

<https://www.halvorsen.blog>



Python and Microsoft Azure Databases

Hans-Petter Halvorsen

Contents

- SQL Server
- Python and SQL server
- Datalogging Example Saving Data to Local SQL Server Database
- Microsoft Azure
- Databases in Microsoft Azure
- Datalogging Example Saving Data to Azure Database



SQL Server

Database Systems

- Oracle
- MySQL
- MariaDB
- Sybase
- Microsoft Access
- Microsoft SQL Server
- ... (we have hundreds different Database Systems)

SQL Server

- SQL Server consists of a Database Engine and a Management Studio.
- The Database Engine has no graphical interface - it is just a service running in the background of your computer (preferable on the server).
- The Management Studio is graphical tool for configuring and viewing the information in the database. It can be installed on the server or on the client (or both).

SQL Server

- SQL Server Express
 - Free version of SQL Server that has all we need for the exercises in this Tutorial
- SQL Server Express consist of 2 parts (separate installation packages):
 - SQL Server Express
 - SQL Server Management Studio (SSMS) – This software can be used to create Databases, create Tables, Insert/Retrieve or Modify Data, etc.
- SQL Server Express Installation:
<https://youtu.be/hhhggAlUYo8>

SQL Server Management Studio

3 New Query

1 Your SQL Server

2 Your Database

4 Write your Query here

5 The result from your Query

SQLQuery1.sql - P...SCHOOL (sa (52))*

```
select * from SCHOOL
```

	SchoolId	SchoolName	Description	Address	Phone	PostCode	PostAddress
1	1	TUC	The best school	Telemark	NULL	NULL	NULL
2	2	MIT	OK School	USA	NULL	NULL	NULL
3	3	NTNU	The second best school	Trondheim	NULL	NULL	NULL
4	4	University of Oslo	The third best school	Oslo	NULL	NULL	NULL

Query executed successfully. | PC88235\DEVELOPMENT (10.50 ... | sa (52) | SCHOOL | 00:00:00 | 4 rows

Ready | Ln 1 | Col 21 | Ch 21 | INS



Python and SQL Server

Hans-Petter Halvorsen

[Table of Contents](#)

Python

- Python is a fairly old Programming Language (1991) compared to many other Programming Languages like C# (2000), Swift (2014), Java (1995), PHP (1995).
- Python has during the last 10 years become more and more popular.
- Today, Python has become one of the most popular Programming Languages.

Software used in this Tutorial:

- Anaconda Distribution (Python + most used Libraries/Packages are included)
- Spyder Python editor (included with Anaconda Distribution)

Python Drivers for SQL Server

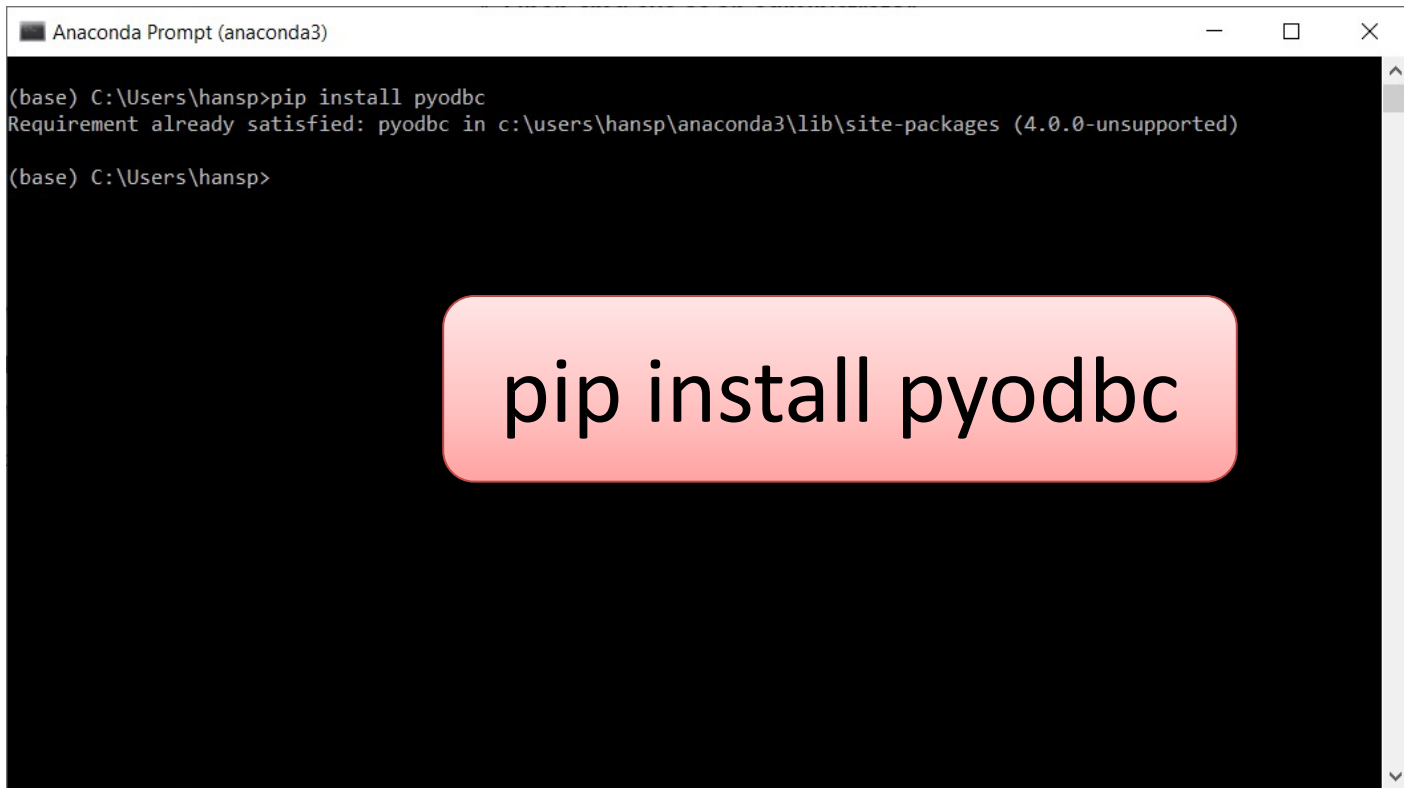
- There are several python SQL drivers available:
 - pyodbc
 - pymssql
- These Drivers are not made made Microsoft but the Python Community.
- However, Microsoft places its testing efforts and its confidence in **pyodbc** driver.
- Microsoft contributes to the pyODBC open-source community and is an active participant in the repository at GitHub

