

<https://www.halvorsen.blog>



# Python and SQL Server

Hans-Petter Halvorsen

# Free Textbook with lots of Practical Examples

## Python for Software Development

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

<https://www.halvorsen.blog/documents/programming/python/>

# Additional Python Resources

## Python Programming

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Science and Engineering

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Control Engineering

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Software Development

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

<https://www.halvorsen.blog/documents/programming/python/>

# Contents

- Database Systems
- SQL Server
- SQL Server and Python
- CRUD Python Examples
- Datalogging Example

# What is a Database?

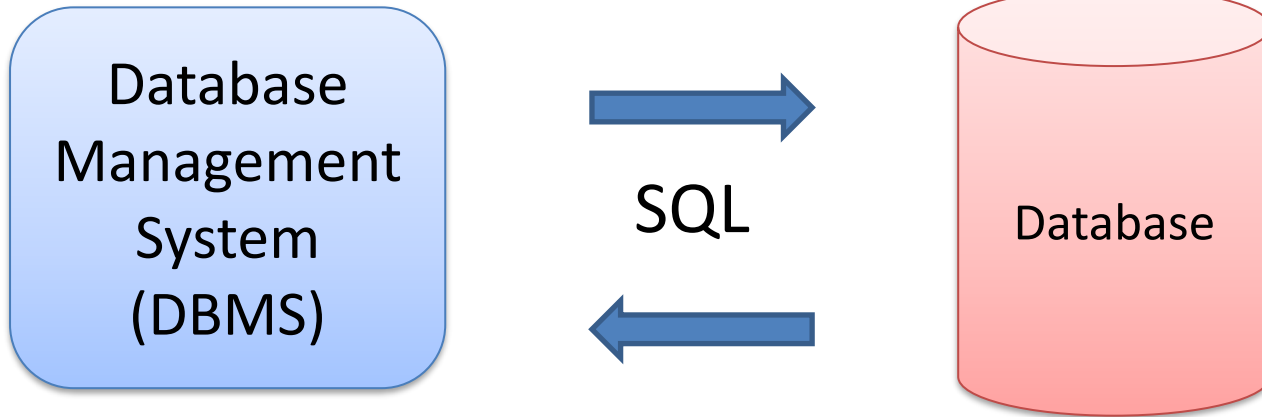
- A Database is a structured way to store lots of information.
- The information inside the database is stored in different tables.
- - “Everything” today is stored in databases!

## Examples:

- Bank/Account systems
- Information in Web pages such as Facebook, Wikipedia, YouTube, etc.
- ... lots of other examples!

# Database Systems

We communicate with the Database using a Database Management System (DBMS). We use the Structured Query Language (SQL) in order to communicate with the Database, i.e., Insert Data, Retrieve Data, Update Data and Delete Data from the Database.



SQL – Structured Query Language

# Database Systems

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

# 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

The screenshot shows the Microsoft SQL Server Management Studio interface. The 'Object Explorer' on the left shows a tree view of a server instance with a database named 'SCHOOL' selected. The 'Query Editor' in the center contains a SQL query: `select * from SCHOOL`. The 'Results' pane at the bottom displays the output of the query as a table with 4 rows. The 'Properties' pane on the right shows connection details for the current session. A status bar at the bottom indicates the query was executed successfully, returning 4 rows.

**1** Your SQL Server

**2** Your Database

**3** New Query

**4** Write your Query here

**5** The result from your Query

	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

# Structured Query Language

- Structured Query Language (SQL) is used to write, read and update data from the Database System
- You can use SQL inside the “SQL Server Management Studio” or inside your Python script.
- SQL Example: `select * from SCHOOL`

# SQL Examples



## Query Examples:

- **insert** into STUDENT (Name , Number, SchoolId)  
values ('John Smith', '100005', 1)
- **select** SchoolId, Name from SCHOOL
- **select** \* from SCHOOL where SchoolId > 100
- **update** STUDENT set Name='John Wayne' **where** StudentId=2
- **delete** from STUDENT **where** SchoolId=3

