```
###  PL-SQL Conditions ====>
============================

* PL-SQL provides the following types of decision making statements.
- IF-THEN
- IF-THEN-ELSE
- IF-THEN-ELSIF
- NESTED-IF
- CASE
- SEARCHED CASE


# The IF-THEN Statement ==>
===========================

* The IF statement associates a condition with a sequence of statements
enclosed by the keywords THEN and END IF.

* Syntax :-
IF condition THEN
    Stmts;
END IF;

* IF the condition is true, the statements get executed and if the condition is
false or NULL then the IF statement does nothing.


# The IF-THEN-ELSE Statement ==>
================================

* A sequence of IF-THEN statements can be followed by an optional sequence of
ELSE statements, which execute when the condition is FALSE.

* Syntax:-
IF condition THEN
    stmts;
ELSE
    Stmts;
END IF;


# The IF-THEN-ELSIF Statement ==>
=================================

* The IF-THEN-ELSIF statement allows us to choose between several alternatives.

* Syntax:-
IF conditionTHEN
    Stmts;
ELSIF conditionTHEN
    Stmts;
ELSE
    Stmts;
END IF;


# The NESTED IF Statement ==>
=============================

* PL/SQL allows us to nest the IF-ELSE statements, which means we can use one
IF or ELSE IF statement inside another IF or ELSE statement(s).
```

* Syntax:-
```
IF condition THEN
    IF condition THEN
        Stmts;
    ELSE
        Stmts;
    END IF;
ELSE
    Stmts;
END IF;
```

# WAP to accept an integer and check whether it is even or odd.

```
declare
a int;
begin
    a := &a;
    if mod(a, 2) = 0 then
        dbms_output.put_line('Even No');
    else
        dbms_output.put_line('Odd No');
    end if;
end;
```

# WAP to accept a character and check whether it is a vowel or not.

```
declare
ch char(1);
begin
   ch := '&ch';
   if lower(ch) in ('a','e','i','o','u') then
      dbms_output.put_line(ch || ' is Vowel.');
   else
      dbms_output.put_line(ch || ' is Consonant.');
   end if;
end;
```

# The CASE Statement ==>
========================

* The CASE statement chooses one sequence of statements to execute out of many possible sequences.

* The CASE statement has two types: simple CASE statement and searched CASE statement.

* Both types of the CASE statements support an optional ELSE clause.

* Syntax:-
```
CASE expression
    WHEN valueTHEN
        Stmts;
    WHEN valueTHEN
        Stmts;
    ...
    ELSE
        Stmts;
```

END CASE;

# WAP to accept an integer and check whether it is even or odd using Simple CASE statement.

```
declare
a int;
begin
  a := &a;
  case mod(a, 2)
      when 0 then
          dbms_output.put_line('Even No');
      else
          dbms_output.put_line('Odd No');
   end case;
end;
```

# Searched CASE Statement ==>
=============================

* The searched CASE statement evaluates multiple Boolean expressions and executes the sequence of statements associated with the first condition that evaluates to TRUE.

* Syntax:-
```
CASE
    WHEN condition THEN
        Stmts;
    WHEN condition THEN
        Stmts;
    ...
    ELSE
        Stmts;
END CASE;
```

# WAP to accept an integer and check whether it is even or odd using Searched CASE statement.

```
declare
    a int;
    begin
      a := &a;
      case
          when mod(a, 2) = 0 then
              dbms_output.put_line('Even No');
          else
              dbms_output.put_line('Odd No');
      end case;
 end;
```

# WAP to accept an integer and check whether it is a single digit number or a double digit number or a triple digit number or a number with more than 3 digits.

```
declare
    a int;
    begin
```

```
      a := &a;
      case
          when a <= 9 then
              dbms_output.put_line('Single Digit');
          when a <= 99 then
              dbms_output.put_line('Double Digit');
          when a <= 999 then
               dbms_output.put_line('Triple Digit');
          else
               dbms_output.put_line('More then Three Digit.');
      end case;
end;
```