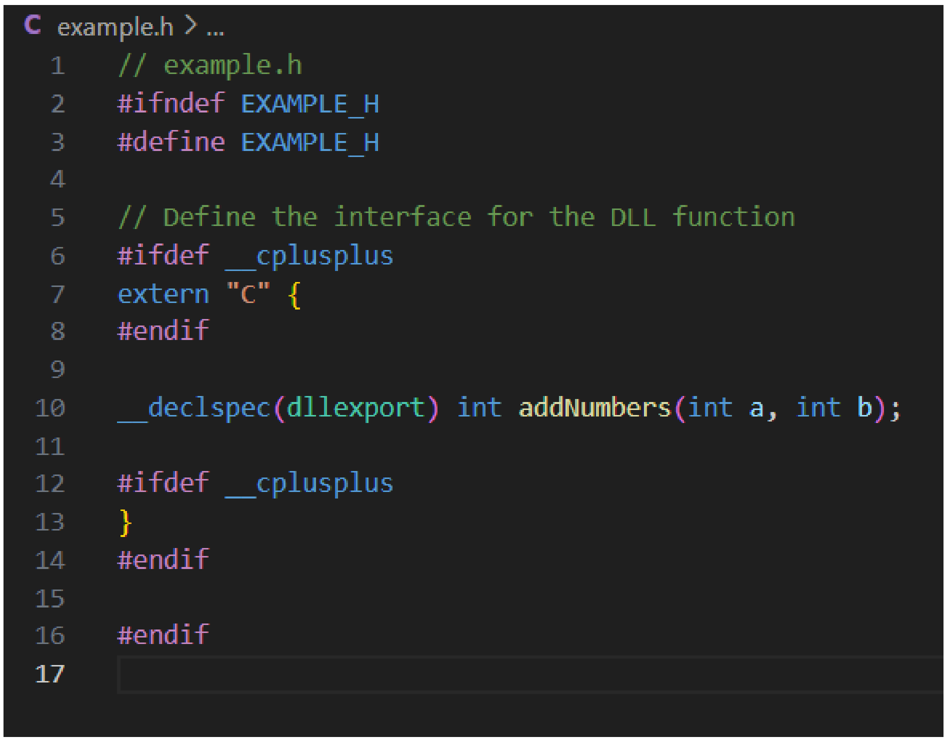
How to use .dll file in python using ctypes.

STEP 1 :-
Make the header file and save it with “.h” extension.



example.h > ...

```
// example.h
#ifndef EXAMPLE_H
#define EXAMPLE_H

// Define the interface for the DLL function
#ifdef __cplusplus
extern "C" {
#endif

__declspec(dllexport) int addNumbers(int a, int b);

#ifdef __cplusplus
}
#endif

#endif
```

```
// example.h
#ifndef EXAMPLE_H
#define EXAMPLE_H

// Define the interface for the DLL function
#ifdef __cplusplus
extern "C" {
#endif

__declspec(dllexport) int addNumbers(int a, int b);

#ifdef __cplusplus
}
#endif

#endif
```

This code is a header file (example.h) for a C++ Dynamic Link Library (DLL). Let's break down its components and their purposes:

1. Header Guards (#ifndef, #define, #endif):

- These lines prevent the header file from being included more than once in the same translation unit (source file) during the compilation process. This is done to avoid issues with redefinition of symbols and to improve compilation efficiency.

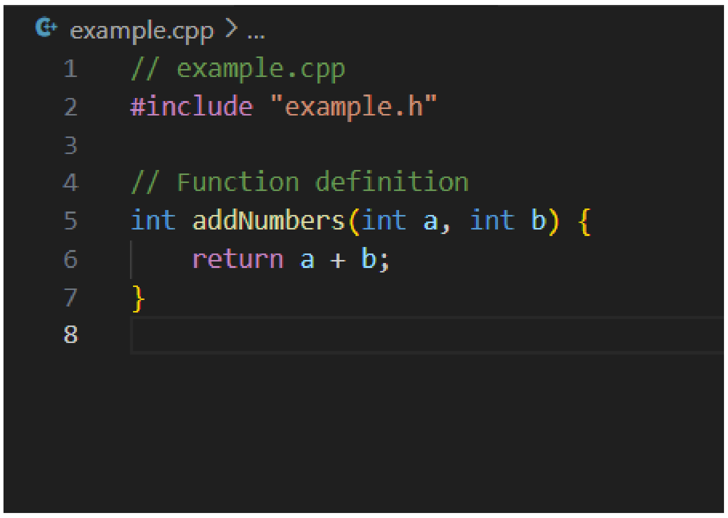
2. Extern "C" Directive:
    - The extern "C" block is used to declare that the functions declared within it should have C linkage. This ensures that the function names are not "mangled" by the C++ compiler, making them accessible to other programming languages, such as C.

