Parts of a PL/SQL Subprogram/ Procedures

Each PL/SQL subprogram has a name, and may also have a parameter list. Like anonymous PL/SQL blocks, the named blocks will also have the following three parts −

|  |  |
| --- | --- |
| **S.No** | **Parts & Description** |
| 1 | **Declarative Part**  It is an optional part. However, the declarative part for a subprogram does not start with the DECLARE keyword. It contains declarations of types, cursors, constants, variables, exceptions, and nested subprograms. These items are local to the subprogram and cease to exist when the subprogram completes execution. |
| 2 | **Executable Part**  This is a mandatory part and contains statements that perform the designated action. |
| 3 | **Exception-handling**  This is again an optional part. It contains the code that handles run-time errors. |

CREATE [OR REPLACE] PROCEDURE procedure\_name

[(parameter\_name [IN | OUT | IN OUT] type [, ...])]

{IS | AS}

BEGIN

< procedure\_body >

END procedure\_name;

CREATE OR REPLACE PROCEDURE greetings

AS

BEGIN

dbms\_output.put\_line('Hello World!');

END;

/

## Executing a Standalone Procedure

A standalone procedure can be called in two ways −

* Using the **EXECUTE** keyword
* Calling the name of the procedure from a PL/SQL block

The above procedure named **'greetings'** can be called with the EXECUTE keyword as −

EXECUTE greetings;

## Deleting a Standalone Procedure

A standalone procedure is deleted with the **DROP PROCEDURE** statement. Syntax for deleting a procedure is −

DROP PROCEDURE procedure-name;

## Parameter Modes in PL/SQL Subprograms

The following table lists out the parameter modes in PL/SQL subprograms −

|  |  |
| --- | --- |
| **S.No** | **Parameter Mode & Description** |
| 1 | **IN**  An IN parameter lets you pass a value to the subprogram. **It is a read-only parameter**. Inside the subprogram, an IN parameter acts like a constant. It cannot be assigned a value. You can pass a constant, literal, initialized variable, or expression as an IN parameter. You can also initialize it to a default value; however, in that case, it is omitted from the subprogram call. **It is the default mode of parameter passing. Parameters are passed by reference**. |
| 2 | **OUT**  An OUT parameter returns a value to the calling program. Inside the subprogram, an OUT parameter acts like a variable. You can change its value and reference the value after assigning it. **The actual parameter must be variable and it is passed by value**. |
| 3 | **IN OUT**  An **IN OUT** parameter passes an initial value to a subprogram and returns an updated value to the caller. It can be assigned a value and the value can be read.  The actual parameter corresponding to an IN OUT formal parameter must be a variable, not a constant or an expression. Formal parameter must be assigned a value. **Actual parameter is passed by value.** |

DECLARE

a number;

b number;

c number;

PROCEDURE findMin(x IN number, y IN number, z OUT number) IS

BEGIN

IF x < y THEN

z:= x;

ELSE

z:= y;

END IF;

END;

BEGIN

a:= 23;

b:= 45;

findMin(a, b, c);

dbms\_output.put\_line(' Minimum of (23, 45) : ' || c);

END;

/

DECLARE

a number;

PROCEDURE squareNum(x IN OUT number) IS

BEGIN

x := x \* x;

END;

BEGIN

a:= 23;

squareNum(a);

dbms\_output.put\_line(' Square of (23): ' || a);

END;

/