ORACLE

Python-oracledb: the new, Thin driver for Oracle Database

Christopher Jones

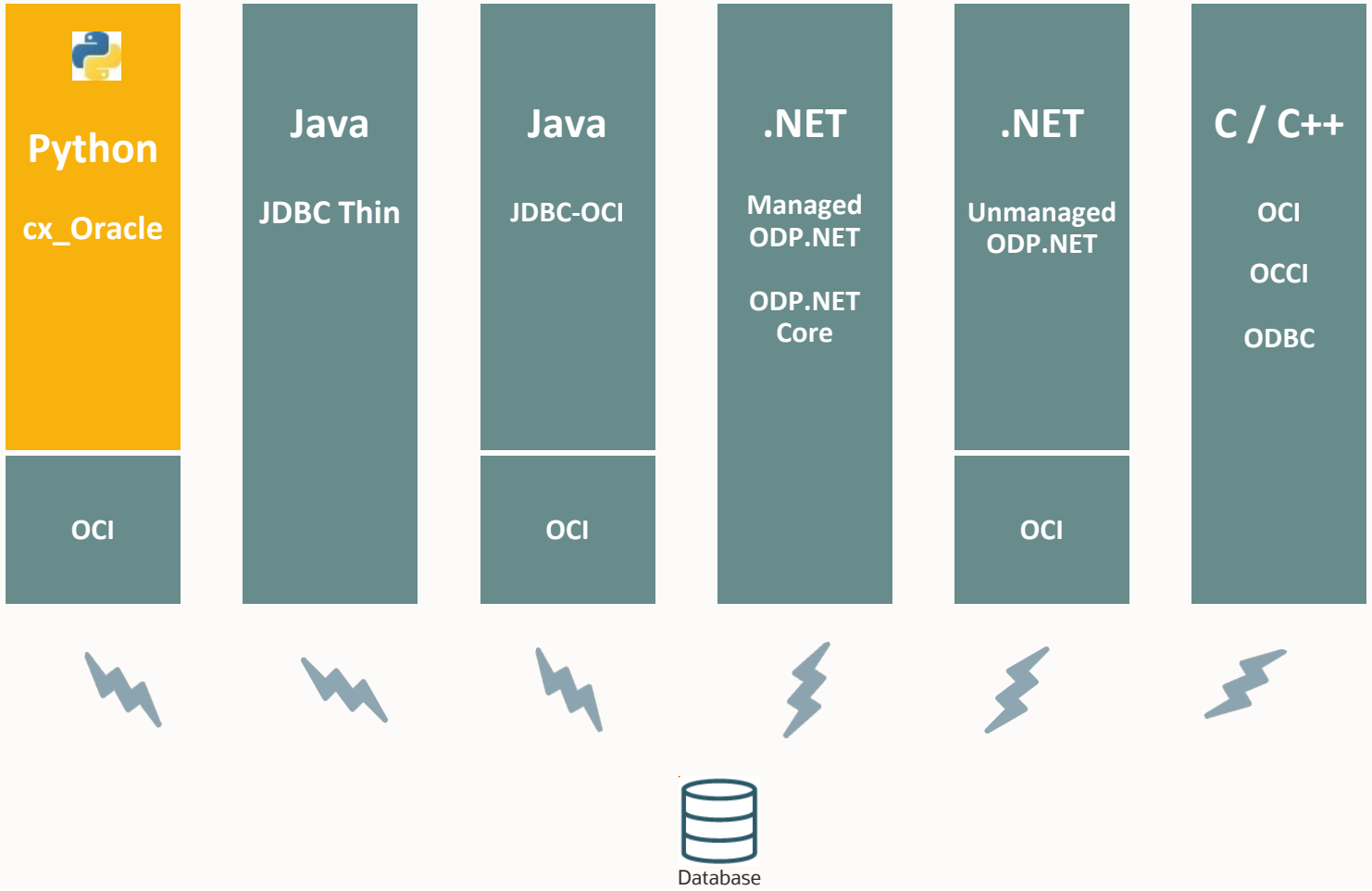
Senior Principal Product Manager
Oracle Database Data Access team
October 2022