We have 4 different Query Types: **INSERT**, **SELECT**, **UPDATE** and **DELETE**

**CRUD**: **C** – Create or Insert Data, **R** – Retrieve (Select) Data, **U** – Update Data, **D** – Delete Data

<https://www.halvorsen.blog>



# Python

Hans-Petter Halvorsen

# 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

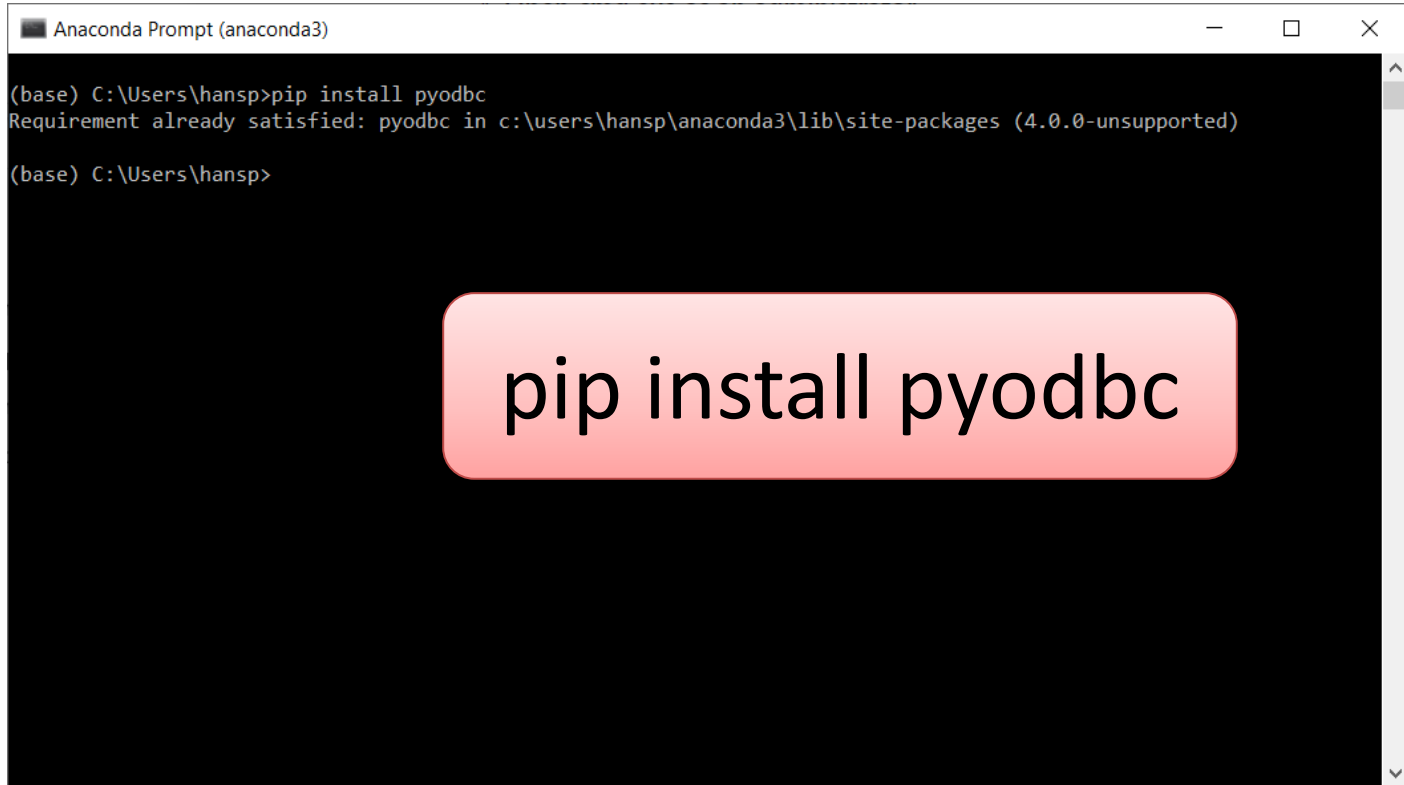
<https://docs.microsoft.com/sql/connect/python/python-driver-for-sql-server>