<https://docs.microsoft.com/sql/connect/python/python-driver-for-sql-server>

pyodbc

- pyodbc is an open-source Python module that can access ODBC databases, e.g., SQL Server
- <https://pypi.org/project/pyodbc/>
- Installation: `pip install pyodbc`

pyodbc




```
Anaconda Prompt (anaconda3)
(base) C:\Users\hansp>pip install pyodbc
Requirement already satisfied: pyodbc in c:\users\hansp\anaconda3\lib\site-packages (4.0.0-unsupported)
(base) C:\Users\hansp>
```

pip install pyodbc

Connect to Database from Python

The newest and
recommend driver



```
import pyodbc

driver = "{ODBC Driver 17 for SQL Server}"
server = "xxxxxx"
database = "xxxxxx"
username = "xxxxxx"
password = "xxxxxx"
conn = pyodbc.connect("DRIVER=" + driver
                      + ";SERVER=" + server
                      + ";DATABASE=" + database
                      + ";UID=" + username
                      + ";PWD=" + password )
```

Connect to Database from Python

Example:

```
import pyodbc
```

```
driver = "{ODBC Driver 17 for SQL Server}"
```

```
server = "TESTPC\\SQLEXPRESS"
```

```
database = "BOOKSTORE"
```

```
username = "sa"
```

```
password = "Test123"
```

```
conn = pyodbc.connect("DRIVER=" + driver  
                        + ";SERVER=" + server  
                        + ";DATABASE=" + database  
                        + ";UID=" + username  
                        + ";PWD=" + password )
```

Server Name

If Server is on your local PC,
you can use LOCALHOST

Instance Name (you can have
multiple instances of SQL Server
on the same computer)

Here is the built-in "sa" user (System Administrator) used to connect to the Database. In general, you should use another user than the sa user. The sa user is used here for simplicity. You can easily create a new user in SQL Server Management Studio



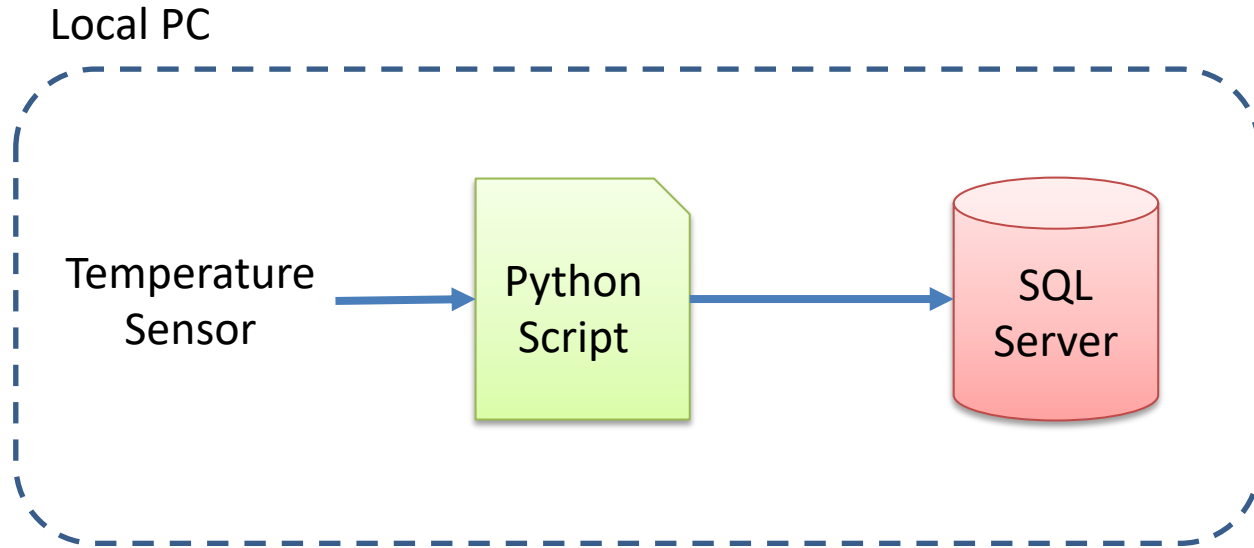
Datalogging Example

Saving Data to Local SQL Server Database

Datalogging Example

- We can log data from a DAQ device or similar
- We start by creating a simple Random Generator that simulates a Temperature Sensor and log these data to the SQL Server database
- Then we will in another script read the data from the database and plot them.

System Overview



SQL Server Database

Let's create a New Database called, e.g., “LOGGINGSYSTEM”

We insert the following Table:

```
CREATE TABLE [MEASUREMENTDATA]
(
    [MeasurmentId] [int] IDENTITY(1, 1) NOT NULL PRIMARY KEY,
    [SensorName] [varchar](50) NOT NULL,
    [MeasurementValue] float NOT NULL,
    [MeasurementDateTime] datetime NOT NULL
)
GO
```

Note! This is a very simplified Database to show the basic principles. It does not reflect best practice. Typically, you have multiple tables that are related to each other and more columns like Unit, etc.

