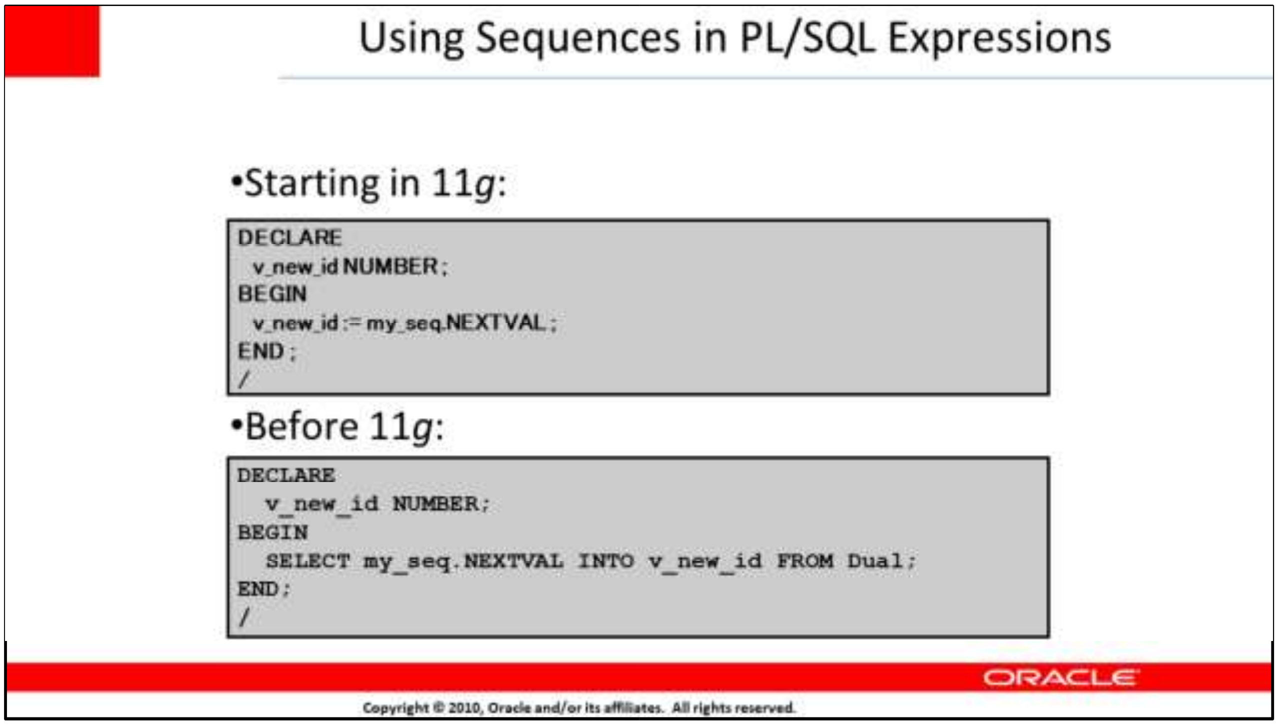


Using Sequences in PL/SQL Expressions

- 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;
/
```

Accessing Sequence Values

In Oracle Database 11g, you can use the NEXTVAL and CURRVAL pseudocolumns in any PL/SQL context, where an expression of the NUMBER data type may legally appear. Although the old style of using a SELECT statement to query a sequence is still valid, it is recommended that you do not use it.

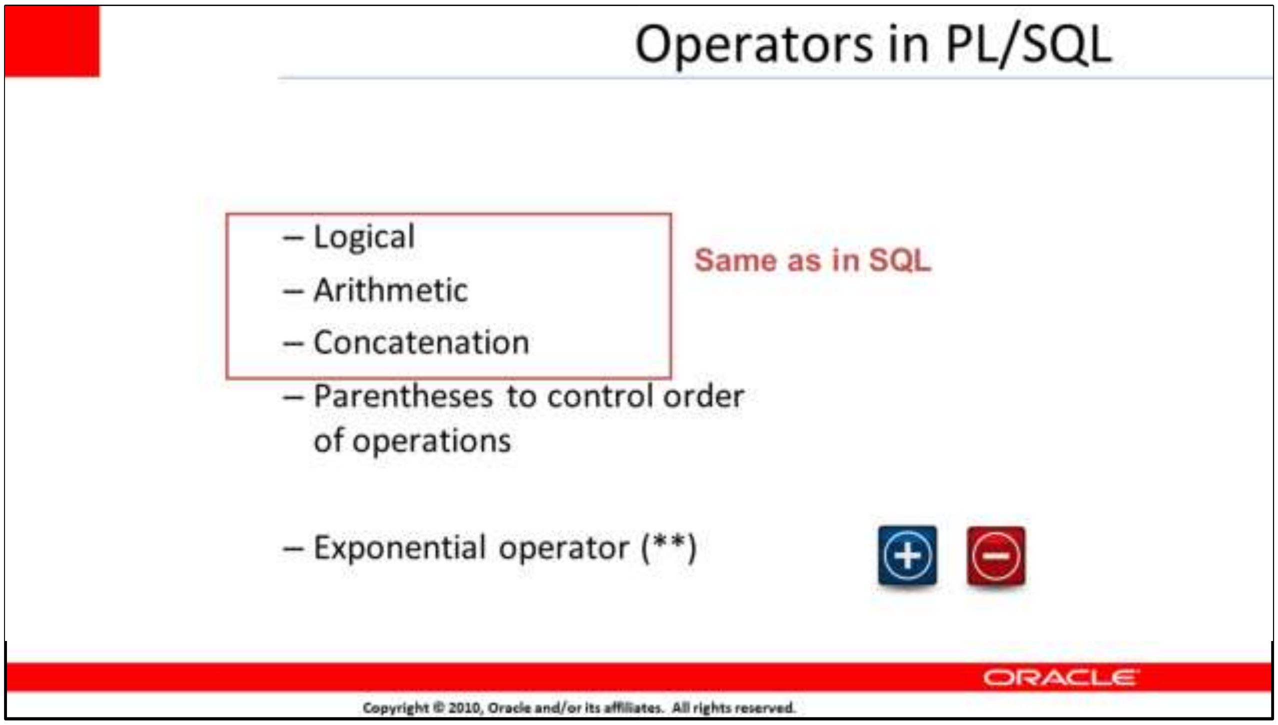
Before Oracle Database 11g, you were forced to write a SQL statement in order to use a sequence object value in a PL/SQL subroutine. Typically, you would write a SELECT statement to reference the pseudocolumns of NEXTVAL and CURRVAL to obtain a sequence number. This method created a usability problem.

In Oracle Database 11g, the limitation of forcing you to write a SQL statement to retrieve a sequence value is eliminated. With the sequence enhancement feature:

