Writing Executable Statements:

Lexical Unites: (4 types of)

1. Identifiers: A Name assigned to a variable (ex, v_first_name. Character and Date literals must be enclosed in single quotation marks)
2. Delimiters: Any characters which have special meaning associated with it. → (; , + -)
    a. : → Bind Variables
    b. & → substitute
3. Literals: Value assigned to a variable →(Jhon, 42 ,True)
    a. Use Single Quotation Marks to use it → v_name := 'ANKUR'
4. Comments: A description for a code.
    a. Single Line Comments → --
    b. Multiple Line Comments → /*……..*/

SQL Function in PL/SQL: (Availability of Function in procedural language)

1. All single row Function are **available**, but
2. DECODE and Group Functions are **not Available** (AVG,MAX,MIN,COUNT, SUM etc.)

Using Sequence in PL/SQL:

To define PK → by default it increments by 1 value

```
DECLARE
  v_new_id NUMBER;
BEGIN
  v_new_id := my_seq.NEXTVAL;
END;
/
```

Data Type Conversion: (Converts data to comparable data types)

1. Implicit Conversion →
    a. Characters and numbers
    b. Characters and Dates
2. Explicit Conversion→
    a. TO_CHAR
    b. TO_DATE
    c. TO_NUMBER
    d. TO_TIMESTAMP

Data Type Conversion

1
```
-- implicit data type conversion
v_date_of_joining DATE:= '02-Feb-2000';
```

2
```
-- error in data type conversion
v_date_of_joining DATE:= 'February 02,2000';
```

3
```
-- explicit data type conversion
v_date_of_joining DATE:= TO_DATE('February
02,2000','Month DD, YYYY');
```


Note: Use **SELECT SYSDATE from DUAL;** to select SYSDATE from dual.

Nested Block:

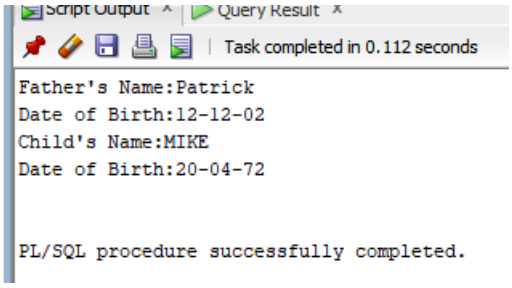
- An executable section (Begin…. END) can contain nested block.
    - Inner block can see the outer block. We can call inner block variable also.

Try to Understand the control flow of below codes:

Variable Scope and Visibility

```
DECLARE
 v_father_name VARCHAR2(20):='Patrick';
 v_date_of_birth DATE:='20-Apr-1972';
BEGIN
  DECLARE
   v_child_name VARCHAR2(20):='Mike';
   v_date_of_birth DATE:='12-Dec-2002';
  BEGIN
   DBMS_OUTPUT.PUT_LINE('Father''s Name: '||v_father_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
   DBMS_OUTPUT.PUT_LINE('Child''s Name: '||v_child_name);
  END;
 DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
END;
/
```

- The *scope* of a variable is the portion of the program in which the variable is declared and is accessible.
- The *visibility* of a variable is the portion of the program where the variable can be accessed without using a qualifier.

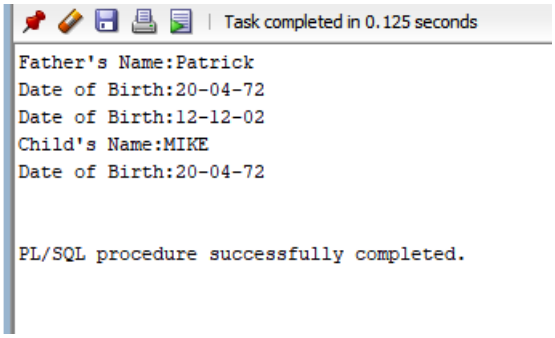


<< >> → Known as Gullimets → Used to create Qualifiers

A qualifier is a label given to block. You can use qualifier to access the variables that have scope but are not visible.

Using a Qualifier with Nested Blocks

```
BEGIN <<outer>>
DECLARE
 v_father_name VARCHAR2(20):='Patrick';
 v_date_of_birth DATE:='20-Apr-1972';
BEGIN
  DECLARE
   v_child_name VARCHAR2(20):='Mike';
   v_date_of_birth DATE:='12-Dec-2002';
  BEGIN
   DBMS_OUTPUT.PUT_LINE('Father''s Name: '||v_father_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '
                        ||outer.v_date_of_birth);
   DBMS_OUTPUT.PUT_LINE('Child''s Name: '||v_child_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
  END;
END;
END outer;
```



Notes:

For control over nest blocks, Whenever the same name as variable name is encountered follow that formula for output or follow that control flow.

Go to page 104 and 105

Operators in PL/SQL:

1. Logical
2. Arithmetic
3. Concatenation
4. Parentheses to control order of operations

Operator	Operation
**	Exponentiation
+, -	Identity, negation
*, /	Multiplication, division
+, -, ||	Addition, subtraction, concatenation
=, <, >, <=, >=, <>, !=, ~=, ^=, IS NULL, LIKE, BETWEEN, IN	Comparison
NOT	Logical negation
AND	Conjunction
OR	Inclusion

Notes:

1. Comparisons involving null always yield NULL.
2. Applying Logical operator to NULL yields NULL.
3. If the condition is NULL then the sequence of code associated with it won't run.

Programming Guidelines: (An advice for doing codes)

1. Indentation
2. Spacing
3. Casing
4. Naming Convention