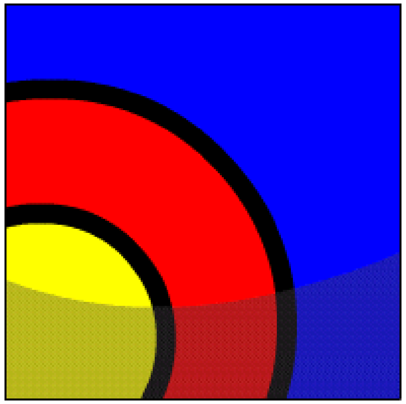
# Creating PL/SQL Blocks

## What Will I Learn?

In this lesson, you will learn how to:

- Describe the structure of a PL/SQL block
- Identify the different types of PL/SQL blocks
- Identify PL/SQL programming environments
- Create and execute an anonymous PL/SQL block
- Output messages in PL/SQL



## Why Learn It?

This lesson introduces the PL/SQL block structure, the basic unit in PL/SQL. All PL/SQL programs are comprised of blocks, so you use blocks a lot! You will learn the structure of a PL/SQL block and create one kind of block: an anonymous block.

After learning about the different environments you can use to develop your PL/SQL programs, you will also begin coding PL/SQL in the Application Express development environment.

## Tell Me/Show Me

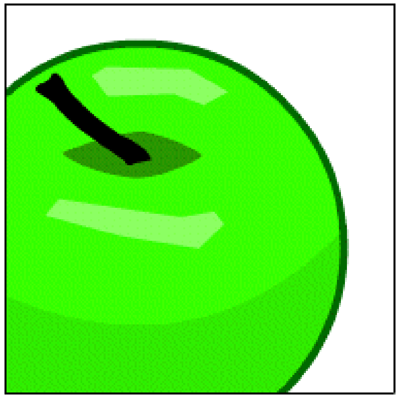
### PL/SQL Block Structure

A PL/SQL block consists of three sections:

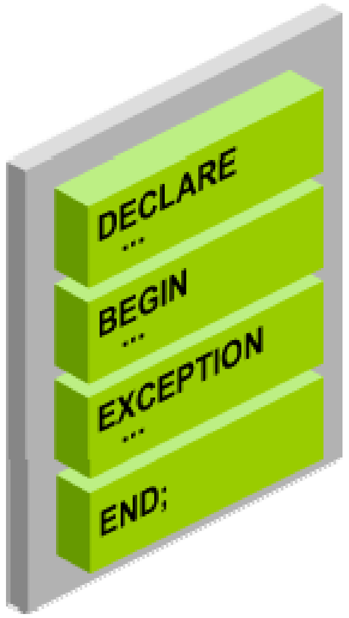
**Declarative (optional):** The declarative section begins with the keyword `DECLARE` and ends when your executable section starts.

**Executable (mandatory):** The executable section begins with the keyword `BEGIN` and ends with `END`. Observe that `END` is terminated with a semicolon. The executable section of a PL/SQL block can include any number of nested PL/SQL blocks.

**Exception handling (optional):** The exception section is nested within the executable section. This section begins with the keyword `EXCEPTION`.



## Tell Me/Show Me



## Tell Me/Show Me

### PL/SQL Block Structure (continued)

| Section | Description | Inclusion |
|---|---|---|
| Declarative (`DECLARE`) | Contains declarations of all variables, constants, cursors, and user-defined exceptions that are referenced in the executable and exception sections. | Optional |
| Executable (`BEGIN … END;`) | Contains SQL statements to retrieve data from the database and PL/SQL statements to manipulate data in the block. Must contain at least one statement. | Mandatory |
| Exception (`EXCEPTION`) | Specifies the actions to perform when errors and abnormal conditions arise in the executable section. | Optional |

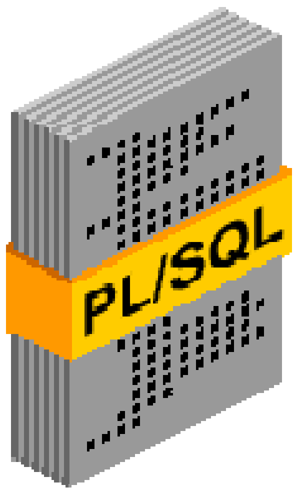
## Tell Me/Show Me

**The PL/SQL Compiler**

Every program written in a high-level programming language (C, Java, PL/SQL and so on) must be checked and translated into binary code (ones and zeros) before it can execute. The software that does this checking and translation is called a compiler.

