**Analyzing Syntax and Semantics**

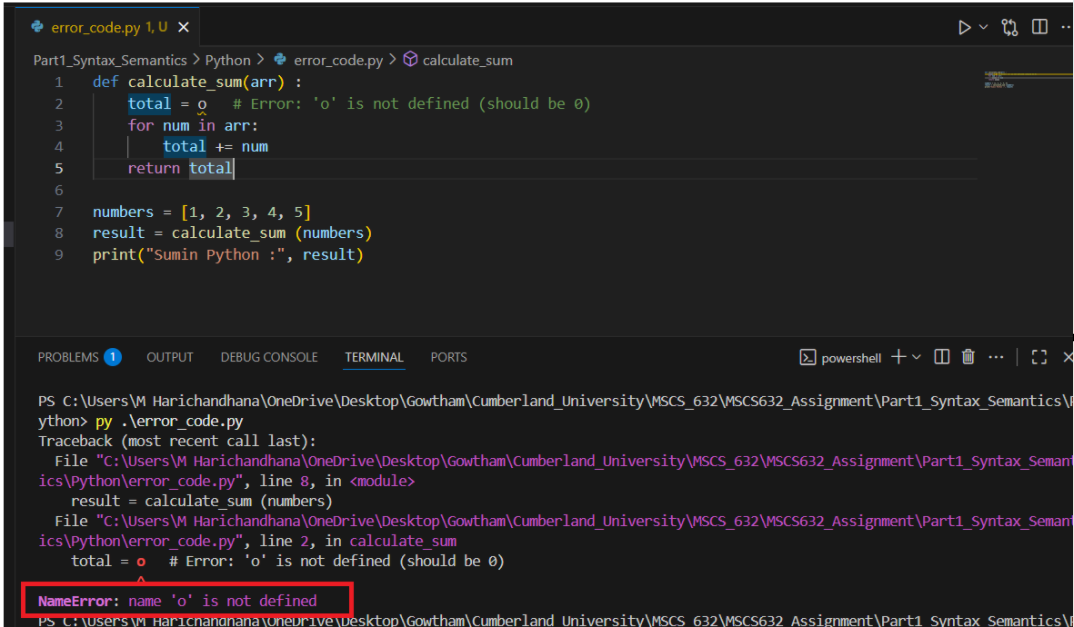
**Submitted by**

**Shiva Gowtham Kumar Vidiyala**

**Instructor Name : Jay Thom**

**Advanced Programming Language**

***Python Error Explanation***



The error message is usually clear and includes the line number, file, and type of error (NameError, SyntaxError)

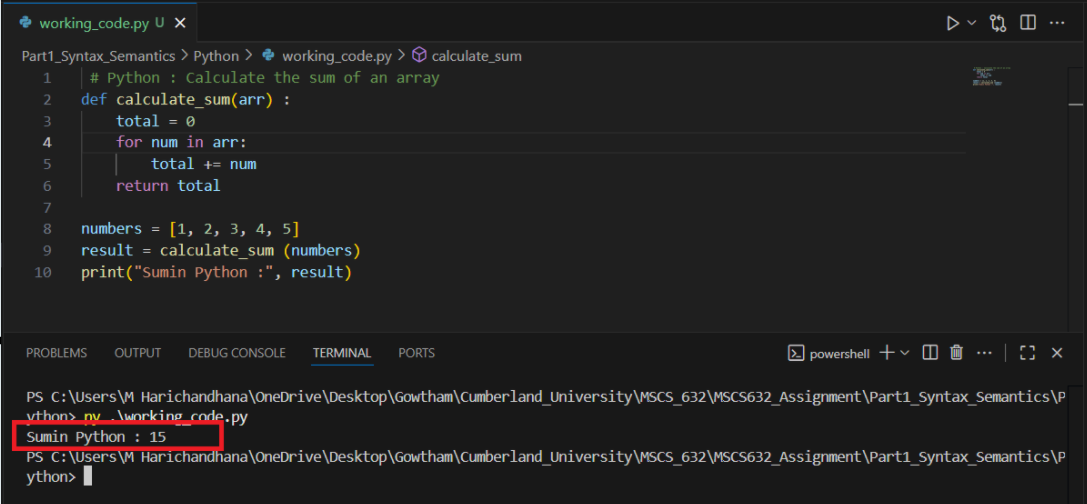
total = o — Here, o is a typo. It should be the number 0.

Python raises a **NameError**, as it treats o as an undefined variable.