Logging Data

```
import pyodbc
import random
import time
from datetime import datetime
import database

# Connect to Database
connectionString = database.GetConnectionString()
conn = pyodbc.connect(connectionString)
cursor = conn.cursor()
query = "INSERT INTO MEASUREMENTDATA (SensorName, MeasurementValue, MeasurementDateTime) VALUES (?, ?, ?)"

sensorName = "Temperature"
Ts = 10 # Sampling Time
N = 20
for k in range(N):
    # Generate Random Data
    LowLimit = 20
    UpperLimit = 25
    measurementValue = random.randint(LowLimit, UpperLimit)

    #Find Date and Time
    now = datetime.now()
    datetimeformat = "%Y-%m-%d %H:%M:%S"
    measurementDateTime = now.strftime(datetimeformat)

    # Insert Data into Database
    parameters = sensorName, measurementValue, measurementDateTime
    cursor.execute(query, parameters)
    cursor.commit()

# Wait
time.sleep(Ts)
```

Connection String

The Connection string has been put in a separate Python File called “database.py”:

```
def GetConnectionString():  
    driver = "{ODBC Driver 17 for SQL Server}"  
    server = "xxxxxxx"  
    database = "LOGGINGSYSTEM"  
    username = "sa"  
    password = "xxxxxxx"  
  
    connectionString = "DRIVER=" + driver + ";SERVER=" + server + ";DATABASE=" + database + ";UID=" + username + ";PWD=" + password  
  
    return connectionString
```

Logged Data

The screenshot shows Microsoft SQL Server Management Studio with a query window containing the following SQL statement:

```
select * from MEASUREMENTDATA
```

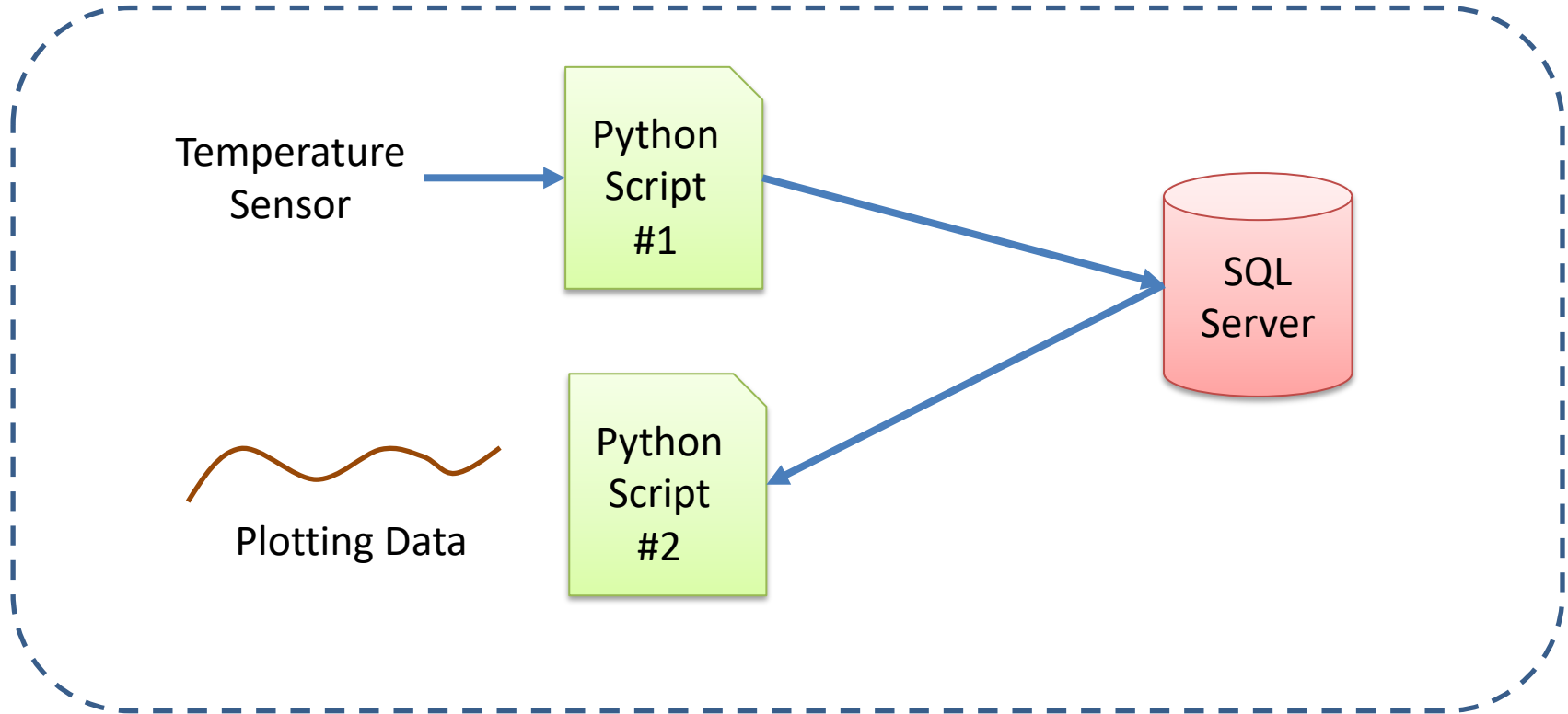
The query results are displayed in a table with the following columns: MeasurementId, SensorName, Measurement Value, and MeasurementDateTime. The results show 20 rows of temperature measurements.

MeasurementId	SensorName	Measurement Value	MeasurementDateTime
4	Temperature	25	2021-11-25 13:46:11.000
5	Temperature	22	2021-11-25 13:46:21.000
6	Temperature	20	2021-11-25 13:46:31.000
7	Temperature	22	2021-11-25 13:46:41.000
8	Temperature	25	2021-11-25 13:46:51.000
9	Temperature	21	2021-11-25 13:47:01.000
10	Temperature	23	2021-11-25 13:47:12.000
11	Temperature	25	2021-11-25 13:47:22.000
12	Temperature	22	2021-11-25 13:47:32.000
13	Temperature	24	2021-11-25 13:47:42.000
14	Temperature	24	2021-11-25 13:47:52.000
15	Temperature	22	2021-11-25 13:48:02.000
16	Temperature	21	2021-11-25 13:48:12.000
17	Temperature	23	2021-11-25 13:48:22.000
18	Temperature	20	2021-11-25 13:48:32.000
19	Temperature	23	2021-11-25 13:48:42.000
20	Temperature	25	2021-11-25 13:48:52.000
21	Temperature	24	2021-11-25 13:49:02.000
22	Temperature	21	2021-11-25 13:49:12.000
23	Temperature	20	2021-11-25 13:49:22.000

The status bar at the bottom indicates: Query executed successfully. XPS15HPH\SQLEXPRESS (13.0 RTM) sa (53) LOGGINGSYSTEM 00:00:00 20 rows

