Introduction

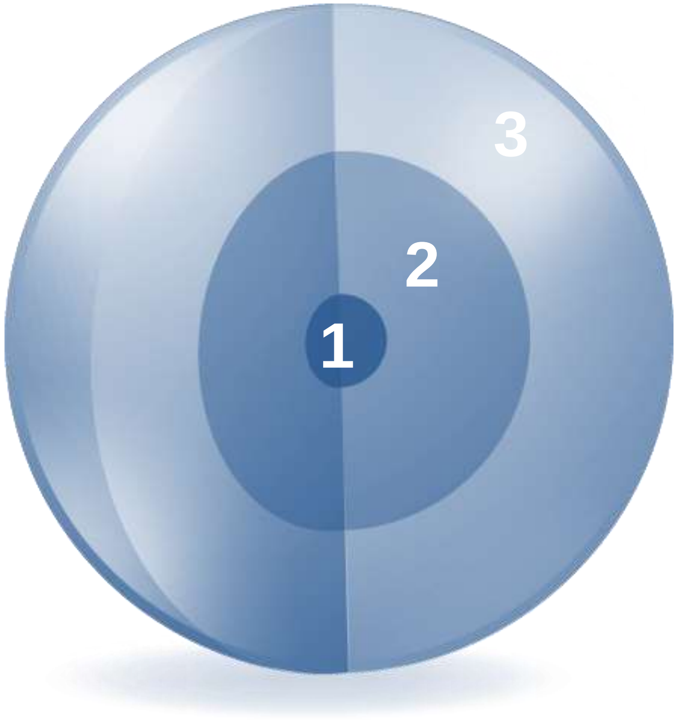
INTRODUCTION

TO

PL/SQL

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

What You will Learn at the end of this Session ?



1. Benefits and need of PL/SQL

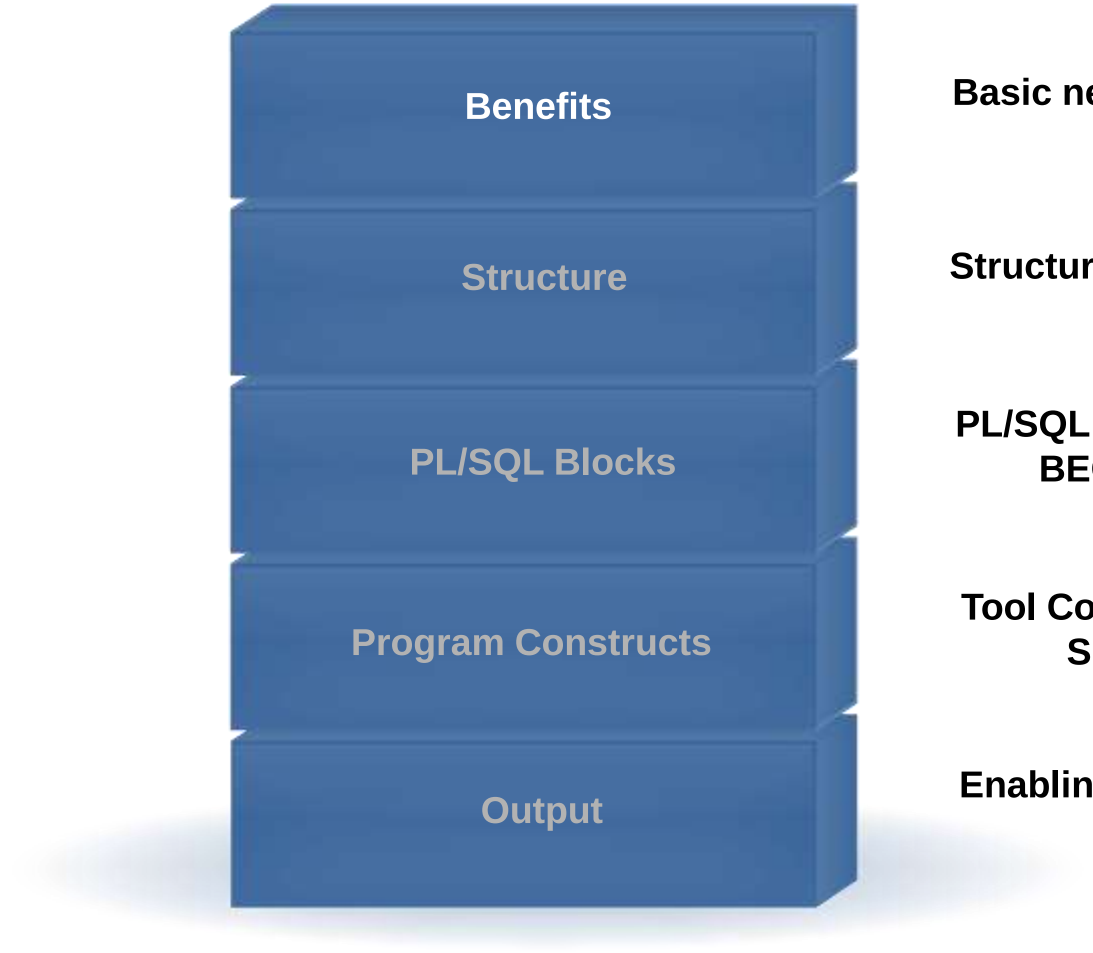
2. Explain PL/SQL blocks

3. Generating output messages in PL/SQL

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Plan



Basic need and benefits of PL/SQL

Structure of a PL/SQL program

PL/SQL block structure with DECLARE, BEGIN, EXECPTION and END.

Tool Constructs and Database Server Constructs

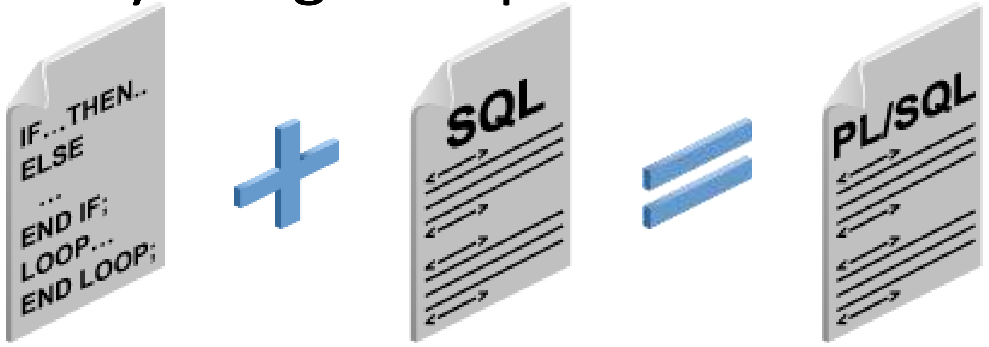
Enabling and Viewing output of a PL/SQL block

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

About PL/SQL

- PL/SQL:
  - Stands for "Procedural Language extension to SQL"
  - Is Oracle Corporation's standard data access language for relational databases
  - Seamlessly integrates procedural constructs with SQL



ORACLE
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

About PL/SQL

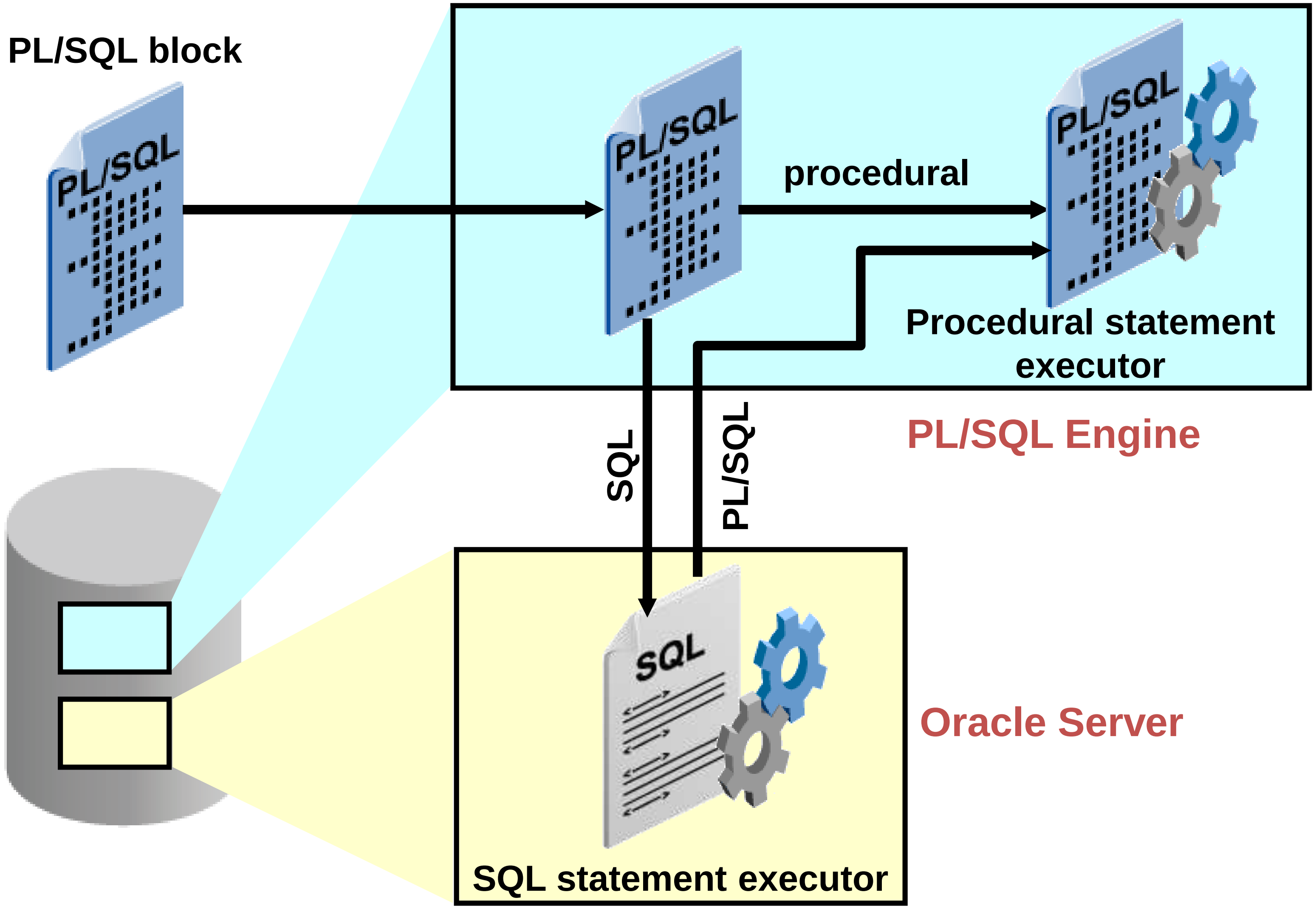
– Provides a block structure for executable units of code. Maintenance of code is made easier with such a well-defined structure.

