https://python-oracledb.readthedocs.io/en/latest/user\_guide/installation.html

**2. Installing python-oracledb**

The python-oracledb driver allows Python 3 applications to connect to Oracle Database.

Python-oracledb is the new name for the Python [cx\_Oracle driver](https://oracle.github.io/python-cx_Oracle/). If you are upgrading from cx\_Oracle, see [Upgrading from cx\_Oracle 8.3 to python-oracledb](https://python-oracledb.readthedocs.io/en/latest/user_guide/appendix_c.html#upgrading83).

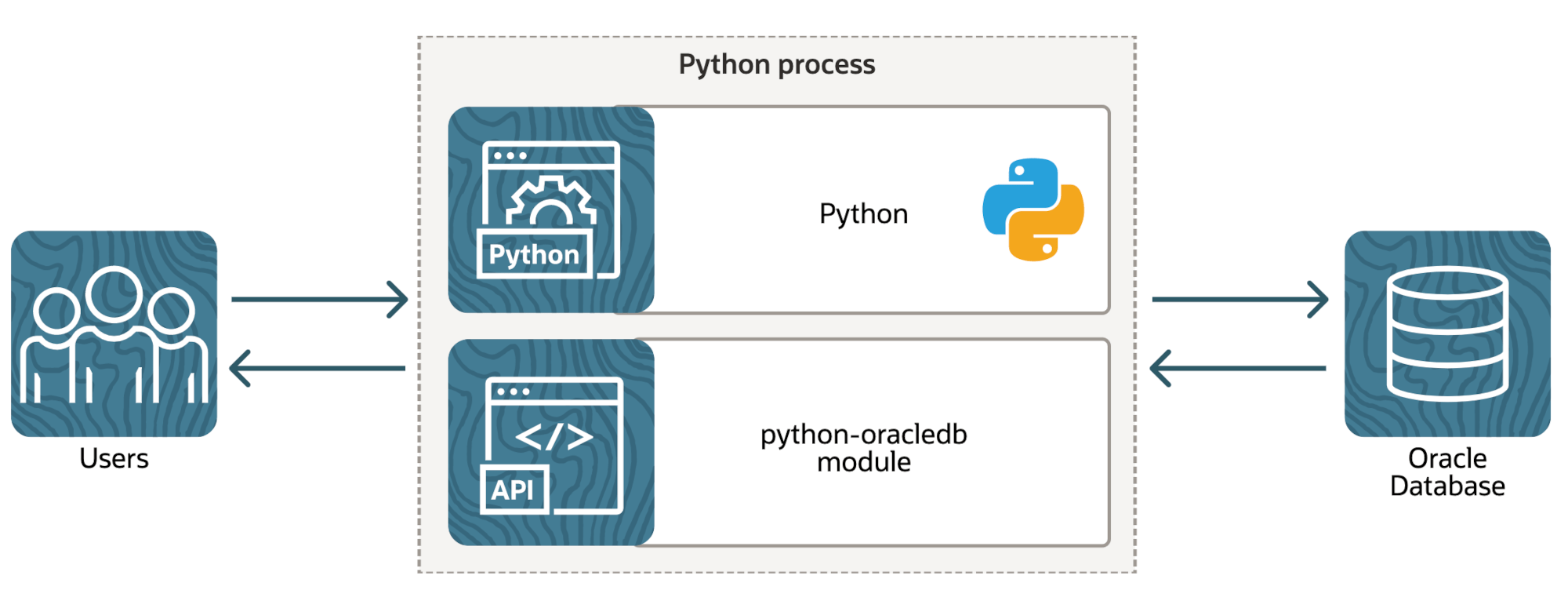


Fig. 2.1 Architecture of the python-oracledb driver[¶](https://python-oracledb.readthedocs.io/en/latest/user_guide/installation.html#id9)

By default, python-oracledb runs in a ‘Thin’ mode which connects directly to Oracle Database. This mode does not need Oracle Client libraries. However, some [additional functionality](https://python-oracledb.readthedocs.io/en/latest/user_guide/appendix_a.html#featuresummary) is available when python-oracledb uses them. Python-oracledb is said to be in ‘Thick’ mode when Oracle Client libraries are used. See [Enabling python-oracledb Thick mode](https://python-oracledb.readthedocs.io/en/latest/user_guide/initialization.html#enablingthick). Both modes have comprehensive functionality supporting the Python Database API v2.0 Specification.

**2.1. Quick Start python-oracledb Installation**

This section contains the steps that you need to perform to install python-oracledb quickly.