# 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

<https://www.halvorsen.blog>



# Python Examples

Hans-Petter Halvorsen

# Database CRUD

All Database Systems supports CRUD

**C** – Create or Insert Data

**R** – Retrieve Data

**U** – Update Data

**D** – Delete Data

Let's go through some Python examples

# Python Examples

## Note!

- The examples provided can be considered as a “proof of concept”
- The sample code is simplified for clarity and doesn't necessarily represent best practices.

# SQL Server Database

Let's Create a simple Table called "BOOK":

```
CREATE TABLE [BOOK]
(
    [BookId] [int] IDENTITY(1, 1) NOT NULL PRIMARY KEY,
    [Title] [varchar](50) NOT NULL UNIQUE,
    [Author] [varchar](50) NOT NULL,
    [Category] [varchar](50) NOT NULL
)
GO
```

# SQL Server Database

Let's insert some data into the BOOK Table:

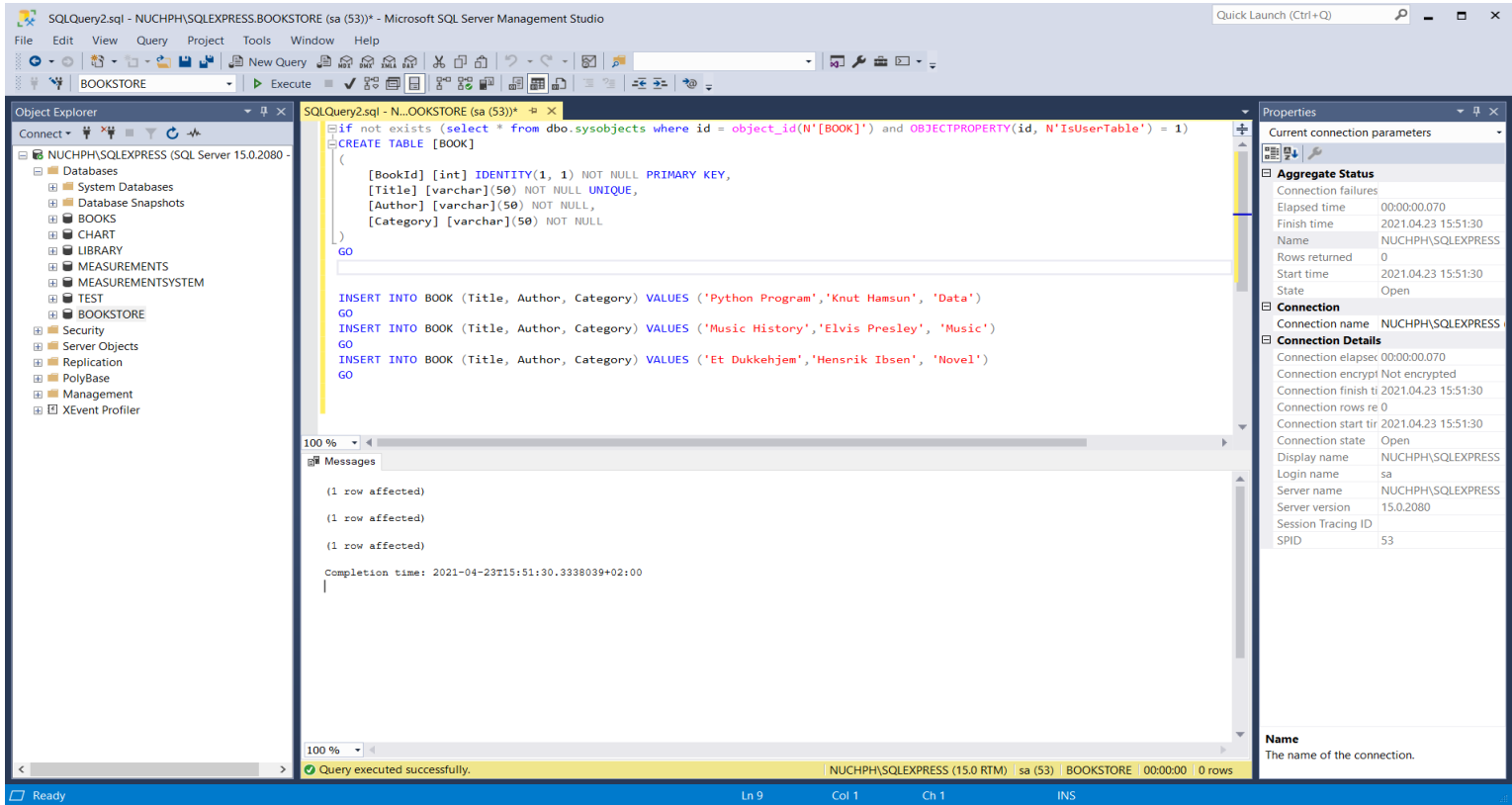
```
INSERT INTO BOOK (Title, Author, Category)
VALUES ('Python Program', 'Knut Hamsun', 'Data')
Go
```

```
INSERT INTO BOOK (Title, Author, Category)
VALUES ('Music History', 'Elvis Presley', 'Music')
GO
```

```
INSERT INTO BOOK (Title, Author, Category)
VALUES ('Et Dukkehjem', 'Henrik Ibsen', 'Novel')
GO
```