System Overview

Local PC



Plotting Data

```
import pyodbc
import matplotlib.pyplot as plt
import database

sensorName = "Temperature"

# Connect to Database
connectionString = database.GetConnectionString()
conn = pyodbc.connect(connectionString)
cursor = conn.cursor()
query = "SELECT MeasurementValue, MeasurementDateTime FROM MEASUREMENTDATA WHERE SensorName=?"
parameters = [sensorName]

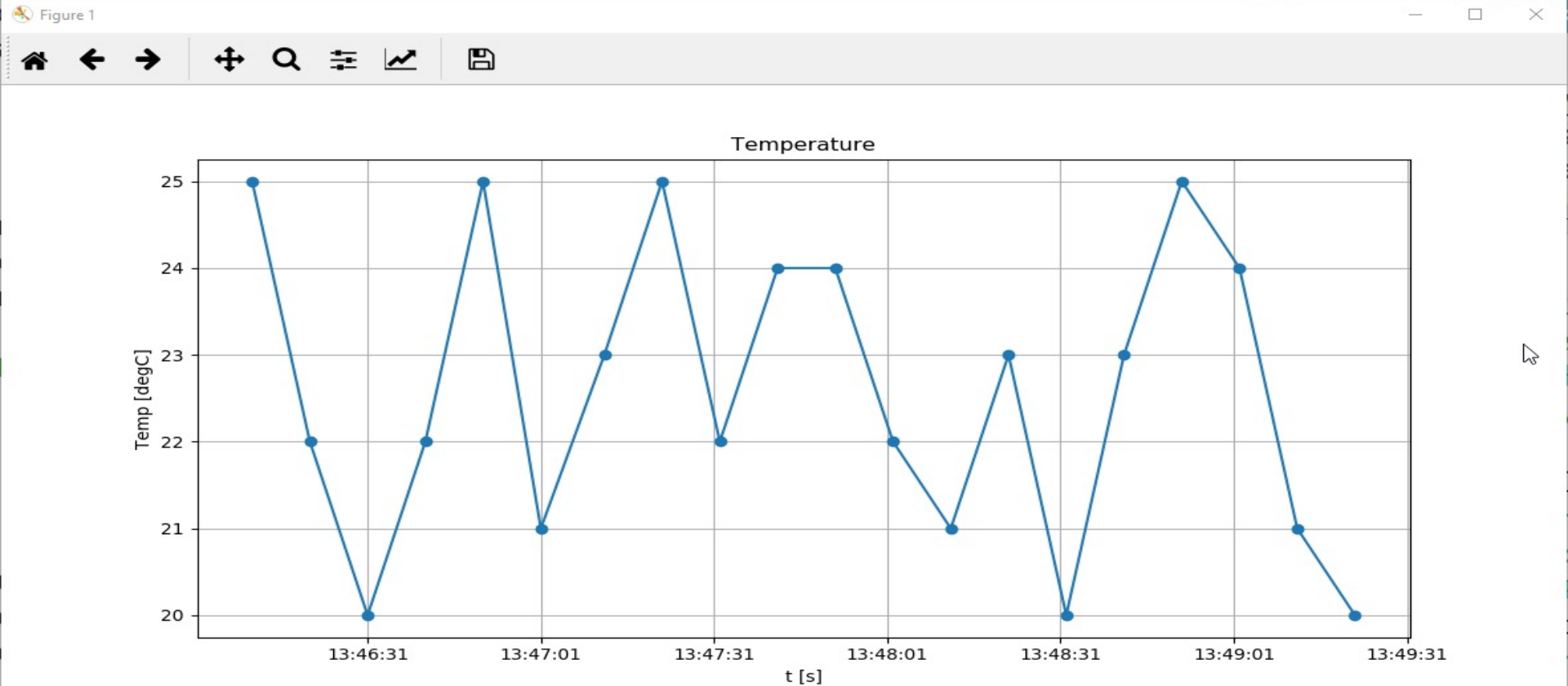
t = []; data = []

# Retrieving and Formatting Data
for row in cursor.execute(query, parameters):
    measurementValue = row.MeasurementValue
    measurementDateTime = row.MeasurementDateTime

    data.append(measurementValue)
    t.append(measurementDateTime)

# Plotting
plt.plot(t, data, 'o-')
plt.title('Temperature')
plt.xlabel('t [s]')
plt.ylabel('Temp [degC]')
plt.grid()
plt.show()
```