The PL/SQL compiler executes automatically when needed. It checks not only that every word is spelled correctly, but also that any referenced database objects (such as tables) exist, and that the user has the necessary privileges on them.

## Tell Me/Show Me



## Tell Me/Show Me

**Anonymous Blocks**

- Unnamed blocks
- Not stored in the database
- Declared inline at the point in an application where they are executed
- Compiled each time the application is executed
- Passed to the PL/SQL engine for execution at run time
- You cannot invoke or call an anonymous block because it does not have a name and does not exist after it is executed

```
[DECLARE]


BEGIN
  --statements

[EXCEPTION]

END;
```

## Tell Me/Show Me

### Examples of Anonymous Blocks

1. No Declaration or exception sections, execution only

```
BEGIN
  DBMS_OUTPUT.PUT_LINE('PL/SQL is easy!');
END;
```

2. Declaration and execution sections, but no exception section

```
DECLARE
  v_date DATE := SYSDATE;
BEGIN
  DBMS_OUTPUT.PUT_LINE(v_date);
END;
```

## Tell Me/Show Me

### Examples of Anonymous Blocks

3. Declaration and exception sections

```
DECLARE
  v_country_name VARCHAR2(40);
  v_region_id     NUMBER;
BEGIN
  SELECT country_name, region_id
    INTO v_country_name, v_region_id
    FROM countries WHERE country_id=‘CA’;
  DBMS_OUTPUT.PUT_LINE ('The country name is: '
             ||v_country_name||' and is located in '
             ||v_region_id||'.') ;
EXCEPTION
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE (' Your select statement retrieved
      multiple rows. Consider using a cursor.');
END;
```

## Tell Me/Show Me

**Subprograms**

- Are named PL/SQL blocks
- Are stored in the database
- Can be invoked whenever you want depending on your application
- Can be declared as procedures or as functions
  - Procedure: Performs an action
  - Function: Computes and returns a value

```
PROCEDURE name
IS
  --variable declaration(s)
BEGIN
  --statements

[EXCEPTION]

END;
```

```
FUNCTION name
RETURN datatype
  --variable declaration(s)
IS
BEGIN
  --statements
  RETURN value;

[EXCEPTION]

END;
```

## Tell Me/Show Me

### Examples of Subprograms

1. Procedure to print the current date

```
CREATE PROCEDURE print_date IS
  v_date VARCHAR2(30);
BEGIN
  SELECT TO_CHAR(SYSDATE,'Mon DD, YYYY')
    INTO v_date
    FROM DUAL;
  DBMS_OUTPUT.PUT_LINE(v_date);
END;
```

2. Function to return the number of characters in a string

```
CREATE FUNCTION num_characters (p_string IN VARCHAR2)
  RETURN INTEGER IS
  v_num_characters INTEGER;
BEGIN
  SELECT LENGTH(p_string) INTO v_num_characters
    FROM DUAL;
    RETURN v_num_characters;
END;
```

## Tell Me/Show Me

### Program Constructs

The following table outlines a variety of different PL/SQL program constructs that use the basic PL/SQL block. The constructs are available based on the environment in which they are executed.

| ORACLE DEVELOPER SUITE 10g | ORACLE DATABASE 10g |
|---|---|
| **Tools constructs** | **Database server constructs** |
| **Anonymous blocks** | **Stored procedures or functions** |
| **Application procedures or functions** | **Stored packages** |
| **Application packages** | **Database triggers** |
| **Application triggers** | **Object types** |
| **Object types** | |

