Python Development on Oracle DB for beginners and Python's enthusiasts

Iwona Rajca
Oracle
Malaga, Spain

Keywords

python, API, database

Introduction

There is live beyond Java and SQL: developers and data scientists know this already and so does Oracle. In this session we present Oracle's capabilities in the wild-wide world of APIs and focus on Python in particular. We'll start with simple tasks like installing Python (we'll throw in some best practices too), to then move onto working with tables, JSON files, spatial data and data visualisation with Python libraries. The following will be talked about / presented on a live environment:

- Setting up Python 2.6: software download and installation
- Working with Python: Hello World
- Querying data in 12C from Python
- Parse JSON data stored in 12C with Python
- Working with SDO_GEOMETRY objects in Python

You do not need a previous experience with Python to join the session however some familiarity with another programming language would enhance your understanding of the demo.

Prerequisites and Setup

This section describes Python 2.6 installation and presents tools and libraries required to enable interaction with the Oracle Database.
cx_Oracle is a Python module that enables access to Oracle databases and conforms to the Python database API specification. This module is currently built against Oracle 11.2 and 12.1 and works for both Python 2.X and 3.X.

To access cx_Oracle documentation go to: https://cx-oracle.readthedocs.org/en/latest/

A Hello World application with Python

There are several ways to execute Python code. Here, we start with the presentation of two ways to execute Python code from the command line. The first is to execute code interactively i.e. executing commands directly in the interpreter. The other way is to save your code in a .py file and invoke the interpreter to execute the file.

Querying Oracle database

This exercise is an introduction to interacting with Oracle database via Python's interface. You will learn what tools are required to establish a connection to Oracle database and run a simple query on Oracle Database 12c.

Querying Oracle database from Python leverages cursor technology and follows the standard cursor execution cycle: opening a cursor, the fetching stage and closing a cursor to flush off the allocated memory. The cursor syntax cx_Oracle uses can be found under: http://cx-oracle.readthedocs.org/en/latest/index.html

Retrieving records from Oracle database using cursors is a simple as embedding the SQL statement within the `cursor().execute` statement.

Querying a JSON table from Python

This section provides detail on how to work with JSON data in Oracle Database 12c using Python's interface. The exercises include creating a table in 12C, loading data into the table, validating documents with IS JSON check, and querying data from Python.

The following sample JSON file is a collection of user reviews of a laptop downloaded from Amazon. The records have been collected in a JSON format. Each record consists of the rating given to the product, the comment title, the customer name, the colour of the product, the indication of whether the purchase was verified, and – finally – the actual comment.


```
{"rating": "3.0 out of 5 stars","title": "Three stars","customer_name": "Alex Scott","date": "on 12 July 2015","colour": "Colour Name: Envoy colour","purchase_type": "Verified Purchase","comment": "not enough memory\\"},
{"rating": "4.0 out of 5 stars","title": "... one week later despite a promise o 2-3 days Happy with the product","customer_name": "Diego Díaz","date": "on 27 August 2014","colour": "Colour Name: Envoy colour","purchase_type": "Verified Purchase","comment": " Happy with the product\\"},
{"rating": "4.0 out of 5 stars","title": "Airbook protective cover","customer_name": "Delpy Maryline","date": "on 5 April 2015","colour": "Colour Name: Envoy colour","purchase_type": "Verified Purchase","comment": "Lovely Cover. Just as I expected."},
{"rating": "4.0 out of 5 stars","title": "Cheaper from Envoy for a student but excellent product!","customer_name": "JasonReview","date": "on 7 December 2014","colour": "Colour Name: Envoy colour","purchase_type": "?","comment": "Excellent product but at the moment\\"},
{"rating": "4.0 out of 5 stars","title": "Four stars","customer_name": "christopher welsh","date": "on 5 April 2015","colour": "Colour Name: Envoy colour","purchase_type": "Verified Purchase","comment": "Yup it's the business"},
{"rating": "4.0 out of 5 stars","title": "Four stars","customer_name": "Kindle Customer","date": "on 26 September 2014","colour": "Colour Name: Envoy colour","purchase_type": "Verified Purchase","comment": "needed an instuction book"},
{"rating": "4.0 out of 5 stars","title": "Four stars","customer_name": "Robin O.","date": "on 20 December 2014","colour": "Colour Name: Envoy colour","purchase_type": "Verified Purchase","comment": "Excellent product and service."},
{"rating": "4.0 out of 5 stars","title": "Four stars","customer_name": "YONNA","date": "on 6 October 2014","colour": "Colour Name: Envoy colour","purchase_type": "Verified Purchase","comment": "very nice"},
```


Illustration 1. A json file containing customer comments from Amazon

Load JSON data into Oracle table

It is likely that rather than writing one JSON row to the database at the time, you will require to load many JSON records at once. For that purpose, you can leverage Oracle External Tables functionality.

In this section we will create a new JSON external table that points to our comments file and query the records from Python's shell.

To create an external table pointing to the JSON file, execute the following in the Python shell:

```
> cur.execute('create table json (doc clob) organization external (type
oracle_loader default directory ext_tab_dir access parameters (records
delimited by newline nobadfile nologfile fields (doc char(50000))) location
(ext_tab_dir: \'amazon_comments.json\')) parallel reject limit unlimited')
```

- External tables are created using the SQL CREATE TABLE...ORGANIZATION EXTERNAL statement.
- Type oracle_loader specifies the external table access driver to ORACLE_LOADER that can work with text files.
- Default directory points to the default directory to use for input files.
- Access parameters specify the structure of the input file – in the above example we are specifying that the records are delimited by newline, the BADFILE clause names the file to which records are written when they cannot be loaded because of errors (here NOBADFILE is set for simplicity). Logfile stores logs generated while accessing the external table (here: NOLOGFILE). FIELDS parameter specifies extra details about the fields of the source file: here we are overwriting the default char(255) constraint to char(50000) to be able to accommodate for longer comments.
- Finally, location specifies what files should be used for the external table. The files should be located under the default directory otherwise the directory has to be explicitly specified.

