Python for Oracle

Arup Nanda
Longtime Oracle Technologist
And
Python Explorer

Oracle Database

Great for Data Store

Critical for Business Operations

Performance?

Run in laptop?

But how do you share it?

What you will learn?

- Why Python
- How can you learn Python pretty fast, for Oracle Developers
- Plotting Charts
- Interaction with Oracle
- Tons of code, a free tutorial series on OTN, videos
- Three real-world example applications
- What's next

Why Python

- Used in data science, machine learning, AI
  - Powerful and math plotting tools, free and paid
- Spark has a PySpark
- General purpose
  - Not like R, which is very data oriented
- Convenience of Interpreted Language
  - Rapid Application Development
- Default language of Raspberry Pi
- Has libraries for all types of access. Including Oracle DB.

Installation

- Python is freely available for most OS'es
  - Download from python.org
  - For many platforms, even Rasperry Pi
- Three components:
  - Command line version
  - Command Line in a Window
  - IDLE: Interactive DeveLopment Environment

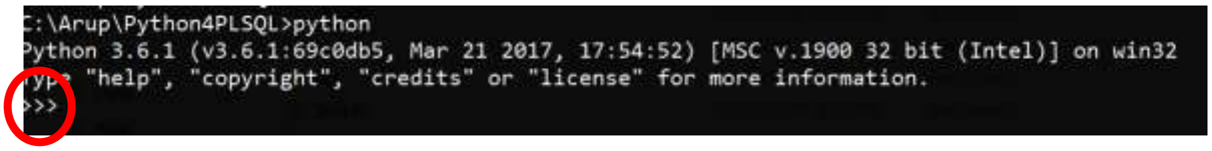
Bring up command line

- From OS prompt:

  `C:\> python`

  (Same command executable on any of the OS'es)

  Brings up the python command line prompt:



```
:\Arup\Python4PLSQL>python
ython 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)] on win32
ype "help", "copyright", "credits" or "license" for more information.
>>>
```

Python prompt

How difficult is learning Python?

- It's *similar*—not same—as PL/SQL
- Some language elements are even the same
- Approach: Jumpstarting the learning by using examples of PL/SQL

PL/SQL	Python

`if Condition then`
`  … statement …`
`end if;`
	`if Condition:`
	`    … statement …`

Help Me!

- Command help brings up the help interface

```
>>> help()
help> Command_You_Need_Help_On
```

- Or

```
>>> help Command_You_Need_Help_On
```

Basics

- Python is case sensitive
  - So, v1 and V1 are different.
- Comments
  - Starts with #
- How to quit:
  - It's a function. So, quit()
  - Or Control-Z

Learning Tool

bit.ly/python4plsql

- 5 Part Article Series
- Complete tutorials
- Video
- Quizzes
- Free!

Arrays

- Three types
  - **List**
    ```
    x1 = [1,2,3]
    ```
  - Can be any mix of datatypes [1,"s",1.5]
  - Address elements by x1[*position*]
  - **Tuple**—same as list but immutable
    ```
    x1 = (1,2,3)
    ```
  - **Dictionary**—key-value pairs
    ```
    x1 = {'k1':'v1','k2':'v2'}
    ```

If

- Python is positional

```
if Condition:
    Statement…
elif Condition:
    Statement
else:
    Statement
```

This indentation is necessary

Looping

- FOR Loop

```
for i in range(1,11):
   print('i=',i)
```

- Looping through arrays

```
x1 = [‘a’,’e’,’i’,’o’,’u’]
for i in range(len(x1)):
   print(x1[i])
```

- WHILE Loop

Adding Modules

- Python Package Index (PyPI)
- Install it my calling it as a module in python

```
C:\> python -m pip ModuleName
```

- Modules we will install

```
python -m pip pandas
python -m pip numpy
python -m pip matplotlib
python –m pip scipy
```