# SQL Server Database

We use SQL Server management Studio in order to create the Table and Data:



The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a query window with the following SQL code:

```
if not exists (select * from dbo.sysobjects where id = object_id(N'[BOOK]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
CREATE TABLE [BOOK]
(
    [BookId] [int] IDENTITY(1, 1) NOT NULL PRIMARY KEY,
    [Title] [varchar](50) NOT NULL UNIQUE,
    [Author] [varchar](50) NOT NULL,
    [Category] [varchar](50) NOT NULL
)
GO

INSERT INTO BOOK (Title, Author, Category) VALUES ('Python Program', 'Knut Hamsun', 'Data')
GO
INSERT INTO BOOK (Title, Author, Category) VALUES ('Music History', 'Elvis Presley', 'Music')
GO
INSERT INTO BOOK (Title, Author, Category) VALUES ('Et Dukkehjem', 'Hensrik Ibsen', 'Novel')
GO
```

The Messages pane shows the following output:

```
(1 row affected)
(1 row affected)
(1 row affected)

Completion time: 2021-04-23T15:51:30.3338809+02:00
```

The Properties pane shows the following connection details:

Current connection parameters	
<b>Aggregate Status</b>	
Connection failures	
Elapsed time	00:00:00.070
Finish time	2021.04.23 15:51:30
Name	NUCHPH\SQLEXPRESS
Rows returned	0
Start time	2021.04.23 15:51:30
State	Open
<b>Connection</b>	
Connection name	NUCHPH\SQLEXPRESS
<b>Connection Details</b>	
Connection elapsed	00:00:00.070
Connection encrypt	Not encrypted
Connection finish ti	2021.04.23 15:51:30
Connection rows re	0
Connection start tir	2021.04.23 15:51:30
Connection state	Open
Display name	NUCHPH\SQLEXPRESS
Login name	sa
Server name	NUCHPH\SQLEXPRESS
Server version	15.0.2080
Session Tracing ID	
SPID	53

The status bar at the bottom indicates: Query executed successfully. NUCHPH\SQLEXPRESS (15.0 RTM) sa (53) BOOKSTORE 00:00:00 0 rows

# SQL Server Database

We see that the Table and the Data have been created:

The screenshot displays the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the hierarchy: Databases > BOOKSTORE > Tables > dbo.BOOK. The central query window contains the SQL statement: `select * from BOOK`. The Results pane at the bottom shows the output of the query as a table with 3 rows and 4 columns: BookId, Title, Author, and Category.