Agenda

New Features

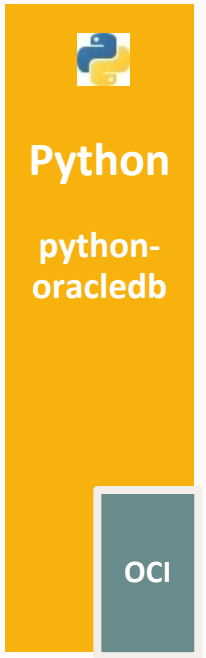
Examples

The popular language ecosystems



python-oracledb: the new cx_Oracle

python-oracledb: “need to know”



- Renamed, major upgrade of cx_Oracle
- New, default Thin mode: no Oracle Client
- Runtime choice to use Thick Mode
- Python 3.6 – 3.10
- Dual Apache 2 or UPL open source license
- Binary module for performance
- Python Database API V2 support

python-oracledb: “need to know”

“python-oracledb works great [...], improved more than 60% of data transfer performance!”

”This library is super cool since it doesn't require Instant Client and is compatible with M1 Mac.”

- Renamed, major upgrade of cx_Oracle
- New, default Thin mode: no Oracle Client
- Runtime choice to use Thick Mode
- Python 3.6 – 3.10
- Dual Apache 2 or UPL open source license
- Binary module for performance
- Python Database API V2 support

