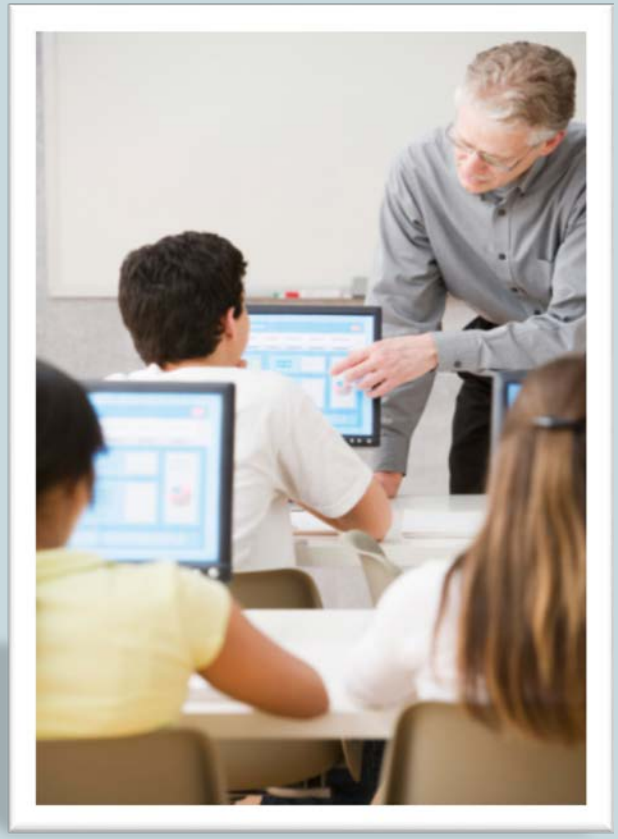

ORACLE
Academy

Database Programming with PL/SQL

2-2
Recognizing PL/SQL Lexical Units



ORACLE Academy

Objectives

This lesson covers the following objectives:

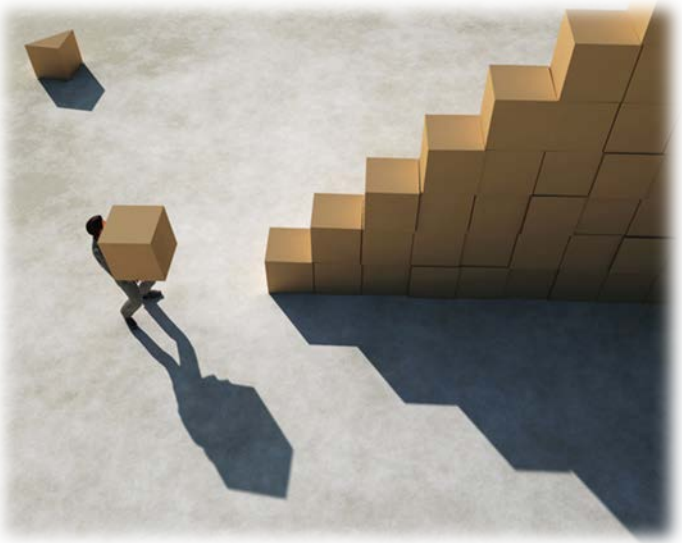
- List and define the different types of lexical units available in PL/SQL
- Describe identifiers and identify valid and invalid identifiers in PL/SQL
- Describe and identify reserved words, delimiters, literals, and comments in PL/SQL

Purpose

- A spoken language has different parts of speech.
- Each part of speech (such as an adjective, noun, and verb) is used differently and must follow rules.
- Similarly, a programming language has different parts of speech that are used differently and must follow rules.
- These parts of speech are called lexical units.

Lexical Units in a PL/SQL Block

- Lexical units:
- Are the building blocks of any PL/SQL block
- Are sequences of characters including letters, digits, tabs, returns, and symbols
- Can be classified as:
  - Identifiers
  - Reserved words
  - Delimiters
  - Literals
  - Comments



Identifiers

- An identifier is the name given to a PL/SQL object, including any of the following:

Procedure	**Function**	**Variable**
Exception	**Constant**	**Package**
Record	**PL/SQL table**	**Cursor**

- Do not be concerned if you do not know what all of the above objects are.
- You will learn about PL/SQL objects throughout this course.

Identifiers Highlighted

- Several identifiers are highlighted in the PL/SQL code shown below.

```
CREATE PROCEDURE print_date IS

    v_date VARCHAR2(30);

BEGIN

    SELECT TO_CHAR(SYSDATE,'Mon DD, YYYY')
        INTO v_date
        FROM DUAL;
    DBMS_OUTPUT.PUT_LINE(v_date);

END;
```

- Key: Variables Packages Procedures Functions

Identifier Properties

- Identifiers:
  - Maximum 30 characters in length
  - Must begin with a letter
  - May include $ (dollar sign), _ (underscore), or # (hashtag)
  - May not contain spaces
  - Identifiers are NOT case sensitive

- Be sure to name your objects carefully. Ideally, the identifier name should describe the object and its purpose. Avoid using identifier names such as A, X, Y1, temp, etc., because they make your code more difficult to read.

Valid and Invalid Identifiers

- Examples of valid identifiers:

First_Name	LastName	address_1
ID#	Total_$	primary_department_contact

- Examples of invalid identifiers:

First Name	Contains a space
Last-Name	Contains invalid symbol "-"
1st_address_line	Begins with a number
Total_%	Contains invalid symbol "%"
primary_building_department_contact	More than 30 characters

Reserved Words

- Reserved words are words that have special meaning to the Oracle database.
- Reserved words cannot be used as identifiers in a PL/SQL program.