1. Install [Python 3](https://www.python.org/downloads), if it is not already available. The version of Python to be used depends on the operating system (OS):

* On Windows, use Python 3.7 to 3.11  
  On macOS, use Python 3.7 to 3.11  
  On Linux, use Python 3.6 to 3.11

By default, python-oracledb connects directly to Oracle Database. This lets it be used when Oracle Client libraries are not available (such Apple M1 or Alpine Linux), or where the client libraries are not easily installable (such as some cloud environments). Note not all environments are tested.

1. Install python-oracledb from [PyPI](https://pypi.org/project/oracledb/):

python -m pip install oracledb --upgrade

If a binary package is not available for your platform, the source package will be downloaded instead. This will be compiled and the resulting binary installed.

The --user option may be useful if you do not have permission to write to system directories:

python -m pip install oracledb --upgrade --user

If you are behind a proxy, add a proxy server to the command, for example add --proxy=http://proxy.example.com:80

1. Create a file test.py such as:

*# test.py*

**import** **oracledb**

**import** **os**

un = os.environ.get('PYTHON\_USERNAME')

pw = os.environ.get('PYTHON\_PASSWORD')

cs = os.environ.get('PYTHON\_CONNECTSTRING')

**with** oracledb.connect(user=un, password=pw, dsn=cs) **as** connection:

**with** connection.cursor() **as** cursor:

sql = """select sysdate from dual"""

**for** r **in** cursor.execute(sql):

print(r)

1. In your integrated development environment (IDE) or terminal window, set the three environment variables used by the test program.

A simple [connection](https://python-oracledb.readthedocs.io/en/latest/user_guide/connection_handling.html#connhandling) to the database requires an Oracle Database [user name and password](https://www.youtube.com/watch?v=WDJacg0NuLo) and a database [connection string](https://python-oracledb.readthedocs.io/en/latest/user_guide/connection_handling.html#connstr). Set the environment variables to your values. For python-oracledb, the connection string is commonly of the format hostname/servicename, using the host name where the database is running and the Oracle Database service name of the database instance. The database can be on-premises or in the Cloud. It should be version 12.1 or later. The python-oracledb driver does not include a database.

1. Run the program as shown below:

python test.py

The date will be shown.

You can learn more about python-oracledb from the [python-oracledb documentation](https://python-oracledb.readthedocs.io/en/latest/index.html) and [samples](https://github.com/oracle/python-oracledb/tree/main/samples).

If you run into installation trouble, see [Troubleshooting](https://python-oracledb.readthedocs.io/en/latest/user_guide/installation.html#troubleshooting).

**2.2. Supported Oracle Database Versions**

When python-oracledb is used in the default Thin mode, it connects directly to the Oracle Database and does not require Oracle Client libraries. Connections in this mode can be made to Oracle Database 12.1 or later.

To use the [Thick mode features](https://python-oracledb.readthedocs.io/en/latest/user_guide/appendix_a.html#featuresummary) of python-oracledb, additional Oracle Client libraries must be installed, as detailed in the subsequent sections. Connections in this mode can be made to Oracle Database 9.2, or later, depending on the Oracle Client library version.

Oracle’s standard client-server network interoperability allows connections between different versions of Oracle Client libraries and Oracle Database. For currently certified configurations, see Oracle Support’s [Doc ID 207303.1](https://support.oracle.com/epmos/faces/DocumentDisplay?id=207303.1). In summary:

Oracle Client 21 can connect to Oracle Database 12.1 or later  
Oracle Client 19, 18 and 12.2 can connect to Oracle Database 11.2 or later  
Oracle Client 12.1 can connect to Oracle Database 10.2 or later  
Oracle Client 11.2 can connect to Oracle Database 9.2 or later

The technical restrictions on creating connections may be more flexible. For example, Oracle Client 12.2 can successfully connect to Oracle Database 10.2.

The python-oracledb attribute [**Connection.thin**](https://python-oracledb.readthedocs.io/en/latest/api_manual/connection.html#Connection.thin) can be used to see what mode a connection is in. In the Thick mode, the function [**oracledb.clientversion()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.clientversion) can be used to determine which Oracle Client version is in use. The attribute [**Connection.version**](https://python-oracledb.readthedocs.io/en/latest/api_manual/connection.html#Connection.version) can be used to determine which Oracle Database version a connection is accessing. These can then be used to adjust the application behavior accordingly. Any attempt to use Oracle features that are not supported by a particular mode or client library/database combination will result in runtime errors.