## Tell Me / Show Me

### PL/SQL Programming Environments

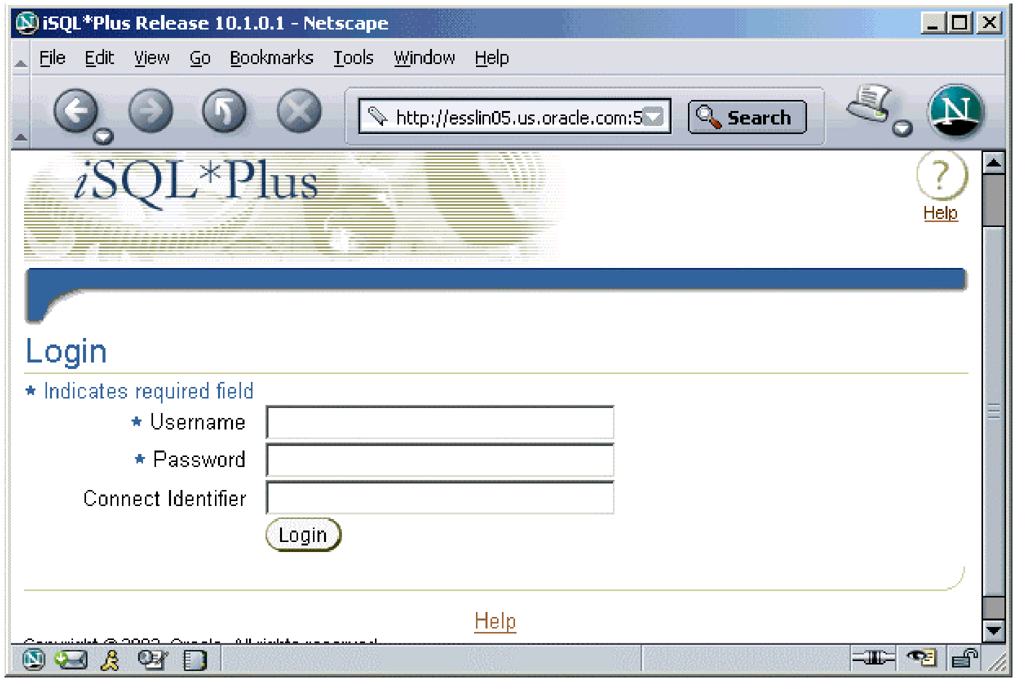
There are many tools that provide an environment for developing PL/SQL code. Oracle provides several tools that you can use to write PL/SQL code. Some of the Oracle development tools are:

| | |
|---|---|
| **SQL*Workshop** | A component of Application Express |
| **SQL*Plus** | A command-line application |
| **Jdeveloper & SQL Developer** | GUI-integrated development environments (IDEs) |
| **iSQL*Plus** | A browser-based development environment |

## Tell Me / Show Me

### PL/SQL Programming Environments: *i*SQL*Plus

***i*SQL*Plus:** *i*SQL*Plus is a browser-based interface to SQL*Plus. You can connect to a local or remote database by using *i*SQL*Plus. It allows you to perform all the operations that you can perform with the command-line version of SQL*Plus.


iSQL*Plus Release 10.1.0.1 - Netscape
File Edit View Go Bookmarks Tools Window Help
http://esslin05.us.oracle.com:5
Search
iSQL*Plus
Help
Login
* Indicates required field
* Username
* Password
Connect Identifier
Login
Help

