**Basic Block**

begin

DBMS\_OUTPUT.PUT\_LINE('Hello World !');

end;

**Declaring Variables**

declare

num1 number:=1;

num2 number:=2;

res number:=0;

begin

res := num1 + num2;

DBMS\_OUTPUT.PUT\_LINE('Result = ' || res);

end;

**If Statement**

declare

num number:=1;

begin

if num > 0 then

DBMS\_OUTPUT.PUT\_LINE('Positive');

end if;

end;

**If then elseif Statement**

declare

num number:=0;

begin

if num > 0 then

DBMS\_OUTPUT.PUT\_LINE('Positive');

elsif num < 0 then

DBMS\_OUTPUT.PUT\_LINE('Negative');

else

DBMS\_OUTPUT.PUT\_LINE('Number is 0');

end if;

end;

**CASE Statement**

DECLARE

grade CHAR(1);

BEGIN

grade := 'B';

CASE grade

WHEN 'A' THEN DBMS\_OUTPUT.PUT\_LINE('Excellent');

WHEN 'B' THEN DBMS\_OUTPUT.PUT\_LINE('Very Good');

WHEN 'C' THEN DBMS\_OUTPUT.PUT\_LINE('Good');

WHEN 'D' THEN DBMS\_OUTPUT.PUT\_LINE('Fair');

WHEN 'F' THEN DBMS\_OUTPUT.PUT\_LINE('Poor');

ELSE DBMS\_OUTPUT.PUT\_LINE('No such grade');

END CASE;

END;

**CASE Statement using If-ElsIf-Else**

DECLARE

grade CHAR(1);

BEGIN

grade := 'B';

IF grade = 'A' THEN

DBMS\_OUTPUT.PUT\_LINE('Excellent');

ELSIF grade = 'B' THEN

DBMS\_OUTPUT.PUT\_LINE('Very Good');

ELSIF grade = 'C' THEN

DBMS\_OUTPUT.PUT\_LINE('Good');

ELSIF grade = 'D' THEN

DBMS\_OUTPUT.PUT\_LINE('Fair');

ELSIF grade = 'F' THEN

DBMS\_OUTPUT.PUT\_LINE('Poor');

ELSE

DBMS\_OUTPUT.PUT\_LINE('No such grade');

END IF;

END;

**Use SQL Statement in PL SQL Blocks**

DECLARE

job\_count NUMBER;

emp\_count NUMBER;

BEGIN

SELECT COUNT(DISTINCT job\_id) INTO job\_count FROM HR.employees;

SELECT COUNT(\*) INTO emp\_count FROM HR.employees;

DBMS\_OUTPUT.PUT\_LINE(emp\_count);

END;

**Simple Loop**

declare

i number :=0;

begin

loop

DBMS\_Output.PUT\_LINE(i);

i := i + 1;

exit when i = 10;

end loop;

end;

**While Loop**

DECLARE

i INTEGER := 1;

BEGIN

WHILE i <= 10 LOOP

DBMS\_OUTPUT.PUT\_LINE(i);

i := i+1;

END LOOP;

END;

**For Loop**

BEGIN

FOR i IN 1..10 LOOP

DBMS\_OUTPUT.PUT\_LINE(i);

END LOOP;

END;

**Associative array indexed by string**

DECLARE

TYPE population IS TABLE OF NUMBER

INDEX BY VARCHAR2(64);

city\_population population;

i VARCHAR2(64);

BEGIN

city\_population('Pune') := 20000;

city\_population('Mumbai') := 30000;

city\_population('Delhi') := 10000;

i:= city\_population.first;

loop

DBMS\_Output.PUT\_LINE(city\_population(i));

i := city\_population.next(i);

exit when i is null;

END LOOP;

END;

**Nested Table array**

DECLARE

TYPE emp\_salaries IS TABLE OF number;

e\_sal emp\_salaries;

BEGIN

e\_sal:=emp\_salaries(10000,20000,13000,12780);