**2.3. Installation Requirements**

To use python-oracledb, you need:

* Python 3.6, 3.7, 3.8, 3.9, 3.10 or 3.11 depending on the operating system:
  + Windows: Use Python 3.7 to 3.11
  + macOS: Use Python 3.7 to 3.11
  + Linux: Use Python 3.6 to 3.11
* The Python cryptography package. This package is automatically installed as a dependency of python-oracledb. It is strongly recommended that you keep the cryptography package up to date whenever new versions are released. If the cryptography package is not available, you can still install python-oracledb but can only use it in Thick mode, see [Installing python-oracledb without the Cryptography Package](https://python-oracledb.readthedocs.io/en/latest/user_guide/installation.html#nocrypto).
* Optionally, Oracle Client libraries can be installed to enable some additional advanced functionality. These can be from the free [Oracle Instant Client](https://www.oracle.com/database/technologies/instant-client.html), from a full Oracle Client installation (such as installed by Oracle’s GUI installer), or from those included in Oracle Database if Python is on the same machine as the database. Oracle Client libraries versions 21, 19, 18, 12, and 11.2 are supported where available on Linux, Windows and macOS (Intel x86). Oracle’s standard client-server version interoperability allows connection to both older and newer databases.
* An Oracle Database either local or remote, on-premises or in the Cloud.

**2.4. Installing python-oracledb on Linux**

This section discusses the generic installation methods on Linux.

**2.4.1. Install python-oracledb**

The generic way to install python-oracledb on Linux is to use Python’s [pip](https://pip.pypa.io/en/latest/) package to install from Python’s package repository [PyPI](https://pypi.org/project/oracledb/):

python -m pip install oracledb

This will download and install a pre-compiled binary from [PyPI](https://pypi.org/project/oracledb/) if one is available for your architecture. Otherwise, the source will be downloaded, compiled, and the resulting binary installed. Compiling python-oracledb requires the Python.h header file. If you are using the default python package, this file is in the python-devel package or equivalent.

On Oracle Linux 8, to use the default Python 3.6 installation, install with:

python3 -m pip install oracledb --user

The --user option is useful when you do not have permission to write to system directories.

Other versions of Python can be used on Oracle Linux, see [Python for Oracle Linux](https://yum.oracle.com/oracle-linux-python.html).

If you are behind a proxy, add a proxy server to the command, for example add --proxy=http://proxy.example.com:80

**2.4.2. Optionally Install Oracle Client**

By default, python-oracledb runs in a Thin mode which connects directly to Oracle Database so no further installation steps are required. However, to use additional features available in [Thick mode](https://python-oracledb.readthedocs.io/en/latest/user_guide/appendix_a.html#featuresummary) you need Oracle Client libraries installed. Oracle Client versions 21, 19, 18, 12 and 11.2 are supported.

* If your database is on a remote computer, then download the free [Oracle Instant Client](https://www.oracle.com/database/technologies/instant-client.html) “Basic” or “Basic Light” package for your operating system architecture.
* Alternatively, use the client libraries already available in a locally installed database such as the free [Oracle Database Express Edition (“XE”)](https://www.oracle.com/database/technologies/appdev/xe.html) release.

To use python-oracledb in Thick mode you must call [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) in your application, see [Enabling python-oracledb Thick mode](https://python-oracledb.readthedocs.io/en/latest/user_guide/initialization.html#enablingthick). For example:

**import** **oracledb**

oracledb.init\_oracle\_client()

On Linux, do not pass the lib\_dir parameter in the call: the Oracle Client libraries on Linux must be in the system library search path *before* the Python process starts.

**2.4.2.1. Oracle Instant Client Zip Files**

To use python-oracledb Thick mode with Oracle Instant Client zip files:

1. Download an Oracle 21, 19, 18, 12, or 11.2 “Basic” or “Basic Light” zip file matching your Python 64-bit or 32-bit architecture:

* [x86-64 64-bit](https://www.oracle.com/database/technologies/instant-client/linux-x86-64-downloads.html)
* [x86 32-bit](https://www.oracle.com/database/technologies/instant-client/linux-x86-32-downloads.html)
* [ARM (aarch64) 64-bit](https://www.oracle.com/database/technologies/instant-client/linux-arm-aarch64-downloads.html)

The latest version is recommended. Oracle Instant Client 21 will connect to Oracle Database 12.1 or later.

1. Unzip the package into a single directory that is accessible to your application. For example:
2. mkdir -p /opt/oracle
3. cd /opt/oracle
4. unzip instantclient-basic-linux.x64-21.6.0.0.0.zip

Note OS restrictions may prevent the opening of Oracle Client libraries installed in unsafe paths, such as from a user directory. You may need to install under a directory like /opt or /usr/local.

1. Install the libaio package with sudo or as the root user. For example:
2. sudo yum install libaio

On some Linux distributions this package is called libaio1 instead.

On recent Linux versions such as Oracle Linux 8, you may also need to install the libnsl package when using Oracle Instant Client 19.

