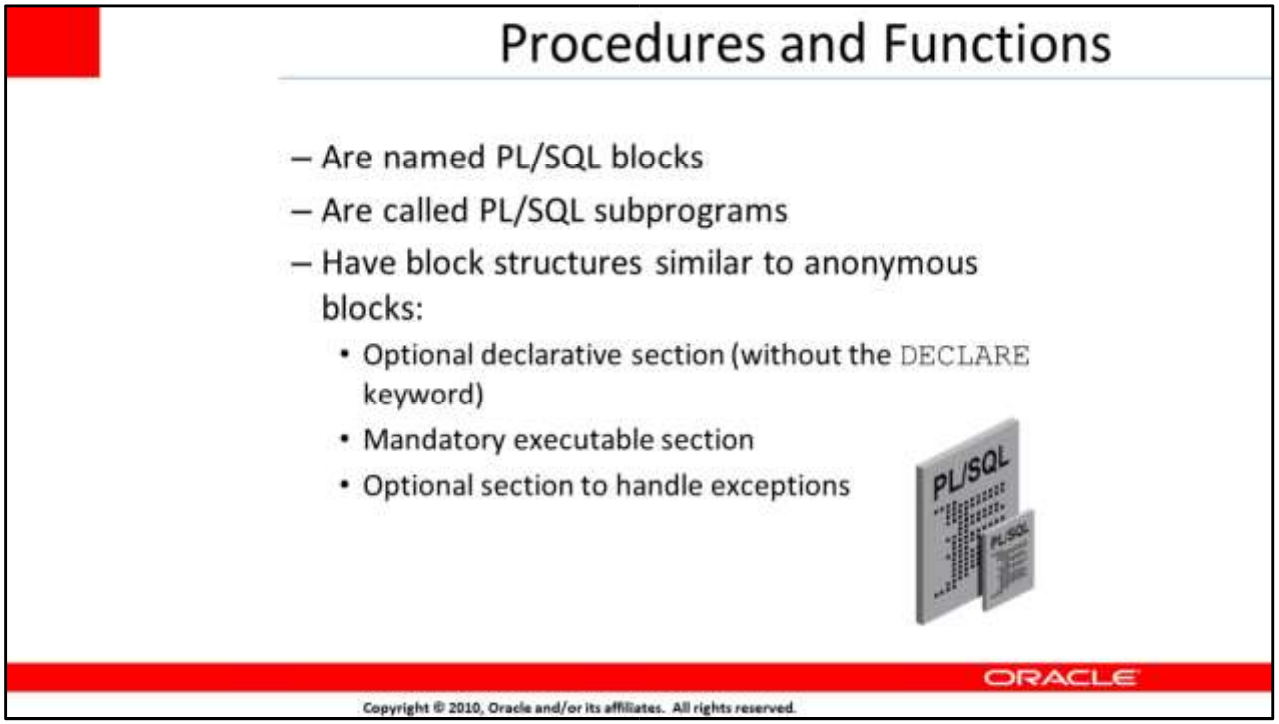


Procedures and Functions

Up to this point, anonymous blocks were the only examples of PL/SQL code covered in this course. As the name indicates, anonymous blocks are unnamed executable PL/SQL blocks. Because they are unnamed, they can be neither reused nor stored for later use.

