CASE

```
DECLARE
    couleur_drapeau VARCHAR(30) := 'VERT';
BEGIN
   CASE couleur_drapeau
        WHEN 'ROUGE'  THEN
             DBMS_OUTPUT.PUT_LINE('Baignade interdite !!');
        WHEN 'ORANGE' THEN
             DBMS_OUTPUT.PUT_LINE('Attention, la mer est dangereuse');
        WHEN 'VERT'   THEN
             DBMS_OUTPUT.PUT_LINE('Tous à l''eau !!');
        WHEN 'NOIR'   THEN
             DBMS_OUTPUT.PUT_LINE('Marée noire');
        ELSE
             DBMS_OUTPUT.PUT_LINE('Drapeau non repertorié');
   END CASE;
END;
```

```
DECLARE
    couleur_drapeau VARCHAR(30) := 'VERT';
BEGIN
  CASE
     WHEN couleur_drapeau = 'VERT' THEN
          DBMS_OUTPUT.PUT_LINE('Le drapeau est vert.');
     WHEN couleur_drapeau = 'ORANGE' THEN
          DBMS_OUTPUT.PUT_LINE('Le drapeau est orange');
     WHEN couleur_drapeau = 'ROUGE' THEN
          DBMS_OUTPUT.PUT_LINE('Le drapeau est rouge.');
     ELSE
          DBMS_OUTPUT.PUT_LINE('Je ne connais pas cette couleur.');
  END CASE;
END;
```

LOOP

```
DECLARE
    index_loop INTEGER := 0;
BEGIN
    LOOP
            IF index_loop = 3 THEN
               EXIT;
            ELSE
               DBMS_OUTPUT.PUT_LINE(index_loop);
            END IF;
            index_loop := index_loop +1;
    END LOOP;
END;
```

SUBTYPE _____

```
SET SERVEROUTPUT ON;

DECLARE
  SUBTYPE salaire IS NUMBER(4);
  SUBTYPE date_nn IS DATE NOT NULL;

  premiere_annee salaire := 2000;
  premier_jour date_nn := SYSDATE;

BEGIN

  DBMS_OUTPUT.PUT_LINE(premiere_annee);
  DBMS_OUTPUT.PUT_LINE(premier_jour);

END;
```

%TYPE _____

```
SET SERVEROUTPUT ON;

DECLARE

  continent regions.region_name%TYPE;

BEGIN

  continent := 'Europe';
  DBMS_OUTPUT.PUT_LINE(continent);

END;
```

RECORD _____

```
SET SERVEROUTPUT ON;

DECLARE

  TYPE fiche_parking IS RECORD
  (
    prenom VARCHAR2(50),
    nom VARCHAR2(50) NOT NULL := 'XXX',
    nombre NUMBER NOT NULL DEFAULT 1
  );

  fiche1 fiche_parking;

BEGIN

  fiche1.prenom := 'Anthony';
  fiche1.nom := 'Cosson';

  DBMS_OUTPUT.PUT_LINE('Prenom : ' || fiche1.prenom);
  DBMS_OUTPUT.PUT_LINE('Nom : ' || fiche1.nom);
  DBMS_OUTPUT.PUT_LINE('Nombre voiture : ' || fiche1.nombre);

END;
```

%ROWTYPE _____

```
SET SERVEROUTPUT ON;

DECLARE

  region_rec regions%ROWTYPE;

BEGIN

  region_rec.region_id = 5;
  region_rec.region_name = 'Antarctique';

  DBMS_OUTPUT.PUT_LINE('Id de la region : ' || region_rec.region_id);
  DBMS_OUTPUT.PUT_LINE('Nom de la region : ' || region_rec.region_name);

END;
```

COLLECTION INDEX BY TABLE _____

```
SET SERVEROUTPUT ON;

DECLARE

  TYPE population IS TABLE OF NUMBER INDEX BY VARCHAR(50);

  population_ville population;

  ville VARCHAR2(50);
BEGIN

  population_ville('Soliers') := 2151;
  population_ville('Caen') := 106538;
  population_ville('Limoges') := 134577;
  population_ville('Rennes') := 213454;

  ville := population_ville.FIRST;

  WHILE ville IS NOT NULL LOOP
      DBMS_Output.PUT_LINE('La population de ' || ville
      || ' est de ' || TO_CHAR(population_ville(ville)) || ' habitants');

      ville := population_ville.NEXT(ville);
  END LOOP;
 END;
```