FOR i in e\_sal.first..e\_sal.last

LOOP

DBMS\_Output.PUT\_LINE(e\_sal(i));

END LOOP;

END;

DECLARE

TYPE emp\_salaries IS TABLE OF number;

e\_sal emp\_salaries;

BEGIN

select salary bulk collect into e\_sal from HR.employees;

FOR i in e\_sal.first..e\_sal.last

LOOP

DBMS\_Output.PUT\_LINE(e\_sal(i));

END LOOP;

END;

**Varray**

DECLARE

TYPE e\_salaries IS VARRAY(5) OF INTEGER;

e\_sal e\_salaries;

BEGIN

e\_sal:= e\_salaries(10000,13000,12000,22000);

FOR i in e\_sal.first..e\_sal.last

LOOP

DBMS\_Output.PUT\_LINE(e\_sal(i));

END LOOP;

END;

**Functions**

create or replace function print\_hello return varchar2

is

begin

return 'Hello World !';

end print\_hello;

-- call the function and assign the value to a variable

declare

msg varchar(100);

begin

msg := print\_hello();

dbms\_output.put\_line(msg);

end;

-- function to find square of a number

create or replace function square(num number) return number

is

begin

return num \* num;

end square;

declare

num number;

begin

num := square(10);

dbms\_output.put\_line(num);

end;

-- find avg salary for a department

create or replace function findAvg(dept\_id number) return number

as

avgSal number;

begin

select avg(salary) into avgSal from HR.employees where

department\_id = dept\_id;

return avgSal;

end findAvg;

declare

avgSal number;

begin

avgSal := findAvg(90);

dbms\_output.put\_line(avgSal);

end;

**Procedures**

create or replace procedure print\_hello

is

begin

dbms\_output.put\_line('Hello World !');

end print\_hello;

-- invoke the procedure

begin

print\_hello();

end;

-- procedure to swap two numbers

create or replace procedure swap (

num1 in out number,

num2 in out number

)

is

temp number;

begin

temp := num1;

num1 := num2;

num2 := temp;

end swap;

-- invoke the procedure

declare

num1 number;

num2 number;

begin

num1 := 10;

num2 := 20;

dbms\_output.put\_line('num1 = ' || num1 || ', num2 = ' || num2);

swap(num1,num2);

dbms\_output.put\_line('num1 = ' || num1 || ', num2 = ' || num2);

end;

**Defining and using an explicit Cursor**

DECLARE

CURSOR c1 IS SELECT last\_name, job\_id FROM HR.employees;

BEGIN

FOR item IN c1

LOOP

DBMS\_OUTPUT.PUT\_LINE('Name = ' || item.last\_name || ', Job = ' || item.job\_id);

END LOOP;

END;

-- using arrays %TYPE with cursors

DECLARE

v\_jobid HR.employees.job\_id%TYPE;

v\_lastname HR.employees.last\_name%TYPE;

CURSOR c1 IS SELECT HR.employees.last\_name, HR.employees.job\_id FROM HR.employees;

BEGIN

OPEN c1;

LOOP

FETCH c1 INTO v\_lastname, v\_jobid;

EXIT WHEN c1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_lastname || ',' || v\_jobid);

END LOOP;

CLOSE c1;

END;

**Defining and using a Trigger**

create table account\_details (

acc\_no number constraint account\_details\_pk primary key,

acc\_name varchar2(50),

acc\_balance number

);

-- creating a trigger to restrict value of balance below 1000

create or replace trigger balance\_trigger\_account\_details

before insert or update on account\_details

for each row

begin

if updating then

if :new.acc\_balance < 1000 then

Raise\_Application\_Error (-20100, 'Reached Minimum Balance.');

end if;

end if;

end;

-- updating to set acc\_balance = 100

update account\_details set acc\_balance = 100 where acc\_no = 1;

ORA-20100: Reached Minimum Balance. ORA-06512: at "SQL\_VSKNDCEMZVMMJDZSEULYLTCFD.BALANCE\_TRIGGER\_ACCOUNT\_DETAILS"