Active python-oracledb ecosystem

SQLAlchemy

Airflow

Django

PyPi

Conda-Forge

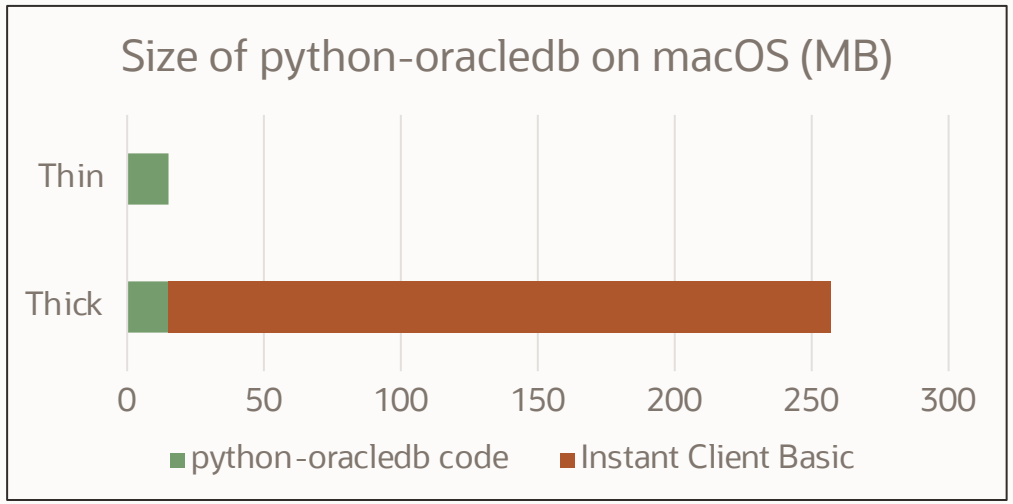
yum.oracle.com

Installing python-oracledb

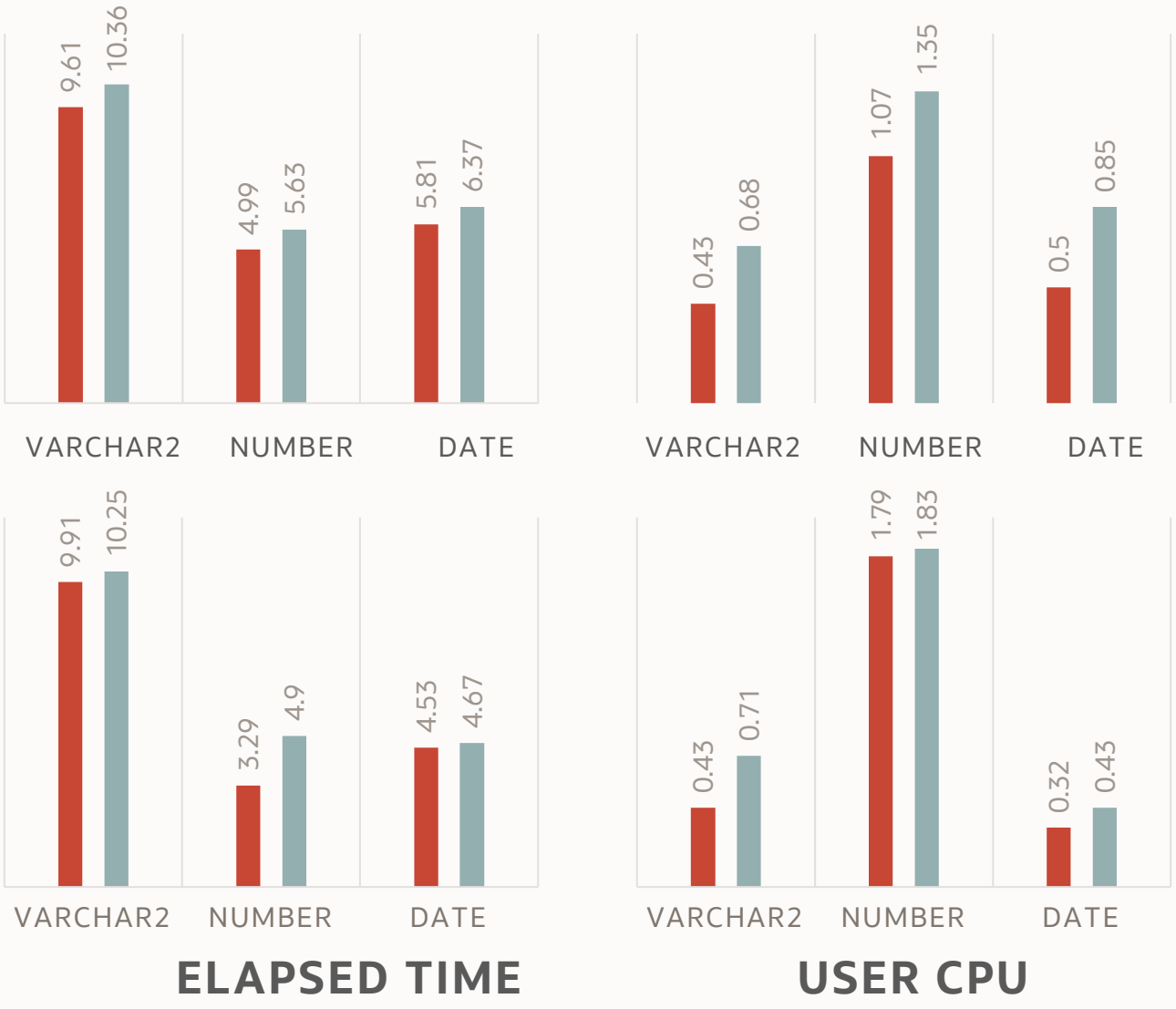
- One quick command installs a small, functional driver

```
python -m pip install oracledb --user
```

- Optionally install Oracle Client libraries for Thick mode



Atomic performance tests



Selecting 50,000 rows, 10 columns, using default `arraysize`

Thin

Inserting 500,000 rows, 10 columns at 100,000 per batch using `executemany()`

Thin Thick **Less is better**

Features of python-oracledb

Thin vs Thick comparison

Thin and Thick:

- Python DB API V2
- SQL, PL/SQL execution with data binding
- Array fetching, batch execution
- Standard datatypes including LOBs and JSON
- Connection pooling, in-band notifications for planned maintenance, DRCP
- 1-way TLS and mTLS

Thin:

- Connects directly to Oracle Database 12.1+

Thick:

- Oracle Client libraries allow access to Oracle Database 9.2+, depending on Client library version
- AC & TAC, AQ, CQN, SODA, Named Objects & Collections

Some new features in python-oracledb

- Azure AD OAuth 2.0 and IAM Token Authentication
- New `oracledb.defaults` object for setting common, application-wide defaults
- Two-phase commits (TPC)
- Advanced Queuing (AQ): recipients lists, JSON payloads, getting message IDs
- _Error class has a `full_code` attribute, e.g. 'ORA-01476'
- `oracledb.init_oracle_client()` may be called multiple times
- Light-weight `connection.is_healthy()` method