Plotted Data





Microsoft Azure

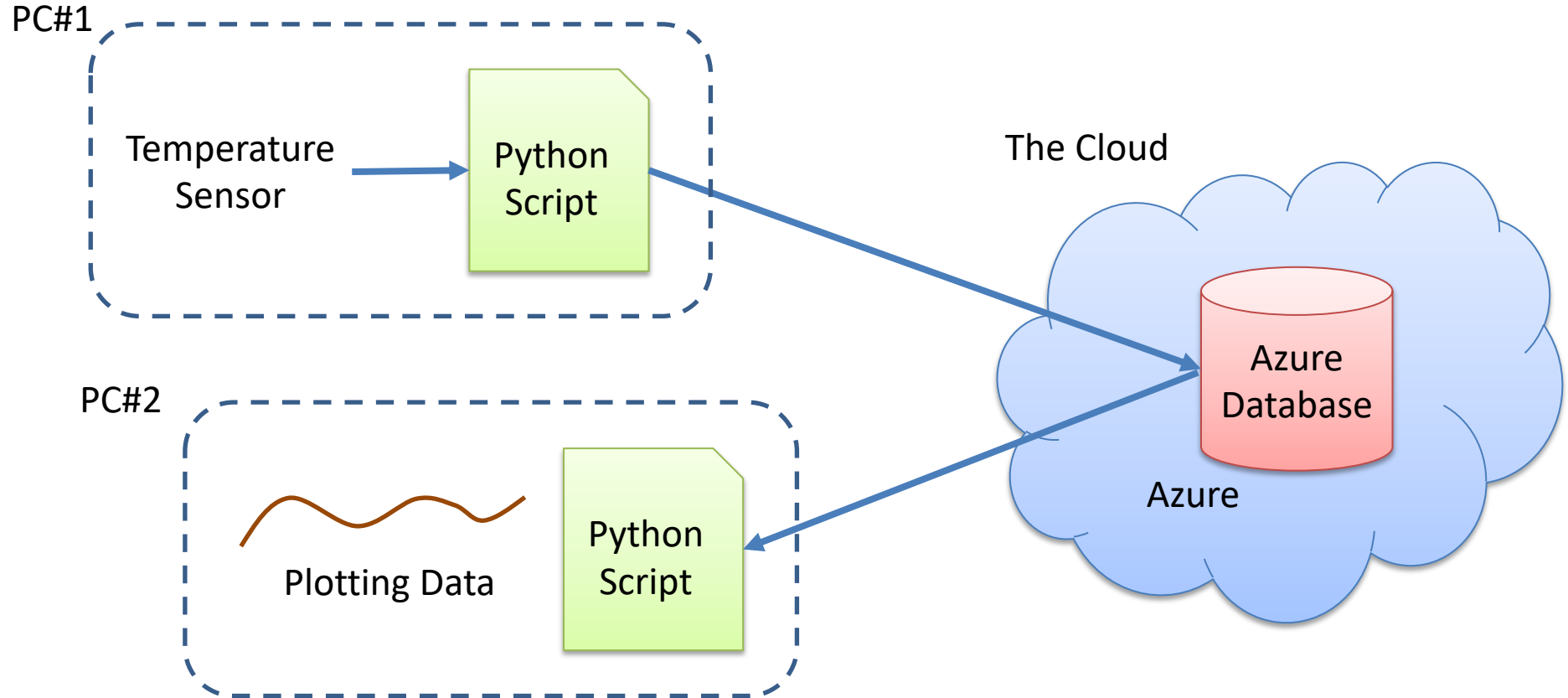
Microsoft Azure

- Microsoft Azure is a Cloud Platform from Microsoft
- You could say it is “Windows running in the Cloud”
- Here you can host Databases, Web Applications, Virtual Machines, etc.
- Azure Portal:
<https://portal.azure.com>

Next Step

- We have created a local Datalogging System
- Next, we want to replace the local SQL Server Database with a Database in the Cloud
- We will use Microsoft Azure
- In that way others can get access to the logged data as well

System Overview





Databases in Microsoft Azure

Configure Database in Azure

Microsoft Azure Search resources, services, and docs (G+)

Home > SQL databases


SQL databases

Default Directory

[+ Create](#) [Reservations](#) [Manage view](#) [Refresh](#) [Export to CSV](#) [Open query](#) [Assign tags](#)

Filter for any field... Subscription == **Azure for Students** Resource group == **all** Location == **all**

Showing 1 to 1 of 1 records.

<input type="checkbox"/> Name ↑↓	Server ↑↓	Replica type ↑↓
<input type="checkbox"/>  LOGGINGSYSTEM (hph/LOGGINGSYSTEM)	hph	--

Microsoft Azure Search resources, services, and docs (G+)

Home > SQL databases >

Create SQL Database

Microsoft

[Basics](#) [Networking](#) [Security](#) [Additional settings](#) [Tags](#) [Review + create](#)

Create a SQL database with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *
Resource group *
[Create new](#)

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name *
Server *
[Create new](#)


Want to use SQL elastic pool? * Yes No

Compute + storage *
General Purpose
Gen5, 2 vCores, 32 GB storage, zone redundant disabled
[Configure database](#)

Backup storage redundancy

Choose how your PITR and LTR backups are replicated. Geo restore or ability to recover from regional outage is only available when geo-redundant storage is selected.

Backup storage redundancy Locally-redundant backup storage
 Zone-redundant backup storage
 Geo-redundant backup storage

 Selected value for backup storage redundancy is Geo-redundant backup

[Review + create](#) [Next: Networking >](#)

Create Table

We will use SQL Server Management Studio and connect to the Azure Database:

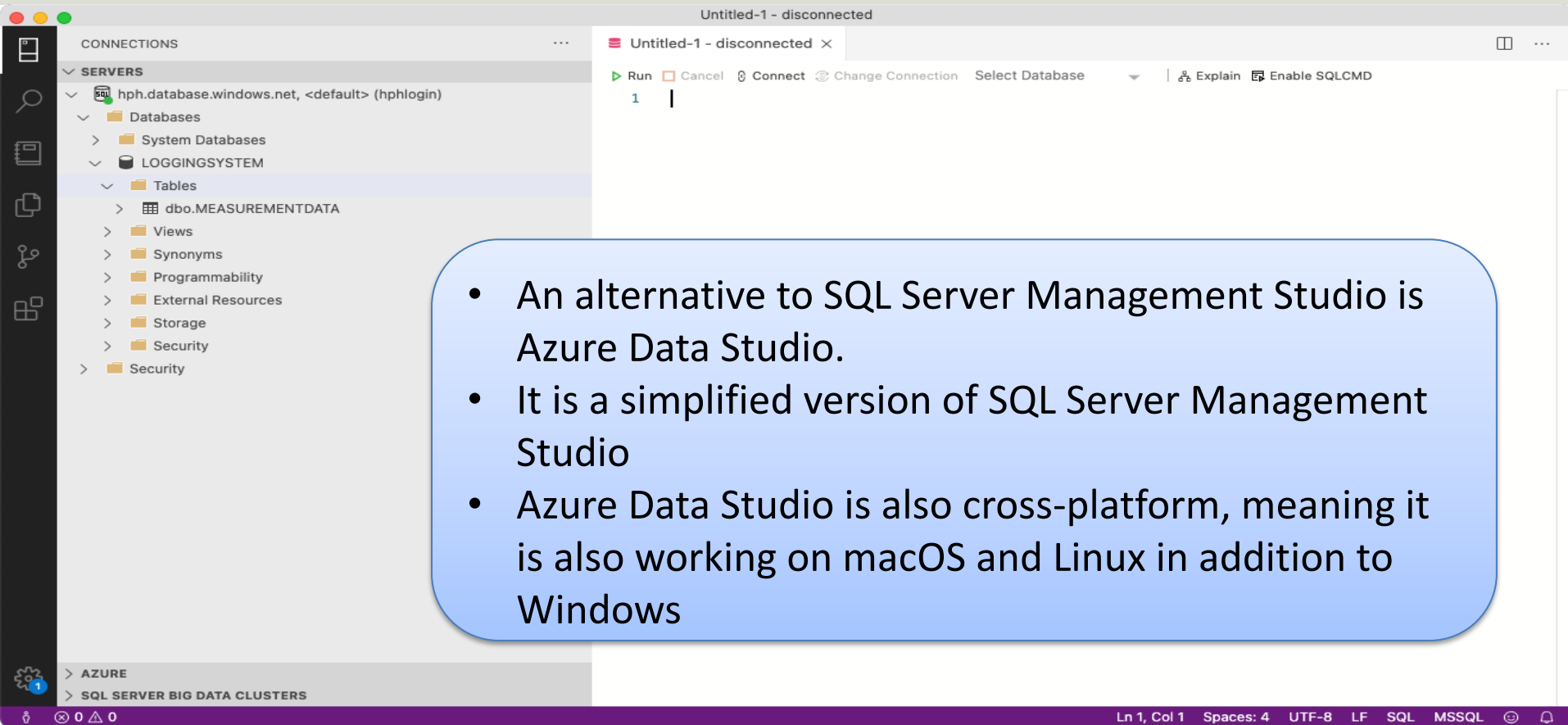
The screenshot displays the Microsoft SQL Server Management Studio (SSMS) interface. In the foreground, the 'Connect to Server' dialog box is open, titled 'SQL Server'. The 'Server type' is set to 'Database Engine'. The 'Server name' is 'hph.database.windows.net', 'Authentication' is 'SQL Server Authentication', and the 'Login' is 'hphlogin'. The 'Remember password' checkbox is checked. The 'Connect' button is highlighted.