Data Analysis

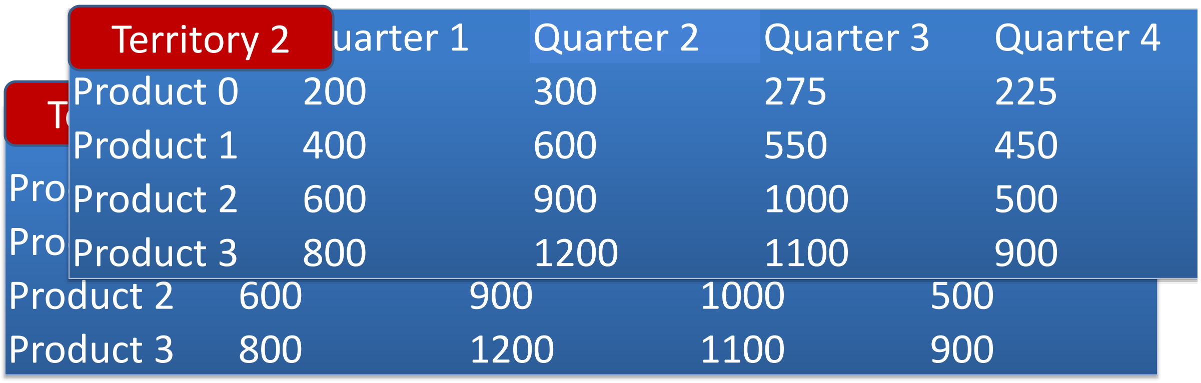
Multi-dimensional Array

- Sales Data
  - ProductID
  - Quarter
  - Territory
  - Amount

2D Array

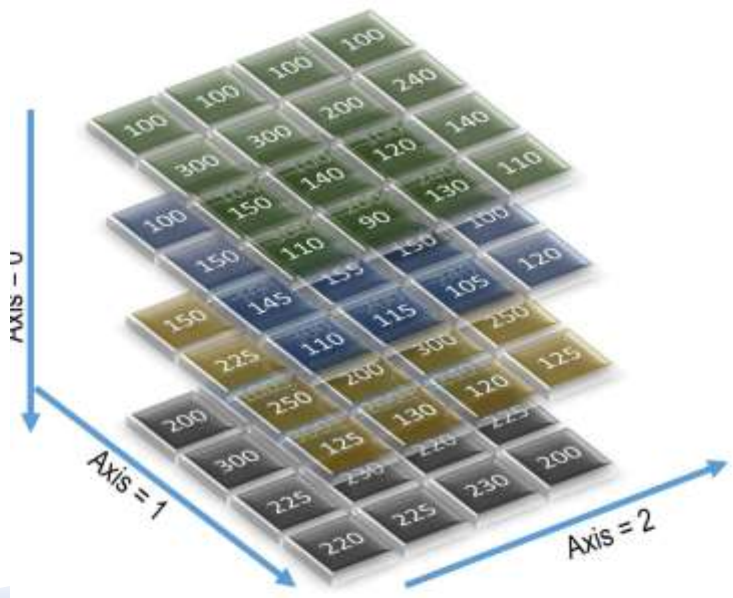
	Quarter 1	Quarter 2	Quarter 3	Quarter 4
Product 0	200	300	275	225
Product 1	400	600	550	450
Product 2	600	900	1000	500
Product 3	800	1200	1100	900

3D Array



Territory 2	Quarter 1	Quarter 2	Quarter 3	Quarter 4
Product 0	200	300	275	225
Product 1	400	600	550	450
Product 2	600	900	1000	500
Product 3	800	1200	1100	900

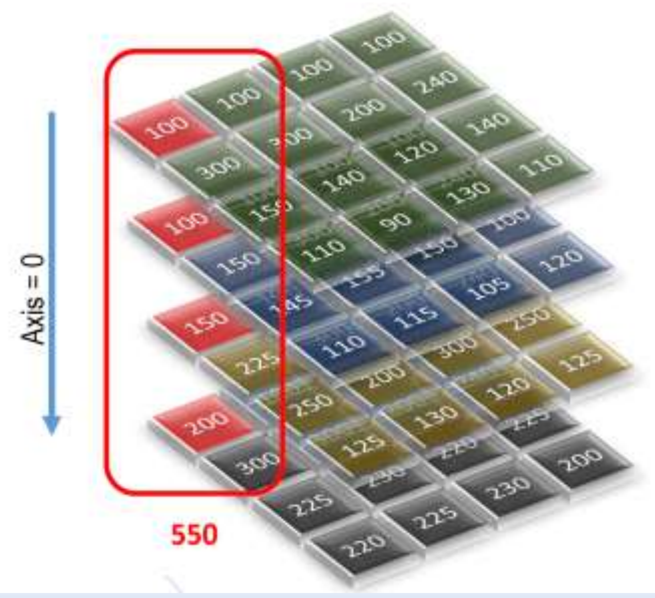
T				
Pro				
Pro				
Product 2	600	900	1000	500
Product 3	800	1200	1100	900



Numpy Array

```
sales = np.array([
    [
        [50,50,50,50],
        [150,150,180,120],
        [75,70,60,70],
        [55,45,65,55]
    ],
    [
        [100,90,110,100],
        [150,160,130,170],
        [145,155,150,100],
        [110,115,105,120]
    ],
```

```
    [
        [150,140,160,150],
        [225,220,230,225],
        [250,200,300,250],
        [125,130,120,125]
    ],
    [
        [200,250,150,200],
        [300,350,250,300],
        [225,230,220,225],
        [220,225,230,200]
    ]
    ])
```



```
sales.sum(axis=0)

sales.min(axis=0)
```

numpy1.txt

Visualization

Plotting

- Package matplotlib

Oracle DB Connection

Connecting to Oracle DB

- A module called cx_Oracle
- Basic Operation

```
>>> import cx_Oracle as cxo
>>> conn = cxo.connect('hr','hr','server:1521/AL122')
>>> c1 = conn.cursor()
>>> c1.execute('select * from employees where rownum <11')
>>> for row in c1:
...    print(row)
```