Some new features in python-oracledb

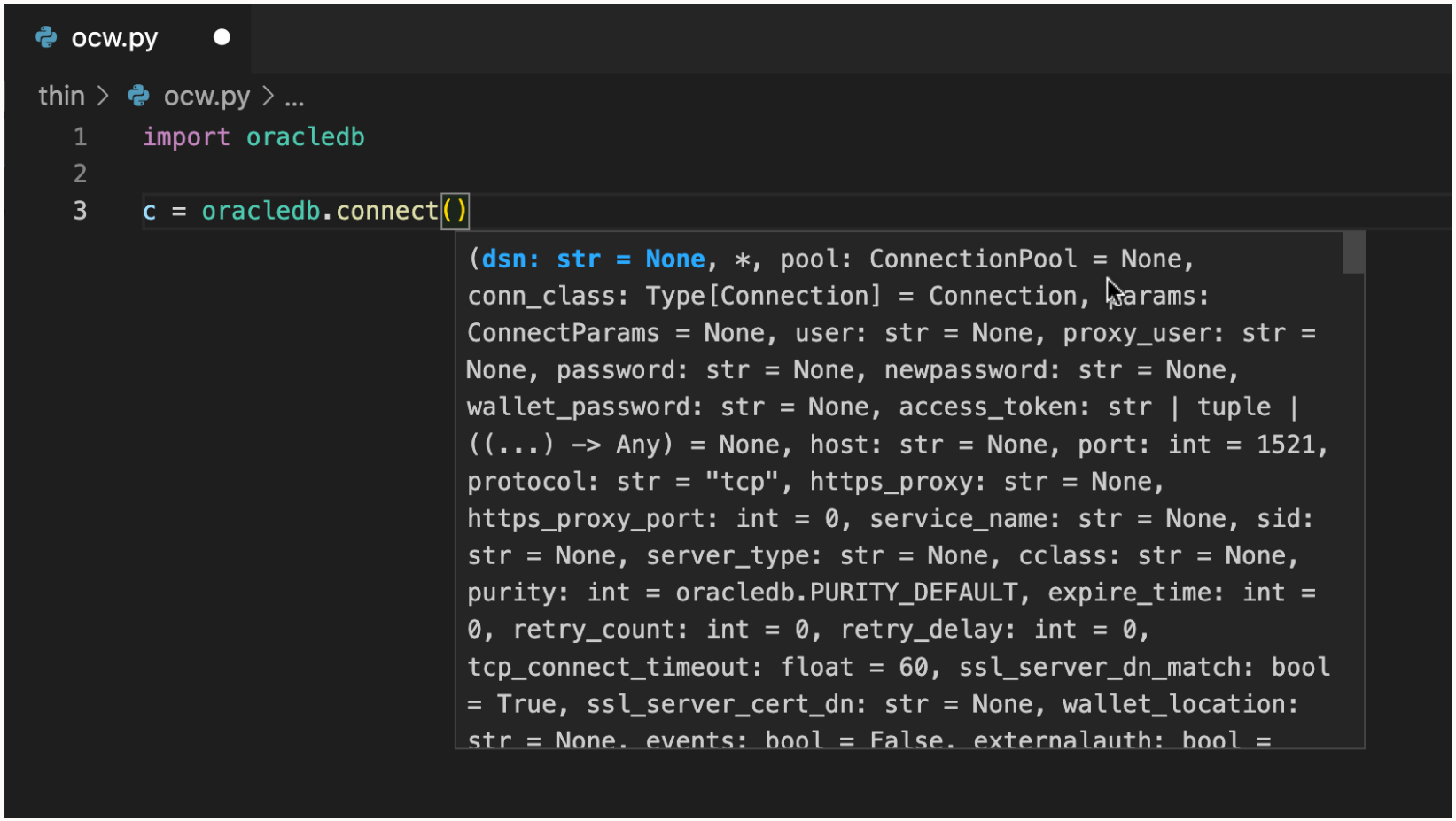
- Many more connection parameters can be specified:

  ```
  user, host, port, dsn, conn_class, params, proxy_user, password, newpassword,
  wallet_password, access_token, protocol, https_proxy, https_proxy_port, service_name, sid,
  server_type, cclass, purity, expire_time, retry_count, retry_delay, tcp_connect_timeout,
  ssl_server_dn_match, ssl_server_cert_dn, wallet_location, events, externalauth, mode,
  disable_oob, stmtcachesize, edition, tag, matchanytag, config_dir, appcontext, shardingkey,
  supershardingkey, debug_jdwp
  ```