1. If there is no other Oracle software on the machine that will be impacted, permanently add Instant Client to the runtime link path. For example, with sudo or as the root user:
2. sudo sh -c "echo /opt/oracle/instantclient\_21\_6 > /etc/ld.so.conf.d/oracle-instantclient.conf"
3. sudo ldconfig

Alternatively, set the environment variable LD\_LIBRARY\_PATH to the appropriate directory for the Instant Client version. For example:

export LD\_LIBRARY\_PATH=/opt/oracle/instantclient\_21\_6:$LD\_LIBRARY\_PATH

1. If you use optional Oracle configuration files such as tnsnames.ora, sqlnet.ora, or oraaccess.xml with Instant Client, then put the files in an accessible directory, for example in /opt/oracle/your\_config\_dir. Then use:
2. **import** **oracledb**
3. oracledb.init\_oracle\_client(config\_dir="/home/your\_username/oracle/your\_config\_dir")

or set the environment variable TNS\_ADMIN to that directory name.

Alternatively, put the files in the network/admin subdirectory of Instant Client, for example in /opt/oracle/instantclient\_21\_6/network/admin. This is the default Oracle configuration directory for executables linked with this Instant Client.

1. Call [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) in your application, if it is not already used.

**2.4.2.2. Oracle Instant Client RPMs**

To use python-oracledb with Oracle Instant Client RPMs:

1. Download an Oracle 21, 19, 18, 12, or 11.2 “Basic” or “Basic Light” RPM matching your Python architecture:

* [x86-64 64-bit](https://www.oracle.com/database/technologies/instant-client/linux-x86-64-downloads.html)
* [x86 32-bit](https://www.oracle.com/database/technologies/instant-client/linux-x86-32-downloads.html)
* [ARM (aarch64) 64-bit](https://www.oracle.com/database/technologies/instant-client/linux-arm-aarch64-downloads.html)

Oracle’s yum server has convenient repositories:

* [Instant Client 21 RPMs for Oracle Linux x86-64 8](https://yum.oracle.com/repo/OracleLinux/OL8/oracle/instantclient21/x86_64/index.html), [Older Instant Client RPMs for Oracle Linux x86-64 8](https://yum.oracle.com/repo/OracleLinux/OL8/oracle/instantclient/x86_64/index.html)
* [Instant Client 21 RPMs for Oracle Linux x86-64 7](https://yum.oracle.com/repo/OracleLinux/OL7/oracle/instantclient21/x86_64/index.html), [Older Instant Client RPMs for Oracle Linux x86-64 7](https://yum.oracle.com/repo/OracleLinux/OL7/oracle/instantclient/x86_64/index.html)
* [Instant Client RPMs for Oracle Linux x86-64 6](https://yum.oracle.com/repo/OracleLinux/OL6/oracle/instantclient/x86_64/index.html)
* [Instant Client RPMs for Oracle Linux ARM (aarch64) 8](https://yum.oracle.com/repo/OracleLinux/OL8/oracle/instantclient/aarch64/index.html)
* [Instant Client RPMs for Oracle Linux ARM (aarch64) 7](https://yum.oracle.com/repo/OracleLinux/OL7/oracle/instantclient/aarch64/index.html)

The latest version is recommended. Oracle Instant Client 21 will connect to Oracle Database 12.1 or later.

1. Install the downloaded RPM with sudo or as the root user. For example:
2. sudo yum install oracle-instantclient-basic-21.6.0.0.0-1.x86\_64.rpm

Yum will automatically install required dependencies, such as libaio.

On recent Linux versions such as Oracle Linux 8, you may need to manually install the libnsl package when using Oracle Instant Client 19.

1. For Instant Client 19 or later, the system library search path is automatically configured during installation.

For older versions, if there is no other Oracle software on the machine that will be impacted, permanently add Instant Client to the runtime link path. For example, with sudo or as the root user:

sudo sh -c "echo /usr/lib/oracle/18.5/client64/lib > /etc/ld.so.conf.d/oracle-instantclient.conf"

sudo ldconfig

Alternatively, for version 18 and earlier, every shell running Python will need to have the environment variable LD\_LIBRARY\_PATH set to the appropriate directory for the Instant Client version. For example:

export LD\_LIBRARY\_PATH=/usr/lib/oracle/18.5/client64/lib:$LD\_LIBRARY\_PATH

1. If you use optional Oracle configuration files such as tnsnames.ora, sqlnet.ora or oraaccess.xml with Instant Client, then put the files in an accessible directory, for example in /opt/oracle/your\_config\_dir. Then use:
2. **import** **oracledb**
3. oracledb.init\_oracle\_client(config\_dir="/opt/oracle/your\_config\_dir")

or set the environment variable TNS\_ADMIN to that directory name.