Partial List of Reserved Words

- The following is a partial list of reserved words.

ALL	CREATE	FROM	MODIFY	SELECT
ALTER	DATE	GROUP	NOT	SYNONYM
AND	DEFAULT	HAVING	NULL	SYSDATE
ANY	DELETE	IN	NUMBER	TABLE
AS	DESC	INDEX	OR	THEN
ASC	DISTINCT	INSERT	ORDER	UPDATE
BETWEEN	DROP	INTEGER	RENAME	VALUES
CHAR	ELSE	INTO	ROW	VARCHAR2
COLUMN	EXISTS	IS	ROWID	VIEW
COMMENT	FOR	LIKE	ROWNUM	WHERE

- Note: For more information, refer to the “PL/SQL User’s Guide and Reference.”

Using Reserved Words

- What happens when you try to use a reserved word as an identifier in a PL/SQL program?

```
DECLARE
      date DATE;
BEGIN
      SELECT ADD_MONTHS(SYSDATE,3) INTO date
      FROM dual;
END;
```

```
ORA-06550: line 4, column 37:
PL/SQL: ORA-00936: missing expression
ORA-06550: line 4, column 3:
PL/SQL: SQL Statement ignored
2.          date DATE;
3. BEGIN
4.          SELECT ADD_MONTHS(SYSDATE,3) INTO date
5.          FROM DUAL;
6. END;
```

Delimiters

- Delimiters are symbols that have special meaning.
- Simple delimiters consist of one character.

Symbol	Meaning
+	addition operator
-	subtraction/negation operator
*	multiplication operator
/	division operator
=	equality operator
'	character string delimiter
;	statement terminator

Delimiters

- Compound delimiters consist of two characters.

Symbol	Meaning
<>	inequality operator
!=	inequality operator
||	concatenation operator
- -	single-line comment indicator
/*	beginning comment delimiter
*/	ending comment delimiter
**	exponent
:=	assignment operator

Literals

- A literal is an explicit numeric, character string, date, or Boolean value that might be stored in a variable.
- Literals are classified as:
  - Character (also known as string literals)
  - Numeric
  - Boolean

Character Literals

- May include any printable character in the PL/SQL character set: letters, numerals, spaces, and symbols
- Typically defined using the VARCHAR2 data type
- Must be enclosed by character string delimiters (')
- Can be composed of zero or more characters
- Are case sensitive; therefore, PL/SQL is NOT equivalent to pl/sql

Character Literals

- The following are examples of character literals being assigned to variables.
- The literals are the characters between the single quotes (the character string delimiters) and are shown here in red text for emphasis.

```
DECLARE
   v_firstname   VARCHAR2(30) := 'John';
   v_classroom   VARCHAR2(4) := '12C';
   v_course_id   VARCHAR2(8 := 'CS 101';
BEGIN
   …
```

Numeric Literals

- Literals that represent numbers are numeric literals.
- Numeric literals can be a simple value (ex. 5, -32.5, 127634, 3.141592)
- Scientific notation also may be used (ex. 2E5, meaning 2*(10 to the power of 5).
- Typically defined using the NUMBER data type

Numeric Literals

- The following are examples of numeric literals being assigned to variables (and one constant).
- The literals are shown here in red text for emphasis.

```
DECLARE
   v_classroom   NUMBER(3) := 327;
   v_grade       NUMBER(3) := 95;
   v_price       NUMBER(5) := 150;
   v_salary      NUMBER(8) := 2E5;
   c_pi          CONSTANT NUMBER(7,6) := 3.141592;
BEGIN
   …
```

Boolean Literals

- Values that are assigned to Boolean variables are Boolean literals.
- TRUE, FALSE, and NULL are the Boolean literals.

```
DECLARE
   v_new_customer   BOOLEAN := FALSE;
   v_fee_paid       BOOLEAN := TRUE;
   v_diploma        BOOLEAN := NULL;
BEGIN
   …
```

- Note that character string delimiters are not required.

Comments

- Comments explain what a piece of code is trying to achieve.
- Well-placed comments are extremely valuable for code readability and future code maintenance.
- Comments should describe the purpose and use of each block of code.
- It is good programming practice to comment code.
- Comments are ignored by PL/SQL.
- They make no difference to how a PL/SQL block executes or the results it displays.

Syntax for Commenting Code

- Two ways to indicate comments in PL/SQL
- When commenting a single line, use two dashes (--)
- When commenting multiple lines, begin the comment with /* and end the comment with */

```
DECLARE
-- converts monthly salary to annual salary
   v_montly_sal NUMBER(9,2);
   v_annual_sal NUMBER(9,2);
BEGIN      -- begin executable section
   …
/* Compute the annual salary based on the
  monthly salary input from the user */
   v_annual_sal := v_monthly_sal * 12;
END;       -- end block
```

Terminology

Key terms used in this lesson included:

- Lexical units
- Identifiers
- Reserved words
- Delimiters
- Literals
- Comments

Summary

In this lesson, you should have learned how to:

- List and define the different types of lexical units available in PL/SQL
- Describe identifiers and identify valid and invalid identifiers in PL/SQL
- Describe and identify reserved words, delimiters, literals, and comments in PL/SQL


ORACLE
Academy