Sequence usability is improved

The developer has to type less

The resulting code is clearer



Operators in PL/SQL

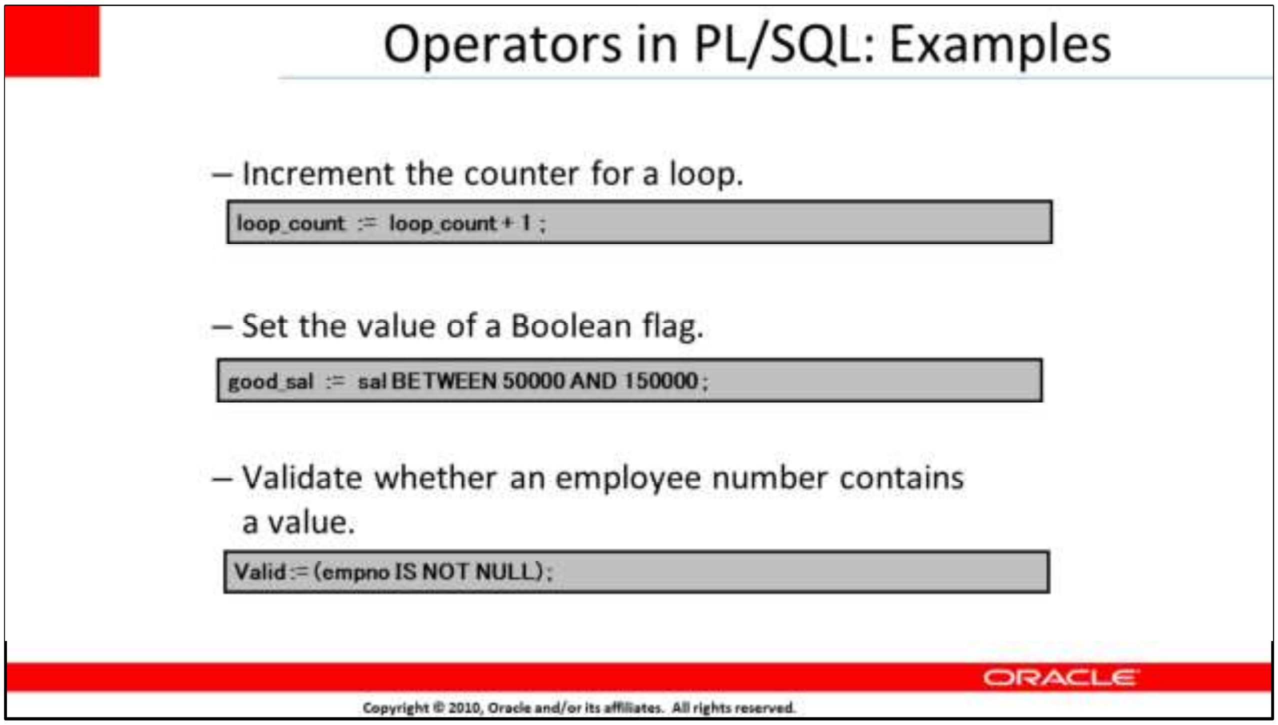
- Logical
- Arithmetic
- Concatenation
- Parentheses to control order of operations
- Exponential operator (**)

Same as in SQL

Operators in PL/SQL

The operations in an expression are performed in a particular order depending on their precedence (priority). The following table shows the default order of operations from high priority to low priority:

Operator	Operation
**	Exponentiation
+, -	Identity, negation
*, /	Multiplication, division
+, -, ||	Addition, subtraction, concatenation
=, <, >, <=, >=, <>, !=, ~=, ^=, IS NULL, LIKE, BETWEEN, IN	Comparison
NOT	Logical negation
AND	Conjunction
OR	Inclusion



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);
```