- New ConnectParams and PoolParams classes
  - can be passed to connection creation and pool creation functions

Some new features in python-oracledb

Code annotations for intelligent IDE code completion



Using python-oracledb

Installing python-oracledb

- Install

  ```
  python -m pip install oracledb --user
  ```

  - Has dependency on ‘cryptography’ package
  - Optionally install Oracle Client libraries for Thick mode

- Platforms without ‘cryptography’

  ```
  python -m pip install oracledb --user --no-deps
  ```

  - Install Oracle Client libraries
  - Use Thick mode only

Upgrading cx_Oracle code

Import the new driver:

```
import oracledb as cx_Oracle
```

Use named parameters in calls to `connect()`, `Connection()`, and `SessionPool()`:

Change: `c = cx_Oracle.connect("un", "pw", "cs")`

to:

```
c = cx_Oracle.connect(user="un", password="pw", dsn="cs")
```


Enabling python-oracledb Thick mode

Calling `init_oracle_client()` enables Thick mode

Enabling python-oracledb Thick mode

```
if platform.system() == "Darwin" and platform.machine() == "x86_64":
    oracledb.init_oracle_client(lib_dir="/Users/cj/instantclient_19_8")
elif platform.system() == "Windows":
    oracledb.init_oracle_client(lib_dir=r"C:\oracle\instantclient_19_16")
elif platform.system() == "Linux":
    oracledb.init_oracle_client()
# else default to Thin mode
```


Error messages in python-oracledb's control may be better

Thin mode:

```
DPY-4010: a bind variable replacement value for placeholder ":mybv" was not provided
```

Thick mode and cx_Oracle:

```
ORA-01008: not all variables bound
```

Thin mode:

```
DPY-6001: cannot connect to database. Service "orclpdb" is not registered with the listener at host "localhost"
port 1521.  (Similar to ORA-12514)
```

Thick mode and cx_Oracle:

```
ORA-12514: TNS:listener does not currently know of service requested in connect descriptor
```

Some Python library messages are too ambiguous to wrap in Thin mode:

```
DPY-6005: cannot connect to database. Connection failed with "[Errno 61] Connection refused"
```


Thin connection parameters replace Thick Client settings

Thick mode and cx_Oracle:

```
oracledb.init_oracle_client(config_dir='/opt/oracle/config')
connection = oracledb.connect(user=un, password=pw, dsn=cs)
```

sqlnet.ora:

```
sqlnet.tcp_timeout=5
```

Thin mode:

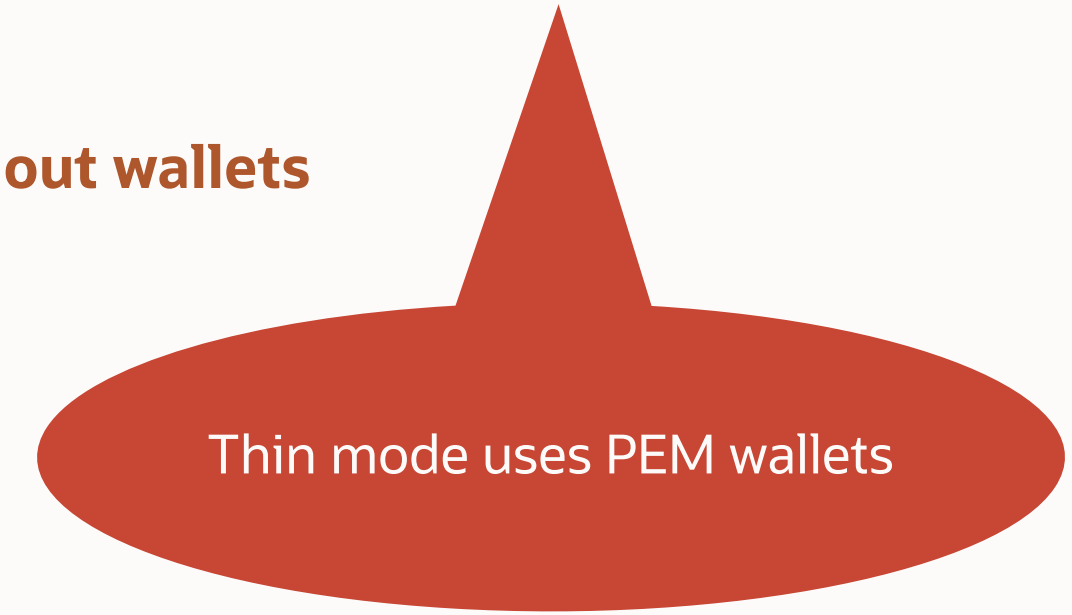
```
connection = oracledb.connect(
      user=un, password=pw, dsn=cs,
      tcp_connect_timeout=5)
```


