PL/SQL memo

PL/SQL block		`DECLARE`
		`  -- Constants/variables declaration`
		`BEGIN`
		`  -- Statements`
		`EXCEPTION`
		`  -- Run-time error handling`
		`END;`

Variable declaration		`my_variable VARIABLE_TYPE;`
Affectation		`my_variable := value;` `SELECT attribute INTO my_variable FROM table;`

Tests		`IF condition1 THEN`
		`  -- Statements`
		`ELSIF condition2 THEN -- (Optional)`
		`  -- Statements`
		`ELSE -- (Optional)`
		`  -- Statements`
		`END IF;`

Loops		`FOR iterator IN [REVERSE] min..max LOOP`
		`  -- Statements`
		`END LOOP;`
		`WHILE condition LOOP`
		`  -- Statements`
		`END LOOP;`

Cursors	- *Declaration*	`CURSOR my_cursor IS SQL_query;`

	- *Usage*	`FOR tuple IN my_cursor LOOP`
		`  -- Statements`
		`  -- Ex. my_variable := tuple.attribute;`
		`END LOOP;-- Note: tuple is of type my_cursor%ROWTYPE`

Exceptions	- *Declare*	`my_exception EXCEPTION;`
	- *Raise*	`RAISE my_exception;`
	- *Process*	`WHEN my_exception THEN -- Statements;`
PL/SQL documentation		http://docs.oracle.com/database/121/LNPLS/toc.htm

Client software: SQL Developer

- Menu "Démarrer / Tous les programmes / Oracle – OraClient*version* / Développement d'applications".
- Execute a PL/SQL block from the SQL window: F5.
- Activate output in result window: DBMS Output ("Sortie SGBD") tab / icon 

Exercise #1

1. Copy table PARTS of user DARMONT into table PARTS on your account (in SQL: `CREATE TABLE parts AS SELECT * FROM darmont.parts`).

2. Write an anonymous PL/SQL block that displays a message on screen. The message must be stored in a string variable initialized to "Hello World!", for instance. Test!

3. In the same PL/SQL block, define an integer variable and store in it the total number of parts from table PARTS. Display the result with respect to format "Number of parts = XX". Test!

4. In case the number of parts is greater than 5, raise an exception and do not display the number of parts. Processing this exception must interrupt the program (fatal error; too many parts in the warehouse): use the predefined procedure `RAISE_APPLICATION_ERROR`. Test!

5. At the end of the PL/SQL block, add in the code to display on screen the description of a part whose number is entered on keyboard (use the condition `WHERE partno = '&enter_partno'`). Modify the exception condition such that this part of the code is executed (e.g., number of parts greater than 50). Test! What happens when the input part number does not exist?

Exercise #2

1. Create any view (e.g., in SQL: `CREATE VIEW part_desc AS SELECT description FROM parts`).

2. Write an anonymous PL/SQL block that displays your system catalog (list of tables and views on your account — available through system view TAB) with respect to the following format.
   Object ONE_TABLE is of type TABLE.
   Object ANOTHER_TABLE is of type TABLE.
   Object ONE_VIEW is of type VIEW.
   …

Hints:

- First check with SQL the result of query `SELECT * FROM TAB` to see what the contents of view TAB is.
- Exploiting a query that returns more than one row requires the use of a cursor.

Exercise #3

1. Copy table DEPT of user DARMONT into table DEPT on your account.

2. Write an anonymous PL/SQL block that displays on screen departments of table DEPT with respect to format "number: name (location)". Test!

3. In case table DEPT is empty, display a warning on screen (exception), but do not raise a fatal error. Test whether your code works by following the procedure below.
   1. Validate all previous updates with the `COMMIT` SQL statement (or F11).
   2. Delete table DEPT's contents (in SQL: `DELETE FROM dept`).
   3. Run the PL/SQL block.

4. Cancel the deletion of table DEPT's contents with `ROLLBACK` SQL statement (or F12).

Exercise #4

1. Copy table EMP of user DARMONT into table EMP on your account.

2. Write an anonymous PL/SQL block that adds $1000 to the salary of all employees from department 20 whose salary is already greater than $1500. Test!

3. Add an exception to your code: if the new salary of an employee becomes greater than or equal to the highest salary in the company, raise a fatal error. Run your program until this exception is triggered. Were salary updates taken into account?