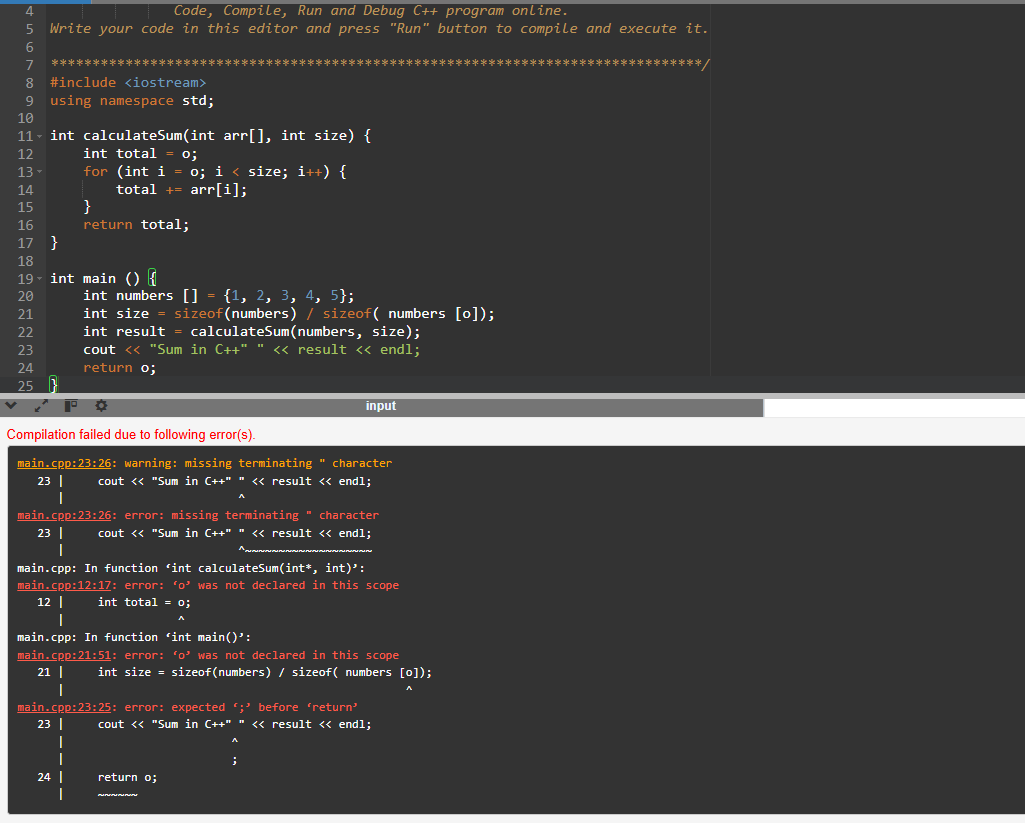
Python is **interpreted**, so the error is caught at **runtime**, not at compilation.

After changing o to 0, we got the correct result.

Here the python interpreter parses the whole file first and stops at the first syntax error, showing the file/line/column and a caret ^ pointing near the offending token.



***C++ Error Explanation***



o is undefined in all places; it should be 0.

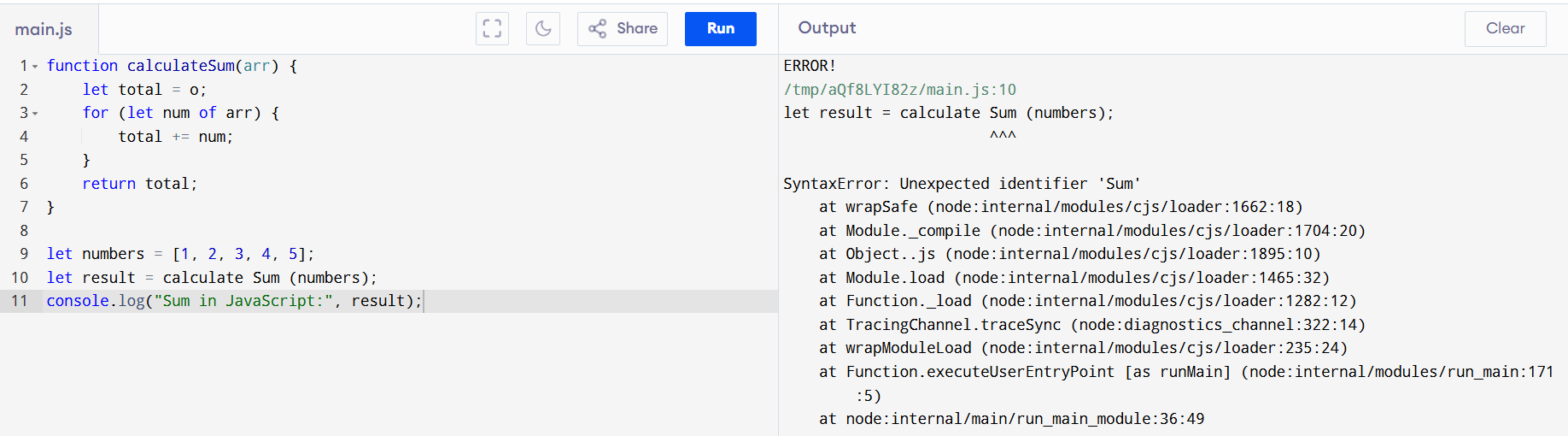
cout << "Sum in C++" " << result << endl; — invalid string concatenation.

return o; — again undefined.

