-- Now let's process two rows.

declare

    anumber number;
    asquare number;
    acube number;

    cursor jane
        is select * from x;

begin

    open jane;

    fetch jane into
        anumber, asquare, acube;
    dbms_output.put_line(anumber || ' ' || asquare || ' ' || acube);

    fetch jane into -- Now get the second line
        anumber, asquare, acube;
    dbms_output.put_line(anumber || 'XX' || asquare || 'YY' || acube);

    close jane;

end;
/

-- A few boolean cursor attributes are automatically maintained by
-- the system.

declare
delilah number;
martin number;
acube number;
cursor eric is select * from x;
begin
  open eric;
  loop  -- This is an endless loop
    fetch eric into
      delilah, martin, acube;
    if eric%found  -- boolean attribute of the cursor
      -- set automatically by the system.
      -- If there was a line it is
      -- set to true (1)
    then
      dbms_output.put_line(' ' || delilah || ' --- ' ||
                           martin || ' --- ' || acube);
    else
      exit;  -- This jumps out of the loop
    end if;
  end loop;
  close eric;
end;
/
/*
Now let's finally do something that is hard to do with a
select statement. But easy to do with a program.
How about displaying the values with rows of stars.

So, this is basically the same program, but now I
display the values by rows of stars, = and > signs.
*/

declare

   anumber number;
asquare number;
acube number;
i number;
outline varchar2(80);
cursor line is select * from x;
begin
  open line;
  loop
    fetch line into
      anumber, asquare, acube;
    if line%found then
      dbms_output.put_line(anumber || ' ' || asquare || ' ' || acube);
      outline := "";
      for i in 1 .. anumber
        loop
          outline := outline || '>'; 
        end loop;
      dbms_output.put_line(outline);
      outline := "";
      for i in 1 .. asquare
        loop
          outline := outline || '=';
        end loop;
      dbms_output.put_line(outline);
      outline := "";
      for i in 1 .. acube
        loop
          outline := outline || '*';
        end loop;
      dbms_output.put_line(outline);
else
    exit;
end if;
end loop;
close line;
end;
/

/*

Other cursor attributes
automatically set by the
system.

cursormame%NOTFOUND
cursormame%ROWCOUNT  number of rows fetched so far
cursormame%ISOPEN

*/
/*
It's hard to believe how much shorter this can be done.
But some people don't "understand" it.
*/

declare
cursor cathy is select * from x; -- Declare cursor
begin

  for counter in cathy -- counter is automatically declared.
    -- As variable that loops over rows.
    -- This FOR loops over all rows.
    -- Inside this loop the column names
    -- are known without declaration!
    -- And... I don't need any % syntax
  loop
    dbms_output.put_line(counter.num ||
      ' ' || counter.sqr || ' ' || counter.cube);
  end loop;

end;
/
/*
Now the whole line drawing in the short form.
*/

declare
cursor line
   is select * from x;
outline varchar2(80);
i number;
begin

for counter in line
loop
    dbms_output.put_line(counter.num || 
    ' ' || counter.sqr || ' ' || counter.cube);

    outline := ';
    for i in 1 .. counter.num
    loop
        outline := outline || '*';
    end loop;
    dbms_output.put_line(outline);
    outline := ';
    for i in 1 .. counter.sqr
    loop
        outline := outline || '*';
    end loop;
    dbms_output.put_line(outline);
    outline := ';
    for i in 1 .. counter.cube
    loop
        outline := outline || '*';
    end loop;
    dbms_output.put_line(outline);
end loop;

end;
/

You can do it even shorter. You don’t even need to declare a cursor. It is declared implicitly when you use a FOR loop.

This is the shortest form.

*/

begin

  for joseph in (select * from x)
  loop
    dbms_output.put_line(joseph.num || ' ' || joseph.sqr || ' ' || joseph.cube);
  end loop;

end;
/*