In the background, the SSMS main window is visible. The 'Object Explorer' on the left shows the server hierarchy for 'hph.database.windows.net (SQL Server 12.0.2000.8 - hphlogin)', with 'dbo.MEASUREMENTDATA' selected. The main query editor window shows the following SQL code:

```
CREATE TABLE [MEASUREMENTDATA]
(
    [MeasurementId] [int] IDENTITY(1, 1) NOT NULL PRIMARY KEY,
    [SensorName] [varchar](50) NOT NULL,
    [MeasurementValue] float NOT NULL,
    [MeasurementDateTime] datetime NOT NULL
)
GO
```

The status bar at the bottom indicates the current position is 'Ln 9 Col 1 Ch 1 INS' and the connection status is 'Connected. (1/1)'. The server name and database are 'hph.database.windows.net (1... hphlogin (90) LOGGINGSYSTEM' and the execution time is '00:00:00 | 0 rows'.

Azure Data Studio



The screenshot displays the Azure Data Studio interface. On the left, the 'CONNECTIONS' pane shows a tree view under 'SERVERS' for 'hph.database.windows.net, <default> (hphlogin)'. The tree includes folders for 'Databases', 'System Databases', 'LOGGINGSYSTEM', 'Tables', 'Views', 'Synonyms', 'Programmability', 'External Resources', 'Storage', 'Security', and 'dbo.MEASUREMENTDATA'. The main editor area shows a query editor with a toolbar containing 'Run', 'Cancel', 'Connect', 'Change Connection', and 'Select Database'. The query text is '1 |'. The bottom status bar shows 'Ln 1, Col 1 Spaces: 4 UTF-8 LF SQL MSSQL'.

- An alternative to SQL Server Management Studio is Azure Data Studio.
- It is a simplified version of SQL Server Management Studio
- Azure Data Studio is also cross-platform, meaning it is also working on macOS and Linux in addition to Windows

Azure Query Editor

The screenshot displays the Azure Query Editor interface for a SQL database named LOGGINGSYSTEM (hph/LOGGINGSYSTEM). The interface is divided into several sections:

- Left Navigation Panel:** Contains a search bar and a list of navigation options including Overview, Activity log, Tags, Diagnose and solve problems, Quick start, Query editor (preview) (highlighted), Power Platform (Power BI, Power Apps, Power Automate), Settings (Compute + storage, Connection strings, Properties, Locks), and Data management (Replicas, Sync to other databases). Integrations like Stream analytics (preview) and Add Azure Search are also listed.
- Top Bar:** Shows the breadcrumb path Home > SQL databases > LOGGINGSYSTEM (hph/LOGGINGSYSTEM) and the title LOGGINGSYSTEM (hph/LOGGINGSYSTEM) | Query editor (preview). It includes a search bar (Search (Cmd+)), Login, New Query, Open query, and Feedback buttons.
- Object Explorer:** Displays the database structure for LOGGINGSYSTEM (hphlogin), including Tables, Views, and Stored Procedures. A message indicates: "Showing limited object explorer here. For full capability please open SSDT."
- Query Editor:** Features a toolbar with Run, Cancel query, Save query, Export data as, and Show only Editor buttons. The query text area contains a single line with the number "1".
- Results and Messages:** Includes tabs for Results and Messages, and a search bar labeled "Search to filter items..."

A 3.alternative is the Query Editor
in the Microsoft Azure Portal

Firewall

We need to give access to the computers running the Python Scripts

The screenshot shows the Microsoft Azure portal interface. At the top, there is a search bar and navigation links. The main content area is divided into a left sidebar with navigation options like 'Overview', 'Activity log', and 'Tags', and a main panel. The main panel shows the 'LOGGINGSYSTEM (hph/LOGGINGSYSTEM)' resource, which is an SQL database. A red box highlights the 'Set server firewall' button in the top action bar. Below this, the 'Firewall settings' page is visible, showing various configuration options: 'Deny public network access' is unchecked, 'Minimum TLS Version' is set to 1.2, 'Connection Policy' is set to Default, and 'Allow Azure services and resources to access this server' is set to No. At the bottom, there is a table for 'Client IP address' with columns for 'Rule name', 'Start IP', and 'End IP', and a table with three empty input fields.