C++ handles it: A separate compiler parses and type-checks first. It may produce multiple cascading diagnostics, but the first error is usually the true cause (missing ;).



***JavaScript Error Explanation***



The error happens due to **syntax error**

Here JavaScript treats calculate and Sum as two separate identifiers because of the space

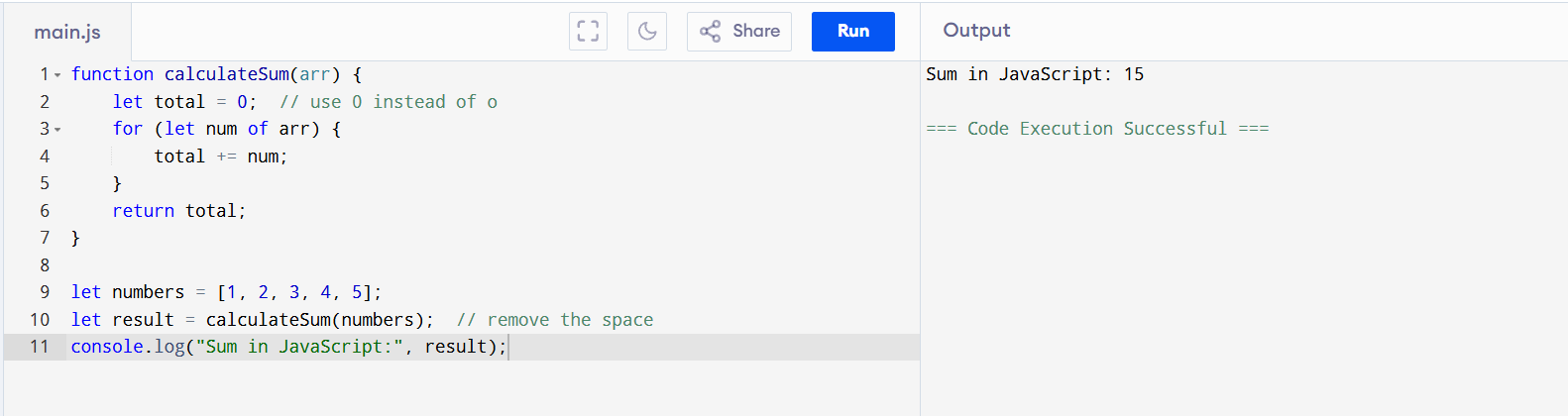
Function names cannot have spaces, so the interpreter throws:

SyntaxError: Unexpected identifier 'Sum'

There is one more error: o is not defined. It should be the number 0 (zero), not the letter o.

Here **JavaScript** engine parses the script before execution and fails fast on malformed structure (e.g., unmatched braces), usually reporting the first unexpected token.

Working Code



**Quick comparison: syntax-error handling**

**Python**: stops at first syntax error during parsing; message is short and points to the exact spot (often “expected …”).

**JavaScript**: also stops at first parsing error; messages often say “Unexpected token …” and highlight the first structural inconsistency.

**C++:** compiler may emit several diagnostics due to cascading errors after the first real mistake; messages are detailed (sometimes verbose) and can include notes and hints.

**Type System**

**Python**: Dynamic → easy and flexible, but type errors show up only when running; slower because compiler knows less.

**JavaScript**: Dynamic at runtime, but **TypeScript** adds optional compile-time checks → safer and better tooling without changing runtime.

**C++**: Static → errors caught early, highly optimized code, but more verbose and complex.

**Why it matters**: Affects development speed, safety, and performance.

**Closures & Scoping**

**Python**: Captures names (late binding) → loop gotchas unless fixed with defaults.

**JavaScript**: Closures are everywhere; let/const give safer block scope than old var.

**C++**: Lambdas with explicit capture lists (by value or ref) → powerful but must manage correctness and performance.

**Memory Management**

**Python & JS**: Garbage collection → no manual cleanup, easier to code, but occasional pauses.

**C++**: Manual control (stack, RAII, smart pointers) → predictable and efficient, but more work for developer.

**Conclusion:** Python favours speed of writing, JavaScript balances flexibility with optional safety, and C++ emphasizes control and performance but requires discipline.

GITHUB LINK:

https://github.com/gowthamvidi/MSCS632\_Assignment.git