BookId	Title	Author	Category
1	Python Program	Knut Hamsun	Data
2	Music History	Elvis Presley	Music
3	Et Dukkehjem	Hensrik Ibsen	Novel

At the bottom of the window, a status bar indicates: "Query executed successfully." and "NUCHPH\SQLEXPRESS (15.0 RTM) | sa (55) | BOOKSTORE | 00:00:00 | 3 rows".



<https://www.halvorsen.blog>




# Connect to Database using Python

Hans-Petter Halvorsen

# 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 new user in SQL Server Management Studio

<https://www.halvorsen.blog>



# Retrieve Data using Python

Hans-Petter Halvorsen

# Python

```
import pyodbc

driver = "{ODBC Driver 17 for SQL Server}"
server = "NUCHPH\\SQLEXPRESS"
database = "BOOKSTORE"
username = "sa"
password = "xxxxxxx"

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

cursor = conn.cursor()

for row in cursor.execute("select BookId, Title, Author, Category from BOOK"):
    print(row.BookId, row.Title, row.Author, row.Category)
```

- 1 Python Program Knut Hamsun Data
- 2 Music History Elvis Presley Music
- 3 Et Dukkehjem Henrik Ibsen Novel

# Python - Alternative

```
import pyodbc

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

cursor = conn.cursor()

cursor.execute("select BookId, Title, Author, Category from BOOK")
row = cursor.fetchone()
while row:
    print(row[0], row[1], row[2], row[3])
    row = cursor.fetchone()
```

- 1 Python Program Knut Hamsun Data
- 2 Music History Elvis Presley Music
- 3 Et Dukkehjem Henrik Ibsen Novel

# “Hide” Connection String

You can put the Connection string in a separate Python File, e.g., “database.py”:

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

```
import pyodbc  
import database  
  
connectionString = database.GetConnectionString()  
  
conn = pyodbc.connect(connectionString)  
  
cursor = conn.cursor()  
  
for row in cursor.execute("select BookId, Title, Author, Category from BOOK"):  
    print(row.BookId, row.Title, row.Author, row.Category)
```

# SELECT ... WHERE ..

Using a SELECT statement with a WHERE clause

Example:

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)

cursor = conn.cursor()

query = "select BookId, Title, Author, Category from BOOK where Category='Data'"

for row in cursor.execute(query):
    print(row.BookId, row.Title, row.Author, row.Category)
```



# Using Parameters- Avoid SQL Injection

- ODBC supports parameters using a question mark as a place holder in the SQL. You provide the values for the question marks by passing them after the SQL
- This is safer than putting the values into the string because the parameters are passed to the database separately, protecting against SQL injection attacks.
- It is also be more efficient if you execute the same SQL repeatedly with different parameters.

# Using Parameters- Avoid SQL Injection

## Example:

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)

cursor = conn.cursor()

query = "select BookId, Title, Author, Category from BOOK where Category=?"

parameters = ['Data']

for row in cursor.execute(query, parameters):
    print(row.BookId, row.Title, row.Author, row.Category)
```

<https://www.halvorsen.blog>



# Insert Data using Python

Hans-Petter Halvorsen

# INSERT

## Basic Example:

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)

cursor = conn.cursor()

query = "INSERT INTO BOOK (Title, Author, Category) VALUES ('Python for Experts', 'Halvorsen', 'Data')"
```

# INSERT with Row Count

With Row Count: You often want to know how many records were inserted. Then you can use the Cursor **rowcount** attribute:

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)

cursor = conn.cursor()

query = "INSERT INTO BOOK (Title, Author, Category) VALUES ('Python for Fun', 'Halvorsen', 'Data')"count = cursor.execute(query).rowcount

cursor.commit()

print('Rows inserted: ' + str(count))
```

# INSERT with Parameters

In this example, you see how to run an INSERT statement safely, and pass parameters. The parameters protect your application from SQL injection.