– Provides procedural constructs such as:
  - Variables, constants, and data types
  - Control structures such as conditional statements and loops
  - Reusable program units that are written once and executed many times

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

PL/SQL Run-Time Architecture



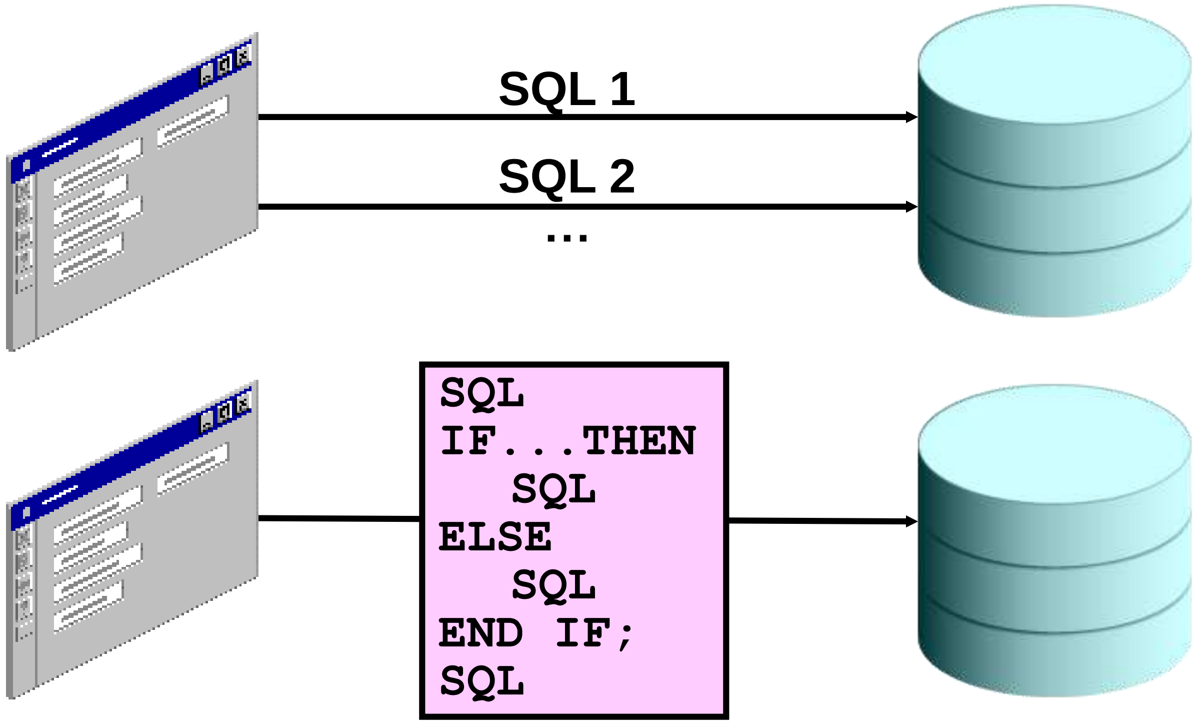
ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Benefits of PL/SQL

– Integration of procedural constructs with SQL

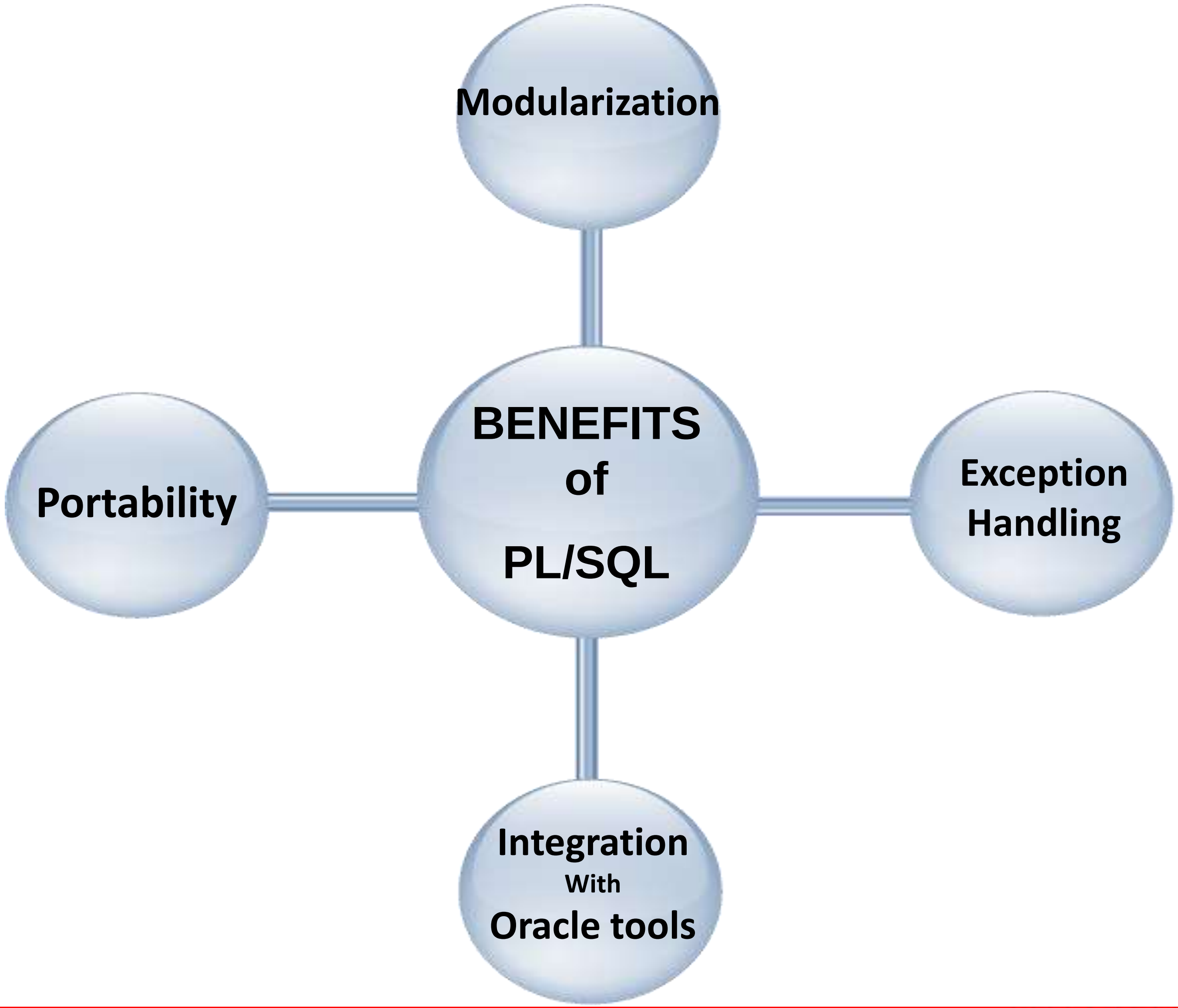
– Improved performance



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

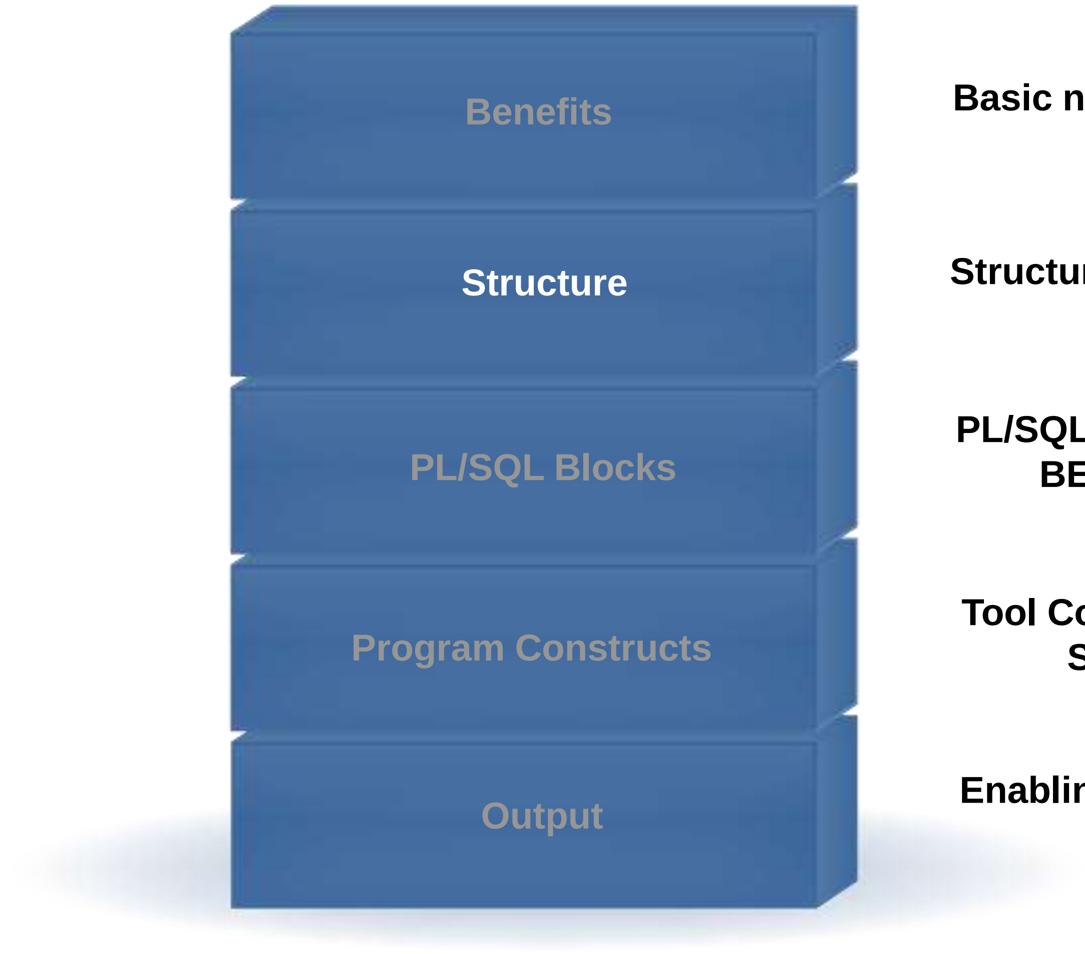
Benefits



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Plan



Basic need and benefits of PL/SQL

Structure of a PL/SQL program

PL/SQL block structure with DECLARE, BEGIN, EXECPTION and END.