Thin mode: Mutual TLS with Oracle Autonomous Database

```
connection = oracledb.connect(
      user=un, password=pw, dsn=cs,
      config_dir='/opt/oracle/config',       # dir with tnsnames.ora
      wallet_location='/opt/oracle/config', # dir with ewallet.pem
      wallet_password=wpw)
```

Don’t forget ADB supports “1-way” TLS without wallets

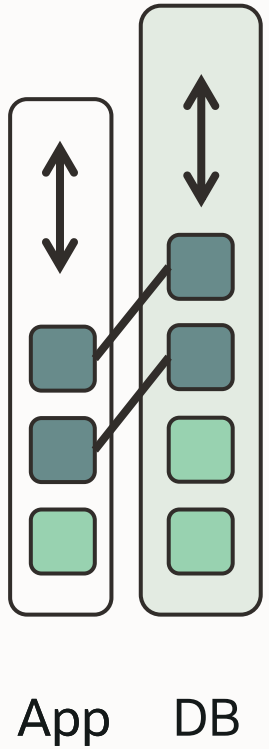
```
connection = oracledb.connect(
      user=un, password=pw, dsn=cs)
```



Thin mode: Connection pooling and DRCP

```
pool = oracledb.create_pool(     # SessionPool() still works
    user=un, password=pw,
    host="myhost", port=1521, service_name="orclpdb1",
    min=1, max=3, increment=1,
    getmode=oracledb.POOL_GETMODE_WAIT,
    server_type="pooled", cclass="MYDEMO",
    purity=oracledb.PURITY_SELF)

connection = pool.acquire()
```



Thin mode Globalization: Locale type handler

```
for row in cursor.execute("select sysdate from dual"):
      print(row)
```

How do we get 'Mi 15 Dez 19:57:56 2021' ???

Thin mode Globalization: Locale type handler

```
locale.setlocale(locale.LC_ALL, 'de_DE.UTF-8')
locale_date_format = locale.nl_langinfo(locale.D_T_FMT)

def type_handler(cursor, name, default_type, size, precision, scale):
    if default_type == oracledb.DB_TYPE_DATE:
        return cursor.var(default_type, arraysize=cursor.arraysize,
                outconverter=lambda v: v.strftime(locale_date_format))

connection.outputtypehandler = type_handler
```


Using python-oracledb in Frameworks, ORMs etc

```
import sys
import oracledb
oracledb.version = "8.3.0"
sys.modules["cx_Oracle"] = oracledb
import cx_Oracle
```


Python-oracledb Futures

- Ongoing feature additions to Thin mode
- Support for future Oracle Database features
- AsyncIO ?? ! - a big project
- All new features will be in python-oracledb, not cx_Oracle
- cx_Oracle packages only for critical fixes and new Python versions (circa 2 years)

Summary

- Python-oracledb is the new name for cx_Oracle
- Compatible with common frameworks and ORMs e.g. SQLAlchemy, Django, Pandas
- “Thin driver by default, with an optional Thick mode”
- Greatly improved deployment experience
- Improved development experience

python-oracledb: New namespace, new URLs

Announcement:	`tinyurl.com/dpy-rel`
Homepage:	`oracle.github.io/python-oracledb`
Documentation:	`python-oracledb.readthedocs.io`
GitHub:	`github.com/oracle/python-oracledb`
Blogs:	`cjones-oracle.medium.com, medium.com/@sharad-chandran`

Christopher Jones
christopher.jones@oracle.com
cjones-oracle.medium.com


ORACLE