### Introduction To Packages ====>
===================================

* A package is a way of logically storing the subprograms like procedure, function or cursor into a single common unit.

* A package can be defined as an Oracle object that is compiled and stored in the database.

* Once it is compiled and stored in the database it can be used by all the users of database who have executable permissions on Oracle database.

# Benefits Of Using A Package ==>
=================================

* `REUSABILITY` :- Whenever a package is created, it is compiled and stored in the database. So, we write the code once which can be reused by other applications.

* `OVERLOADING` :- Two or more procedures or functions can be created in a package with the same name.

* `IMPROVES PERFORMANCE` :- Package code gets loaded inside the SGA(system global area) of Oracle at first call itself due to which other subsequent calls will work very fast.

## Components Of A Package ===>
===============================

* Package has two basic components:
- `Specification` :- It is the declaration section of a Package
- `Body` :- It is the definition section of a Package.

# Package Specification ==>
===========================

* The package specification is where we declare public items.
* The scope of package items is the schema of the package.
* In other words, we can access items declared in a package specification from anywhere in the schema e.g., we can access items in a package specification from other packages.
* A package specification does not contain any implementations of the public items.
* For example, in case of procedures or functions, the package specification contains only their headers, but not their bodies.

* Syntax :-

```
CREATE OR REPLACE PACKAGE package_name IS | AS
    variable_declaration ... ;
    constant_declaration ...;
    exception_declaration ...;
    cursor_specification ...;
PROCEDURE procedure_name (param_name [IN|OUT|N OUT]
    datatype ,…);
FUNCTION function_name (param_name [IN|OUT|N OUT]
    datatype ,…)RETURN datatype ;
END package_name;
```

# Package Body ==>
==================

* If the package specification has cursors or subprograms, then the package body is mandatory.
* Otherwise, it is optional.

* Syntax :-

```
CREATE OR REPLACE PACKAGE BODY <package_name> IS/AS
    FUNCTION <function_name> (<list of arguments>) RETURN <datatype>IS/AS
        -- local variable declaration;
    BEGIN
        -- executable statements;
    EXCEPTION
        -- error handling statements;
    END <function_name>;

    PROCEDURE <procedure_name> (<list of arguments>)IS/AS
        -- local variable declaration;
    BEGIN
        -- executable statements;
    EXCEPTION
        -- error handling statements;
    END <procedure_name>;
END <package_name>;
```

# Using A Package ==>
=====================

* Creating a package only defines it, to use it we must refer it using the package object.

* Following is the syntax for referring a package object:
# Packagename.objectname;

* The Object can be a function, procedure, cursor, exception that has been declared in the package specification and defined in the package body.

# Example ==>

* Below we have a table called Students with Student's data in it.

ROLLNO	SNAME	AGE	COURSE
11	Anu	20	BSC
12	Asha	21	BCOM
13	Arpit	18	BCA
14	Chetan	20	BCA

# Create a package called Student_pkg containing a procedure called update_course to update the course name for a student with given roll no and a function called delete_stud to remove a student of given roll no.

# Package Spec ==>

```
CREATE OR REPLACE PACKAGE pkg_student IS
    PROCEDURE update_course (sno students.rollno%type ,
    cname students.course%type);
    FUNCTION delete_stud (sno students.rollno%type) RETURN
    boolean;
END pkg_student;
```

# Package Body ==>

```
CREATE OR REPLACE PACKAGE BODY pkg_student IS
    PROCEDURE update_course (sno students.rollno%type,
    cname students.course%type)IS
    BEGIN
        Update students set course=cname where rollno=sno;
        IF SQL%FOUND THEN
            dbms_output.put_line('RECORD UPDATED');
        ELSE
            dbms_output.put_line('RECORD NOT FOUND');
        END IF;
    END updateRecord;

    FUNCTION delete_stud (sno students.rollno%type) RETURN
    boolean IS
        BEGIN
            Delete from student where rollno=sno;
            RETURN SQL%FOUND;
    END deleteRecord;
END pkg_student;
```

# Calling ==>

```
DECLARE
    sno student.rollno%type;
    cname student.course%type;
BEGIN
    sno := &sno;
    cname:='&course';
    pkg_student.update_course(sno,cname);
    IF pkg_student.delete_stud(sno) THEN
        dbms_output.put_line('RECORD DELETED');
    ELSE
        dbms_output.put_line('RECORD NOT FOUND');
    END IF;
END;
```

## Overloading ===>
====================

* When more than one program in the same scope share the same name, the programs are said to be overloaded.

* PL/SQL supports the overloading of procedures and functions in package specifications and bodies.

* Each overloaded version must differ from all other overloaded versions in at least one of the following respects:
- Number of parameters
- Order of the parameters
- Data types of the parameters

# Example ==>

```
CREATE OR REPLACE PACKAGE addition IS
    FUNCTION adding (n integer,m integer) RETURN integer;
    FUNCTION adding (n date,m integer) RETURN date;
END addition;

CREATE OR REPLACE PACKAGE BODY addition IS
    FUNCTION adding (n integer,m integer) RETURN integer IS
        BEGIN
            return n + m;
    END;
    FUNCTION adding (n date,m integer) RETURN date IS
        BEGIN
            return n + m;
    END;
END addition;
```

# Calling ==>

```
BEGIN
    dbms_output.put_line(‘adding 3 and 6 using overloaded
    function adding:’ || addition.adding(3,6));

    dbms_output.put_line(’10 days from today using overloaded
    function adding:’ || addition.adding(sysdate,10));
END;
```

## Dropping A Package ==>
==========================

* To drop a package, we use the DROP PACKAGE statement with the following syntax:
# DROP PACKAGE [BODY] package_name;

* If we want to drop only the body of the package, we need to specify the BODY keyword.

* If we omit the BODY keyword, then the statement drops both the body and specification of the package.