CHULETA PL/SQL from http://www.devjoker.com/gru/tutorial-PL-SQL/PLSQ/Tutorial-PL-SQL.aspx

```
IF (expresion) THEN
    -- Instrucciones
ELSIF (expresion) THEN
    -- Instrucciones
ELSE
    -- Instrucciones
END IF;
```

```
WHILE (expresion) LOOP
    -- Instrucciones
END LOOP;
```

```
FOR contador IN [REVERSE] inicio..final LOOP
    -- Instrucciones
END LOOP;
```

```
RAISE_APPLICATION_ERROR(<error_num>,<mensaje>);
        error_num es un entero negativo comprendido entre -20001 y -20999
```

```
CREATE [OR REPLACE] PROCEDURE <procedure_name> [(<param1> [IN|OUT|IN OUT] <type>,...)]
IS
  -- Declaracion de variables locales
BEGIN
  -- Sentencias
[EXCEPTION]
  -- Sentencias control de excepcion
END [<procedure_name>];
```

```
CREATE [OR REPLACE] FUNCTION <fn_name>[(<param1> IN <type>,...)]
RETURN <return_type>
IS
  result <return_type>;
BEGIN

  return(result);
[EXCEPTION]
  -- Sentencias control de excepcion
END [<fn_name>];
```

CURSORES:

CURSOR nombre_cursor IS
instrucción_SELECT;

CURSOR nombre_cursor*(param1 tipo1, ..., paramN tipoN)* IS
instrucción_SELECT;

OPEN nombre_cursor*;*
o bien (en el caso de un cursor con parámetros)
OPEN nombre_cursor*(valor1, valor2, ..., valorN);*

FETCH nombre_cursor INTO lista_variables;
-- o bien ...
FETCH nombre_cursor INTO registro_PL/SQL;

CLOSE nombre_cursor;

CURSOR**%ISOPEN**
CURSOR**%FOUND**
CURSOR**%NOTFOUND**
CURSOR**%ROWCOUNT**

```
CREATE [OR REPLACE] TRIGGER <nombre_trigger>
{BEFORE|AFTER}
                {DELETE|INSERT|UPDATE [OF col1, col2, ..., colN]
             [OR {DELETE|INSERT|UPDATE [OF col1, col2, ..., colN]...]}
ON <nombre_tabla>
[FOR EACH ROW [WHEN (<condicion>)]]
DECLARE
  -- variables locales
BEGIN
  -- Sentencias
[EXCEPTION]
  -- Sentencias control de excepcion
END <nombre_trigger>;


       IF INSERTING THEN
         <BLOQUE pl/sql>
       END IF;
       IF DELETING THEN
         <BLOQUE pl/sql>
       END IF;
       IF UPDATING THEN
         <BLOQUE pl/sql>
       END IF;
```

ALTER TRIGGER <TRIGGER_NAME> {DISABLE|ENABLE};

```
CREATE [OR REPLACE] PACKAGE <pkgName>
IS

  -- Declaraciones de tipos y registros públicas
  {[TYPE <TypeName> IS <Datatype>;]}

  -- Declaraciones de variables y constantes publicas
  -- También podemos declarar cursores
  {[<ConstantName> CONSTANT <Datatype> := <valor>;]}
  {[<VariableName> <Datatype>;]}
  -- Declaraciones de procedimientos y funciones públicas

  {[FUNCTION <FunctionName>(<Parameter> <Datatype>,...)
    RETURN <Datatype>;]}
  {[PROCEDURE <ProcedureName>(<Parameter> <Datatype>, ...);]}
END <pkgName>;


CREATE [OR REPLACE] PACKAGE BODY <pkgName>
IS

  -- Declaraciones de tipos y registros privados
  {[TYPE <TypeName> IS <Datatype>;]}

  -- Declaraciones de variables y constantes privadas
  -- También podemos declarar cursores
  {[<ConstantName> CONSTANT <Datatype> := <valor>;]}
  {[<VariableName> <Datatype>;]}

  -- Implementacion de procedimientos y funciones
  FUNCTION <FunctionName>(<Parameter> <Datatype>,...)
  RETURN <Datatype>
  IS
    -- Variables locales de la funcion
  BEGIN
    -- Implementacion de la funcion
    return(<Result>);
  [EXCEPTION]
    -- Control de excepciones
  END;

  PROCEDURE <ProcedureName>(<Parameter> <Datatype>, ...)
  IS
   -- Variables locales de la funcion
  BEGIN
    -- Implementacion de procedimiento
  [EXCEPTION]
    -- Control de excepciones
  END;
END <pkgName>;
```