Microsoft Azure

Search resources, services, and docs (G+)

Home >

LOGGINGSYSTEM (hph/LOGGINGSYSTEM) SQL database

Search (Cmd+/) <<

Copy Restore Export Set server firewall Delete Connect with... Feedback

Overview

Activity log

Tags

Diagnose and solve problems

Quick start

Query editor (preview)

Essentials

Resource group (Move) : halvorsen

Status : Online

Location : West Europe

Subscription (Move) : Azure for Students

Subscription ID : 3c6a9d07-b932-4d

Tags (Edit) : Click here to add t

Microsoft Azure

Search resources, services, and docs (G+)

Home > LOGGINGSYSTEM (hph/LOGGINGSYSTEM) >

Firewall settings hph (SQL server)

Save Discard Add client IP

Deny public network access

Minimum TLS Version ⓘ

1.0 1.1 1.2

Connection Policy ⓘ

Default Proxy Redirect

Allow Azure services and resources to access this server ⓘ

Yes No

Client IP address 128.39.132.145

Rule name	Start IP	End IP

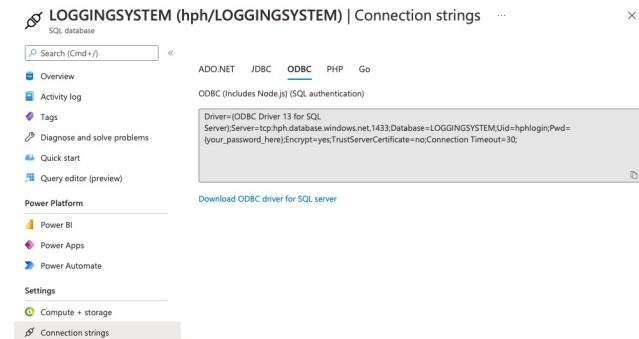


Datalogging Example

Saving Data to Azure Database

Python Code

- The Python Code is 100% the same
- The only thing we need to change is the Connection String
- You find the Connection String in the Azure Portal



Connection String

The Connection string has been put in a separate Python File called “database.py”:

```
def GetConnectionString():
    driver = "{ODBC Driver 17 for SQL Server}"
    server = "xxxxxxx"
    database = "LOGGINGSYSTEM"
    username = "sa"
    password = "xxxxxxx"

    connectionString = "DRIVER=" + driver + ";SERVER=" + server + ";DATABASE=" + database + ";UID=" + username + ";PWD=" + password

    return connectionString

def GetConnectionStringAzure():
    driver = "{ODBC Driver 17 for SQL Server}"
    server = "xxx.database.windows.net"
    database = "LOGGINGSYSTEM"
    username = "xxxxxxx"
    password = "xxxxxxx"

    connectionString = "DRIVER=" + driver + ";SERVER=" + server + ";DATABASE=" + database + ";UID=" + username + ";PWD=" + password

    return connectionString
```

```
import pyodbc
import random
import time
from datetime import datetime
import database

# Connect to Database
connectionString = database.GetConnectionStringAzure()
conn = pyodbc.connect(connectionString)
cursor = conn.cursor()
query = "INSERT INTO MEASUREMENTDATA (SensorName, MeasurementValue, MeasurementDateTime) VALUES (?, ?, ?)"

sensorName = "Temperature"
Ts = 10 # Sampling Time
N = 20
for k in range(N):
    # Generate Random Data
    LowLimit = 20
    UpperLimit = 25
    measurementValue = random.randint(LowLimit, UpperLimit)

    #Find Date and Time
    now = datetime.now()
    datetimeformat = "%Y-%m-%d %H:%M:%S"
    measurementDateTime = now.strftime(datetimeformat)

    # Insert Data into Database
    parameters = sensorName, measurementValue, measurementDateTime
    cursor.execute(query, parameters)
    cursor.commit()

# Wait
time.sleep(Ts)
```

Final Results

SQLQuery4.sql - hph.database.windows.net.LOGGINGSYSTEM (hphlogin (60)) - Microsoft SQL Server Management Studio

Object Explorer

- hph.database.windows.net (SQL Server 12.0.2000.8 - hphlogin)
 - Databases
 - System Databases
 - LOGGINGSYSTEM
 - Database Diagrams
 - Tables
 - System Tables
 - External Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store
 - Extended Events
 - Storage
 - Security

```
select * from MEASUREMENTDATA
```

Results

	MeasurementId	SensorName	MeasurementValue	MeasurementDateTime
1	1	Temperature	22	2021-11-25 14:36:24.000
2	2	Temperature	20	2021-11-25 14:36:34.000
3	3	Temperature	25	2021-11-25 14:36:44.000
4	4	Temperature	21	2021-11-25 14:36:54.000
5	5	Temperature	21	2021-11-25 14:37:04.000
6	6	Temperature	25	2021-11-25 14:37:14.000
7	7	Temperature	24	2021-11-25 14:37:24.000
8	8	Temperature	23	2021-11-25 14:37:34.000
9	9	Temperature	21	2021-11-25 14:37:45.000
10	10	Temperature	25	2021-11-25 14:37:55.000
11	11	Temperature	20	2021-11-25 14:38:05.000
12	12	Temperature	25	2021-11-25 14:38:15.000
13	13	Temperature	21	2021-11-25 14:38:25.000
14	14	Temperature	24	2021-11-25 14:38:35.000
15	15	Temperature	20	2021-11-25 14:38:45.000
16	16	Temperature	21	2021-11-25 14:38:55.000
17	17	Temperature	22	2021-11-25 14:39:05.000
18	18	Temperature	23	2021-11-25 14:39:15.000
19	19	Temperature	20	2021-11-25 14:39:25.000
20	20	Temperature	22	2021-11-25 14:39:35.000

Query executed successfully. | hph.database.windows.net (1... | hphlogin (60) | LOGGINGSYSTEM | 00:00:00 | 20 rows

Ready Ln 1 Col 30 Ch 30 INS

Final Results

Home > SQL databases > LOGGINGSYSTEM (hph/LOGGINGSYSTEM)

SQL databases

Default Directory

+ Create Reservations ...

Filter for any field...

Name ↑

LOGGINGSYSTEM (hph/LOGGINGSYSTE...

LOGGINGSYSTEM (hph/LOGGINGSYSTEM) | Query editor (preview)

Search (Cmd+/)

Login + New Query ↑ Open query Feedback

- Overview
- Activity log
- Tags
- Diagnose and solve problems
- Quick start
- Query editor (preview)

Power Platform

- Power BI
- Power Apps
- Power Automate

Settings

- Compute + storage
- Connection strings
- Properties
- Locks

Data management

- Replicas
- Sync to other databases

Integrations

- Stream analytics (preview)
- Add Azure Search

Security

- Auditing
- Ledger
- Data Discovery & Classification
- Dynamic Data Masking
- Microsoft Defender for Cloud
- Transparent data encryption

Intelligent Performance

- Performance overview

LOGGINGSYSTEM (hphlogin)

Showing limited object explorer here. For full capability please open SSDT.