Alternatively, put the files in the network/admin subdirectory of Instant Client, for example in /usr/lib/oracle/21/client64/lib/network/admin. This is the default Oracle configuration directory for executables linked with this Instant Client.

1. Call [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) in your application, if it is not already used.

**2.4.2.3. Local Database or Full Oracle Client**

Python-oracledb applications can use Oracle Client 21, 19, 18, 12, or 11.2 libraries from a local Oracle Database or full Oracle Client installation (such as installed by Oracle’s GUI installer).

The libraries must be either 32-bit or 64-bit, matching your Python architecture.

1. Set required Oracle environment variables by running the Oracle environment script. For example:
2. source /usr/local/bin/oraenv

For Oracle Database Express Edition (“XE”) 11.2, run:

source /u01/app/oracle/product/11.2.0/xe/bin/oracle\_env.sh

1. Optional Oracle configuration files such as tnsnames.ora, sqlnet.ora, or oraaccess.xml can be placed in $ORACLE\_HOME/network/admin.

Alternatively, Oracle configuration files can be put in another, accessible directory. Then set the environment variable TNS\_ADMIN to that directory name.

1. Call [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) in your application, if it is not already used.

**2.5. Installing python-oracledb on Windows**

**2.5.1. Install python-oracledb**

Use Python’s [pip](https://pip.pypa.io/en/latest/installation/) package to install python-oracledb from Python’s package repository [PyPI](https://pypi.org/project/oracledb/):

python -m pip install oracledb

If you are behind a proxy, add a proxy server to the command, for example add --proxy=http://proxy.example.com:80

python -m pip install oracledb --proxy=http://proxy.example.com:80 --upgrade

This will download and install a pre-compiled binary [if one is available](https://pypi.org/project/oracledb/) for your architecture. If a pre-compiled binary is not available, the source will be downloaded, compiled, and the resulting binary installed.

**2.5.2. Optionally Install Oracle Client**

By default, python-oracledb runs in a Thin mode which connects directly to Oracle Database so no further installation steps are required. However, to use additional features available in [Thick mode](https://python-oracledb.readthedocs.io/en/latest/user_guide/appendix_a.html#featuresummary) you need Oracle Client libraries installed. Oracle Client versions 21, 19, 18, 12, and 11.2 are supported.

* If your database is on a remote computer, then download the free [Oracle Instant Client](https://www.oracle.com/database/technologies/instant-client.html) “Basic” or “Basic Light” package for your operating system architecture.
* Alternatively, use the client libraries already available in a locally installed database such as the free [Oracle Database Express Edition (“XE”)](https://www.oracle.com/database/technologies/appdev/xe.html) release.

To use python-oracledb in Thick mode you must call [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) in your application, see [Enabling python-oracledb Thick mode](https://python-oracledb.readthedocs.io/en/latest/user_guide/initialization.html#enablingthick). For example:

**import** **oracledb**

oracledb.init\_oracle\_client()

On Windows, you may prefer to pass the lib\_dir parameter in the call as shown below.

**2.5.2.1. Oracle Instant Client Zip Files**

To use python-oracledb in Thick mode with Oracle Instant Client zip files:

1. Download an Oracle 21, 19, 18, 12, or 11.2 “Basic” or “Basic Light” zip file: [64-bit](https://www.oracle.com/database/technologies/instant-client/winx64-64-downloads.html) or [32-bit](https://www.oracle.com/database/technologies/instant-client/microsoft-windows-32-downloads.html), matching your Python architecture.

The latest version is recommended. Oracle Instant Client 19 will connect to Oracle Database 11.2 or later.

Windows 7 users: Note that Oracle 19c is not supported on Windows 7.

1. Unzip the package into a directory that is accessible to your application. For example unzip instantclient-basic-windows.x64-19.11.0.0.0dbru.zip to C:\oracle\instantclient\_19\_11.
2. Oracle Instant Client libraries require a Visual Studio redistributable with a 64-bit or 32-bit architecture to match Instant Client’s architecture. Each Instant Client version requires a different redistributable version:

* For Instant Client 21, install [VS 2019](https://docs.microsoft.com/en-US/cpp/windows/latest-supported-vc-redist?view=msvc-170) or later
* For Instant Client 19, install [VS 2017](https://docs.microsoft.com/en-US/cpp/windows/latest-supported-vc-redist?view=msvc-170)
* For Instant Client 18 or 12.2, install [VS 2013](https://docs.microsoft.com/en-US/cpp/windows/latest-supported-vc-redist?view=msvc-170#visual-studio-2013-vc-120)
* For Instant Client 12.1, install [VS 2010](https://docs.microsoft.com/en-US/cpp/windows/latest-supported-vc-redist?view=msvc-170#visual-studio-2010-vc-100-sp1-no-longer-supported)
* For Instant Client 11.2, install [VS 2005 64-bit](https://docs.microsoft.com/en-US/cpp/windows/latest-supported-vc-redist?view=msvc-170#visual-studio-2005-vc-80-sp1-no-longer-supported)

**2.5.2.1.1. Configure Oracle Instant Client**

1. There are several alternative ways to tell python-oracledb where your Oracle Client libraries are, see [Initializing python-oracledb](https://python-oracledb.readthedocs.io/en/latest/user_guide/initialization.html#initialization).