Tool Constructs and Database Server Constructs

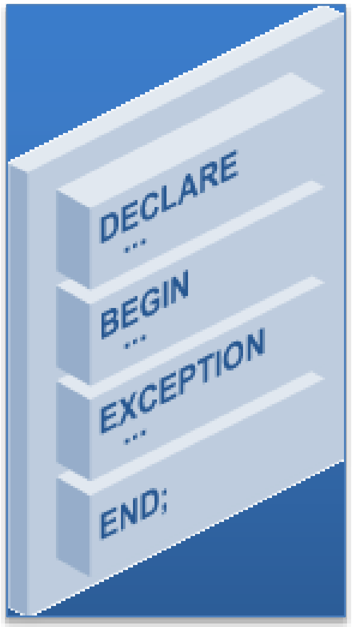
Enabling and Viewing output of a PL/SQL block

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

PL/SQL Block Structure

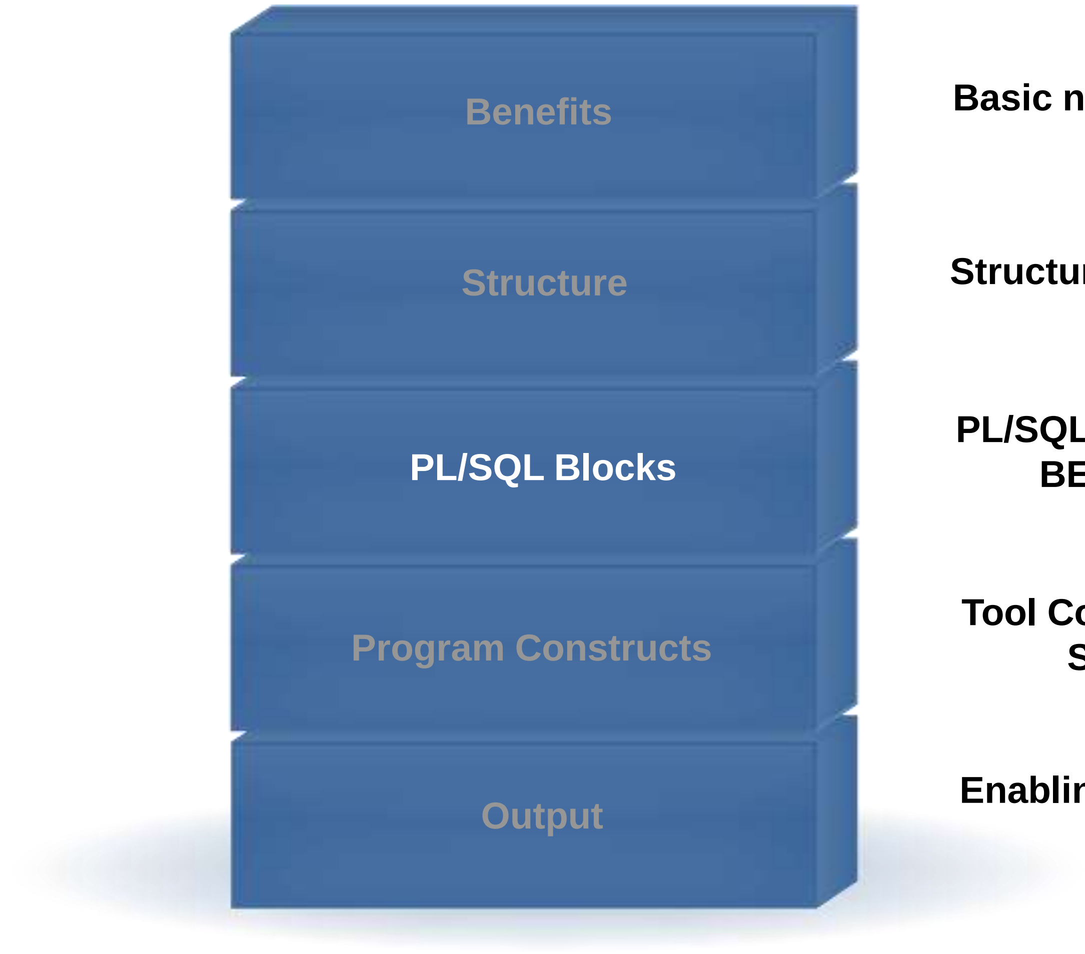
- `DECLARE` (optional)
  - Variables, cursors, user-defined exceptions
- `BEGIN` (mandatory)
  - SQL statements
  - PL/SQL statements
- `EXCEPTION` (optional)
  - Actions to perform when exceptions occur
- `END;` (mandatory)



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Plan



Basic need and benefits of PL/SQL

Structure of a PL/SQL program

PL/SQL block structure with DECLARE, BEGIN, EXECPTION and END.

Tool Constructs and Database Server Constructs

Enabling and Viewing output of a PL/SQL block

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Block Types

Procedure	Function	Anonymous

```
PROCEDURE name
IS

BEGIN
  --statements

[EXCEPTION]

END;
```

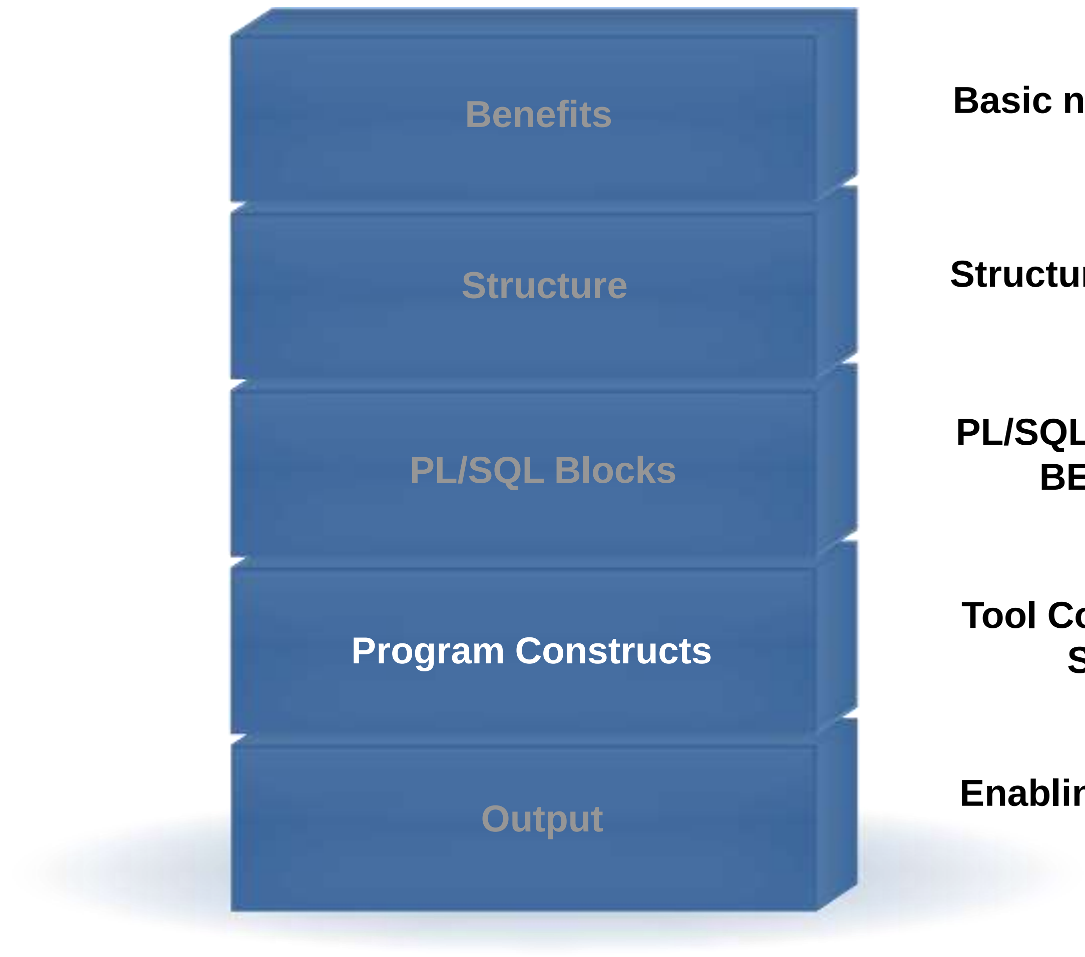
```
FUNCTION name
RETURN datatype
IS
BEGIN
  --statements
  RETURN value;
[EXCEPTION]

END;
```

```
[DECLARE]


BEGIN
  --statements

[EXCEPTION]

END;
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Plan



Basic need and benefits of PL/SQL

Structure of a PL/SQL program

PL/SQL block structure with DECLARE, BEGIN, EXECPTION and END.

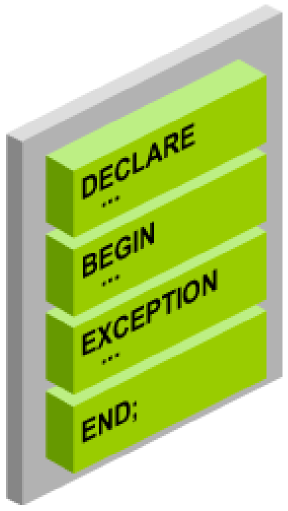
Tool Constructs and Database Server Constructs

Enabling and Viewing output of a PL/SQL block

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Program Constructs



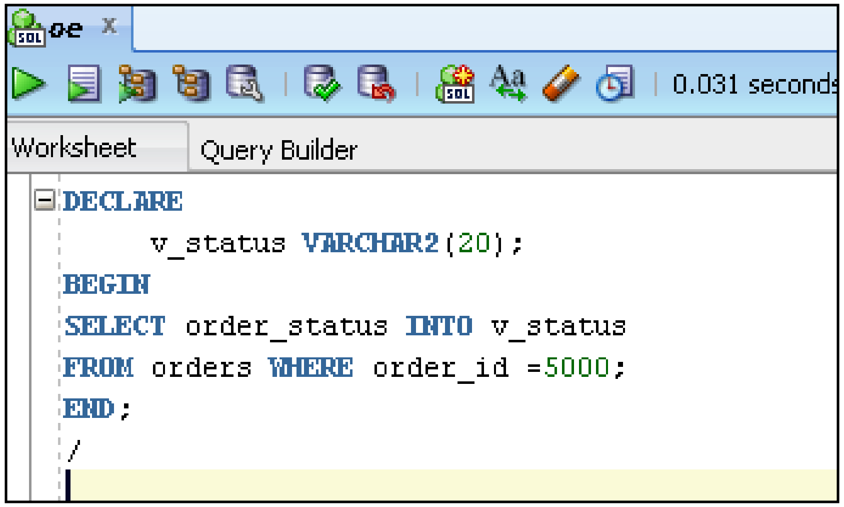
Tools Constructs	Database Server Constructs
Anonymous blocks	Anonymous blocks
Application procedures or functions	Stored procedures or functions
Application packages	Stored packages
Application triggers	Database triggers
Object types	Object types

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Examining an Anonymous Block

An anonymous block in the SQL Developer workspace:



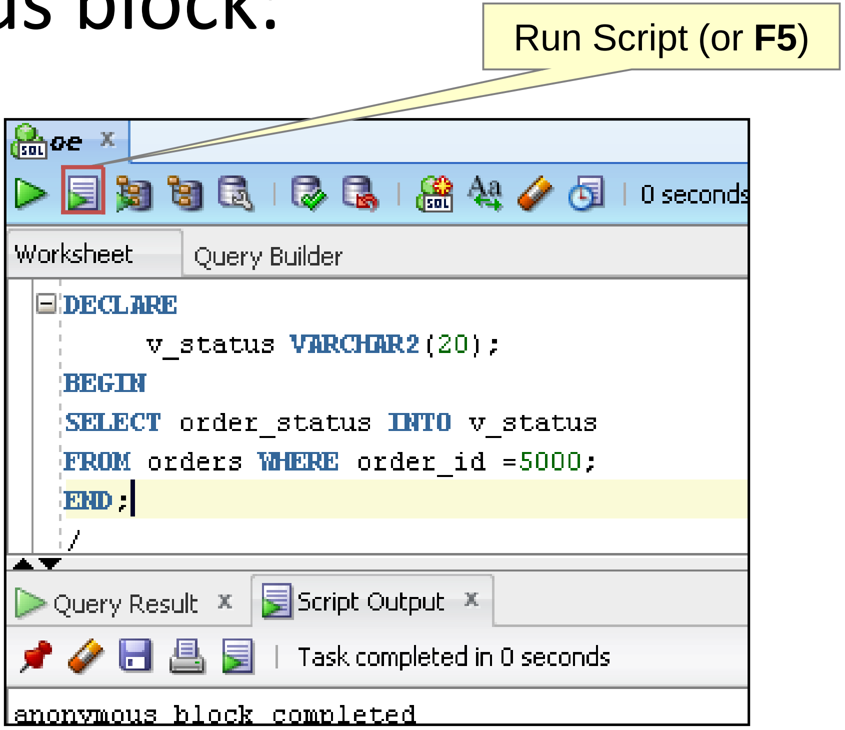
```
DECLARE
      v_status VARCHAR2(20);
BEGIN
SELECT order_status INTO v_status
FROM orders WHERE order_id =5000;
END;
/
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Executing an Anonymous Block

