Emscripten and Python can interact in a few ways:

1. Compiling C/C++ Code to WebAssembly with Emscripten and Using it from Python

* **Write your C/C++ code.**

Create the necessary C or C++ files that you want to compile to WebAssembly.

* **Compile with Emscripten.**

Use the emcc command to compile your code to WebAssembly. This will generate a .wasm file and a corresponding JavaScript glue code file.

* **Use from Python.**

Utilize the wasmtime Python library to load and execute the WebAssembly module in your Python environment.

Example:

C++

*// c\_function.cpp*  
#include <iostream>  
  
extern "C" {  
 int add(int a, int b) {  
 return a + b;  
 }  
}

Code

emcc c\_function.cpp -o c\_function.wasm

Python

import wasmtime  
  
*# Load the WebAssembly module*  
engine = wasmtime.Engine()  
store = wasmtime.Store(engine)  
module = wasmtime.Module.from\_file(store, "c\_function.wasm")  
  
*# Create an instance of the module*  
instance = wasmtime.Instance(store, module, [])  
  
*# Call the exported function*  
add\_func = instance.exports["add"]  
result = add\_func(2, 3)  
  
print(result) # Output: 5

2. Compiling Python to WebAssembly (Experimental)

* **Python 3.11 and later:**

Starting with Python 3.11, there is experimental support for compiling Python code directly to WebAssembly. You can use the Tools/wasm directory in the CPython source code to explore this feature.

* **Pyodide:**

Pyodide is a project that allows you to run Python code in the browser using WebAssembly. It provides a full Python scientific stack, including popular libraries like NumPy, Pandas, and Matplotlib.