

Function	Packages (cont)	Conditional and Loops (cont)	Conditional and Loops (cont)
<pre>CREATE OR REPLACE FUNCTION function_name (param eter_1 data_type, Param eter_2 data_type) RETURN data_type { IS AS } [decla rat ion _se - ction] BEGIN execut abl e_s ection [EXCEPTION except ion _se ction] END [funct ion _name];</pre>	<pre>> procedure_or_function_speci- fication_2; END [package_name]; Package body CREATE PACKAGE BODY package_name { IS AS } procedure_or_function_body_1; procedure_or_function_body_2; END [package_name];</pre>	<pre>> dbms_output.put_line(myvar); DECLARE myvar number; BEGIN myvar:=2; dbms_output.put_line(myvar); END; dbms_output.put_line(myvar); END; IF THEN ELSE END IF DECLARE v_number NUMBER; BEGIN IF v_number<=0 THEN dbms_output.put_line('it is less than 0'); ELSIF v_number>=0 THEN dbms_output.put_line('it is greater than 0'); ELSE dbms_output.put_line('not either of the case'); END IF; END; Loops FOR IN .. LOOP {statements}; END LOOP; WHILE condition LOOP</pre>	<pre>> {statements}; END LOOP; LOOP {statemens}; EXIT WHEN condition; CONTINUE WHEN condition; END LOOP; Loops DECLARE i NUMBER :=10; BEGIN FOR i IN 1..5 LOOP dbms_output.put_line(i); END LOOP; dbms_output.put_line(i); END; CASE – Simple Case CASE expression WHEN value_1 THEN .. WHEN value_2 THEN ELSE END CASE; CASE – Searched Case WHEN boolean_expression THEN ELSE END CASE;</pre>
Procedures	Bind variable Need to specify type Need to wrap around quote when assign string value No need quote when reference the variable Value can only be assigned in a PL, via exec or Begin / End block Use PRINT to list out bind variable	Conditional and Loops Declare and use of variable <pre>%TYPE %ROWTYPE VARCHAR2 NUMBER DATE Assignment operator := Nested block variable scope DECLARE myvar number; BEGIN myvar:=1;</pre>	
Packages			
<pre>CREATE PACKAGE package_name { IS AS } proced ure _or _fu nct - ion _sp eci fic ati - on_1;</pre>			



By **juliosueiras**

Published 17th December, 2015.

Last updated 12th May, 2016.

Page 1 of 3.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Function vs Procedures

Function must return a value.
Procedure can not return a value

Function and procedure can both return data in OUT and IN OUT parameters

Function can be called from SQL, but not for procedure

Can not perform a DML DDL within function, while allowed in procedure

Substitution variable

No need to specify type, as it is always character type

No need to wrap around quote when assign value

Need quote when reference the variable

ACCEPT implicitly defined a substitution type variable

Use DEFINE to list out substitution variable

Procedures Parts (cont)

3 **Exception-handling** This is again an optional part. It contains the code that handles run-time errors.

Parameter Modes in PL/SQL Subprograms

S.N.	Parts & 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.

Parameter Modes in PL/SQL Subprograms (cont)

- 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 its 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.

Trigger

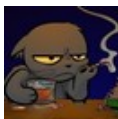
```
CREATE [OR REPLACE]
TRIGGER trigger_name
BEFORE | AFTER
[INSERT, UPDATE, DELETE
[COLUMN NAME...]]
ON table_name
Refere ncing [ OLD AS
OLD | NEW AS NEW ]
FOR EACH ROW | FOR EACH
STATEMENT [ WHEN
Condition ]
DECLARE
[decla rat ion _se -
ction]
BEGIN
[execu tab le _sec tion]
EXCEPTION
[excep tio n _s ection]
END;
```

Procedures Parts

S.N.	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.

Packages Code Example

```
CREATE OR REPLACE
PACKAGE roppkg AS
    PRO CEDURE ropmall
(pi_city varchar2
default 'Missi ssa uga',
```



By **juliosueiras**

Published 17th December, 2015.

Last updated 12th May, 2016.

Page 2 of 3.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Packages Code Example (cont)

```
> pi_mall varchar2,
pi_city_code out varchar2) ;
FUNCTION roppop
(pi_city varchar2 default'Mississ-
auga')
RETURN NUMBER ;
END;
CREATE OR REPLACE
PACKAGE BODY roppkg AS
PROCEDURE ropmall
(pi_city varchar2 default 'Missi ssa uga',
pi_mall varchar2,
pi_city_code out varchar2)
AS
l_cnt NUMBER;
l_cid number;
BEGIN
SELECT count(1) INTO l_cnt
from
mall a
WHERE
a.mall_name=pi_mall
;
IF l_cnt = 0
THEN
SELECT cid into l_cid
FROM rop
WHERE CITY=pi_city;

INSERT INTO mall VALUES
(l_cid, pi_mall);
```

Packages Code Example (cont)

```
> END IF;
COMMIT;
pi_city_code:=l_cid;
END;
FUNCTION roppop
(pi_city varchar2 default'Mississ-
auga')
RETURN NUMBER AS
l_pop NUMBER;
BEGIN
SELECT population INTO l_pop
from
rop WHERE city=pi_city;
RETURN l_pop;
END;
END;
```

Function Example

```
CREATE or REPLACE
FUNCTION roppop
(pi_city varchar2 )
RETURN NUMBER AS
l_pop NUMBER;
BEGIN
SELECT population INTO
l_pop from
rop WHERE city=p -
i_city;
RETURN l_pop;
END;
```

Procedures Example

```
CREATE or REPLACE
PROCEDURE ropmall
(pi_city varchar2
default 'Missi ssa uga',
pi_mall varchar2,
pi_city_code out
varchar2)
AS
l_cnt NUMBER;
l_cid number;
BEGIN
dbms_output.put_line( nvl (pi_ci -
ty_code, ' NUL L'));

SELECT count(1) INTO
l_cnt from
mall a
WHERE
a.mall_name= -
pi_mall
;
IF l_cnt = 0
THEN
SELECT cid into
l_cid
FROM rop
WHERE CITY=p -
i_city;

INSERT INTO mall
VALUES (l_cid, pi_mall);
END IF;
COMMIT;
pi_city_code :=l -
_cid;
END;
```



By julio sueiras

Published 17th December, 2015.

Last updated 12th May, 2016.

Page 3 of 3.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>