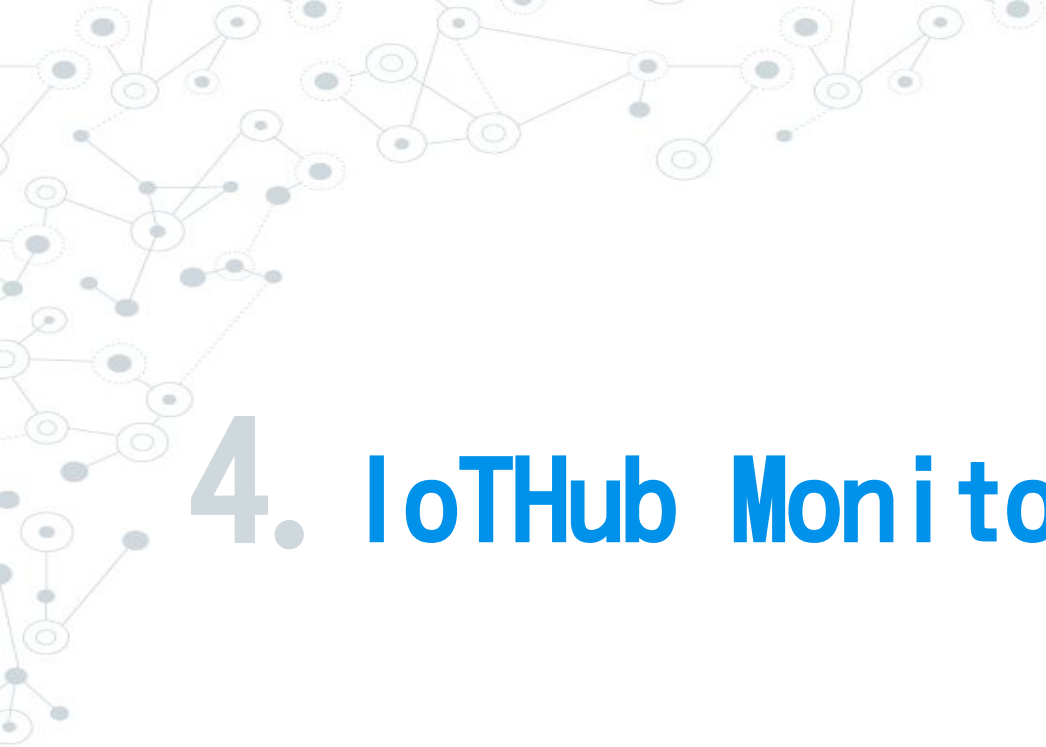
## 実行結果





## 制限事項

- このプログラムは `Ctrl-C`で停止するような処理を入れたつもりだが、マルチスレッド処理の関係で、`CTRL-C`を上手く処理できず、停止できない。
- 停止するには、コマンドプロンプトを閉じること。



## 4. IoT Hub Monitor



## プログラムの修正箇所①

FSUserDefinedProcess.py の以下の  
CONNECTION\_STRの箇所 にIoTHub の次ページの  
「イベントハブ互換エンドポイント」の文字列を入れてください

```
17 # ↓ CONNECTION_STR に自分の IoTHub の「組み込みのエンドポイント」の 「共有アクセスポリシー」を「service」を選択した時に表示される、  
18 # 「イベントハブ互換エンドポイント」の文字列を入れてください  
19 # CONNECTION_STR ="自分の イベントハブ互換エンドポイント"  
20 # 例: CONNECTION_STR ="Endpoint=sb://iothub-ns-jinnouchii-5698076-7882ac2621.servicebus.windows.net/;SharedAccessKeyName=service;SharedAccessKey=8PXMpYiE  
21 CONNECTION_STR = "Endpoint=sb://iothub-ns-jinnouchii-5698076-7882ac2621.servicebus.windows.net/;SharedAccessKeyName=service;SharedAccessKey=8PXMpYiE
```

# イベントハブ互換エンドポイント

Microsoft Azure | リソース、サービス、ドキュメントの検索 (G+/)

ホーム > すべてのリソース > jinnouchiloTHuB

» jinnouchiloTHuB | 組み込みのエンドポイント IoT Hub

検索 (Ctrl+/) << 保存 元に戻す

新しいコンシューマーグループを作成する

価格とスケール

デバイス管理

デバイス

IoT Edge

構成

更新プログラム

クエリ

ハブ設定

組み込みのエンドポイント

メッセージルーティング

ファイルのアップロード

フェールオーバー

プロパティ

イベントハブ互換エンドポイント

サービス接続のアクセス許可を付与するポリシーのみを選択してください。サービス許可は、組み込みのエンドポイントに適用されます。

共有アクセスポリシー ①

service

イベントハブ互換エンドポイント ①

Endpoint=sb://iothub-ns-jinnouchii-5698076-7882ac2621.servicebus.windows.net;/SharedAccessKeyName=service;SharedAccessKe...