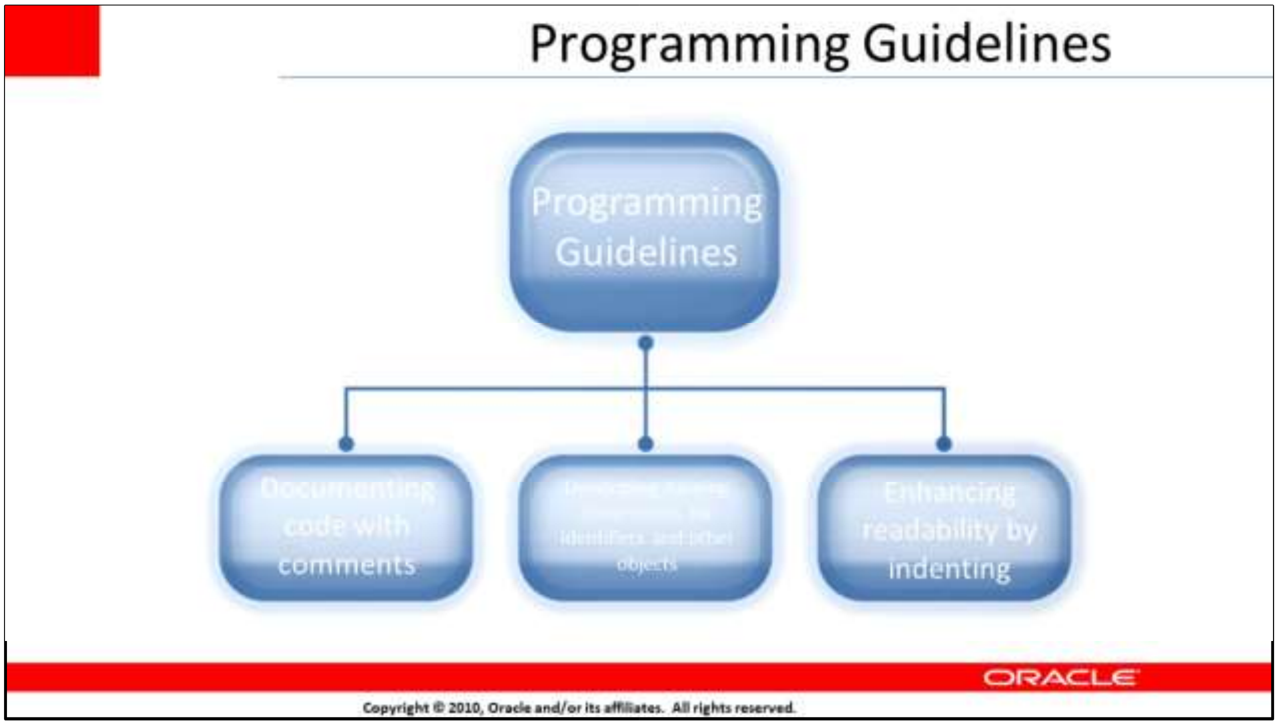
Operators in PL/SQL (continued)

When you are working with nulls, you can avoid some common mistakes by keeping in mind the following rules:

Comparisons involving nulls always yield NULL.