Click the Run Script button to execute the anonymous block:

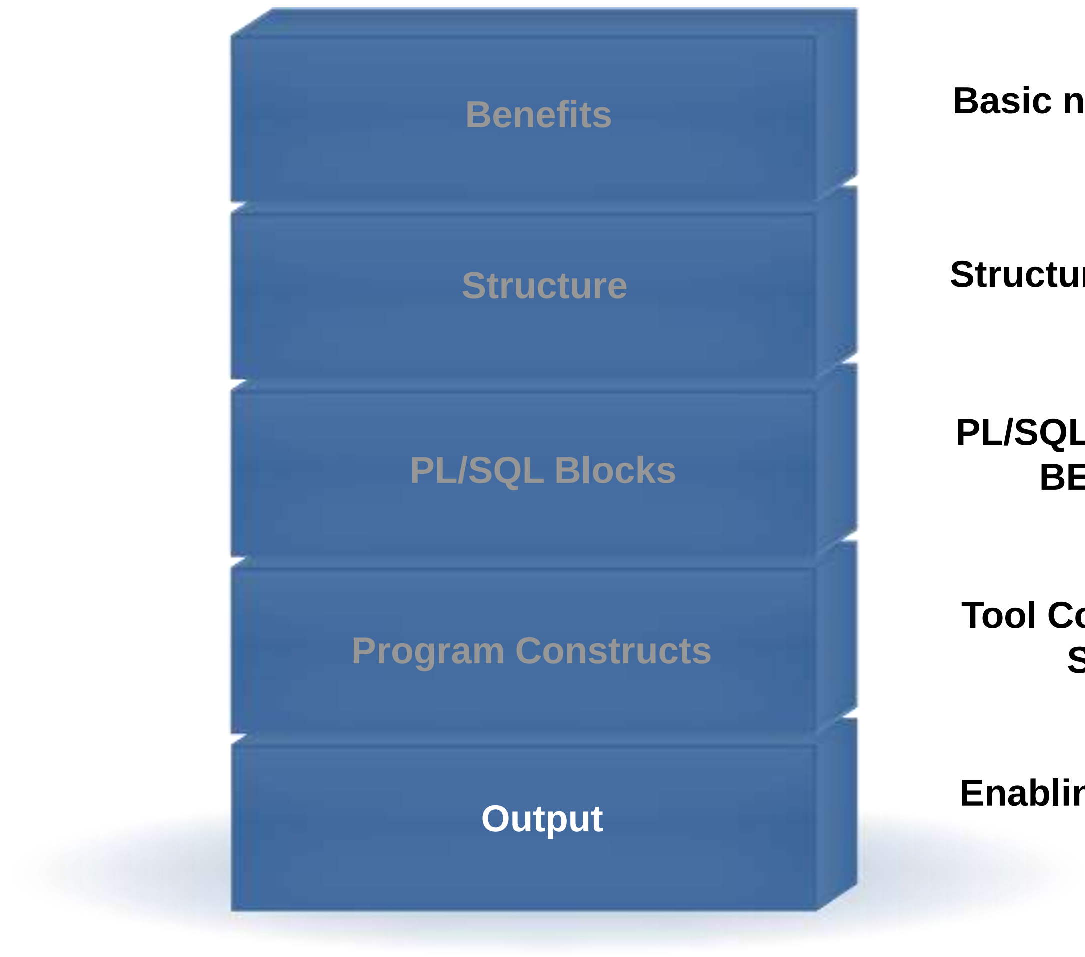


```
DECLARE
      v_status VARCHAR2(20);
BEGIN
SELECT order_status INTO v_status
FROM orders WHERE order_id =5000;
END;
/
```

```
anonymous block completed
```

ORACLE
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Plan



Basic need and benefits of PL/SQL

Structure of a PL/SQL program

PL/SQL block structure with DECLARE, BEGIN, EXECPTION and END.

Tool Constructs and Database Server Constructs

Enabling and Viewing output of a PL/SQL block

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Enabling Output of a PL/SQL Block

1. To enable output in SQL Developer, execute the following command before running the PL/SQL block:

```
SET  SERVEROUTPUT  ON
```

2. Use a predefined Oracle package and its procedure in the anonymous block:

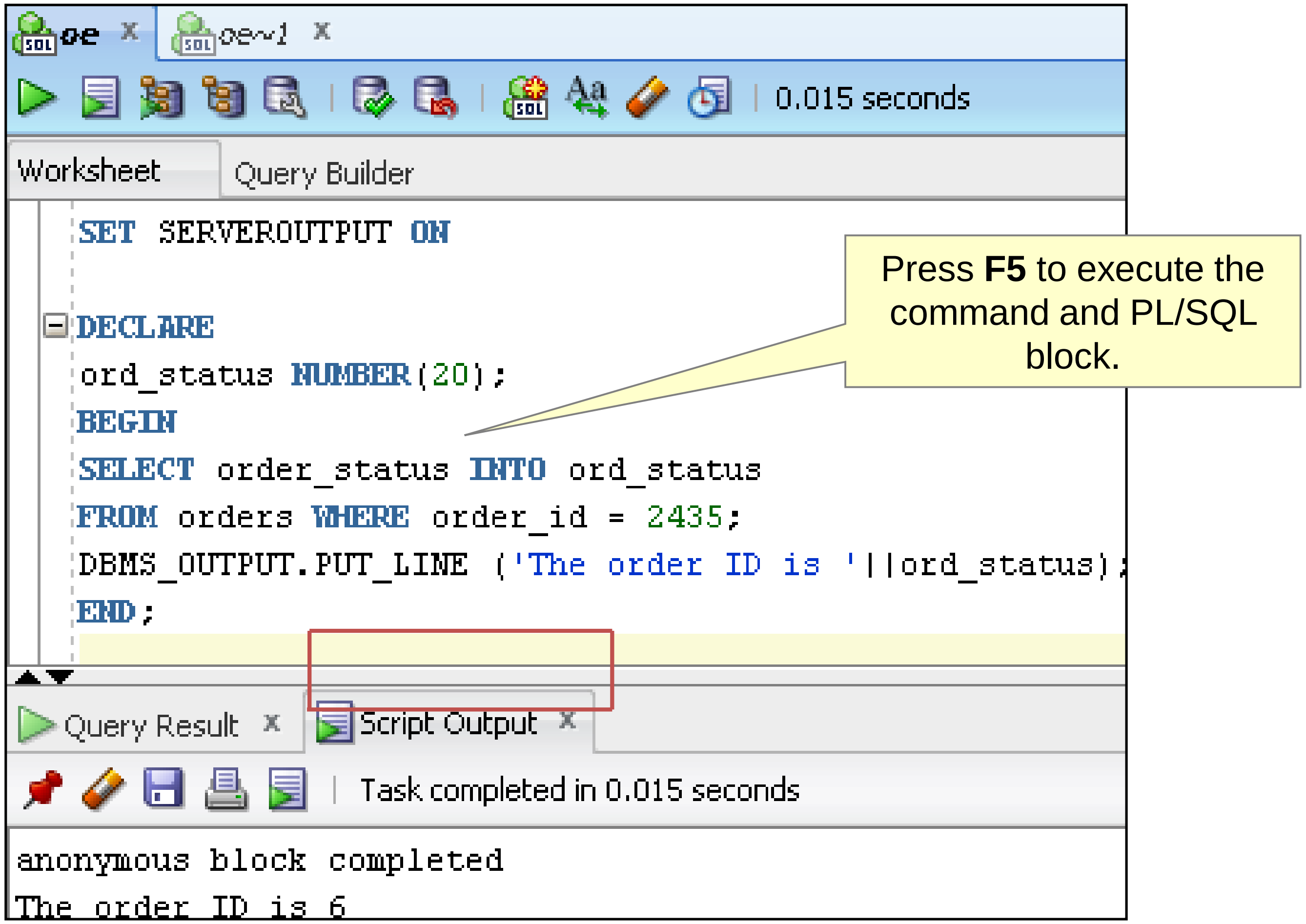
   `DBMS_OUTPUT.PUT_LINE`

```
DBMS_OUTPUT.PUT_LINE (' The First Name of the
Employee is ' || v_fname) ;
…
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Viewing the Output of a PL/SQL Block



```
SET SERVEROUTPUT ON

DECLARE
ord_status NUMBER(20);
BEGIN
SELECT order_status INTO ord_status
FROM orders WHERE order_id = 2435;
DBMS_OUTPUT.PUT_LINE ('The order ID is '||ord_status);
END;
```

Press **F5** to execute the command and PL/SQL block.