## Tell Me/Show Me

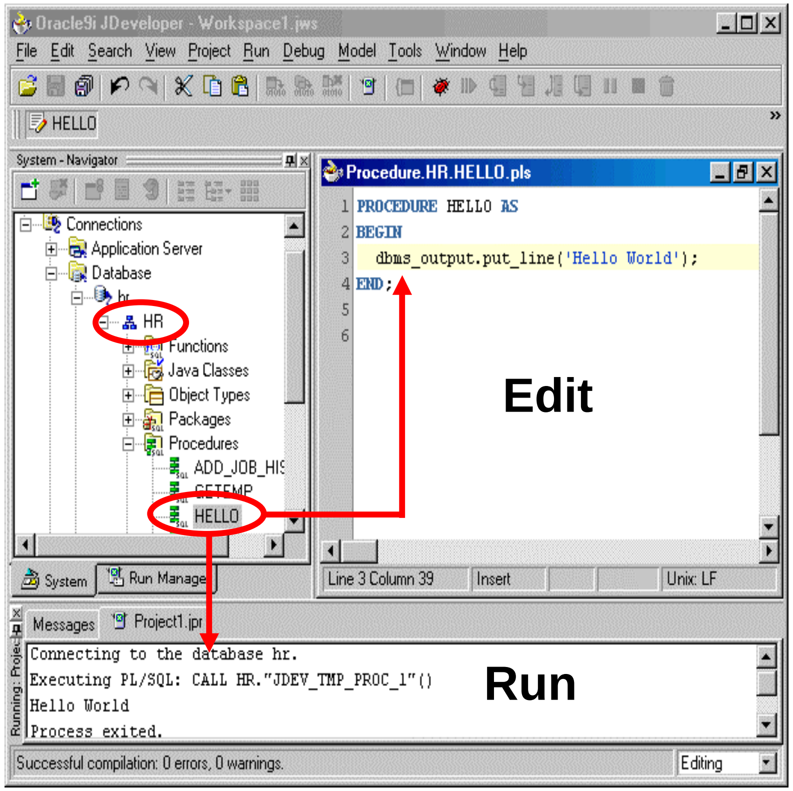
**PL/SQL Programming Environments: Oracle JDeveloper**

JDeveloper is a Windows-based application.

Using JDeveloper, you can create, edit, test, and debug PL/SQL.

The JDeveloper Code Editor assists in PL/SQL code development by offering:

- Different colors for syntactical components of the PL/SQL language
- Features to locate procedures and functions in supplied packages



```
PROCEDURE HELLO AS
BEGIN
  dbms_output.put_line('Hello World');
END;
```

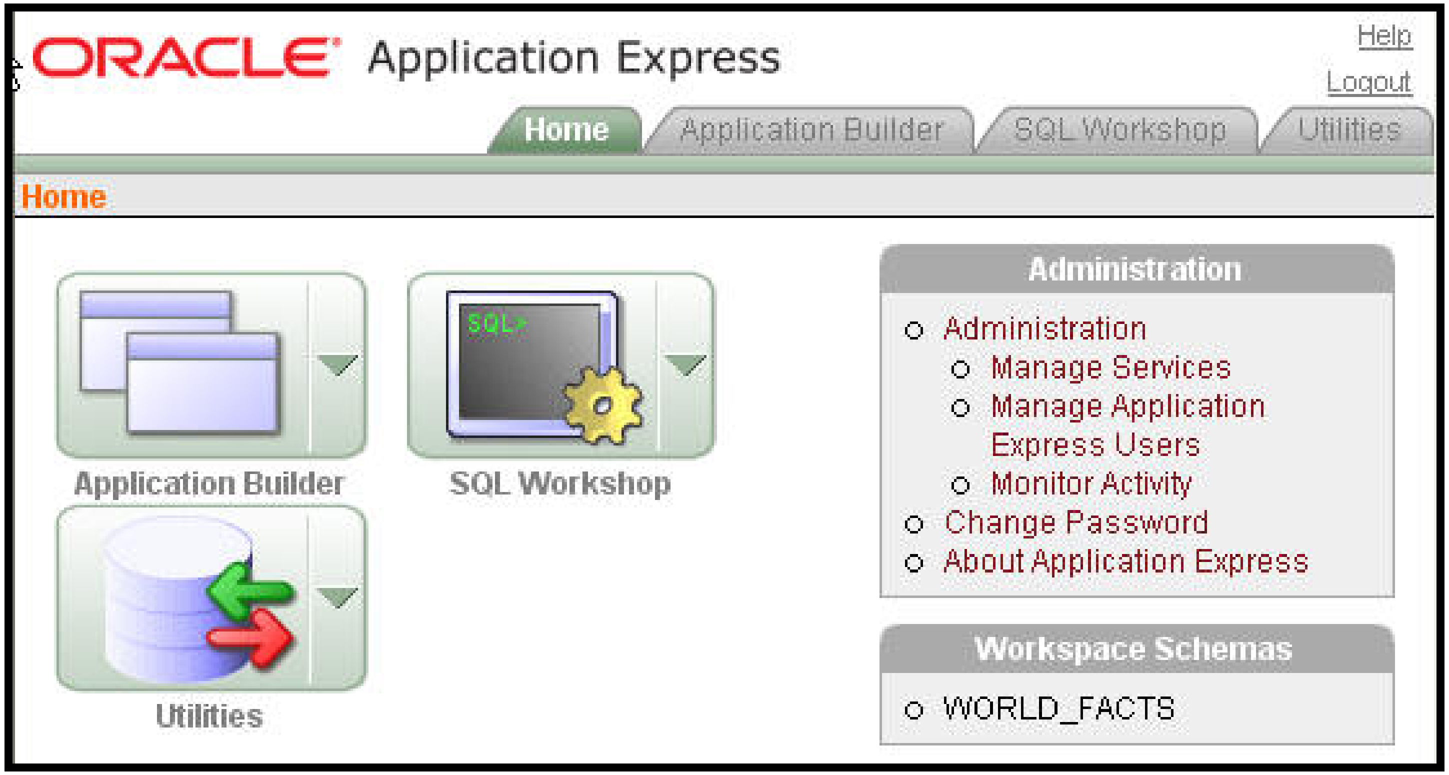
```
Connecting to the database hr.
Executing PL/SQL: CALL HR."JDEV_TMP_PROC_1"()
Hello World
Process exited.
```