* With Oracle Instant Client you can use [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) in your application, for example:
* **import** **oracledb**
* oracledb.init\_oracle\_client(lib\_dir=r"C:\oracle\instantclient\_19\_14")

Note that a ‘raw’ string is used because backslashes occur in the path.

* Alternatively, add the Oracle Instant Client directory to the PATH environment variable. The directory must occur in PATH before any other Oracle directories. Restart any open command prompt windows.

Update your application to call init\_oracle\_client(), which enables python-oracledb Thick mode:

**import** **oracledb**

oracledb.init\_oracle\_client()

* Another way to set PATH is to use a batch file that sets it before Python is executed, for example:
* REM mypy.bat
* SET PATH=C:\oracle\instantclient\_19\_14;%PATH%
* python %\*

Invoke this batch file every time you want to run Python.

Update your application to call init\_oracle\_client(), which enables python-oracledb Thick mode:

**import** **oracledb**

oracledb.init\_oracle\_client()

1. If you use optional Oracle configuration files such as tnsnames.ora, sqlnet.ora, or oraaccess.xml with Instant Client, then put the files in an accessible directory, for example in C:\oracle\your\_config\_dir. Then use:
2. **import** **oracledb**
3. oracledb.init\_oracle\_client(lib\_dir=r"C:\oracle\instantclient\_19\_14",
4. config\_dir=r"C:\oracle\your\_config\_dir")

or set the environment variable TNS\_ADMIN to that directory name.

Alternatively, put the files in a network\admin subdirectory of Instant Client, for example in C:\oracle\instantclient\_19\_11\network\admin. This is the default Oracle configuration directory for executables linked with this Instant Client.

**2.5.2.2. Local Database or Full Oracle Client**

Python-oracledb Thick mode applications can use Oracle Client 21, 19, 18, 12, or 11.2 libraries from a local Oracle Database or full Oracle Client (such as installed by Oracle’s GUI installer).

The Oracle libraries must be either 32-bit or 64-bit, matching your Python architecture.

1. Set the environment variable PATH to include the path that contains OCI.DLL, if it is not already set.

Restart any open command prompt windows.

1. Optional Oracle configuration files such as tnsnames.ora, sqlnet.ora, or oraaccess.xml can be placed in the network\admin subdirectory of the Oracle Database software installation.

Alternatively, pass config\_dir to [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) as shown in the previous section, or set TNS\_ADMIN to the directory name.

1. To use python-oracledb in Thick mode you must call [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) in your application, see [Enabling python-oracledb Thick mode](https://python-oracledb.readthedocs.io/en/latest/user_guide/initialization.html#enablingthick).
2. **import** **oracledb**
3. oracledb.init\_oracle\_client()

**2.6. Installing python-oracledb on macOS**

Python-oracledb is available as a Universal binary for Python 3.8, or later, on Apple Intel and M1 architectures. A binary is also available for Python 3.7 on Apple Intel.

**2.6.1. Install python-oracledb**

Use Python’s [pip](https://pip.pypa.io/en/latest/installation/) package to install python-oracledb from Python’s package repository [PyPI](https://pypi.org/project/oracledb/):

python -m pip install oracledb

The --user option may be useful if you do not have permission to write to system directories:

python -m pip install oracledb --user

To install into the system Python, you may need to use /usr/bin/python3 instead of python:

/usr/bin/python3 -m pip install oracledb --user

If you are behind a proxy, add a proxy server to the command, for example add --proxy=http://proxy.example.com:80

The source will be downloaded, compiled, and the resulting binary installed.

**2.6.2. Optionally Install Oracle Client**

By default, python-oracledb runs in a Thin mode which connects directly to Oracle Database so no further installation steps are required. However, to use additional features available in [Thick mode](https://python-oracledb.readthedocs.io/en/latest/user_guide/appendix_a.html#featuresummary) you need Oracle Client libraries installed. Note that to use Thick mode on the M1 architecture you will need to use Rosetta with Python 64-bit Intel and the Instant Client (Intel x86) libraries.