3. __declspec(dllexport):
    - This is a Microsoft-specific extension used to export symbols (functions or variables) from a DLL. It tells the compiler to export the addNumbers function so that it can be accessed from outside the DLL.

4. int addNumbers(int a, int b);:
    - This is the declaration of the addNumbers function. It specifies that the function takes two integer arguments (a and b) and returns an integer. The function is intended to be used to add two numbers together.

In summary, this header file defines the interface for a DLL function called addNumbers, ensuring that it can be accessed from outside the DLL, even by languages other than C++. The extern "C" block and __declspec(dllexport) attribute are used to achieve this interoperability.

STEP 2 :-
Make the c++ file and add the header file (.h) using #include ”[header.h]” and write the desire code you what and save it with .cpp extension.
For example here we are creating a function to add to integer values.



```cpp
// example.cpp
#include "example.h"

// Function definition
int addNumbers(int a, int b) {
    return a + b;
}
```

```
// example.cpp
#include "example.h"

// Function definition
int addNumbers(int a, int b) {
   return a + b;
}
```

STEP 3 :-
Compile this code into a DLL. If you're using MinGW on Windows, you can compile it like this:

g++ -shared -o example.dll example.cpp

Using terminal /cmd
NOTE :- If you are using cmd make sure you are in the directory where the c++ file is present.

This command is using the GNU Compiler Collection (GCC), specifically the g++ command, to compile a C++ source file (example.cpp) into a shared library, also known as a Dynamic Link Library (DLL) on Windows

- g++: This is the GNU C++ compiler. It is capable of compiling both C and C++ source code.
- -shared: This option tells the compiler to generate a shared library instead of a standalone executable. Shared libraries contain code that can be linked and loaded at runtime by multiple programs.
- -o example.dll: This option specifies the name of the output file. In this case, it's example.dll. The -o flag is followed by the name of the output file.
- example.cpp: This is the source file that the compiler will compile into the shared library. In this example, example.cpp contains the implementation of the functions that will be included in the DLL.
-

STEP 4 :-
Once you have the example.dll DLL file, you can use it in Python with ctypes:

This Python script demonstrates how to load and use a DLL file (example.dll) using the ctypes module. Let's break down each part of the script:

1. Importing ctypes Module: import ctypes:

   - This line imports the ctypes module, which provides a foreign function interface (FFI) for calling functions and manipulating data types in DLLs.

2. Loading the DLL:

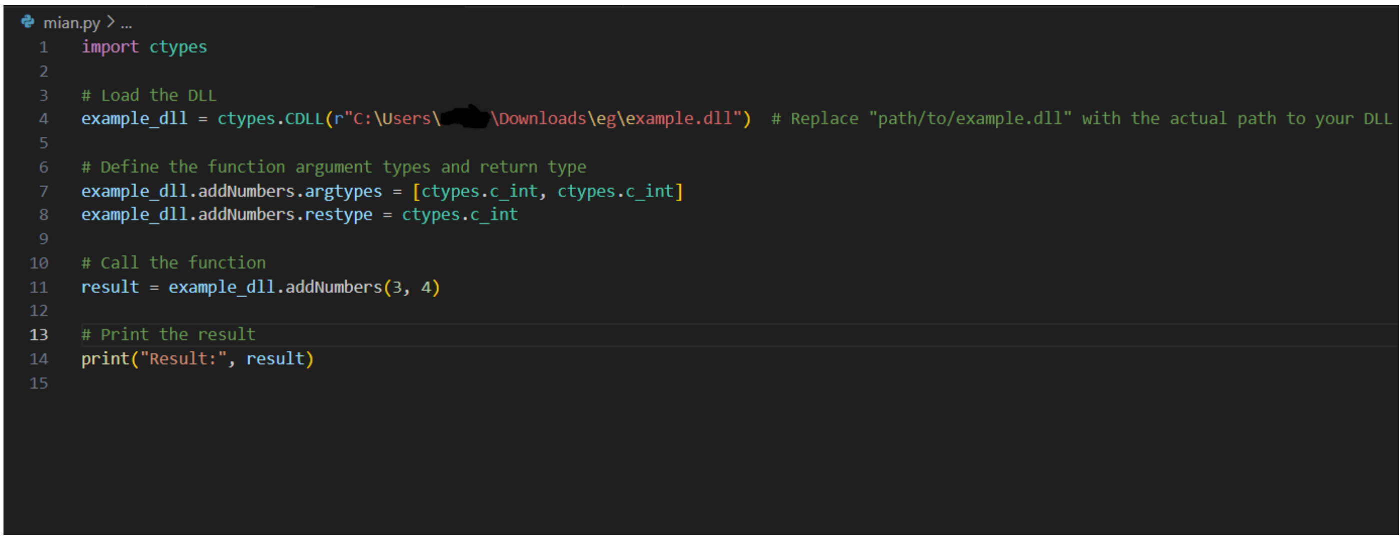
- example_dll = ctypes.CDLL("path/to/example.dll"): This line loads the DLL file named "example.dll" into the Python script. The ctypes.CDLL() function takes the path to the DLL file as an argument. Replace "path/to/example.dll" with the actual path to your DLL file.

