

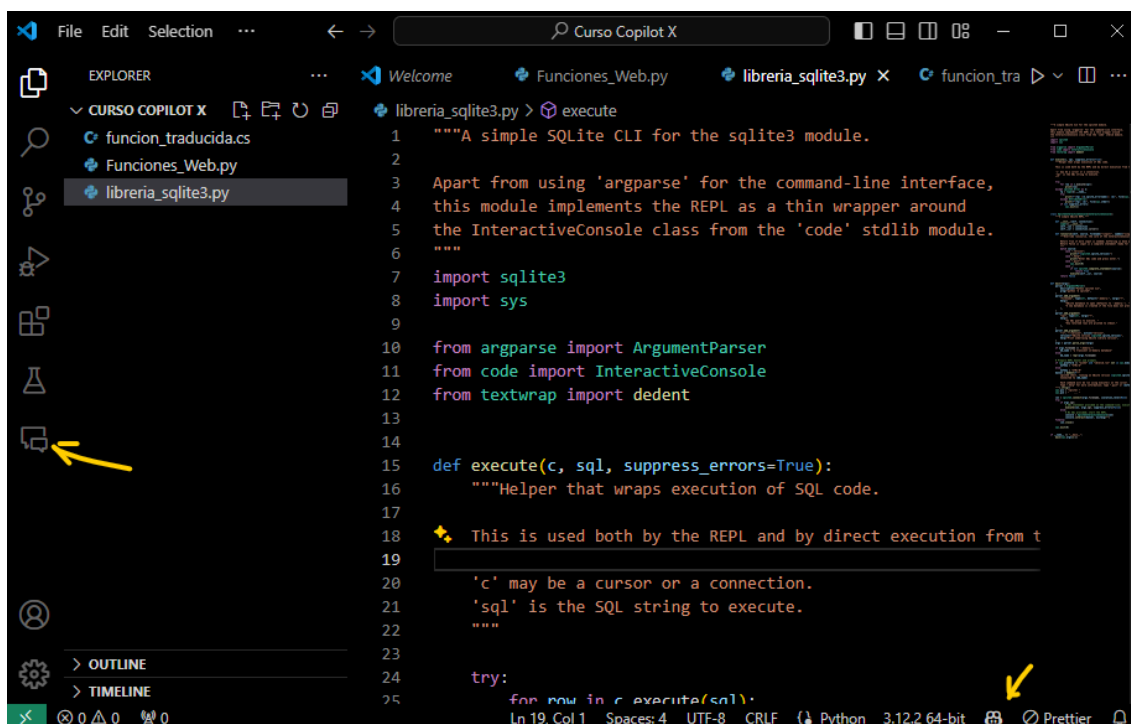
Práctica: Traducción de código con GitHub Copilot

Contenido

Práctica: Traducción de código con GitHub Copilot	1
Selección del código	1
Traducción a otro lenguaje de programación	2
Supervisión y análisis	3

Selección del código

1. Abrir un documento de código propio o descargar y utilizar *libreria_sqlite3.py*
2. En la barra inferior, a la derecha pinchar el icono de GitHub Copilot si no aparece el icono de Chat aún en la barra izquierda



3. Seleccionar parte del código a traducir
4. Consultar qué propósito tiene el código seleccionado en Copilot Chat

GitHub Copilot

Used 1 reference >

El código resaltado es una función de Python llamada `execute` que se utiliza para ejecutar comandos SQL en una base de datos SQLite3.

Aquí está lo que hace cada parte:

1. La función `execute` toma tres argumentos: `c`, `sql` y `suppress_errors`. `c` puede ser un cursor o una conexión a la base de datos SQLite3. `sql` es la cadena de texto que contiene el comando SQL que se va a ejecutar. `suppress_errors` es un valor booleano que determina si la función debe terminar el programa en caso de un error o simplemente imprimir el error y continuar.
2. Dentro de un bloque `try`, la función intenta ejecutar el comando SQL con `c.execute(sql)`. Si la ejecución es exitosa, imprime cada fila del resultado.
3. Si ocurre un error durante la ejecución del comando SQL, se lanza una excepción `sqlite3.Error`. Esta excepción se captura en el bloque `except` y se imprime en la salida de error estándar (`sys.stderr`). El tipo de error y el mensaje de error se incluyen en la salida.
4. Si `suppress_errors` es `False`, la función termina el programa con un código de salida de 1 para indicar que ocurrió un error. Si `suppress_errors` es `True`, la función simplemente imprime el error y continúa, permitiendo que el programa maneje el error de la manera que prefiera.

```

6
7 import sqlite3
8 import sys
9
10 from argparse import ArgumentParser
11 from code import InteractiveConsole
12 from textwrap import dedent
13
14
15 def execute(c, sql, suppress_errors=True):
16     """Helper that wraps execution of SQL code.
17
18     This is used both by the REPL and by direct execution from the CLI.
19
20     'c' may be a cursor or a connection.
21     'sql' is the SQL string to execute.
22     """
23
24     try:
25         for row in c.execute(sql):
26             print(row)
27     except sqlite3.Error as e:
28         tp = type(e).__name__
29         try:
30             print(f"{tp}({e.sqlite_errname}): {e}", file=sys.stderr)
31         except AttributeError:
32             print(f"{tp}: {e}", file=sys.stderr)
33         if not suppress_errors:
34             sys.exit(1)