**2.6.2.1. Manual Installation**

* Download the **Basic** 64-bit DMG from [Oracle](https://www.oracle.com/database/technologies/instant-client/macos-intel-x86-downloads.html).
* In Finder, double-click DMG to mount it.
* Open a terminal window and run the install script in the mounted package, for example:
* /Volumes/instantclient-basic-macos.x64-19.8.0.0.0dbru/install\_ic.sh

This copies the contents to $HOME/Downloads/instantclient\_19\_8. Applications may not have access to the Downloads directory, so you should move Instant Client somewhere convenient.

* In Finder, eject the mounted Instant Client package.

If you have multiple Instant Client DMG packages mounted, you only need to run install\_ic.sh once. It will copy all mounted Instant Client DMG packages at the same time.

**2.6.2.2. Scripted Installation**

Instant Client installation can alternatively be scripted, for example:

cd $HOME/Downloads

curl -O https://download.oracle.com/otn\_software/mac/instantclient/198000/instantclient-basic-macos.x64-19.8.0.0.0dbru.dmg

hdiutil mount instantclient-basic-macos.x64-19.8.0.0.0dbru.dmg

/Volumes/instantclient-basic-macos.x64-19.8.0.0.0dbru/install\_ic.sh

hdiutil unmount /Volumes/instantclient-basic-macos.x64-19.8.0.0.0dbru

The Instant Client directory will be $HOME/Downloads/instantclient\_19\_8. Applications may not have access to the Downloads directory, so you should move Instant Client somewhere convenient.

**2.6.3. Configure Oracle Instant Client**

1. Call [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) in your application:
2. **import** **oracledb**
3. oracledb.init\_oracle\_client(lib\_dir="/Users/your\_username/Downloads/instantclient\_19\_8")
4. If you use optional Oracle configuration files such as tnsnames.ora, sqlnet.ora, or oraaccess.xml with Oracle Instant Client, then put the files in an accessible directory, for example in /Users/your\_username/oracle/your\_config\_dir. Then use:
5. **import** **oracledb**
6. oracledb.init\_oracle\_client(lib\_dir="/Users/your\_username/Downloads/instantclient\_19\_8",
7. config\_dir="/Users/your\_username/oracle/your\_config\_dir")

Or set the environment variable TNS\_ADMIN to that directory name.

Alternatively, put the files in the network/admin subdirectory of Oracle Instant Client, for example in /Users/your\_username/Downloads/instantclient\_19\_8/network/admin. This is the default Oracle configuration directory for executables linked with this Instant Client.

**2.7. Installing python-oracledb without Internet Access**

To install python-oracledb on a computer that is not connected to the internet, download the appropriate python-oracledb file from Python’s package repository [PyPI](https://pypi.org/project/oracledb/#files). Transfer this file to the offline computer and install it with:

python -m pip install "<file\_name>"

Then follow the general python-oracledb platform installation instructions to install Oracle client libraries.

**2.8. Installing python-oracledb without the Cryptography Package**

If the Python cryptography package is not available, python-oracledb can still be installed but can only be used in Thick mode.

To install without the cryptography package, use pip’s --no-deps option, for example:

python -m pip install oracledb --no-deps

Oracle Client libraries must then be installed. See previous sections.

To use python-oracledb in Thick mode you must call [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) in your application, see [Enabling python-oracledb Thick mode](https://python-oracledb.readthedocs.io/en/latest/user_guide/initialization.html#enablingthick). Without this, your application will get the error DPY-3016: python-oracledb thin mode cannot be used because the cryptography package is not installed.

**2.9. Installing from Source Code**

The following dependencies are required to build python-oracledb from source code:

* Cython Package: Cython is a standard Python package from PyPI.
* The Python cryptography package. This will need to be installed manually before building python-oracledb. For example install with pip.
* C Compiler: A C99 compiler is needed.

**2.9.1. Install Using GitHub**

In order to install using the source on GitHub, use the following commands:

git clone --recurse-submodules https://github.com/oracle/python-oracledb.git

cd python-oracledb

python setup.py build

python setup.py install

Note that if you download a source zip file directly from GitHub then you will also need to download an [ODPI-C](https://github.com/oracle/odpi) source zip file and put the extracted contents inside the “odpi” subdirectory, for example in “python-oracledb-main/src/oracledb/impl/thick/odpi”.

