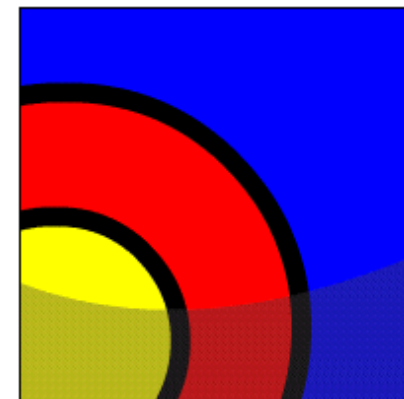


# Recognizing PL/SQL Lexical Units

## What Will I Learn?

In this lesson, you will learn 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



## Why Learn It?

A spoken language has different parts of speech. Each part of speech (such as 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.

This lesson explores the lexical units of the PL/SQL programming language.

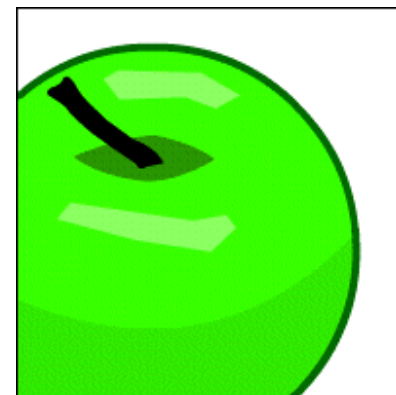


## Tell Me/Show Me

### 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





# Tell Me/Show Me

## 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.)



# Tell Me/Show Me

## Identifiers (continued)

The identifiers in the following PL/SQL code are highlighted::

```
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:  Procedure     Variable     Reserved word



# Tell Me/Show Me

## Properties of an Identifier

Identifiers:

- Are up to 30 characters in length
- Must start with a letter
- Can include \$ (dollar sign), \_ (underscore), and # (pound sign/hash sign)
- Cannot contain spaces
- All identifiers (variables) are case insensitive



# Tell Me/Show Me

## 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 "-"
1st_address_line	Begins with a number
Total_%	Contains invalid "%"
primary_building_department_contact	More than 30 characters





# Tell Me/Show Me

## 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.



# Tell Me/Show Me

## Reserved Words (continued)

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*.”



# Tell Me/Show Me

## Reserved Words (continued)

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;
```



# Tell Me/Show Me

## Delimiters

Delimiters are symbols that have special meaning to the Oracle database.

### Simple delimiters

Symbol	Meaning
+	Addition operator
–	Subtraction/negation operator
*	Multiplication operator
/	Division operator
=	Equality operator
'	Character string delimiter
;	Statement terminator

### Compound delimiters

Symbol	Meaning
<>	Inequality operator
!=	Inequality operator
	Concatenation operator
--	Single-line comment indicator
/*	Beginning comment delimiter
*/	Ending comment delimiter
:=	Assignment operator



# Tell Me/Show Me

## Literals

A literal is an explicit numeric, character string, date, or Boolean value that is not represented by an identifier.

Literals are classified as:

- Character (also known as string literals)
- Numeric
- Boolean



# Tell Me/Show Me

## Character Literals

- Character literals include all the printable characters in the PL/SQL character set: letters, numerals, spaces, and special symbols.
- Character literals have the data type `CHAR` and must be enclosed in single quotation marks.
- Character literals can be composed of zero or more characters from the PL/SQL character set.
- Character literals are case sensitive and, therefore, `PL/SQL` is not equivalent to `pl/sql`.

```
v_first_name := 'John';  
v_classroom  := '12C';  
v_date_today := '20-MAY-2005';
```



# Tell Me/Show Me

## Numeric Literals

- Values that represent an integer or real value are numeric literals
- You can represent numeric literals either by a simple value (for example, `-32.5`) or by a scientific notation (for example, `2E5`, meaning  $2 * (10 \text{ to the power of } 5) = 200000$ ).
- Examples: `428`, `1.276`, `2.09E14`

```
v_elevation      := 428;  
v_order_subtotal := 1025.69;  
v_growth_rate    := .56;  
v_distance_sun_to_centauri := 4.3E13;
```



# Tell Me/Show Me

## Boolean Literals

- Values that are assigned to Boolean variables are Boolean literals. They are not surrounded by quotes.
- TRUE, FALSE, and NULL are Boolean literals or keywords.

```
v_new_customer      := FALSE;  
v_paid_in_full      := TRUE;  
v_authorization_approved := FALSE;  
v_high_school_diploma := NULL;  
v_island            := FALSE;
```





# Tell Me/Show Me

## 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. 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.



## Tell Me/Show Me

### Syntax for Commenting Code

Commenting a single line:

- Two dashes -- are used for commenting a single line.

Commenting multiple lines:

- /\* \*/ is used for commenting multiple lines.

```
DECLARE
...
    v_annual_sal NUMBER (9,2);
BEGIN    -- Begin the executable section

/* Compute the annual salary based on the
    monthly salary input from the user */
    v_annual_sal := v_monthly_sal * 12;
END;    -- This is the end of the block
```

# Tell Me / Show Me

## Terminology

Key terms used in this lesson include:

Lexical units

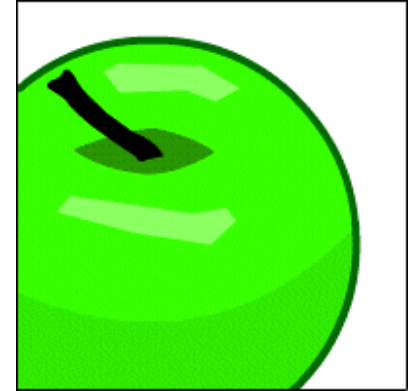
Identifiers

Reserved words

Delimiters

Literals

Comments

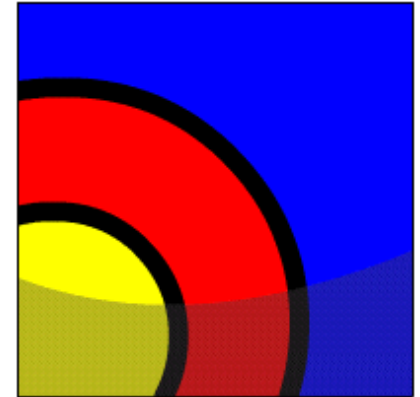




# Summary

In this lesson, you 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





## Try It/Solve It

The exercises for this lesson cover the following topics:

- Listing and defining the different types of lexical units available in PL/SQL
- Describing identifiers and identifying valid and invalid identifiers in PL/SQL
- Describing and identifying reserved words, delimiters, literals, and comments in PL/SQL