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)

cursor = conn.cursor()

query = "INSERT INTO BOOK (Title, Author, Category) VALUES (?, ?, ?)"

parameters = 'Python for Beginners', 'Hans-Petter Halvorsen', 'Data'

count = cursor.execute(query, parameters).rowcount
cursor.commit()

print('Rows inserted: ' + str(count))
```

<https://www.halvorsen.blog>



# Modify Data using Python

Hans-Petter Halvorsen

# UPDATE

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)
cursor = conn.cursor()

query = "UPDATE BOOK SET Title='Python Tutorial' WHERE BookId=5"

cursor.execute(query)
cursor.commit()
```



# UPDATE with Row Count

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)
cursor = conn.cursor()

query = "UPDATE BOOK SET Title='Python Tutorial' WHERE BookId=5"

count = cursor.execute(query).rowcount
cursor.commit()

print('Rows updated: ' + str(count))
```

# UPDATE with Parameter

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)
cursor = conn.cursor()

query = "UPDATE BOOK SET Title='Python Tutorial' WHERE BookId=?"
bookId = 5
parameters = bookId

count = cursor.execute(query, parameters).rowcount
cursor.commit()
print('Rows updated: ' + str(count))
```

<https://www.halvorsen.blog>



# Delete Data using Python

Hans-Petter Halvorsen

# DELETE

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)

cursor = conn.cursor()

query = "DELETE FROM BOOK WHERE BookId=10"

cursor.execute(query)
cursor.commit()
```

# DELETE with Row Count

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)
cursor = conn.cursor()

query = "DELETE FROM BOOK WHERE BookId=8"

count = cursor.execute(query).rowcount
cursor.commit()

print('Rows deleted: ' + str(count))
```

# DELETE with Parameter

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)
cursor = conn.cursor()

query = "DELETE FROM BOOK WHERE BookId=?"
parameters = 12

count = cursor.execute(query, parameters).rowcount
cursor.commit()

print('Rows deleted: ' + str(count))
```

<https://www.halvorsen.blog>



# Datalogging Example

Hans-Petter Halvorsen

# 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.



# 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
```