```

Traducción a otro lenguaje de programación

5. Con un fragmento de código seleccionado:
 - a. Pulsar en las estrellas en la parte superior para abrir un cuadro de diálogo
 - b. Escribir en el chat directamente una petición de traducción

✦

🔊
▶
✕
⋮

🔗 You can also type / for commands

```

def execute(c, sql, suppress_errors=True):
    """Helper that wraps execution of SQL code.

    This is used both by the REPL and by direct execution from the CLI.

    'c' may be a cursor or a connection.
    'sql' is the SQL string to execute.
    """
    try:
        for row in c.execute(sql):
            print(row)
    except sqlite3.Error as e:
        tp = type(e).__name__
        try:
            print(f"{tp}({e.sqlite_errname}): {e}", file=sys.stderr)
        except AttributeError:
            print(f"{tp}: {e}", file=sys.stderr)
        if not suppress_errors:
            sys.exit(1)

```

```

9
10 from argparse import ArgumentParser
11 from code import InteractiveConsole
12 from textwrap import dedent
13
14
15 def execute(c, sql, suppress_errors=True):
16     """Helper that wraps execution of SQL code.
17     This is used both by the REPL and by direct execution from the CLI.
18     'c' may be a cursor or a connection.
19     'sql' is the SQL string to execute.
20
21     try:
22         for row in c.execute(sql):
23             print(row)
24     except sqlite3.Error as e:
25         tp = type(e).__name__
26         print(f"{tp} ({e.sqlite_errname}): {e}", file=sys.stderr)
27     except AttributeError:
28         print(f"{tp}: {e}", file=sys.stderr)
29     if not suppress_errors:
30         sys.exit(1)
31
32
33
34
35
36

```

6. Copiar el código traducido acercando el cursor a la primera línea y seleccionando Copiar, insertar en la posición del cursor o crear un archivo nuevo
7. La opción más simple será crear un nuevo archivo con extensión en el lenguaje de destino
8. Copiar el código traducido

```

1 using System;
2 using System.Data.SQLite;
3
4 public class SQLiteHelper
5 {
6     public static void Execute(SQLiteCommand cmd, string sql, bool suppressErrors = true)
7     {
8         try
9         {
10             cmd.CommandText = sql;
11             using (var reader = cmd.ExecuteReader())
12             {
13                 while (reader.Read())
14                 {
15                     for (int i = 0; i < reader.FieldCount; i++)
16                     {
17                         Console.Write($"{reader.GetValue(i)} ");
18                     }
19                     Console.WriteLine();
20                 }
21             }
22         }
23         catch (SQLiteException e)
24         {
25             Console.Error.WriteLine($"{e.GetType().Name} ({e.ErrorCode}): {e.Message}");
26             if (!suppressErrors)
27             {
28                 Environment.Exit(1);
29             }
30         }
31     }
32 }

```

Supervisión y análisis

9. Seleccionar el código traducido

10. Consultar en Copilot Chat el propósito del código seleccionado

Used 1 reference >

El código resaltado es una clase de C# llamada `SQLiteHelper` que contiene un método estático `Execute`. Este método se utiliza para ejecutar comandos SQL en una base de datos SQLite.

Aquí está lo que hace cada parte:

- El método `Execute` toma tres argumentos: `cmd`, `sql` y `suppressErrors`. `cmd` es un objeto `SQLiteCommand` que representa un comando SQL para ser ejecutado. `sql` es la cadena de texto que contiene el comando SQL que se va a ejecutar. `suppressErrors` es un valor booleano que determina si el método debe terminar el programa en caso de un error o simplemente imprimir el error y continuar.
- Dentro de un bloque `try`, el método establece el texto del comando SQL con `cmd.CommandText = sql` y luego intenta ejecutar el comando SQL con `cmd.ExecuteReader()`. Si la ejecución es exitosa, imprime cada campo de cada fila del resultado.
- Si ocurre un error durante la ejecución del comando SQL, se lanza una excepción `SQLiteException`. Esta excepción se captura en el bloque `catch` y se imprime en la salida de error estándar (`Console.Error.WriteLine`). El tipo de error, el código de error y el mensaje de error se incluyen en la salida.
- Si `suppressErrors` es `false`, el método termina el programa con un código de salida de 1 para indicar que ocurrió un error. Si `suppressErrors` es `true`, el método simplemente imprime el error y continúa, permitiendo que el programa maneje el error de la manera que prefiera.

[How can I handle transactions when executing SQL commands in C#?](#)

```

1 using System;
2 using System.Data.SQLite;
3
4 public class SQLiteHelper
5 {
6     public static void Execute(SQLiteCommand cmd, string sql, bool suppress
7     {
8         try
9         {
10             cmd.CommandText = sql;
11             using (var reader = cmd.ExecuteReader())
12             {
13                 while (reader.Read())
14                 {
15                     for (int i = 0; i < reader.FieldCount; i++)
16                     {
17                         Console.WriteLine($"{reader.GetValue(i)}");
18                     }
19                     Console.WriteLine();
20                 }
21             }
22         }
23         catch (SQLiteException e)
24         {
25             Console.Error.WriteLine($"{e.GetType().Name}-{e.ErrorCode}");
26             if (!suppressErrors)
27             {
28                 Environment.Exit(1);
29             }
30         }
31     }
32 }

```

11. Comparar con el propósito del código original a nivel general

12. Revisar las variables y el flujo de ejecución

13. Comprobar si los métodos o funciones traducidos mantienen el mismo uso $O(n, n^2, \dots)$