```
anonymous block completed
The order ID is 6
```

Quiz

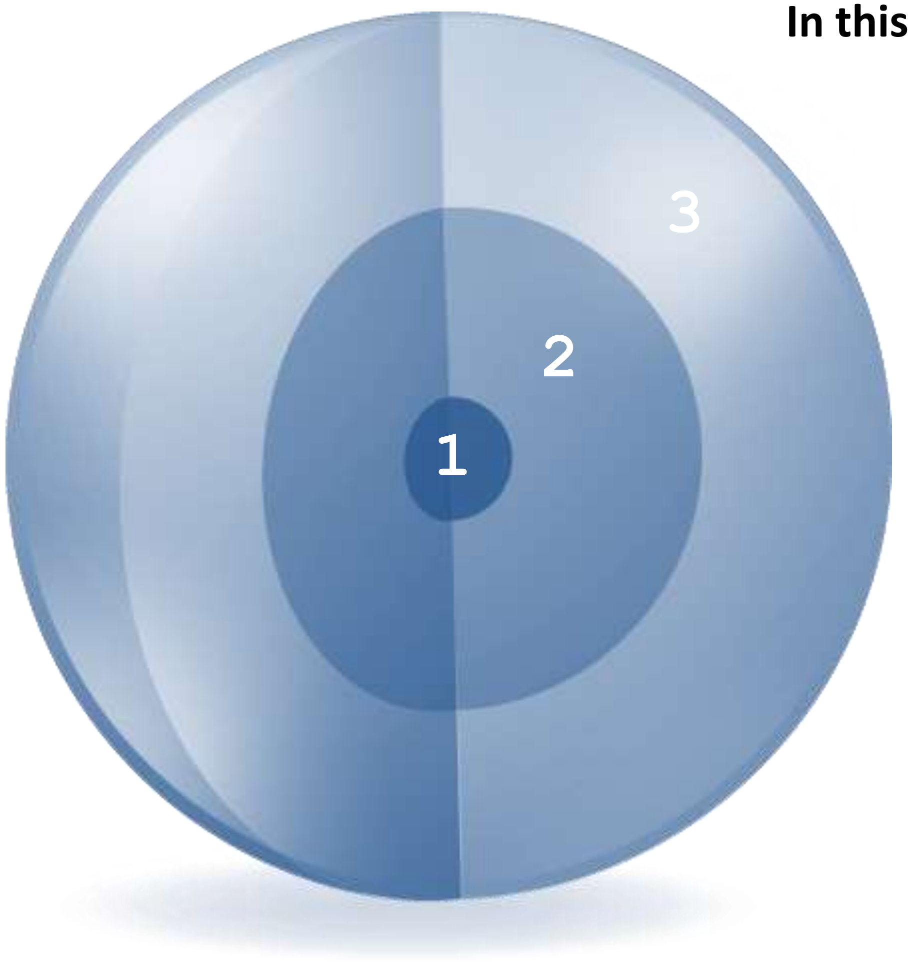
A PL/SQL block *must* consist of the following three sections:

- A Declarative section, which begins with the keyword `DECLARE` and ends when the executable section starts.
- An Executable section, which begins with the keyword `BEGIN` and ends with `END`.
- An Exception handling section, which begins with the keyword `EXCEPTION` and is nested within the executable section.

- True
- False

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Summary



In this lesson, you should have learned how to:

1 **Describe the benefits of PL/SQL**

2 **Differentiate between PL/SQL block types**

3 **Output messages in PL/SQL**

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 1: Overview

Identifying the PL/SQL blocks
Practice 1: Overview
Creating and executing a simple PL/SQL block

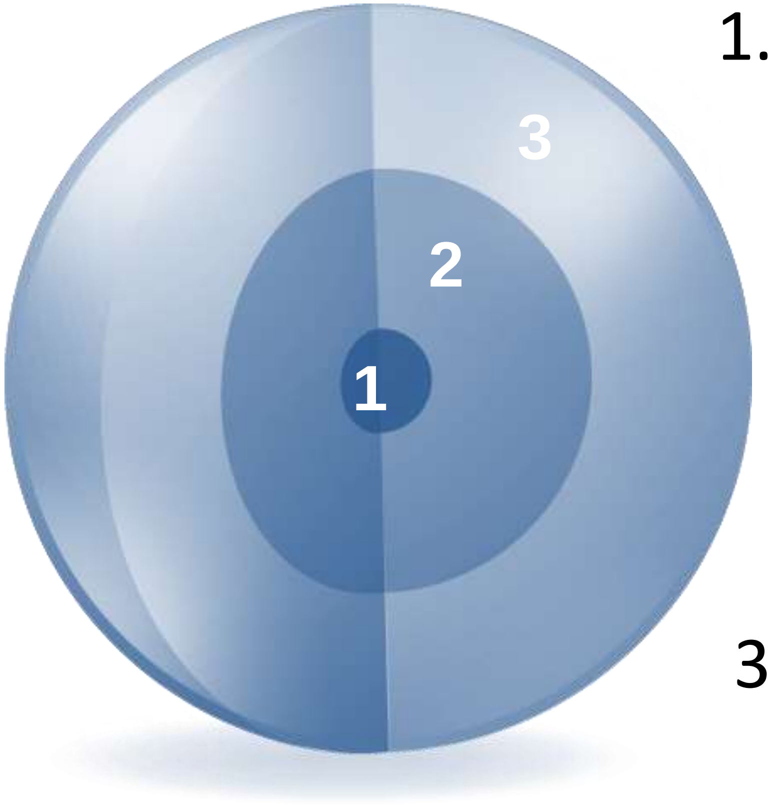
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Lesson 3

Writing Executable Statements

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

What You will Learn at the end of this Session ?



1. Writing executable statements in a PL/SQL block

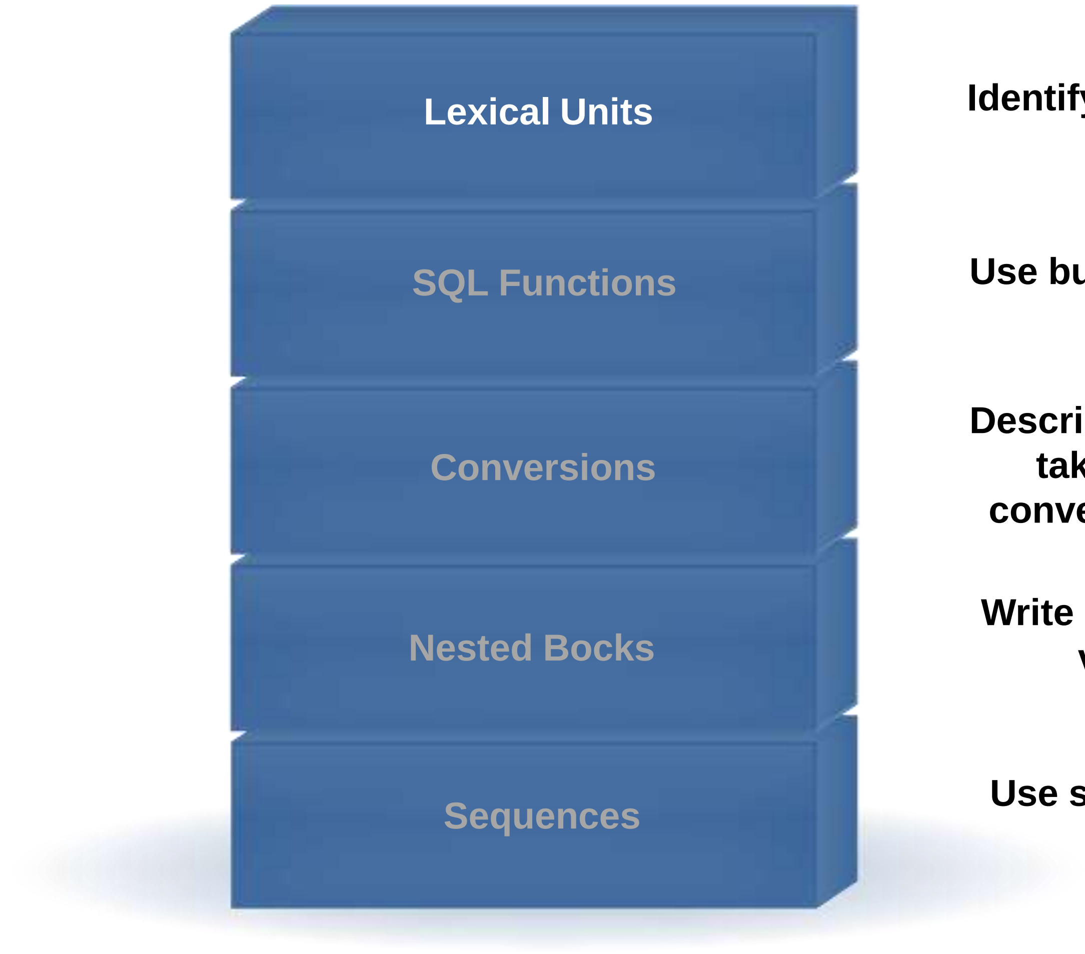
2. Writing nested blocks

3. Using operators and developing readable code

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Plan



Identify lexical units in a PL/SQL block

Use built-in SQL functions in PL/SQL

Describe when implicit conversions take place and when explicit conversions have to be dealt with

Write nested blocks and qualify variables with labels

Use sequences in PL/SQL expressions

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Lexical Units in a PL/SQL Block

Lexical units:

- Are building blocks of any PL/SQL block
- Are sequences of characters including letters, numerals, tabs, spaces, returns, and symbols
- Can be classified as:
  - Identifiers: `v_fname, c_percent`
  - Delimiters: `; , +, -`
  - Literals: `John, 428, True`
  - Comments: `--, /* */`

ORACLE

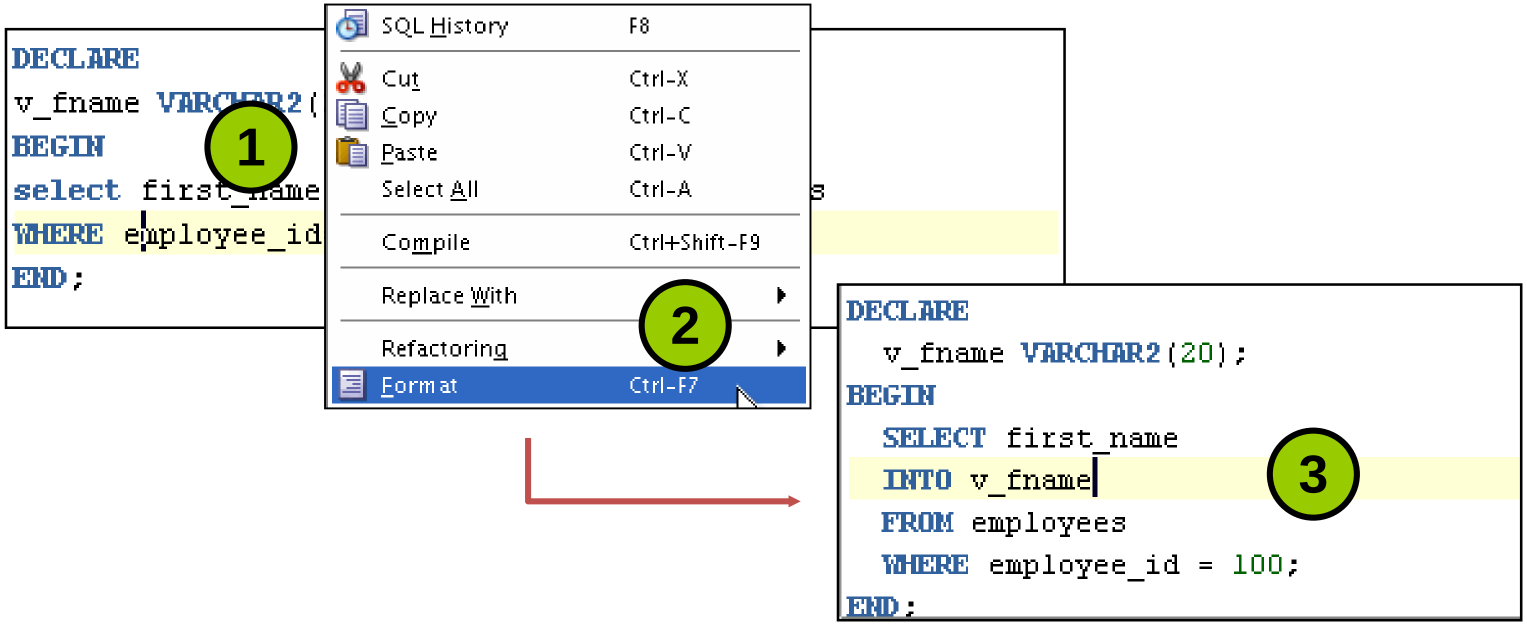
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

PL/SQL Block Syntax and Guidelines

– Using Literals
  - Character and date literals must be enclosed in single quotation marks.
  - Numbers can be simple values or in scientific notation.

