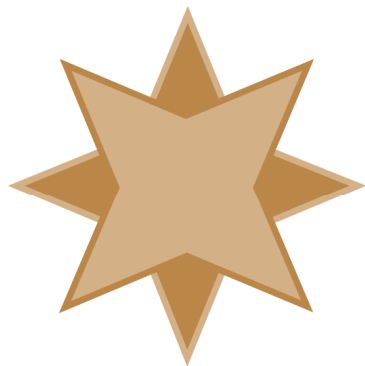


# **Accessing Zen v15 from Python on Windows Using the ODBC Interface**

A White Paper From



**GOLDSTAR  
SOFTWARE**

*www.GoldstarSoftware.com*

For more information, see our web site at  
**<http://www.goldstarsoftware.com>**

# Accessing Zen v15 from Python on Windows Using ODBC

Last Updated: January 2024

The Actian Zen database engine (formerly known as Actian PSQL) supports a wide variety of application programming interfaces (APIs) to access the data. Some of these interfaces leverage the power of SQL to access your data, while others use a lower-level interface, commonly known as the Btrieve API to provide the needed performance and flexibility.

For many applications, SQL/ODBC access is by far the easiest, and the built-in ODBC drivers in Actian Zen and Python make this almost trivial – so much in fact that I almost didn't create this paper in the first place, but here we go anyway.

This paper is broken up into each of the critical steps you need to follow in order to get Python working with Zen v15 via ODBC on Windows. Our example here will use the folder "C:\Develop" as the base folder, and everything else will be installed under that location. Of course, if you use an alternative location, just be sure to change "C:\Develop" to your own root path in all of the sample code.

## Downloading and Installing Actian Zen v15

Accessing the Zen v15 database requires – you guessed it – a Zen v15 database engine. If the Zen engine is already running and you can connect from the Zen Control Center to the database from your development machine, then you are already all set. If the engine is on a different server, though, then you may need to install the Zen v15 Client onto your workstation. If you are working in a disconnected development environment, then you may need to install the Zen v15 Workgroup Engine (which needs to be licensed to use it beyond 30 days). Instructions for all of these downloads can be found on our web site at <http://www.goldstarsoftware.com/ineedzen15.asp>.

## Downloading and Installing Python for Windows

If you don't have Python set up yet, then this becomes the first obvious step.

- 1) In your web browser, go to <https://www.python.org/downloads/windows/> and select the version of Python you want to download. At the time of this writing, the current version is 3.10.7.
- 2) From the second page, click on the link to download the Executable Installer for the platform on which you want to install. For a typical x64 Windows environment, we recommend the link

titled **Windows Installer (64-bit)**.

Files		
Version	Operating System	Description
<a href="#">Gzipped source tarball</a>	Source release	
<a href="#">XZ compressed source tarball</a>	Source release	
<a href="#">macOS 64-bit universal2 installer</a>	macOS	for macOS 10.9 and later
<a href="#">Windows embeddable package (32-bit)</a>	Windows	
<a href="#">Windows embeddable package (64-bit)</a>	Windows	
<a href="#">Windows help file</a>	Windows	
<a href="#">Windows installer (32-bit)</a>	Windows	
<a href="#">Windows installer (64-bit)</a>	Windows	Recommended

- 3) Follow the instructions for the default installation and set the installation folder to C:\Develop. When you are done, you will have Python installed in the C:\Develop\Python310 folder and ready to go.
- 4) Create a folder called **MyPrograms** underneath the Python folder – this will be where we will do all of our development efforts.

---

## Creating your Python Application

Once the environment is ready, you can simply import the pyodbc module into your code:

```
import pyodbc
```

With the module imported, you need to first establish a connection with the **pyodbc.connect()** function. Once you have an active connection, you use that connection to create an ODBC cursor with the **.cursor()** function. You then pass the SQL statement to the cursor with the **.execute()** function.

Once you have the data queried, you can use the **.fetch()** or **.fetchall()** functions to retrieve the data, and then just format the output as needed.

Instead of going into all of the gory details here, just take a look at the sample code in Appendix A, which shows a very simple example of retrieving data from a SQL query that joins the Person and Faculty tables within the DEMODATA database, displaying the top 10 records coming back. If you want to try this sample application, create a new text file called **test\_odbc.py**, then copy and paste in the source code from Appendix A into the file and save it.

## Running Your Python Application

Running the application is even easier. Simply launch Python and pass in the script name!

```
python test_odbc.py
```

In my case, this produced the following data set (your data may vary):

```
C:\Develop\Python310\MyPrograms>python test_odbc.py
Record Returned: ('Andrew', 'Sugar', Decimal('45000.0000'))
Record Returned: ('Robert', 'Reagen', Decimal('80000.0000'))
Record Returned: ('Ismail', 'Badad', Decimal('45000.0000'))
```

Information Provided By **Goldstar Software Inc.**

<http://www.goldstarsoftware.com>

```
Record Returned: ('Mildred', 'Sukara', Decimal('65000.0000'))
Record Returned: ('Elaine', 'Bald', Decimal('65000.0000'))
Record Returned: ('Amy', 'Sylar', Decimal('80000.0000'))
Record Returned: ('Richard', 'Baldwin', Decimal('80000.0000'))
Record Returned: ('Lewis', 'Syra', Decimal('45000.0000'))
Record Returned: ('Randy', 'Beavis', Decimal('125000.0000'))
Record Returned: ('Daniel', 'Tadda', Decimal('65000.0000'))
Closing Cursor...
Closing Connection...
Quitting.
```

From here, you're only limited by your imagination!

## Finding More Help

If you have other problems getting this to work, we urge you to contact Actian directly through their web forums at <https://communities.actian.com/s/> for more help. If you need some additional hand-holding, Goldstar Software may be able to assist you as well. You can contact us at 1-708-647-7665 or via the web at <http://www.goldstarsoftware.com>.

## Appendix A: Sample Application

The following simple application makes use of the ODBC API to read data from a SQL query.

```
import sys
import pyodbc
def main():
    conn_str = 'Driver={Pervasive ODBC Interface};server=localhost;DBQ=demodata;UID=Master;PWD=007;'
    conn = pyodbc.connect(conn_str)
    curs = conn.cursor()
    curs.execute("SELECT top 10 First_Name, Last_Name, Salary FROM Person INNER JOIN Faculty ON (Person.ID =
Faculty.ID) ORDER BY Person.ID")
    rows = curs.fetchall()
    for row in rows:
        print('Record Returned:', row)
    print('Closing Cursor...')
    curs.close
    print('Closing Connection...')
    conn.close
    print('Quitting.')
    return 0
if __name__ == "__main__":
    sys.exit(main())
```