

# PL/SQL - BASIC SYNTAX

[http://www.tutorialspoint.com/plsql/plsql\\_basic\\_syntax.htm](http://www.tutorialspoint.com/plsql/plsql_basic_syntax.htm)

Copyright © tutorialspoint.com

PL/SQL is a block-structured language, meaning that PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts:

S.N.	Sections & Description
1	<b>Declarations</b> This section starts with the keyword <b>DECLARE</b> . It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.
2	<b>Executable Commands</b> This section is enclosed between the keywords <b>BEGIN</b> and <b>END</b> and it is a mandatory section. It consists of the executable PL/SQL statements of the program. It should have at least one executable line of code, which may be just a NULL command to indicate that nothing should be executed.
3	<b>Exception Handling</b> This section starts with the keyword <b>EXCEPTION</b> . This section is again optional and contains exception(s) that handle errors in the program.

Every PL/SQL statement end with a semicolon (;). PL/SQL blocks can be nested within other PL/SQL blocks using **BEGIN** and **END**. Here is the basic structure of a PL/SQL block:

```
DECLARE
    <declarations section>
BEGIN
    <executable command(s)>
EXCEPTION
    <exception handling>
END;
```

## The 'Hello World' Example:

```
DECLARE
    message varchar2(20) := 'Hello, World!';
BEGIN
    dbms_output.put_line(message);
END;
/
```

The **end;** line signals the end of the PL/SQL block. To run the code from SQL command line, you may need to type **/** at the beginning of the first blank line after the last line of the code. When the above code is executed at SQL prompt, it produces following result:

```
Hello World

PL/SQL procedure successfully completed.
```

## The PL/SQL Identifiers

PL/SQL identifiers are constants, variables, exceptions, procedures, cursors, and reserved words. The identifiers consist of a letter optionally followed by more letters, numerals, dollar signs, underscores, and number signs and should not

exceed 30 characters.

By default, **identifiers are not case-sensitive**. So you can use **integer** or **INTEGER** to represent a numeric value. You cannot use a reserved keyword as an identifier.

## The PL/SQL Delimiters

A delimiter is a symbol with a special meaning. Following is the list of delimiters in PL/SQL:

Delimiter	Description
+, -, *, /	Addition, subtraction/negation, multiplication, division
%	Attribute indicator
'	Character string delimiter
.	Component selector
(,)	Expression or list delimiter
:	Host variable indicator
,	Item separator
"	Quoted identifier delimiter
=	Relational operator
@	Remote access indicator
;	Statement terminator
:=	Assignment operator
=>	Association operator
	Concatenation operator
**	Exponentiation operator
<<, >>	Label delimiter (begin and end)
/*, */	Multi-line comment delimiter (begin and end)
--	Single-line comment indicator
..	Range operator
<, >, <=, >=	Relational operators
<>, !=, ~=, ^=	Different versions of NOT EQUAL

## The PL/SQL Comments

Program comments are explanatory statements that you can include in the PL/SQL code that you write and helps anyone reading it's source code. All programming languages allow for some form of comments.

The PL/SQL supports single line and multi-line comments. All characters available inside any comment are ignored by PL/SQL compiler. The PL/SQL single-line comments start with the delimiter -- (double hyphen) and multi-line comments are enclosed by /\* and \*/.

```
DECLARE
  -- variable declaration
  message varchar2(20) := 'Hello, World!';
BEGIN
  /*
   * PL/SQL executable statement(s)
   */
  dbms_output.put_line(message);
END;
/
```

When the above code is executed at SQL prompt, it produces following result:

```
Hello World

PL/SQL procedure successfully completed.
```

## PL/SQL Program Units

A PL/SQL unit is any one of the following:

- PL/SQL block
- Function
- Package
- Package body
- Procedure
- Trigger
- Type
- Type body

Each of these units will be discussed in the forthcoming chapters.