```
v_name  :=  'Henderson' ;
```

– Formatting Code: Statements can span several lines.



```
DECLARE
v_fname VARCHAR2(
BEGIN
select first_name
WHERE employee_id
END;
```

```
DECLARE
  v_fname VARCHAR2(20);
BEGIN
  SELECT first_name
  INTO v_fname
  FROM employees
  WHERE employee_id = 100;
END;
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Commenting Code

- Prefix single-line comments with two hyphens (--).
- Place a block comment between the symbols /* and */.

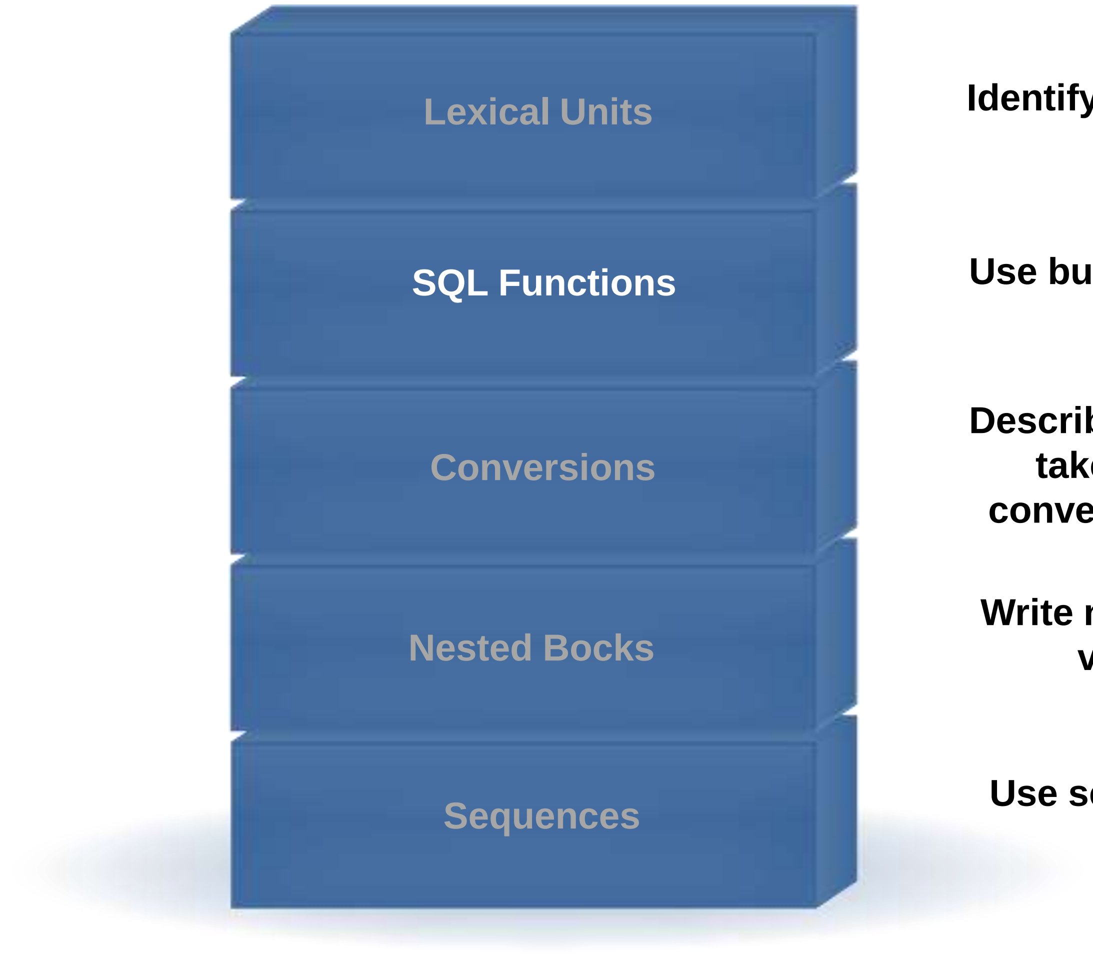
Example:

```
DECLARE
...
v_annual_sal  NUMBER (9,2) ;
BEGIN
/* Compute the annual salary based on the
   monthly salary input from the user */
v_annual_sal := monthly_sal * 12 ;
--The following line displays the annual salary
DBMS_OUTPUT.PUT_LINE (v_annual_sal) ;
END ;
/
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Plan



Identify lexical units in a PL/SQL block

Use built-in SQL functions in PL/SQL

Describe when implicit conversions take place and when explicit conversions have to be dealt with

Write nested blocks and qualify variables with labels

Use sequences in PL/SQL expressions

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

SQL Functions in PL/SQL

- Available in procedural statements:
  - Single-row functions
- Not available in procedural statements:
  - `DECODE`
  - Group functions

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

SQL Functions in PL/SQL: Examples

– Get the length of a string:

```
v_desc_size INTEGER (5) ;
v_prod_description VARCHAR2 (70) := 'You can use this product with your
radios for higher frequency' ;

-- get the length of the string in prod_description
v_desc_size := LENGTH (v_prod_description) ;
```

– Get the number of months an employee has worked:

```
v_tenure := MONTHS_BETWEEN (CURRENT_DATE, v_hiredate) ;
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Using Sequences in PL/SQL Expressions

Starting in 11*g*:

```
DECLARE
  v_new_id NUMBER ;
BEGIN
  v_new_id := my_seq.NEXTVAL;
END ;
/
```