Successful compilation: 0 errors, 0 warnings.

## Tell Me/Show Me

**PL/SQL Programming Environments: Oracle Application Express**

Oracle Application Express is a browser-based web application environment that offers a SQL Workshop component.



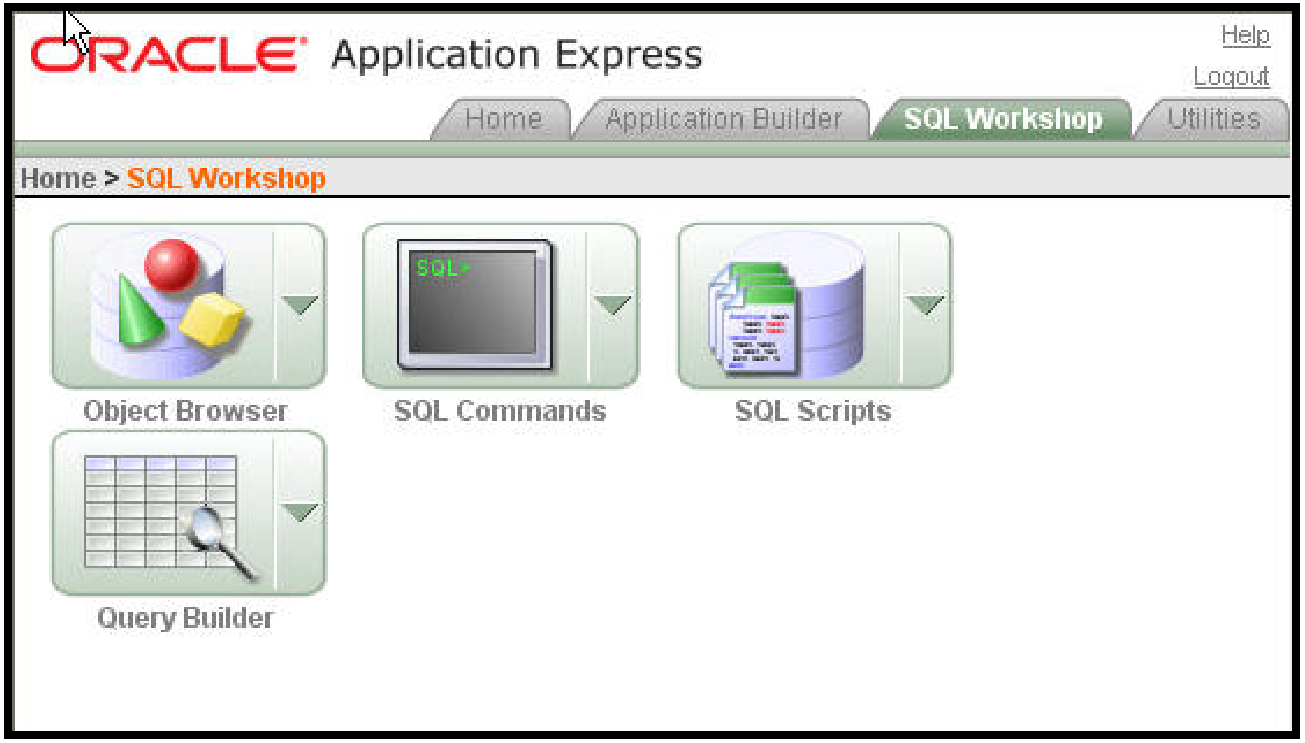
- Administration
  - Manage Services
  - Manage Application Express Users
  - Monitor Activity
- Change Password
- About Application Express

