Variable:

An area in memory to store values.

Use of variables:

1. Temporary storage data,
2. Manipulation of stored values and
3. Reusability.

Variables can be changed but not the constants.

- Variable names should not be an ambiguous, follow meaning full naming convention.
- Name **must start with a letter**, can include numbers and special characters. Must not include **reserved words.**

Variable Handling:

- Declared and initialized in declarative section
- Used and assigned new values
- Passed as parameters
- Used to hold the output

Delimiter:

If you have ‘’ or ! or : then use delimiter to get values in variable.

Delimiter is q’.

If you have many quote like ‘’ or “” and you want to print it put special character after delimiter. → q’ [ ] ‘(Opening as well as closing)

Example,

```
SET SERVEROUTPUT ON
DECLARE
v_event VARCHAR2(15);

BEGIN
v_event := q'!Father's day!';
DBMS_OUTPUT.PUT_LINE('3rd Sunday in June is: ' || v_event);
v_event := q'[Father's day]';
DBMS_OUTPUT.PUT_LINE('2nd Sunday in May is: ' || v_event);
END;
```

Types of PL/SQL Variables:

1. Scaler:
    a. Holds a single value (Like TRUE, FALSE, Atlanta, 234567.567 etc.)
    b. Have no internal components (internal components → Something which is automatic)
    c. Examples: NUMBER, CHAR, DATE, BOOLEAN, VARCHAR2, TIMESTAMP, BINARY_INTEGER, PLS_INTEGER, BINARY_FLOAT, BINARY_DOUBLE
2. Reference:
    a. Holds value of reference (Pointer)
3. Large Object (LOB) (holds value of LOB locator)
    a. BLOB → Binary Large Object (Image)
    b. CLOB → Character Large Object
    c. NCLOB → National Character Large Object (Different Language uses age)
    d. BFile → Binary File (Movies and videos)
4. Composite → can hold many values (Same Collections and Different collection)

Non-PL/SQL Variables: Bind Variables (aka host variables)

- A **VARIABLE** key word must be used to declare bind values.
- Should be declared before declarative section.
- Use :(colon) to reference them in SQL Statements
- For bind variables use **PRINT** to give output.
- Accessed after PL/SQL block is executed
- To display all bind variables, use **PRINT** without a variable
- Use **SET AUTOPRINT ON** to automatically display all bind variables and use &(substitution variable to receive user input)

1. Host Variable
2. Runtime Variable
3. Dynamic Variable

Guideline:

1. Consistent naming convention
2. Use meaningful identifiers for variables (a name to a variable → identifiers)
3. Initialize variables that are designated as **NOT NULL** and **CONSTANT**.
4. Initialize variable with (Assignment Operator) **:=** or the **DEFAULT** keyword.

PL/SQL statements works in 4 phases: parsing (Scanning), Binding, Execute

%TYPE Attribute: (Very much Useful for ETL):

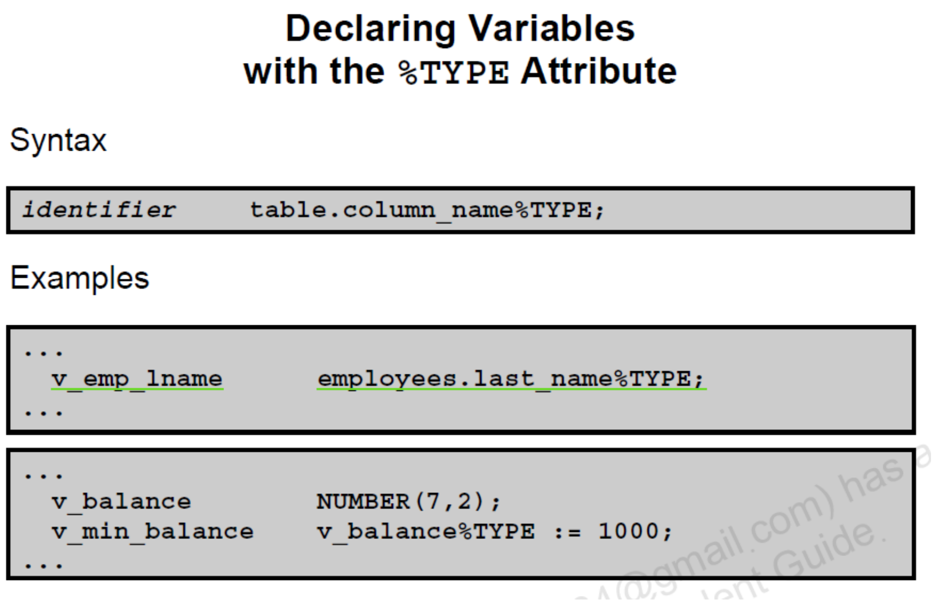
1. Used to declare a variable according to (type of mapping available for %TYPE attribute is):
    a. A database column definition (aka **Default Schema mapping**)
    b. Another declared variable (aka **User Defined variable mapping**)

Syntax: identifier table.column_name%TYPE

Example, v_emp_lastname employee.last_name%TYPE;

Advantages of %TYPE:

1. Error can be avid caused by data type mismatch or wrong precision
2. Hard coding of data type can be avoided
3. No need to change the variable declaration if the column definition changes



Declaring Variables with the %TYPE Attribute

Syntax

```
identifier     table.column_name%TYPE;
```

Examples

```
...
  v_emp_lname       employees.last_name%TYPE;
...
```

```
...
  v_balance         NUMBER(7,2);
  v_min_balance     v_balance%TYPE := 1000;
...
```

Boolean Variables:

Values that can Boolean variable hold: TRUE, FALSE, NULL

Arithmetic, Character and Date Expression can be used to return Boolean variables

Boolean expressions are the basis for conditional control

NULL stands for missing, inapplicable or unknown values

Syntax:

```
identifier [CONSTANT] datatype [NOT NULL]
     [:= | DEFAULT expr];
```

Examples:

```
DECLARE
  v_hiredate      DATE;
  v_deptno        NUMBER(2) NOT NULL := 10;
  v_location      VARCHAR2(13) := 'Atlanta';
  c_comm          CONSTANT NUMBER := 1400;
```

Records and Collections:

Records	Collections
Internal Component can be different data type (aka fields)	Internal components are same data type (aka elements)
record_name.field_name	Unique subscript is used to access elements
Each record fields corresponds to table column	Associative arrays, nested tables and VArrays

The %ROWTYPE Attribute

The %ROWTYPE attribute is used to declare a record that can hold an entire row of a table or view. You learn about this attribute in the lesson titled “Working with Composite Data Types.”