cloud-to-device メッセージング

メッセージの保存期間と再試行回数を制御します。

既定の TTL ①

時間 1

フィードバック維持時間 ①

時間 1

20

## プログラムの修正箇所②

FSC2DMessage.py の以下の箇所を記入。

CONNECTION\_STRING に次ページの接続文字列を設定する。

DEVICE\_ID にアラートを鳴らすデバイスのIDを設定する。

C2D messageを受け取る方法は、FSKit2の  
09\_FSセンサ活用例：ブザー.pdf を参照の事。

```
16 # ここに自分の IoTHubの接続文字列を入れてください。
17 #CONNECTION_STRING = "{IoTHubConnectionString}"
18 #DEVICE_ID = "{deviceId}"
19
20 CONNECTION_STRING = "HostName=jinnouchiIoTHub.azure-devices.net;SharedAccessKeyName=service;SharedAccessKey=8PXMpYiEF9cunSzhq/MshdQZGn0R1Tv+dTAze10J"
21 DEVICE_ID = "jinnouchi03"
22
```

# 接続文字列の取得方法

Microsoft Azure

リソース、サービス、ドキュメントの検索 (G+)

ホーム > すべてのリソース > jinnouchiloTHuB

jinnouchiloTHuB | 共有アクセス ポリシー

検索 (Ctrl+/)

プロパティ

ロック

セキュリティ設定

ID

共有アクセス ポリシー

ネットワーク

証明書

Defender for IoT

概要

セキュリティ通知

推奨事項

設定

監視

警告

メトリック

診断設定

共有アクセス ポリシーは、IoT ハブ機能を使用するためのセキュリティ トークンを生成するために使用できます

共有アクセス ポリシーを使用した接続

保存 変更の破棄

許可

拒否

共有アクセス ポリシーの管理

共有アクセス ポリシーの追加 最新の情報に更新 削除

ポリシー名	アクセス許可
iothubowner	レジストリ読み取り, レジストリ書き込み, サービス接続, デバイス接続
<input checked="" type="checkbox"/> service	サービス接続
device	デバイス接続
registryRead	レジストリ読み取り
registryReadWrite	レジストリ読み取り, レジストリ書き込み