For more information regarding the external tables functionality please refer to
https://docs.oracle.com/cd/E11882_01/server.112/e22490/et_concepts.htm

Working with spatial data in Python

This section describes working with Oracle spatial object in Python. Python is able to work with SDO_GEOMETRY objects stored in the Oracle database.
In this exercise we'll work with HERE (Formerly NAVTEQ) archive, a sample geographical dataset available from the Oracle Technology Network website:
http://www.oracle.com/technetwork/middleware/mapviewer/downloads/navteq-lic-168395.html
We will run a points-in-polygon query and a nearest neighbour search.

Displaying points on a map with Python

Python comes with some data visualization libraries, such as maps. In this module we will use the Python's Basemap library to plot the list of the points generated in the Points in polygon exercise.

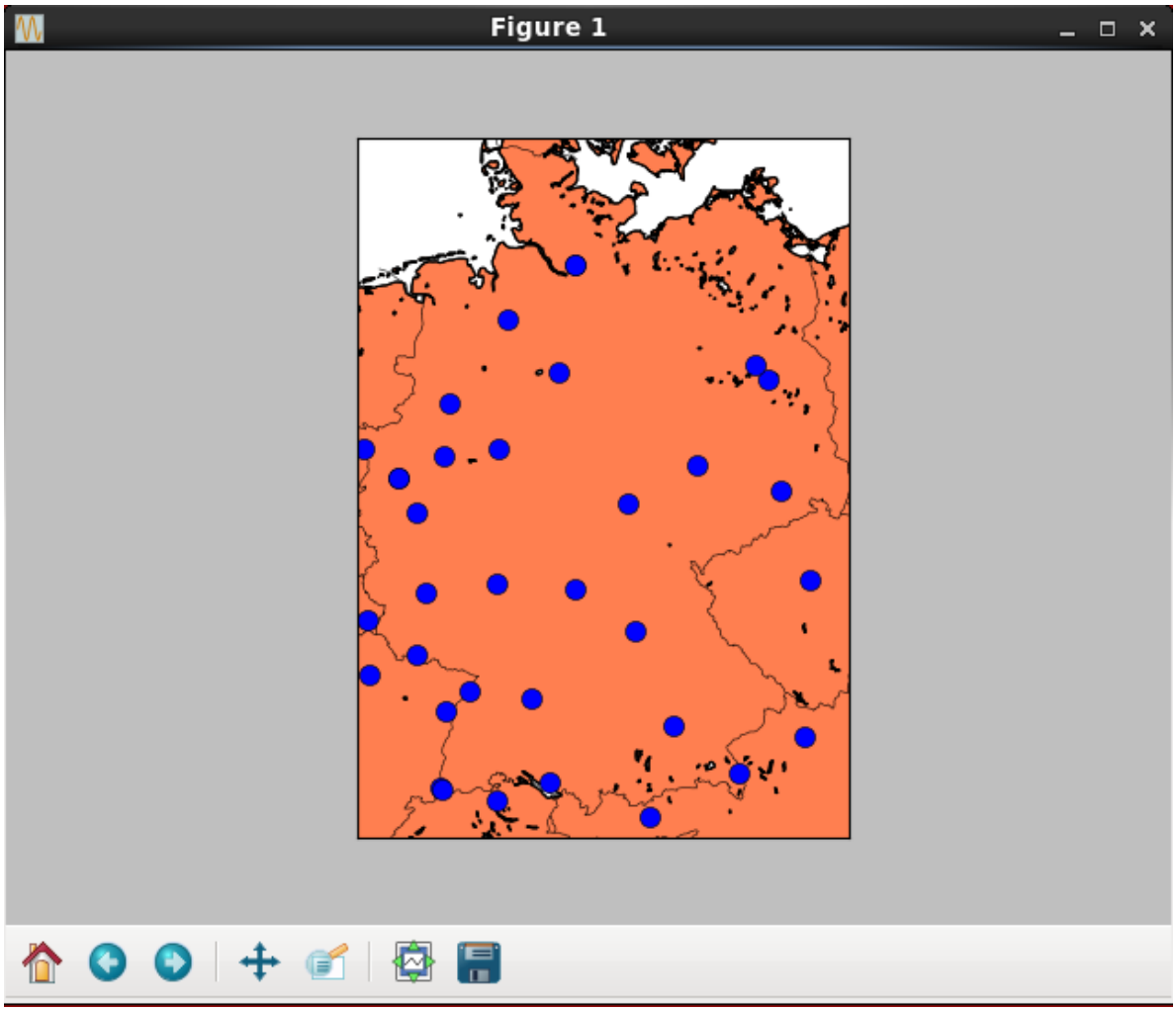

Illustration 2. Map displaying the results of points-in-polygon query on Oracle Database.

To learn more about Basemap's functionality, including the map customization and a list of functions to work with geodata, please refer to its official documentation: http://matplotlib.org/basemap/index.html or review Basemap's tutorial available here: http://basemaptutorial.readthedocs.org/en/latest/index.html

Contact address:

Iwona Rajca

Oracle

Oracle Iberica, 10 Severo Ochoa

29590 Malaga, Spain

Phone: +34 952108924

Email: iwona.rajca@oracle.com