Applying the logical operator NOT to a null yields NULL.

In conditional control statements, if the condition yields NULL, its associated sequence of statements is not executed.



Programming Guidelines

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

Oracle Database: PL/SQL Fundamentals

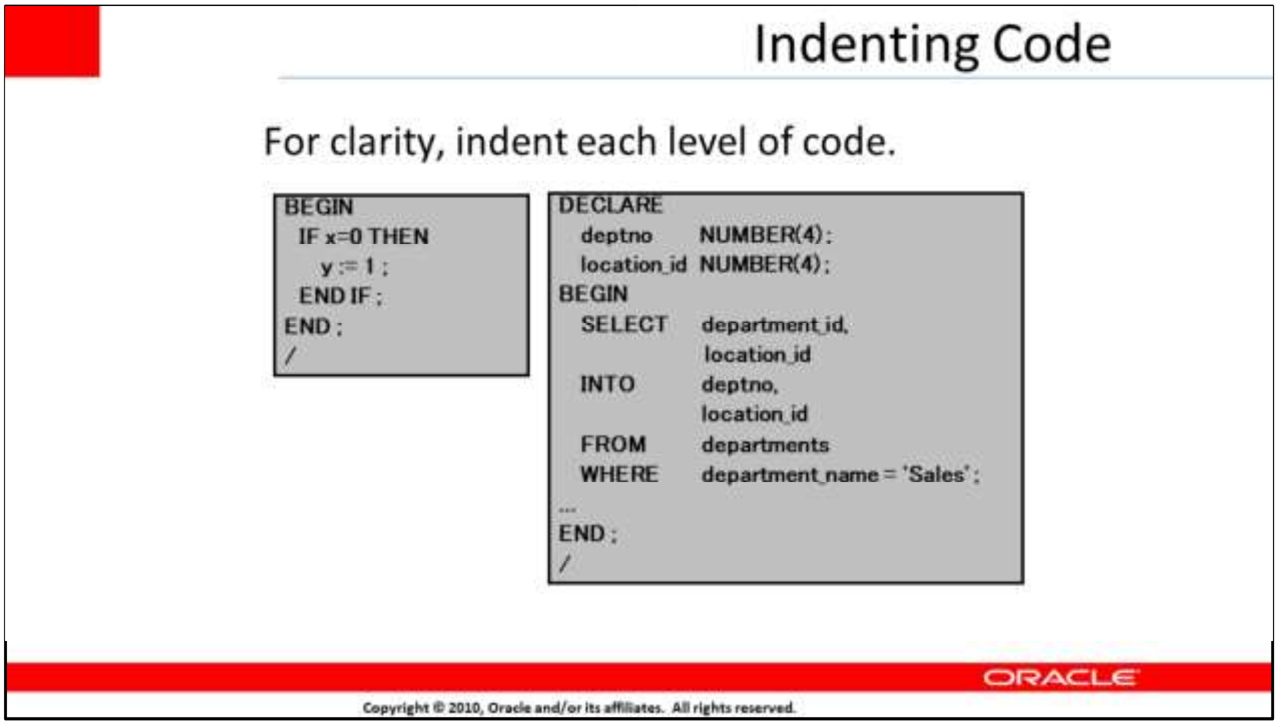
Programming Guidelines

Follow programming guidelines shown in the slide to produce clear code and reduce maintenance when developing a PL/SQL block.

Code Conventions

The following table provides guidelines for writing code in uppercase or lowercase characters to help distinguish keywords from named objects.

Category	Case Convention	Examples
SQL statements	Uppercase	SELECT, INSERT
PL/SQL keywords	Uppercase	DECLARE, BEGIN, IF
Data types	Uppercase	VARCHAR2, BOOLEAN


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 ;
/
```

Identifiers and parameters	Lowercase	v_sal, emp_cursor, g_sal, p_empno
Database tables	Lowercase, plural	employees, departments
Database columns	Lowercase, singular	employee_id, department_id

Indenting Code

For clarity and enhanced readability, indent each level of code. To show structure, you can divide lines by using carriage returns and you can indent lines by using spaces and tabs. Compare the following IF statements for readability:

```
IF x>y THEN max:=x;ELSE max:=y;END IF;


IF x > y THEN
  max := x;
ELSE
  max :=
  y;
END IF;
```