3. Defining Function Argument Types and Return Type:

    - example_dll.addNumbers.argtypes = [ctypes.c_int, ctypes.c_int]: This line specifies the argument types of the addNumbers function in the DLL. In this case, it expects two integer arguments.
    - example_dll.addNumbers.restype = ctypes.c_int: This line specifies the return type of the addNumbers function, which is an integer.

4. Calling the Function:
    - result = example_dll.addNumbers(3, 4): This line calls the addNumbers function from the DLL with arguments 3 and 4. The ctypes module automatically converts Python integers to C integers before passing them to the function.

5. Printing the Result:
    - print("Result:", result): This line prints the result returned by the addNumbers function. The result variable contains the sum of 3 and 4.



```
import ctypes

# Load the DLL
example_dll = ctypes.CDLL(r"C:\Users\[redacted]\Downloads\eg\example.dll")  # Replace "path/to/example.dll" with the actual path to your DLL

# Define the function argument types and return type
example_dll.addNumbers.argtypes = [ctypes.c_int, ctypes.c_int]
example_dll.addNumbers.restype = ctypes.c_int

# Call the function
result = example_dll.addNumbers(3, 4)

# Print the result
print("Result:", result)
```

import ctypes

Load the DLL
example_dll = ctypes.CDLL("path/to/example.dll") # Replace "path/to/example.dll" with the actual path to your DLL

Define the function argument types and return type

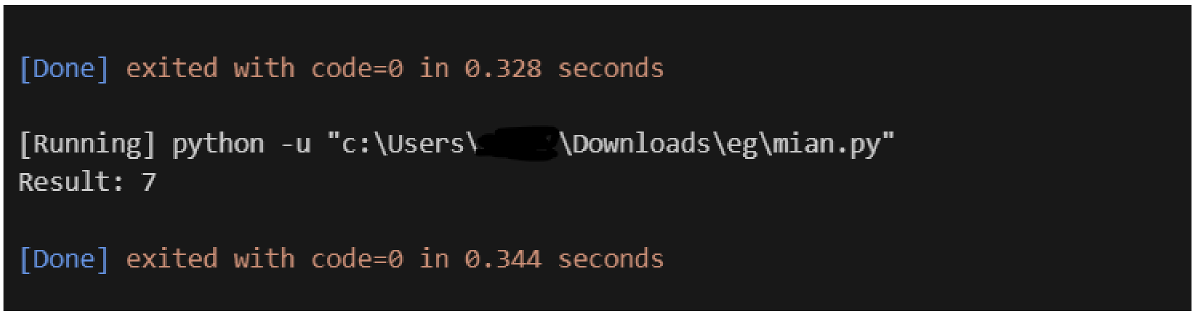
```
example_dll.addNumbers.argtypes = [ctypes.c_int, ctypes.c_int]
example_dll.addNumbers.restype = ctypes.c_int

# Call the function
result = example_dll.addNumbers(3, 4)

# Print the result
print("Result:", result)
```

This Python script loads the DLL and defines the argument types and return type of the addNumbers function. Then it calls the function with arguments 3 and 4, and prints the result.

Ensure to replace "path/to/example.dll" with the actual path to your DLL file. This example assumes you're using Windows, but the approach is similar for other platforms.



```
[Done] exited with code=0 in 0.328 seconds

[Running] python -u "c:\Users\[redacted]\Downloads\eg\mian.py"
Result: 7

[Done] exited with code=0 in 0.344 seconds
```