# 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)
```

# Logged Data

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar indicates the current query is 'SQLQuery1.sql' in the 'LOGGINGSYSTEM' database. The Object Explorer on the left shows the database structure, with 'dbo.MEASUREMENTDATA' selected. The query editor shows the SQL statement: `select * from MEASUREMENTDATA`. The Results pane at the bottom displays a table with 20 rows of data.

	MeasurementId	SensorName	MeasurementValue	MeasurementDateTime
1	21	Temperature	21	2021-04-26 14:46:37.000
2	22	Temperature	25	2021-04-26 14:46:47.000
3	23	Temperature	24	2021-04-26 14:46:57.000
4	24	Temperature	22	2021-04-26 14:47:07.000
5	25	Temperature	20	2021-04-26 14:47:17.000
6	26	Temperature	20	2021-04-26 14:47:27.000
7	27	Temperature	22	2021-04-26 14:47:37.000
8	28	Temperature	20	2021-04-26 14:47:47.000
9	29	Temperature	25	2021-04-26 14:47:57.000
10	30	Temperature	23	2021-04-26 14:48:07.000
11	31	Temperature	24	2021-04-26 14:48:17.000
12	32	Temperature	21	2021-04-26 14:48:27.000
13	33	Temperature	24	2021-04-26 14:48:37.000
14	34	Temperature	24	2021-04-26 14:48:47.000
15	35	Temperature	23	2021-04-26 14:48:57.000
16	36	Temperature	22	2021-04-26 14:49:07.000
17	37	Temperature	23	2021-04-26 14:49:17.000
18	38	Temperature	22	2021-04-26 14:49:27.000
19	39	Temperature	23	2021-04-26 14:49:37.000
20	40	Temperature	22	2021-04-26 14:49:47.000

# 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 = []

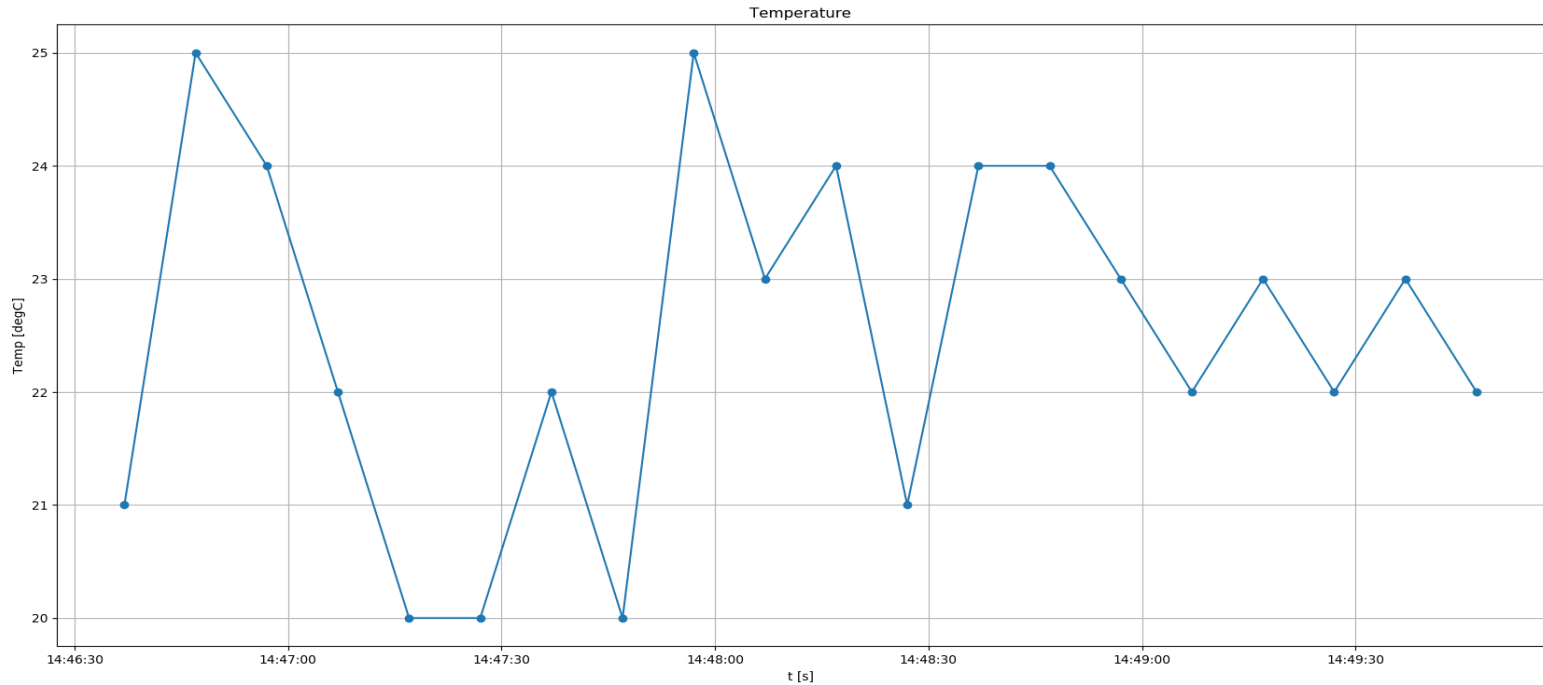
# Retrieving and Formatting Data
for row in cursor.execute(query, parameters):
    measurementValue = row.MeasurementValue
    measurementDateTime = row.MeasurementDateTime

    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

Figure 1



# Additional Python Resources

## Python Programming

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Science and Engineering

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Control Engineering

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

## Python for Software Development

Hans-Petter Halvorsen



<https://www.halvorsen.blog>

<https://www.halvorsen.blog/documents/programming/python/>

# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

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