Python-oracledb source code is also available from opensource.oracle.com. This can be installed with:

git clone --recurse-submodules https://opensource.oracle.com/git/oracle/python-oracledb.git

cd python-oracledb

python setup.py build

python setup.py install

If you do not have access to system directories, the --user option can be used to install into a local directory:

python setup.py install --user

**2.9.2. Install Using Source from PyPI**

The source package can be downloaded manually from [PyPI](https://pypi.org/project/oracledb/#files) and extracted, after which the following commands should be run:

python setup.py build

python setup.py install

If you do not have access to system directories, the --user option can be used to install into a local directory:

python setup.py install --user

**2.10. Troubleshooting**

**2.10.1. Installation Troubleshooting**

If installation fails:

* An error such as not a supported wheel on this platform. indicates that you may be using an older *pip* version. Upgrade it with the following command:
* python -m pip install pip --upgrade --user
* Use option -v with pip. Review your output and logs. Try to install using a different method. **Google anything that looks like an error.** Try some potential solutions.
* If there was a network connection error, check if you need to set the environment variables http\_proxy and/or https\_proxy or try python -m pip install --proxy=http://proxy.example.com:80 oracledb --upgrade.
* If the upgrade did not give any errors but the old version is still installed, try python -m pip install oracledb --upgrade --force-reinstall.
* If you do not have access to modify your system version of Python, then use python -m pip install oracledb --upgrade --user or venv.
* If you get the error No module named pip, it means that the pip module that is built into Python may sometimes be removed by the OS. Use the venv module (built into Python 3.x) or virtualenv module instead.
* If you get the error fatal error: dpi.h: No such file or directory when building from source code, then ensure that your source installation has a subdirectory called “odpi” containing files. If this is missing, review the section on [Install Using GitHub](https://python-oracledb.readthedocs.io/en/latest/user_guide/installation.html#install-using-github).

**2.10.2. Runtime Error Troubleshooting**

If using python-oracledb fails:

* If you have multiple versions of Python installed, ensure that you are using the correct python and pip (or python3 and pip3) executables.
* If you get the error DPI-1047: Oracle Client library cannot be loaded:
  + Review the [features available in python-oracledb’s default Thin mode](https://python-oracledb.readthedocs.io/en/latest/user_guide/appendix_a.html#featuresummary). If Thin mode suits your requirements, then remove calls in your application to [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) since this loads the Oracle Client library to enable Thick mode.
  + On Windows and macOS, pass the lib\_dir library directory parameter in your [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) call. The parameter should be the location of your Oracle Client libraries. Do not pass this parameter on Linux.
  + Check that the Python process has permission to open the Oracle Client libraries. OS restrictions may prevent the opening of libraries installed in unsafe paths, such as from a user directory. On Linux you may need to install the Oracle Client libraries under a directory like /opt or /usr/local.
  + Check if Python and your Oracle Client libraries are both 64-bit or both 32-bit. The DPI-1047 message will tell you whether the 64-bit or 32-bit Oracle Client is needed for your Python.
  + Set the environment variable DPI\_DEBUG\_LEVEL to 64 and restart python-oracledb. The trace messages will show how and where python-oracledb is looking for the Oracle Client libraries.

At a Windows command prompt, this could be done with:

set DPI\_DEBUG\_LEVEL=64

On Linux and macOS, you might use:

export DPI\_DEBUG\_LEVEL=64

* + On Windows, if you have a full database installation, ensure that this database is the [currently configured database](https://www.oracle.com/pls/topic/lookup?ctx=dblatest&id=GUID-33D575DD-47FF-42B1-A82F-049D3F2A8791).
  + On Windows, if you are not using passing a library directory parameter to [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client), then restart your command prompt and use set PATH to check if the environment variable has the correct Oracle Client listed before any other Oracle directories.
  + On Windows, use the DIR command to verify that OCI.DLL exists in the directory passed to [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) or set in PATH.
  + On Windows, check that the correct [Windows Redistributables](https://oracle.github.io/odpi/doc/installation.html#windows) have been installed.
  + On Linux, check if the LD\_LIBRARY\_PATH environment variable contains the Oracle Client library directory. Some environments such as web servers reset environment variables. If you are using Oracle Instant Client, a preferred alternative to LD\_LIBRARY\_PATH is to ensure that a file in the /etc/ld.so.conf.d directory contains the path to the Instant Client directory, and then run ldconfig.
* If you get the error DPY-3010: connections to this database server version are not supported by python-oracledb in thin mode when connecting to Oracle Database 11.2, then you need to enable Thick mode by installing Oracle Client libraries and calling [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) in your code. Alternatively, upgrade your database.
* If you get the error DPI-1072: the Oracle Client library version is unsupported, then review the installation requirements. The Thick mode of python-oracledb needs Oracle Client libraries 11.2 or later. Note that version 19 is not supported on Windows 7. Similar steps shown above for DPI-1047 may help. You may be able to use Thin mode which can be done by removing calls [**oracledb.init\_oracle\_client()**](https://python-oracledb.readthedocs.io/en/latest/api_manual/module.html#oracledb.init_oracle_client) from your code.