- WORLD_FACTS

## Tell Me/Show Me

### Developing with SQL Workshop

When you log in to Oracle Application Express and choose SQL Workshop, you can choose to use the SQL Commands option to use the SQL command-line editor or you can choose the SQL Scripts option to work within the Script Editor.

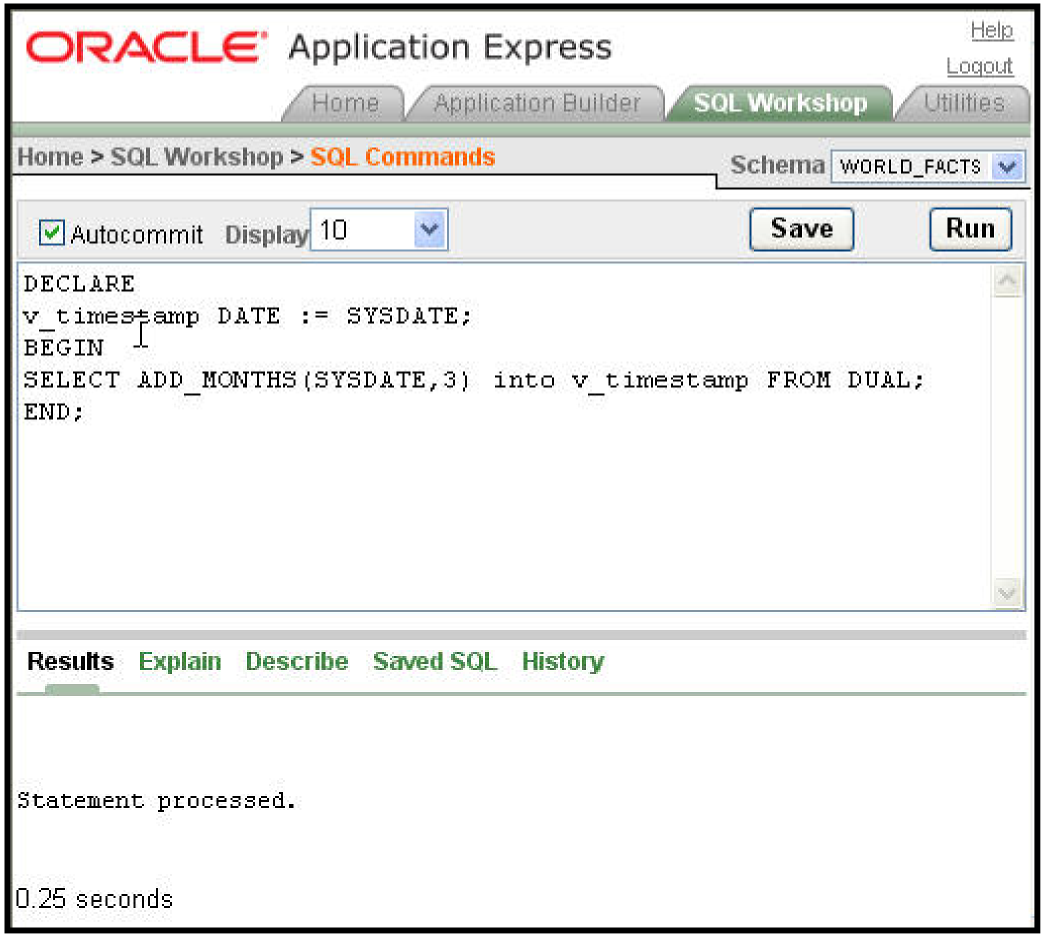


## Tell Me/Show Me

### SQL Commands

You can use **SQL Commands** to enter and run a single SQL statement or a single PL/SQL block.