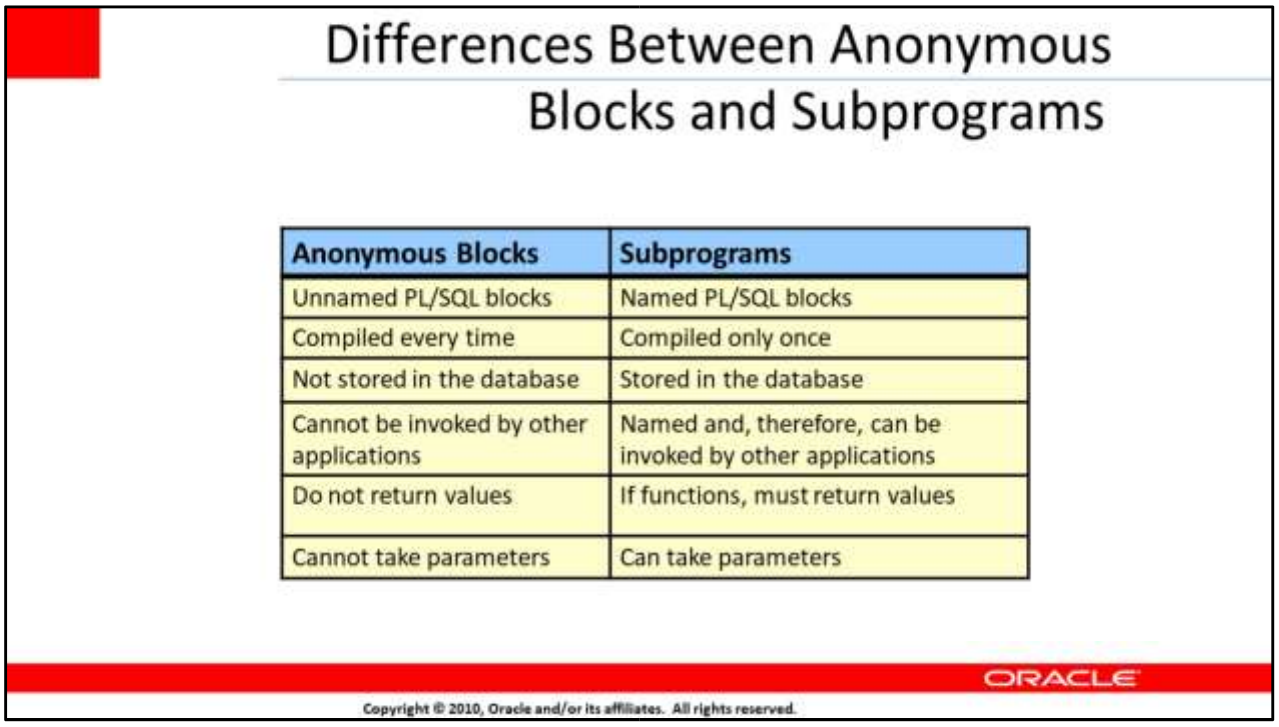
Procedures and functions are named PL/SQL blocks that are also known as subprograms. These subprograms are compiled and stored in the database. The block structure of the subprograms is similar to the structure of anonymous blocks. Subprograms can be declared not only at the schema level but also within any other PL/SQL block. A subprogram contains the following sections:

Declarative section: Subprograms can have an optional declarative section. However, unlike anonymous blocks, the declarative section of a subprogram does not start with the DECLARE keyword. The optional declarative section follows the IS or AS keyword in the subprogram declaration.

Executable section: This is the mandatory section of the subprogram, which contains the implementation of the business logic. Looking at the code in this section, you can easily determine the business

functionality of the subprogram. This section begins and ends with the BEGIN and END keywords, respectively.

Exception section: This is an optional section that is included to handle exceptions.



Differences Between Anonymous Blocks and Subprograms

The table in the slide not only shows the differences between anonymous blocks and subprograms, but also highlights the general benefits of subprograms. Anonymous blocks are not persistent database objects. They are compiled every time they are to be executed. They are not stored in the database for reuse. If you want to reuse them, you must rerun the script that creates the anonymous block, which causes recompilation and execution. Procedures and functions are compiled and stored in the database in a compiled form. They are recompiled only when they are modified. Because they are stored in the database, any application can make use of these subprograms based on appropriate permissions. The calling application can pass parameters to the procedures if the procedure is designed to accept parameters. Similarly, a calling application can retrieve a value if it invokes a function or a procedure.