Python for Oracle Professionals

Arup Nanda
Longtime Oracle Technologist
And
Python Explorer

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
  - Can also use C; but you get tons of readymade projects
- Has libraries for all types of access. Including Oracle DB.

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* …

What I will Cover

- Basics of Python with examples of PL/SQL
- Plotting Charts
- Connecting to Oracle
- RaspberryPi
- Tons of code, a free tutorial on OTN, videos

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:

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

Python prompt

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.
- How to quit:
  - It's a function. So, quit()
  - Or Control-Z

Demos

Learning Tool

http://bit.ly/python4plsql

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

Topics

- Direct input
- Variable assignment
- Type of variable
- Datatype conversion
- Comments
- User Input
- Script execution
- Operators

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'}
    ```

Differences

Operation	PL/SQL	Python
Declaration of variables	DECLARE numvar1 NUMBER;	None required.
Assignment of the values	numvar1 := 1;	numvar1 = 1 n1 = n2 = n3 = 1 n1, n2, n3 = 1,2,3
To know the datatype of a variable	Not required; it's declared explicitly	type(var1)
Boolean datatype	TRUE and FALSE (case-insensitive)	True, False (case-sensitive) Any value that evaluates to 0 is False Any non-zero value is True
Concatenation	'Char1'||'Char2' CONCAT('Char1','Char2')	'Char1','Char2' 'Char1' + 'Char2'
Line termination	;	None required
Display	dbms_output.put_line()	print()
Comments	Lines starting with -- OR between /* and */	Lines starting with # Lines between """ quotes
Equality operator	=	==
Conversions	to_char(n1) to_number(c1)	str(n1) float(c1)
User inputs	ACCEPT n1 NUMBER PROMPT 'Enter value of n1: ' (SQL*Plus, actually)	n1 = input('Enter value of n1: ')
Same operators	+ - * / < > <= >= != <>	+ - * / < > <= >= != <>
Power function	POWER(5,2)	5**2
Modulus or remainder	REMAINDER(n1,n2)	n1%n2
Combination operator	Doesn't exist	n1 += 3 functionally same as n1 = n1 + 3
Array	VARRAY PL/SQL tables Nested tables	list lVar = [1,2,3,3] tuple tVar = (1,2,3,3) set sVar = {1,2,3} # duplicates not allowed dictionary dVar = {"key1":"val1","key2":"val2"}
Array index	Starts with 1	Starts with 0

If

- Python is positional

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

This indentation is necessary

For Loop

- Python has for

```
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])
```


Differences

Functionality	PL/SQL	Python

Basic IF	`if` (*condition*) `then`
	List of statements to be executed
	`end if;`
	list of statements outside of IF block.
		if (*condition*):
		List of statements to be executed
		List of statements outside of if block
		Note the dissimilarities:
		1. There is no end if.
		2. The ":" after the condition.
		3. The indentation before the statements in the if block. There is no beginor end statements to mark the beginning and the end of the blocks under the "if" condition. The indentation determines the block.

ELSE and ELSIF	if (*condition*) then
	List of statements to be executed
	else
	List of statements
	end if;
	list of statements outside of IF block.
		if (*condition*):
		List of statements to be executed
		else:
		List of the statements in else block
		There is no end if and like IF, the blocks of code are determined by indentation.

ELSIF	ELSIF	elif

Differences, cont…

FOR loop	for i in *Start...End* loop *list of statements* end loop;	for range(*Start*,*End*): *list of statements* *statement not inside the for loop* Note the indentation for the statements inside the for loop.

Skipping counters in a loop, for example, skipping one number, that is, 1, 3, 5, and so on	`for i in` *Start..End*
	`loop`
	`if mod(i,2) = 0 then`
	Do something
	`end if;`
	`end loop;`
		for i in range(Start,End,2):
		Do something

Counting backwards in a loop, for example, 3, 2, 1, and so on	`for i in reverse` *Start..End*
	`loop`
	list of statements
	`end loop;`
		for i in range(Start, End, -1):
		list of statements

WHILE loop	while (condition) loop *list of statements;* end loop;	while (*condition*): *list of statements* *statement not inside the while loop* Note the indentation for the statements inside the while loop.
Breaking away from a loop	EXIT WHEN (*Some condition*); or, IF (*Some condition*) THEN EXIT;	Inside a loop: if (*Some condition*): break Note the indentation.

Differences, contd…

ELSE for loops	No PL/SQL equivalent.	while (*Some condition*):
		if (*Some other condition*):
		break
		else:
		statements executed if the loop exits or completes without finding a match

case statement	`case`
	`when` *Some condition* `then`
	List of statements;
	`when` *Some other condition* `then`
	List of statements;
	`end case;`
		There is no case; but you can create similar functionality using if ... elif ...

Continue in a loop	CONTINUE	continue
Do nothing where a legal, syntactically correct statement is needed	NULL;	pass

Adding Modules

- Python Package Index
- 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
```


Numpy multi-dimensional Array

- Sales Data
- ProductID
- Quarter
- Territory
- Amount

Analysis and Plotting

Connecting to Oracle DB

- A module called cx_Oracle
- Basic Operation

```
>>> import cx_Oracle as cxo
>>> conn = cxo.connect('sh','sh','localhost:1521/AL122')
>>> c1 = conn.cursor()
>>> c1.execute('select * from sales 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)
```


Thank You!

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