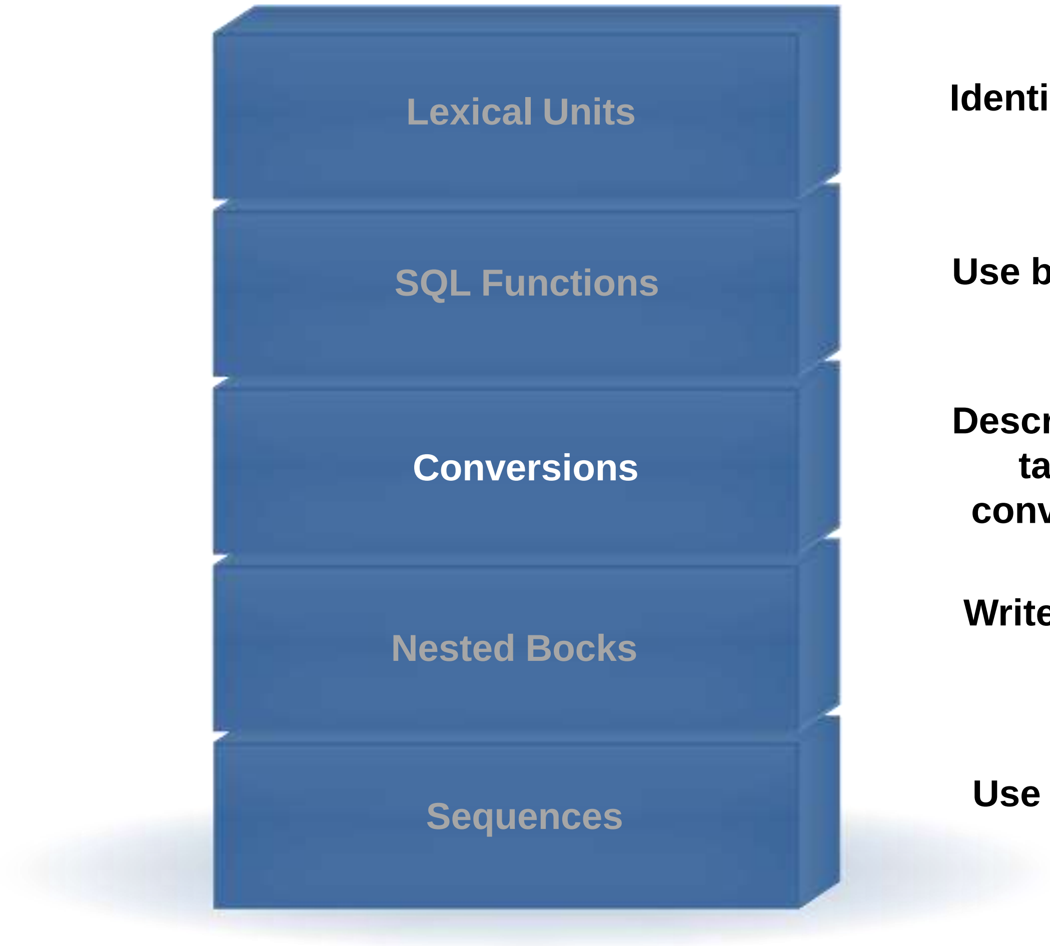
Before 11*g*:

```
DECLARE
  v_new_id NUMBER;
BEGIN
  SELECT my_seq.NEXTVAL INTO v_new_id FROM Dual;
END;
/
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Plan



Identify lexical units in a PL/SQL block

Use built-in SQL functions in PL/SQL

Describe when implicit conversions take place and when explicit conversions have to be dealt with

Write nested blocks and qualify variables with labels

Use sequences in PL/SQL expressions

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Data Type Conversion

- Converts data to comparable data types
- Is of two types:
  - Implicit conversion
  - Explicit conversion
- Functions:
  - `TO_CHAR`
  - `TO_DATE`
  - `TO_NUMBER`
  - `TO_TIMESTAMP`

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Data Type Conversion

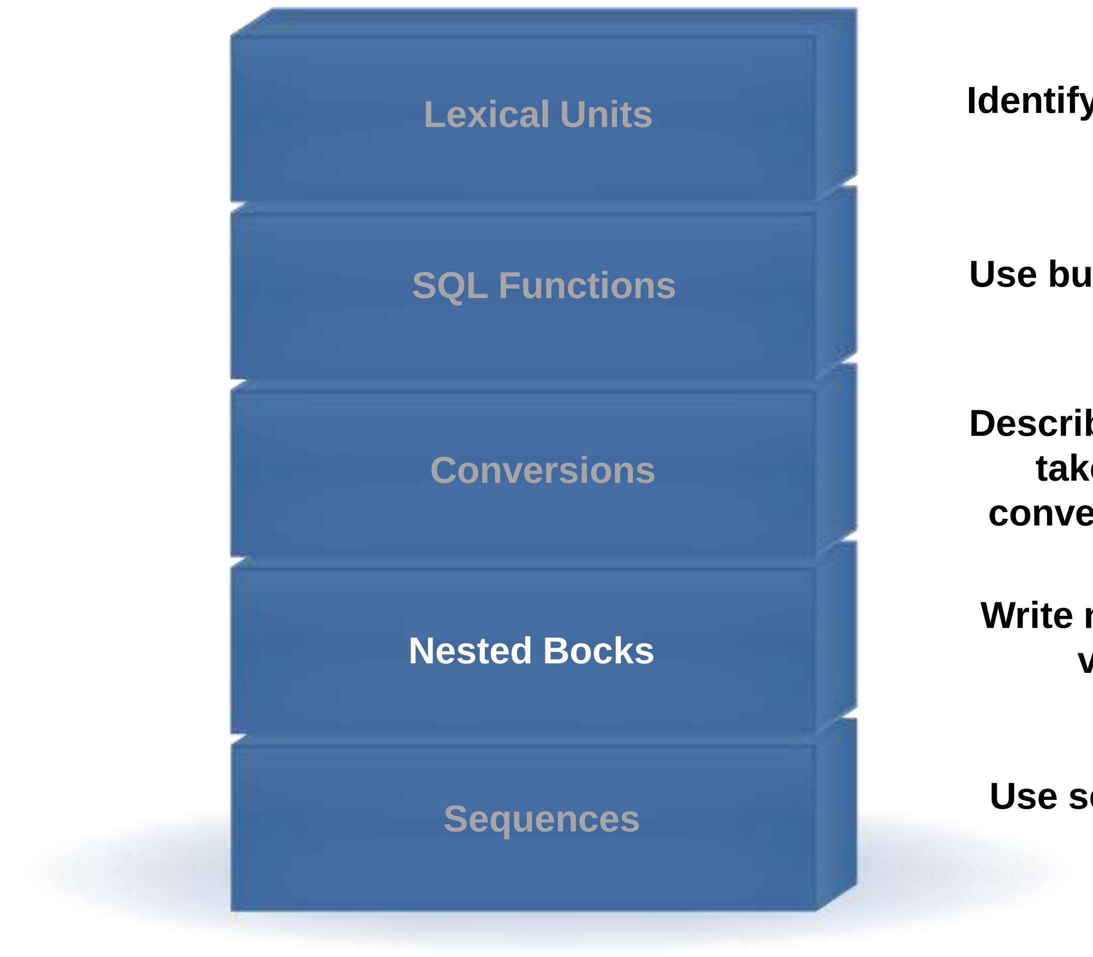
1
```
-- implicit data type conversion
v_date_of_joining DATE := '02-Feb-2000' ;
```

2
```
-- error in data type conversion
v_date_of_joining DATE :=  'February 02,2000‘ ;
```

3
```
-- explicit data type conversion
v_date_of_joining DATE := TO_DATE( 'February 02,2000'
Month DD, YYYY') ;
```

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Plan



Identify lexical units in a PL/SQL block

Use built-in SQL functions in PL/SQL

Describe when implicit conversions take place and when explicit conversions have to be dealt with

Write nested blocks and qualify variables with labels

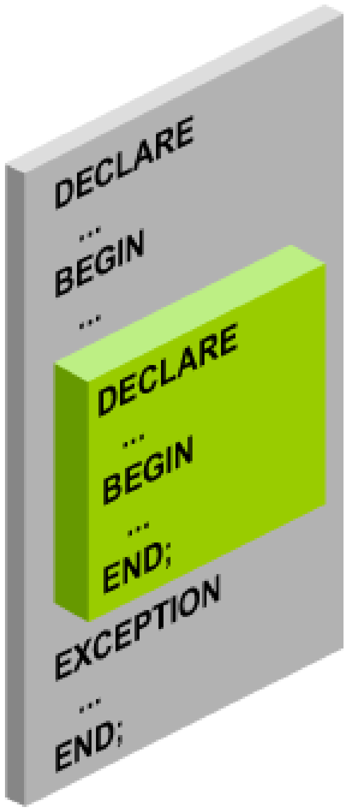
Use sequences in PL/SQL expressions

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Nested Blocks

PL/SQL blocks can be nested.

- An executable section (`BEGIN` … `END`) can contain nested blocks.
- An exception section can contain
  nested blocks.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Nested Blocks: Example

```
DECLARE
 v_outer_variable VARCHAR2(20):='GLOBAL VARIABLE';
BEGIN
  DECLARE
   v_inner_variable VARCHAR2(20):='LOCAL VARIABLE';
  BEGIN
   DBMS_OUTPUT.PUT_LINE(v_inner_variable);
   DBMS_OUTPUT.PUT_LINE(v_outer_variable);
  END;
 DBMS_OUTPUT.PUT_LINE(v_outer_variable);
END;
```

```
anonymous block completed
LOCAL VARIABLE
GLOBAL VARIABLE
GLOBAL VARIABLE
```

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Variable Scope and Visibility

```
DECLARE
 v_father_name VARCHAR2(20):='Patrick';
 v_date_of_birth DATE:='20-Apr-1972';
BEGIN
  DECLARE
   v_child_name VARCHAR2(20):='Mike';
   v_date_of_birth DATE:='12-Dec-2002';
  BEGIN
   DBMS_OUTPUT.PUT_LINE('Father''s Name: '||v_father_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
   DBMS_OUTPUT.PUT_LINE('Child''s Name: '||v_child_name);
  END;
 DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
END;
/
```

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Using a Qualifier with Nested Blocks

```
BEGIN <<outer>>
DECLARE
 v_father_name VARCHAR2(20):='Patrick';
 v_date_of_birth DATE:='20-Apr-1972';
BEGIN
  DECLARE
   v_child_name VARCHAR2(20):='Mike';
   v_date_of_birth DATE:='12-Dec-2002';
  BEGIN
   DBMS_OUTPUT.PUT_LINE('Father''s Name: '||v_father_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '
                        ||outer.v_date_of_birth);
   DBMS_OUTPUT.PUT_LINE('Child''s Name: '||v_child_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
  END;
END;
END outer;
```

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

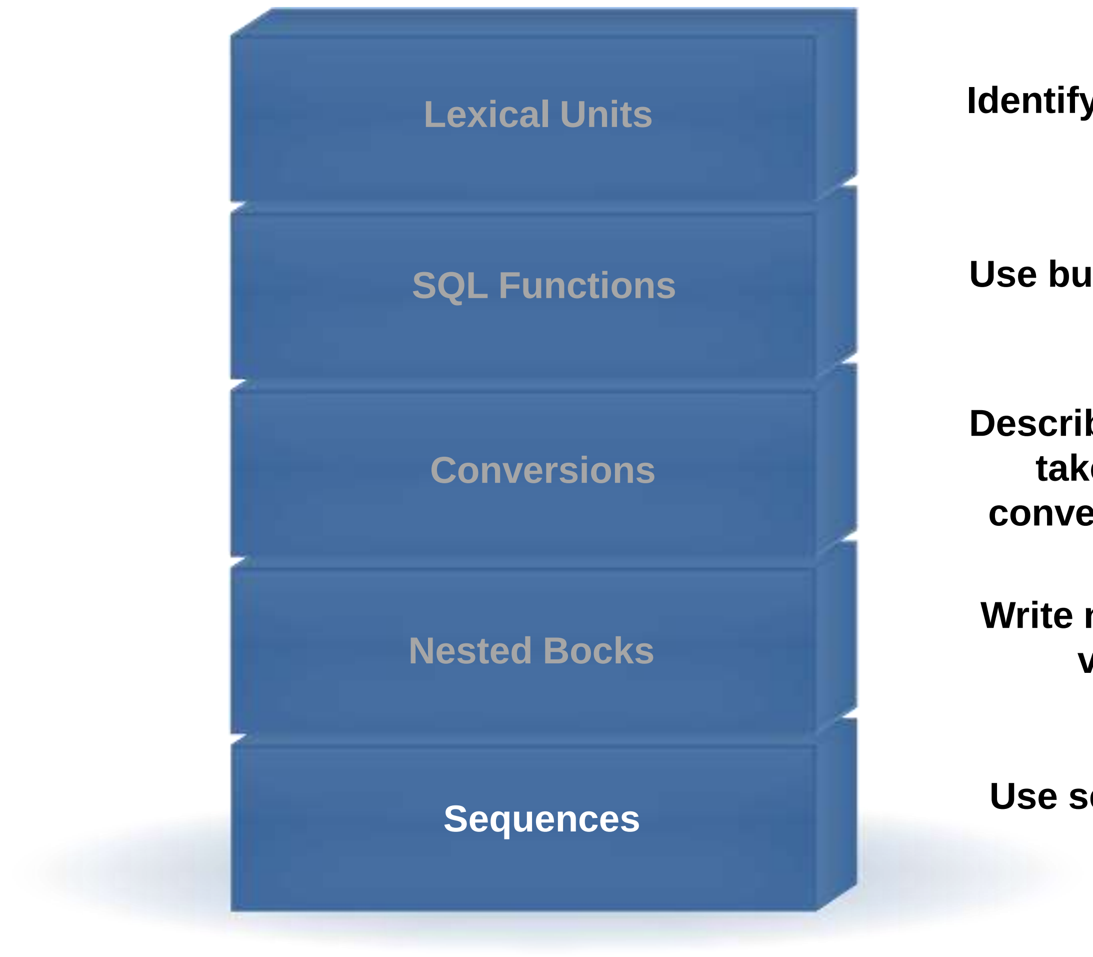
Challenge: Determining Variable Scope

```
  BEGIN <<outer>>
  DECLARE
    v_sal       NUMBER(7,2) := 60000;
    v_comm      NUMBER(7,2) := v_sal * 0.20;
    v_message   VARCHAR2(255) := ' eligible for commission';
  BEGIN
    DECLARE
          v_sal        NUMBER(7,2) := 50000;
          v_comm       NUMBER(7,2) := 0;
          v_total_comp  NUMBER(7,2) := v_sal + v_comm;
    BEGIN
1          v_message := 'CLERK not'||v_message;
          outer.v_comm := v_sal * 0.30;
    END;
2  v_message := 'SALESMAN'||v_message;
  END;
  END outer;
  /
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Plan



Identify lexical units in a PL/SQL block

Use built-in SQL functions in PL/SQL

Describe when implicit conversions take place and when explicit conversions have to be dealt with

Write nested blocks and qualify variables with labels

Use sequences in PL/SQL expressions

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Using Sequences in PL/SQL Expressions

- Starting in 11*g*:

```
DECLARE
  v_new_id NUMBER ;
BEGIN
  v_new_id := my_seq.NEXTVAL ;
END ;
/
```

- Before 11*g*:

```
DECLARE
  v_new_id NUMBER;
BEGIN
  SELECT my_seq.NEXTVAL INTO v_new_id FROM Dual;
END;
/
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Operators in PL/SQL

- Logical
- Arithmetic
- Concatenation
- Parentheses to control order of operations

Same as in SQL

- Exponential operator (**)

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Operators in PL/SQL: Examples

– Increment the counter for a loop.