A SQL script can contain one or more SQL statements and/or PL/SQL blocks. Use **SQL Scripts** to enter and run multi-statement scripts.



Schema WORLD_FACTS

✓ Autocommit Display 10

```
DECLARE
v_timestamp DATE := SYSDATE;
BEGIN
SELECT ADD_MONTHS(SYSDATE,3) into v_timestamp FROM DUAL;
END;
```

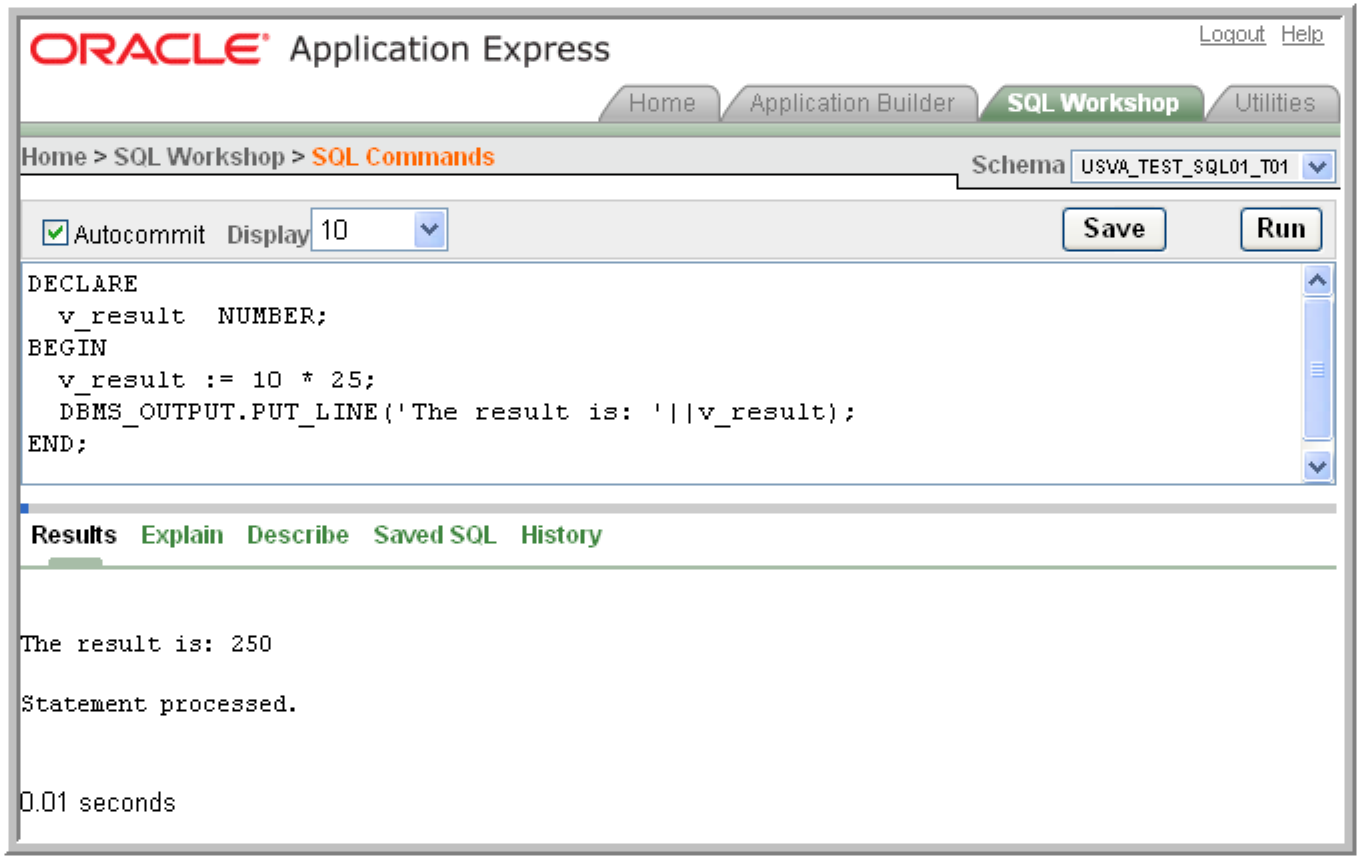
```
Statement processed.

0.25 seconds
```