COLLECTION NESTED TABLE

```
SET SERVEROUTPUT ON;

DECLARE

  TYPE tableau_noms_objets IS TABLE OF VARCHAR2(50);

  tableau_musique tableau_noms_objets;

  objet_courant VARCHAR2(50);
BEGIN

  tableau_musique := tableau_noms_objets('Guitare','Tambour','Batterie','Basse');

  tableau_musique.DELETE(2);

  DBMS_OUTPUT.PUT_LINE(tableau_musique(1));
  --DBMS_OUTPUT.PUT_LINE(tableau_musique(2));
  DBMS_OUTPUT.PUT_LINE(tableau_musique(3));


 END;
```

COLLECTION VARRAY

```
SET SERVEROUTPUT ON;

DECLARE

  TYPE tableau_couleur IS VARRAY(5) OF VARCHAR2(50);

  france tableau_couleur := tableau_couleur('Bleu','Blanc','Rouge');

BEGIN

  DBMS_OUTPUT.PUT_LINE(france(1));
  DBMS_OUTPUT.PUT_LINE(france(2));
  DBMS_OUTPUT.PUT_LINE(france(3));

 END;
```

CURSEUR EXPLICITE 1ere METHODE

```
SET SERVEROUTPUT ON;

DECLARE
  CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60;

  employee employees%ROWTYPE;
BEGIN
  OPEN cursor_employees_it;

  LOOP
    FETCH cursor_employees_it INTO employee;
    DBMS_OUTPUT.PUT_LINE(employee.last_name);
    EXIT WHEN cursor_employees_it%NOTFOUND;
  END LOOP;

  CLOSE cursor_employees_it;
END;
```

CURSEUR EXPLICITE 2eme METHODE

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR cursor_employees_it IS SELECT first_name,last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = 60;

    employee employees%ROWTYPE;
BEGIN

  FOR employee IN cursor_employees_it LOOP

    DBMS_OUTPUT.PUT_LINE(employee.first_name);

  END LOOP;

END;
```

VERROU FOR UPDATE

```
SET SERVEROUTPUT ON;

DECLARE
  CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60 FOR UPDATE;

BEGIN

  FOR employee IN cursor_employees_it LOOP

    UPDATE employees SET  last_name = UPPER(last_name) WHERE employee_id = employee.employee_id;
    DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);

  END LOOP;

  COMMIT;
END;
```

VERROU FOR UPDATE OF _____

```
SET SERVEROUTPUT ON;

DECLARE
  CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60 FOR UPDATE OF last_name;

BEGIN

  FOR employee IN cursor_employees_it LOOP

    UPDATE employees SET  last_name = UPPER(last_name) WHERE employee_id = employee.employee_id;
    DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);

  END LOOP;

  COMMIT;
END;
```

WHERE CURRENT OF _____

```
SET SERVEROUTPUT ON;

DECLARE
  CURSOR cursor_employees_it IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60 FOR UPDATE OF last_name;

BEGIN

  FOR employee IN cursor_employees_it LOOP

    UPDATE employees SET  last_name = UPPER(last_name) WHERE CURRENT OF cursor_employees_it;
    DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);

  END LOOP;

  COMMIT;

END;
```

CURSEUR EXPLICITE PARAMETRABLE _____

```
SET SERVEROUTPUT ON;

DECLARE
  CURSOR cursor_employees_it(dep NUMBER) IS SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = dep FOR UPDATE OF last_name;

BEGIN

  FOR employee IN cursor_employees_it(60) LOOP

    UPDATE employees SET  last_name = UPPER(last_name) WHERE CURRENT OF cursor_employees_it;
    DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name);

  END LOOP;

  COMMIT;

END;
```

CURSEUR EXPLICITE 3ème METHODE _____

```
SET SERVEROUTPUT ON;

BEGIN

  FOR employee IN (SELECT first_name,last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = 60) LOOP

    DBMS_OUTPUT.PUT_LINE(employee.first_name);

  END LOOP;