Tables

dbo.MEASUREMENTDATA

- MeasurementId (PK, int, not null)
- SensorName (varchar, not null)
- MeasurementValue (float, not null)
- MeasurementDateTime (datetime, not null)

Views

Stored Procedures

Query 1

Run Cancel query Save query Export data as Show only Editor

```
1 select * from MEASUREMENTDATA
```

Results Messages

Search to filter items...

MeasurementId	SensorName	MeasurementValue	MeasurementDateTime
1	Temperature	22	2021-11-25T14:36:24.0000000
2	Temperature	20	2021-11-25T14:36:34.0000000
3	Temperature	25	2021-11-25T14:36:44.0000000
4	Temperature	21	2021-11-25T14:36:54.0000000
5	Temperature	21	2021-11-25T14:37:04.0000000
6	Temperature	25	2021-11-25T14:37:14.0000000
7	Temperature	24	2021-11-25T14:37:24.0000000
8	Temperature	23	2021-11-25T14:37:34.0000000
9	Temperature	21	2021-11-25T14:37:45.0000000
10	Temperature	25	2021-11-25T14:37:55.0000000
11	Temperature	20	2021-11-25T14:38:05.0000000
12	Temperature	25	2021-11-25T14:38:15.0000000
13	Temperature	21	2021-11-25T14:38:25.0000000


```
import pyodbc
import matplotlib.pyplot as plt
import database

sensorName = "Temperature"

# Connect to Database
connectionString = database.GetConnectionStringAzure()
conn = pyodbc.connect(connectionString)
cursor = conn.cursor()
query = "SELECT MeasurementValue, MeasurementDateTime FROM MEASUREMENTDATA WHERE SensorName=?"
parameters = [sensorName]

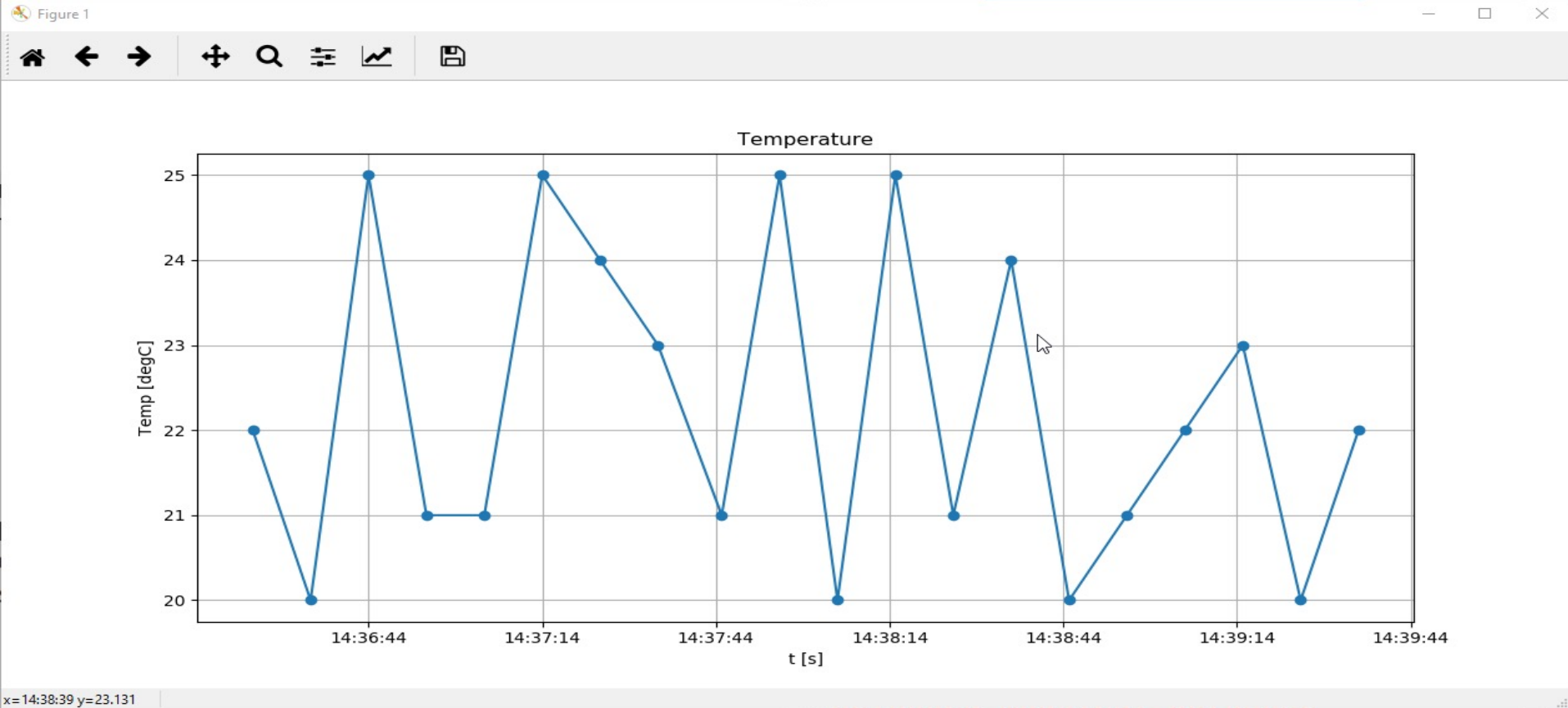
t = []; data = []

# Retrieving and Formatting Data
for row in cursor.execute(query, parameters):
    measurementValue = row.MeasurementValue
    measurementDateTime = row.MeasurementDateTime

    data.append(measurementValue)
    t.append(measurementDateTime)

# Plotting
plt.plot(t, data, 'o-')
plt.title('Temperature')
plt.xlabel('t [s]')
plt.ylabel('Temp [degC]')
plt.grid()
plt.show()
```

Final Results



Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: <https://www.halvorsen.blog>