```
loop_count  :=  loop_count + 1 ;
```

– Set the value of a Boolean flag.

```
good_sal  :=  sal BETWEEN 50000 AND 150000 ;
```

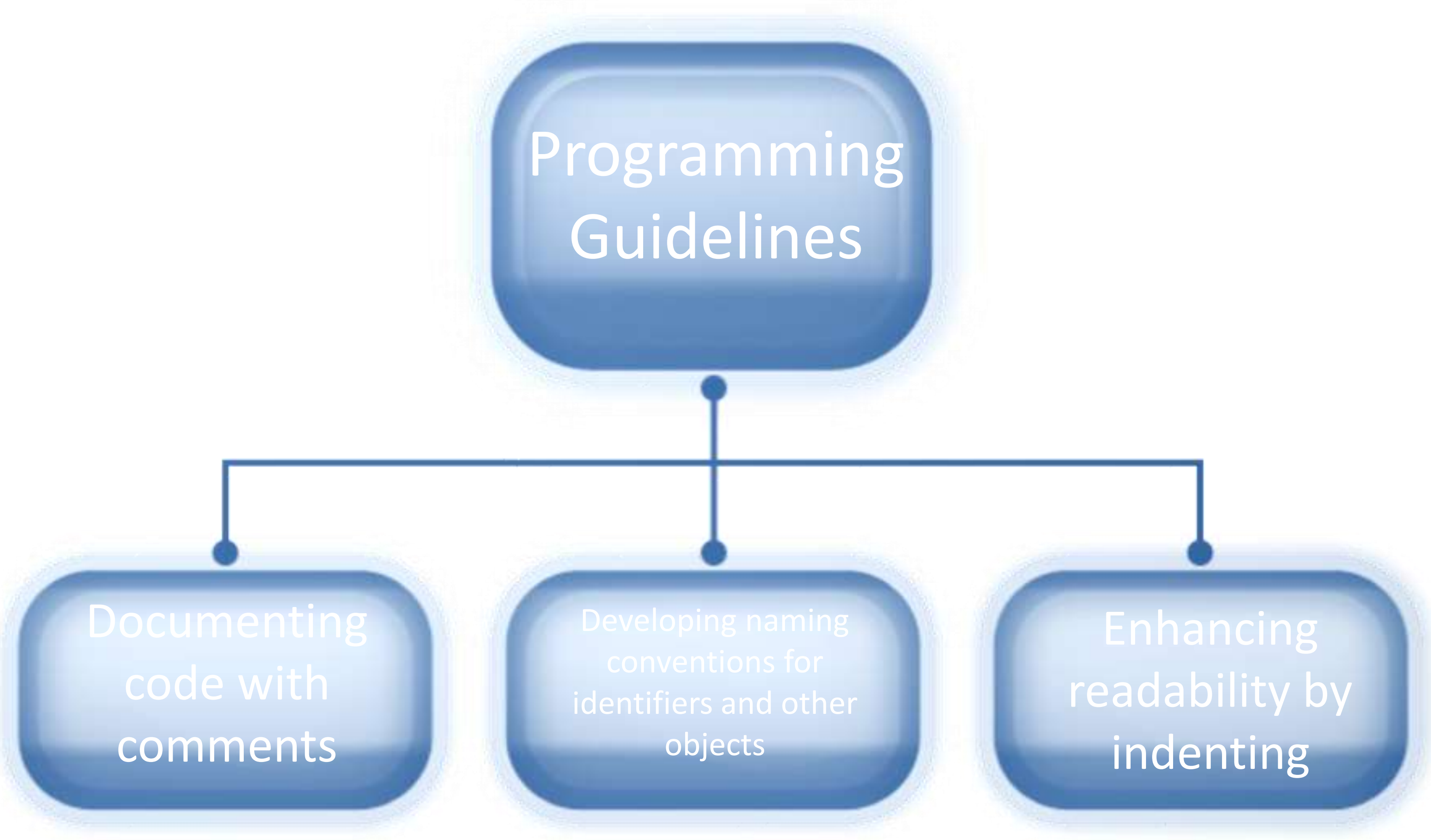
– Validate whether an employee number contains a value.

```
Valid := (empno IS NOT NULL) ;
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Programming Guidelines


Programming Guidelines
Documenting code with comments
Developing naming conventions for identifiers and other objects
Enhancing readability by indenting

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Indenting Code

For clarity, indent each level of code.

```
BEGIN
  IF x=0 THEN
    y := 1 ;
  END IF ;
END ;
/
```

```
DECLARE
   deptno       NUMBER(4) ;
   location_id  NUMBER(4) ;
BEGIN
   SELECT     department_id,
              location_id
   INTO       deptno,
              location_id
   FROM       departments
   WHERE      department_name = 'Sales' ;
...
END ;
/
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Quiz

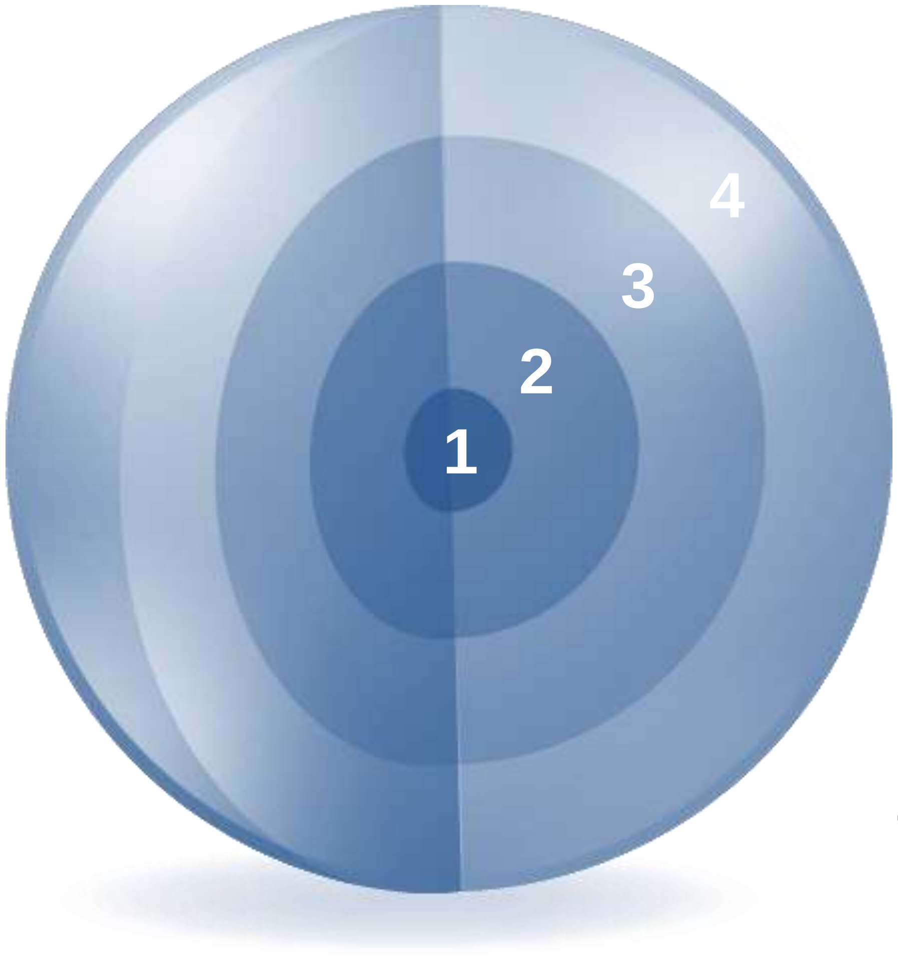
You can use most SQL single-row functions such as number, character, conversion, and date single-row functions in PL/SQL expressions.

a. True

b. False

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Session Summary



1. Identify lexical units in a PL/SQL block and use built-in functions

2. Decide when to perform explicit conversions

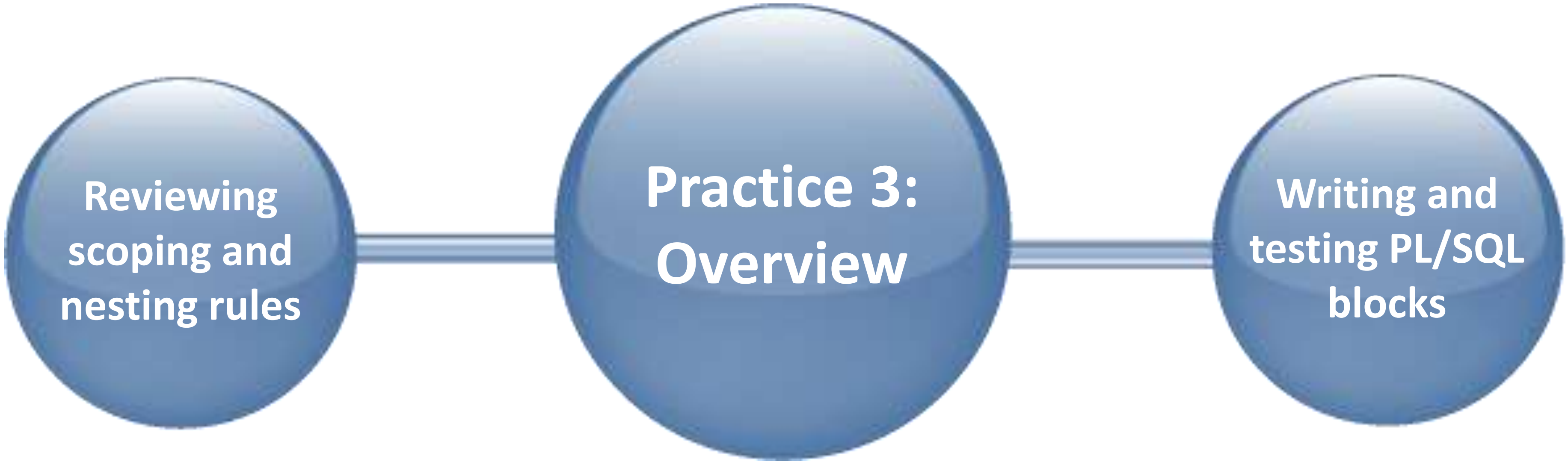
3. Qualify variables in nested blocks

4. Use sequences in PL/SQL expressions

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 3


Reviewing scoping and nesting rules
Practice 3: Overview
Writing and testing PL/SQL blocks

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.