- Fetch one row alone

```
r1 = c1.fetchone()
```

- Or, many

```
r1 = c1.fetchmany(numRows=2)
```

More cx_Oracle operations

- Set the arraysize

```
>>> c1.arraysize = 10
```

- Describe the output

```
>>> c1.description
```

- One step fetch

```
>>> r1 = c1.execute('select * from sales where rownum < 11')
>>> for rec in r1:
...     print (rec)
```

- Close the cursor

```
>>> c1.close()
```

Bind Variables

```
>>> conn = cxo.connect('sh','sh','localhost:1521/AL122')
>>> c1 = conn.cursor()
>>> c1.prepare('select * from sales where rownum < :limiting_rows')
>>> c1.execute(None, {'limiting_rows':11})
>>> c1.fetchall()
```

Dynamically Constructed Queries

- Note the query

  `c1.prepare('select * from...')`

- You can construct the query as a character array

```
>>> s1 = 'select '
>>> s1 +=  '*'
>>> s1 +=  ' from sales '
>>> s1 +=  ' where rownum < '
>>> s1 +=  '11'
>>> s1
'select * from sales  where rownum < 11'
>>> r1 = c1.execute(s1)
```

Example 1

Creating a DB Monitor

- Objective
  - To measure waits on named events in database
  - From AWR repository tables
  - And plot them

```
select sn.end_interval_time,
   (after.total_waits-before.total_waits) "No. of Waits",
   (after.time_waited_micro-before.time_waited_micro)/
   (after.total_waits-before.total_waits) "Avg Wait in
us"
from
   dba_hist_system_event before,
   dba_hist_system_event after,
   dba_hist_snapshot sn
where
   before.event_name=:event_name
and
   after.event_name=before.event_name
and
   after.snap_id=before.snap_id+1
and
   after.snap_id=sn.snap_id
and
   (after.total_waits-before.total_waits) > 0
order by after.snap_id
```

Example 2

Finding Employee Retention

- Objective
  - To model how employees stay in an organization
- Hypothesis
  - The tenure of the employees depends on department, the specific job, the reporting manager.
- Methodology
  - We will find the correlation between tenure and multiple other factors

Example 3

Twitter Analysis

- Objective
  - To find out how many have been tweeting about Oracle Code (i.e. with #OracleCode), and how many times each has been retweeted.
  - Store the data in an Oracle database for further processing
- Methodolgy
  - A package called tweepy allows twitter interaction

Tweepy

- A package for accessing twitter accounts
- You need:
  - Authentication details from twitter
    - Consumer key
    - Consumer secret
    - Authentication token
    - Authentication secret
  - All can be gotten from https://dev.twitter.com/apps
    - To Install tweepy
    - Using command `python –m pip install tweepy`

Authentication in Tweepy

- Define the values

```
ckey = '...'
csecret = '...'
atoken = '...'
asecret = '...'
```

- Authenticate

```
auth = tweepy.OAuthHandler(ckey,csecret)
auth.set_access_token(atoken, asecret)
```

- Declare the API object

```
api = tweepy.API(auth)
```

Accessing objects

- Details about the account

```
>>> iam = api.me()
```

- To find out all about the account:

```
for i in iam._json.keys():
    print("%30s = %s" % (i, iam._json[i]))
```

Finding Tweets

- The search API is used

```
>>> t =
tweepy.Cursor(api.search,q="#OracleCode",lang="en",since
="2018-03-01").items()
```

- We used the “since” parameter to limit the number of items
- The object t now contains the data on the tweets. To get the handle and text of the tweet:

```
>>> for text in t:
...     print(text.author.screen_name, text.text)
```

More Uses

- Connecting to a physical device using Raspberry Pi
- Using a Event Driven Architecture to use fraud detection in transactions
- Using Oracle AQ and cx_Oracle
- Using microservices where each service talks to another using messages

In Summary

- Python is a general purpose language; not just data
- Interpreted, but can also be cached
- Rich data manipulation capabilities exist natively as well as via packages
- Very easy to learn for Oracle PL/SQL developers
- Multiple use cases spanning a variety of applications

Thank You!

Blog: arup.blogspot.com
Tweeter: @ArupNanda
Facebook.com/ArupKNanda
Google Plus: +ArupNanda