service

jinnouchiloTHuB

プライマリ キーの再生成 セカンダリ キーの再生成 キーのスワップ

プライマリ キー

セカンダリ キー

プライマリ接続文字列

セカンダリ接続文字列

アクセス許可

☐ レジストリ読み取り

☐ レジストリ書き込み

☒ サービス接続

☐ デバイス接続

アクセス許可の更新

キャンセル

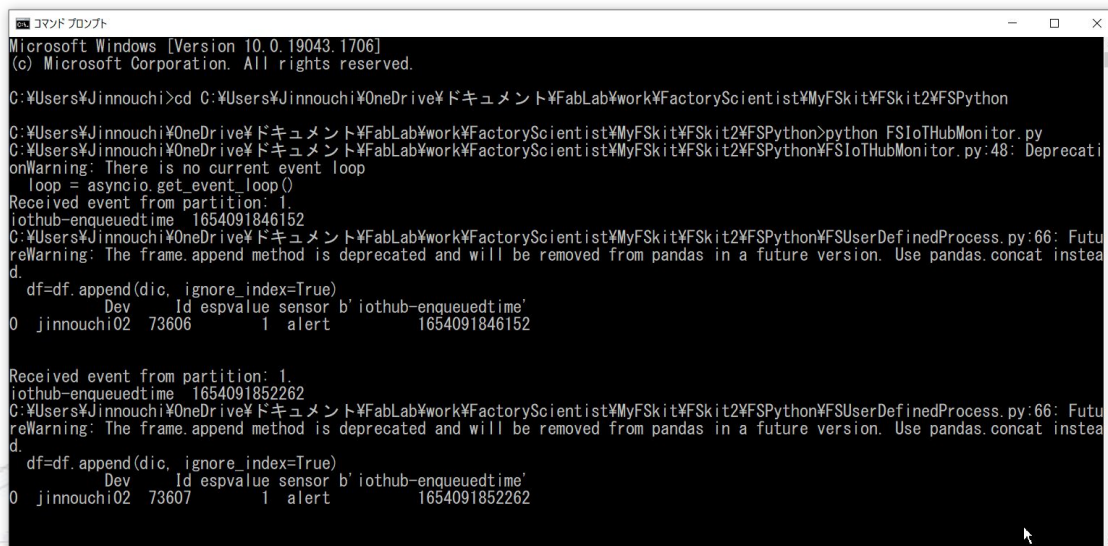
22

## 実行

コマンドプロンプトで以下のコマンドを実行する

\$ cd インストールフォルダ

\$ python FSIoTHubMonitor.py



```
Microsoft Windows [Version 10.0.19043.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Jinnouchi>cd C:\Users\Jinnouchi\OneDrive\ドキュメント\FabLab\work\FactoryScientist\MyFSkit\FSkits\FSPython

C:\Users\Jinnouchi\OneDrive\ドキュメント\FabLab\work\FactoryScientist\MyFSkit\FSkits\FSPython>python FSIoTHubMonitor.py
C:\Users\Jinnouchi\OneDrive\ドキュメント\FabLab\work\FactoryScientist\MyFSkit\FSkits\FSPython\FSIoTHubMonitor.py:48: Deprecati
onWarning: There is no current event loop
  loop = asyncio.get_event_loop()
Received event from partition: 1.
iothub-enqueuedtime 1654091846152
C:\Users\Jinnouchi\OneDrive\ドキュメント\FabLab\work\FactoryScientist\MyFSkit\FSkits\FSPython\FUserDefinedProcess.py:66: Futu
reWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instea
d.
  df=df.append(dic, ignore_index=True)
   Dev  Id espvalue sensor b' iothub-enqueuedtime'
0 jinnouchi02  73606      1  alert      1654091846152

Received event from partition: 1.
iothub-enqueuedtime 1654091852262
C:\Users\Jinnouchi\OneDrive\ドキュメント\FabLab\work\FactoryScientist\MyFSkit\FSkits\FSPython\FUserDefinedProcess.py:66: Futu
reWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instea
d.
  df=df.append(dic, ignore_index=True)
   Dev  Id espvalue sensor b' iothub-enqueuedtime'
0 jinnouchi02  73607      1  alert      1654091852262
```



# 5. CosmosDB Access





## プログラムの修正箇所

FSCosmosDB.pyの以下を設定

endpoint: cosmosDBのURI

key: プライマリキー

database: DB名

container\_name: コンテナ名

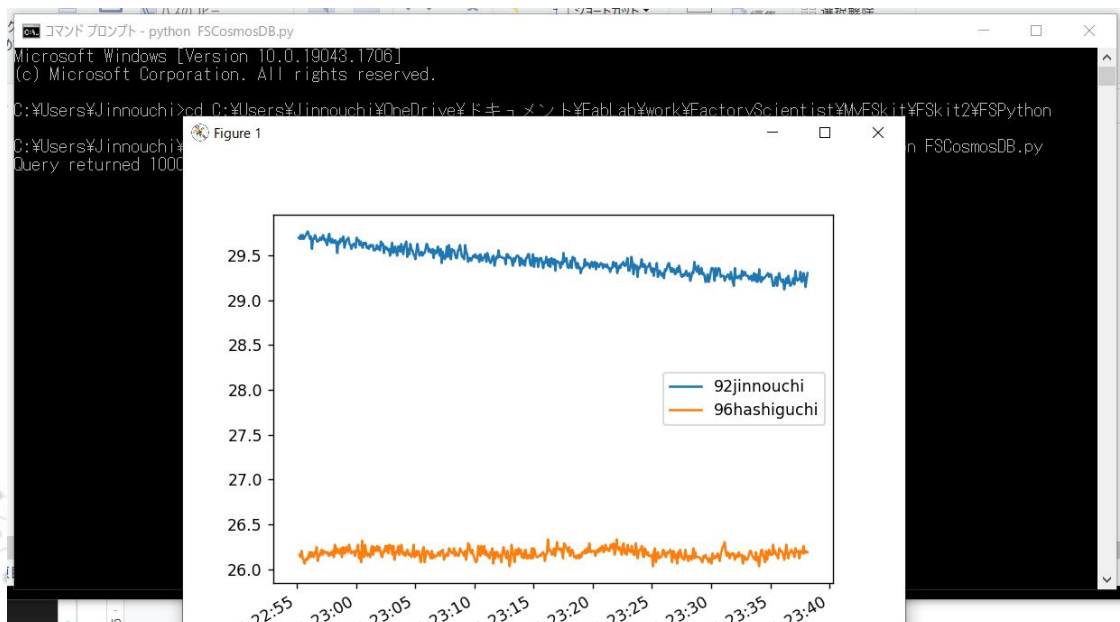
```
4 # CosmosDB の URIとプライマリキー
5 endpoint = "https://fs2021cosmos.documents.azure.com:443/"
6 key = 'dnkYLF67YxVTXGZ0DD0pjz14EBp4qSrjJq6rsiRiybJefPSKaqqmdBdk970uxBLad4pQ1rZ97ziHDqv6HkBAZg=='
7
8
9 client = CosmosClient(endpoint, key)
10 database_name = 'fsdatabase' #DB名を設定する
11 database = client.create_database_if_not_exists(id=database_name)
12 container_name = 'fscontainer' #コンテナ名を設定する
13 container = database.create_container_if_not_exists(
```

実行:

コマンドプロンプトで以下のコマンドを実行する

```
$ cd インストールフォルダ
```

```
$ python FSCosmosDB.py
```



## プログラムの解説とカスタマイズ

プログラムは以下の3つの部分に大きく分かれている

1. CosmosDBに接続して、クエリを発行、データ取得  
結果は `items` (リスト) に入る。
2. グラフ出力  
途中で `items` から `df` (DataFrame) に変換している
3. CSV出力  
2で作成したDFを使用して、CSV出力している。