PACKAGES:

* It is a Named block which contains Variables, Cursors,Procedures & Functions are stored in one location/single unit of memory.

* Generally packages and stored procedures are saves individually if we want to store in a single unit then in that case packages are required.

> Easy to share the Subprograms in Application S/w Tools.

> They Improve performance while accessing subprograms from client location.

> They are stored in "User_Source" system table.

> They supports function Overloading, Encapsulation & Databinding.

> To create package we need to use 2 blocks.

1. **Package Specification Block:**
   - It holds the declaration of Variables,functions,Cursors & Subprograms.

Syntax:

```
Create [or Replace ] package <package name>
is / as
<Declare variables,cursors,sub blocks>;
end;
/
```

2. **Package Body/definition//Package Implementation Block:**
   - It holds the body of subprograms.Implementing the logical code of function and subprogram.

Syntax:

```
Create [or Replace ] package body <package name>
is / as
<implementing sub blocks>;
end;
/
```

Syntax To Call A Stored Procedure from a Package

```
execute package_name.procedure_name(values)
```

Syntax To Call A Stored Function from a Package

select package_name.function_name(values) from dual

Examples on Packages:

Ex:

Package Specification:

```
Create or replace package my_pack
        is
 Result Varchar2(50);   -- public variables
 Procedure emp_exp(tempno emp.empno%type);
 Function emp_netsal(Tempno Emp.Empno%type) return Varchar2;
End my_pack;
```

Package Body:

```
Create or Replace Package Body my_pack
            is
Procedure Emp_Exp(Tempno Emp.Empno%type)
           is
Tdate Emp.Hiredate%type;  -- private variables
Texp Number;
Begin
 Select Hiredate into Tdate from Emp where Empno=Tempno;
 Texp:=round((sysdate-tdate)/365);
 dbms_output.put_line(Tempno||' Employee Experience is '||Texp||'Years.');
End Emp_Exp;

Function Emp_Netsal(Tempno Emp.Empno%type)
         return Varchar2
            is
Tsal Emp.Sal%type;
Tcomm Emp.Comm%type;
```

```
 Begin
  select sal+nvl(comm,0) into Result From Emp
                                    where empno=Tempno;
  Return(Tempno||'Employee Net Salary Rs.'||Result);
 End Emp_Netsal;
End my_pack;
```

To Execute above Package:

```
Exec my_pack.Emp_Exp(7788);
Select my_pack.Emp_Netsal(7788) from dual;
```

Function Overloading using package:

Ex:

```
Create or Replace Package fo_pack
          is
Function addval(a number, b number) return Number;
Function addval(a number, b number, c number) return number;
Function addval(str1 varchar2, str2 varchar2) return Varchar2;
Function addval(str1 varchar2, str2 varchar2, str3 varchar2) return varchar2;
End fo_pack;
```

Package Body:

```
Create or Replace Package Body fo_pack
          is
Function addval(a number, b number) return Number
             is
Begin
 return(a+b);
End addval;

Function addval(a number, b number, c number) return number
                  Is
```

```
Begin
 return(a+b+c);
End addval;

Function addval(str1 varchar2, str2 varchar2) return varchar2
                    is
Begin
 return(str1||str2);
End Addval;

Function addval(str1 varchar2, str2 varchar2, str3 varchar2) return varchar2
                          is
Begin
 return(str1||str2||str3);
End Addval;
End fo_pack;
```

calling package:

```
Select fo_pack.addval(10,20) from dual;
Select fo_pack.addval('Rama ','Krishna ','Raju') from dual;
Select fo_pack.addval(10,20,50) from dual;
```

Note:

* all packages bodies are stored in 'user_source'.

* to see the package body.

ex:

```
sql> select text from USER_SOURCE where name='FO_PACK';
```

Dropping Package body:

syntax:

```
sql> Drop Package Body <package name>;
```

Ex:

sql> Drop Package Body my_pack;

Dropping Packages:

syntax:

sql> Drop Package <package name>;

Ex:

sql> Drop package my_pack;