Introduction To Cursor ====>
==========

* When an SQL statement is processed, Oracle creates a memory area known as context area.

* This context area contains information regarding rows retrieved from table, their count etc.

* A Cursor is basically a pointer to this Context Area i.e. a 'Cursor' is the PL/SQL construct that allows the user to name the work area and access the stored information in it.

Types Of Cursor ===>
==========

* There are two types of cursors:
- Implicit
- Explicit

Implicit Cursor ==>
==========

* They are generated automatically by Oracle server when an SQL statement occurs in the PL/SQL executable part; like SELECT INTO , INSERT , UPDATE or DELETE .

* For INSERT operations, the cursor holds the data that needs to be inserted.

* For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

Implicit Cursor Attributes ==>
==========

Attribute	Description
SQL%FOUND	Its return value is TRUE
SQL%NOTFOUND	Its return value is TRUE
SQL%ISOPEN	It always returns FALSE
SQL%ROWCOUNT	It returns the number of rows affected by DML statements

WA PL-SQL script to accept a EMP name from the user and increase the salary by 10%. Also check whether the update was done or not.

```
1  declare
2  name emp.ename%type;
3  begin
4  name := '&name';
5  update emp set sal = sal + sal * 0.1
6  where ename = name;
7  if sql%found then
8     dbms_output.put_line('Record Updated');
9  else
10     dbms_output.put_line('Not Updated');
11 end if;
12 end;
```

Explicit Cursor ==>
=====================

* Explicit cursors are programmer-defined cursors for gaining more control over the context area.

* An explicit cursor should be defined in the declaration section of the PL/SQL Block.

* It is created on a SELECT Statement which returns more than one row.

* The syntax for creating an explicit cursor is –
- CURSOR cursor_name IS select_statement;

* Working with an explicit cursor includes the following steps –
- Declaring the cursor for initializing the memory
- Opening the cursor for allocating the memory
- Fetching the cursor for retrieving the data
- Closing the cursor to release the allocated memory

Declaring the Cursor ===>

* Declaring the cursor defines the cursor with a name and the associated SELECT statement.

* For example –
- CURSOR c_customers IS SELECT id, name, address FROM customers;

Opening the Cursor ===>

* Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL
statement into it.

* For example, we will open the previously defined cursor as follows –
- OPEN c_customers;

Fetching the Cursor ===>

* Fetching the cursor involves accessing one row at a time.

* For example, we will fetch rows from the c_customers cursor as follows –
- FETCH c_customers INTO c_id, c_name, c_addr;

Closing the Cursor ===>

* Closing the cursor means releasing the allocated memory.
* For example, we will close the c_customers cursor as follows –
- CLOSE c_customers;

WA PL_SQL script to display bookname and subject of all the books belonging to JSE.

1 declare

```
2    cursor javabooks is select bookname, bookprice from allbooks  where
subject= 'JSE';
3    name allbooks.bookname%type;
4    amt allbooks.bookprice%type;
5  begin
6    open javabooks;
7    if javabooks%isopen then
8       loop
9          fetch javabooks into name,amt;
10         exit when javabooks%notfound;
11         dbms_output.put_line(name || ',' || amt);
12      end loop;
13      close javabooks;
14   else
15      dbms_output.put_line('Sorry! can not open the cursot');
16   end if;
17 end;
18 /
```

WA PL_SQL script to display top 3 high paid EMP sal from emp rable.

```
1  declare
2    cursor allemp is select ename, sal from emp order by sal desc;
3    name emp.ename%type;
4    amt emp.sal%type;
5  begin
6    open allemp;
7    if allemp%isopen then
8        loop
9          fetch allemp into name,amt;
10          exit when allemp%notfound or allemp%rowcount=4;
11          dbms_output.put_line(name || ',' || amt);
12       end loop;
13       close allemp;
14   else
15       dbms_output.put_line('Sorry! can not open the cursor');
16   end if;
17 end;
```

Cursor For Loop ===>

==========

* The cursor FOR LOOP statement is an elegant extension of the numeric FOR LOOP statement.

* The numeric FOR LOOP executes the body of a loop only once for every integer value in a specified range.

* Similarly, the cursor FOR LOOP executes the body of the loop only once for each row returned by the query associated with the cursor.

* A nice feature of the cursor FOR LOOP statement is that it allows you to fetch every row from a cursor without manually managing the execution cycle i.e., OPEN, FETCH, and CLOSE.

* The cursor FOR LOOP implicitly creates its loop index as a variable with the %rowtype in which the cursor returns and then opens the cursor.

* In each loop iteration, the cursor FOR LOOP statement fetches a row from the result set into its loop index. If there is no row to fetch, the cursor FOR LOOP closes the cursor.

* Syntax :-

```
FOR record IN cursor_name LOOP
    process_record_statements;
END LOOP;
```

* Note that besides the cursor name, we can use a SELECT statement as shown below:

```
FOR record IN (select_statement) LOOP
    process_record_statements;
END LOOP;
```

* In this case, the cursor FOR LOOP declares, opens, fetches from, and closes an implicit cursor.

* However, the implicit cursor is internal; therefore, we cannot reference it.

WA PL_SQL script to display bookname and subject of all the books belonging to JSE.

```
1 declare
2      cursor cr_java in select bookname, bookprice from
3      allbooks where subject = 'JSE';
4  begin
5      for x in cr_java loop
6          dbms_output.put_line(x.bookname || ',' || x.bookprice);
7      end loop;
8  end;
9  /
```

==> NEW STYLE <==

```
1 begin
2     for x in (select bookname, bookprice from allbooks where subject = 'JSE')
loop
3          dbms_output.put_line(x.bookname || ',' || x.bookprice);
4     end loop;
5 end;
```

Parameterized Cursor ===>

==========

* An explicit cursor may accept a list of parameters.

* Each time we open the cursor, we can pass different arguments to the cursor, which results in different result sets.

* Syntax:

```
CURSOR cursor_name (parameter_list) IS
cursor_query;
```

WA PL-SQL script to accept a subject name from the user and display the bookname and bookprice of all the books of that subject. In case no books of the subject is found then display the message No Books Found.