END;
```

EXCEPTIONS _____

```
DECLARE
  --Déclaration des variables
BEGIN
  --Traitement
EXCEPTION
  --Gestion des erreurs
END;
```

```
SET SERVEROUTPUT ON;
DECLARE
  un_entier NUMBER(1);
  valeur_trop_grande EXCEPTION;
  PRAGMA EXCEPTION_INIT(valeur_trop_grande,-6502);
BEGIN
  un_entier := 15;
EXCEPTION
  WHEN valeur_trop_grande THEN
    DBMS_OUTPUT.PUT_LINE('Nous sommes ici');
    DBMS_OUTPUT.PUT_LINE('Message de l''erreur : ' || SQLERRM);
    DBMS_OUTPUT.PUT_LINE('Code de l''erreur : '  ||  SQLCODE);
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Message de l''erreur : ' || SQLERRM);
    DBMS_OUTPUT.PUT_LINE('Code de l''erreur : '  ||  SQLCODE);
END;
```

Sortie de script

Tâche terminée en 0,015 secondes

```
bloc anonyme terminé
Nous sommes ici
Message de l'erreur : ORA-06502: PL/SQL : erreur numérique ou erreur sur une valeur: précision de NUMBER trop élevée
Code de l'erreur : -6502
```

```
SET SERVEROUTPUT ON;

DECLARE
  exception_test EXCEPTION;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Je suis avant le sous bloc');
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Je suis avant l''erreur');
    RAISE exception_test;
    DBMS_OUTPUT.PUT_LINE('Je suis apres l''erreur');
    EXCEPTION
      WHEN exception_test THEN
    DBMS_OUTPUT.PUT_LINE('Je traite l''erreur dans le sous bloc');
  END;
  DBMS_OUTPUT.PUT_LINE('Je suis après le sous bloc');
EXCEPTION
  WHEN exception_test THEN
    DBMS_OUTPUT.PUT_LINE('Je traite l''erreur');
END;
```

SEQUENCES _____

```
CREATE SEQUENCE locations_seq
       START WITH     3300
       INCREMENT BY   100
       MAXVALUE       9900
       NOCACHE
       NOCYCLE;
```

```
locations_seq.NEXTVAL
```

```
locations_seq.CURRVAL
```

```
DROP SEQUENCE departments_seq;
DROP SEQUENCE employees_seq;
DROP SEQUENCE locations_seq;
```

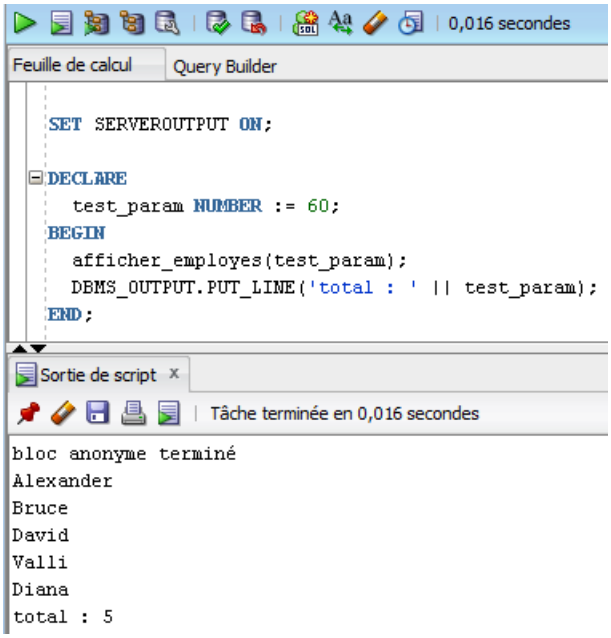
PROCEDURES (avec IN et OUT) _____

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE afficher_employes(info IN OUT NUMBER)
AS
    CURSOR cursor_employees_it IS SELECT first_name,last_name FROM EMPLOYEES WHERE DEPARTMENT_ID = info;
    employee employees%ROWTYPE;
BEGIN

  FOR employee IN cursor_employees_it LOOP
   DBMS_OUTPUT.PUT_LINE(employee.first_name);
  END LOOP;

  SELECT COUNT(*) INTO info FROM employees WHERE DEPARTMENT_ID = info;
END;
```