## Tell Me/Show Me

## Tell Me/Show Me

### Using `DBMS_OUTPUT.PUT_LINE` (continued)

Let's add a call to `DBMS_OUTPUT.PUT_LINE`:


ORACLE Application Express
Logout Help
Home
Application Builder
SQL Workshop
Utilities
Home > SQL Workshop > SQL Commands
Schema USVA_TEST_SQL01_T01
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
DECLARE
  v_result  NUMBER;
BEGIN
  v_result := 10 * 25;
  DBMS_OUTPUT.PUT_LINE('The result is: '||v_result);
END;
```

```
The result is: 250

Statement processed.


0.01 seconds
```

Now you can see the result !

## Tell Me/Show Me

**Using DBMS_OUTPUT.PUT_LINE (continued)**

The DBMS_OUTPUT.PUT_LINE allows you to display results so that you can check that your block is working correctly. It allows you to display one character string at a time, although this can be concatenated. Here are two more examples:

```
DECLARE
  v_emp_count    NUMBER;
BEGIN
  DBMS_OUTPUT.PUT_LINE('PL/SQL is easy so far!');
  SELECT COUNT(*) INTO v_emp_count FROM employees;
  DBMS_OUTPUT.PUT_LINE(‘There are '||v_emp_count||' rows
                        in the employees table’);
END;
```

## Tell Me/Show Me

**Terminology**

Key terms used in this lesson include:

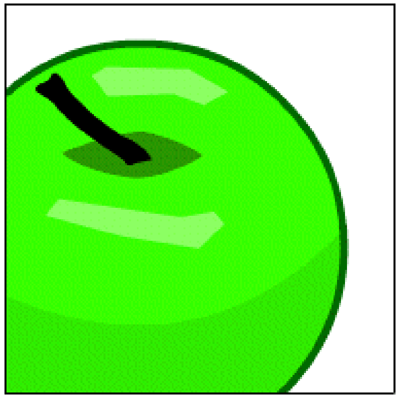
Anonymous PL/SQL block

Compiler

Subprograms

Procedures

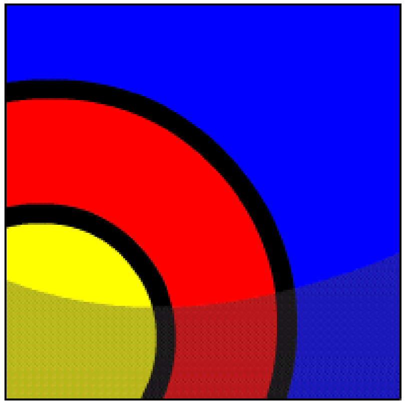
Functions



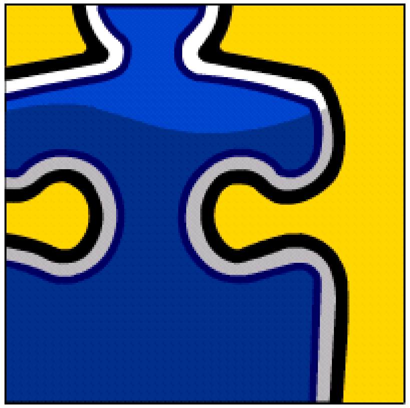
## Summary

In this lesson, you learned how to:

- Describe the structure of a PL/SQL block
- Identify the different types of PL/SQL blocks
- Identify PL/SQL programming environments
- Create and execute an anonymous PL/SQL block
- Output messages in PL/SQL



## Try It/Solve It



The exercises in this lesson cover the following topics:

- Describing the structure of a PL/SQL block
- Identifying the block types of PL/SQL
- Identifying PL/SQL programming environments
- Creating and executing an anonymous PL/SQL block
- Outputting messages in PL/SQL