LES FONCTIONS _____

Les paramètres d'entrées sont exclusivement de type IN

•Les paramètres d'entrées doivent être de type SQL et non PL/SQL

•Le paramètre de retour doit être de type SQL et non PL/SQL

•Les fonctions ne doivent pas faire de DML (INSERT, UPDATE, DELETE)
Les fonctions

```
CREATE OR REPLACE FUNCTION multiplier_par_deux(nombre_a_multiplier IN NUMBER)
RETURN NUMBER
IS
resultat NUMBER;
BEGIN
  resultat := nombre_a_multiplier * 2;
  RETURN resultat;
END;
```

```
SET SERVEROUTPUT ON;

DECLARE
  exception_test EXCEPTION;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Je suis avant le sous bloc');
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Je suis avant l''erreur');
    RAISE exception_test;
    DBMS_OUTPUT.PUT_LINE('Je suis apres l''erreur');
    EXCEPTION
      WHEN exception_test THEN
    DBMS_OUTPUT.PUT_LINE('Je traite l''erreur dans le sous bloc');
  END;
  DBMS_OUTPUT.PUT_LINE('Je suis après le sous bloc');
EXCEPTION
  WHEN exception_test THEN
    DBMS_OUTPUT.PUT_LINE('Je traite l''erreur');
END;
```

LES TRIGGERS_____

Les spécificités
•Blocs associés à un nom

•Peuvent appeler des sous-programmes

•Non paramétrables

•COMMIT et ROLLBACK interdits

```
CREATE OR REPLACE TRIGGER securisation_horaires
  BEFORE INSERT OR UPDATE OR DELETE ON employees
BEGIN
    IF TO_CHAR (SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '18:00' OR TO_CHAR (SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
      IF INSERTING THEN
        RAISE_APPLICATION_ERROR (-20205, 'Vous ne pouvez pas faire d''insertions hors des heures normales de bureau');
      END IF;

      IF DELETING THEN
        RAISE_APPLICATION_ERROR (-20206, 'Vous ne pouvez pas faire de suppressions hors des heures normales de bureau');
      END IF;

      IF UPDATING THEN
        RAISE_APPLICATION_ERROR (-20207, 'Vous ne pouvez pas faire de mises à jour hors des heures normales de bureau');
      END IF;
    END IF;
END;
```

```
CREATE OR REPLACE TRIGGER securisation_salaire
BEFORE UPDATE ON employees
FOR EACH ROW
BEGIN
  IF(:OLD.salary > :NEW.salary) THEN
    RAISE_APPLICATION_ERROR(-20210,'Impossible de réduire un salaire');
  END IF;
END;
```

```
CREATE OR REPLACE TRIGGER securisation_salaire
BEFORE UPDATE ON employees
FOR EACH ROW
WHEN(OLD.salary > NEW.salary)
BEGIN
    RAISE_APPLICATION_ERROR(-20210,'Impossible de réduire un salaire');
END;
```

TRIGGERS DE VUE

```
CREATE OR REPLACE VIEW view_countries_regions
AS
  SELECT
    c.country_name AS country,
    r.region_name AS region
  FROM
    countries c
    JOIN regions r ON c.region_id = r.region_id;
```

```
CREATE OR REPLACE TRIGGER insert_view_countries_regions
INSTEAD OF INSERT ON view_countries_regions
DECLARE
  total_region NUMBER;
  new_id_region NUMBER;
  total_country NUMBER;
  new_id_country NUMBER;
BEGIN

  SELECT COUNT(*) INTO total_region FROM REGIONS WHERE REGION_NAME = :NEW.region;
  IF total_region = 0 THEN
    SELECT (MAX(REGION_ID)+1) INTO new_id_region FROM REGIONS;
    INSERT INTO regions VALUES(new_id_region,:NEW.region);
  END IF;

  SELECT COUNT(*) INTO total_country FROM countries WHERE COUNTRY_NAME = :NEW.country;
  IF total_country = 0 THEN
    SELECT (MAX(REGION_ID)+1) INTO new_id_country FROM countries;
    INSERT INTO countries VALUES(new_id_country,:NEW.country,new_id_region);
  END IF;
END;
```