Final Project: B1. Variable Binding and Scope Analysis

Comparison of Scoping Rules:

While all 3 languages employ static scoping and allow for block, function, global, and class scoping, they differ in their specific scoping applications and binding methods.

C++ is the most static of the 3 languages, as it is both statically scoped and bound. C++ does not allow for hoisting or run-time scope chain manipulation. C++ is a strictly static language in comparison to Javascript and python, which are dynamically bound languages where variables are determined at runtime, and do not have to be explicitly typed.

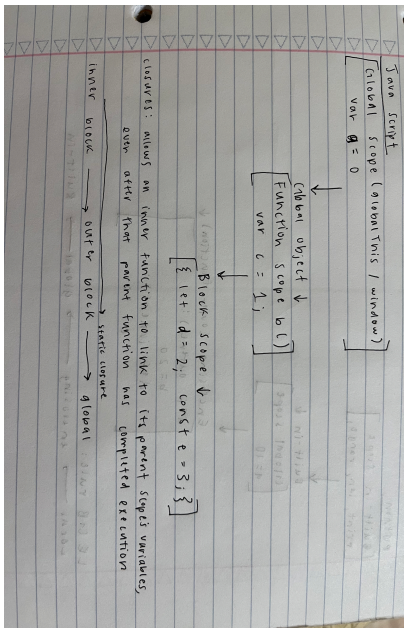
Similar to C++, python executes code line by line and does not allow for hoisting variables or functions. Python is statically scoped but is dynamically bound and thus is considered a 'strongly typed' language because it avoids implicit type conversions between incompatible types, though a single variable can still hold multiple different data types across different sections of the same program through its dynamic binding.

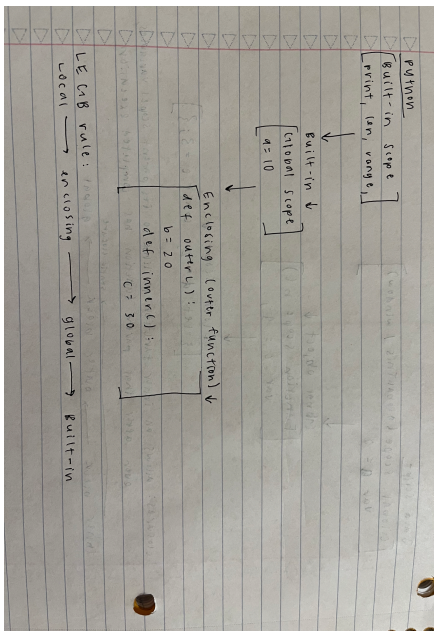
Javascript in particular is unique among the 3 for its generous scoping allowance through hoisting, and its ability to simulate dynamic scoping using both local and global functions despite being a static scoping language. Python is also dynamically bound, and is considered a 'weakly typed' language for these allowances, as variable types can be converted automatically during compilation. Through hoisting, Javascript moves declarations up the code to the top of the scope chain during compilation, allowing for versatile programs that do not require the declaration of variables or functions.

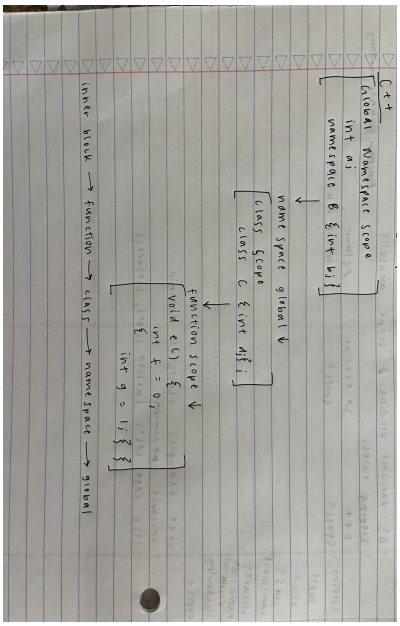
	C++	Javascript	Python
Scoping Types	Block, Function, Class, Namespace, Global	Block, Function, Global, Module	Local, Enclosing, Global, Built-in (LEGB)
Block scoping	Brackets{} always create a block scope	Let and const are block scoped, var is function scoped.	Normal blocks do not create scopes
Class scoping	Class has its own scope	Method binding is dependant on call site	Class body is executed like a normal block that defines a new namespace
Namespace scoping	Namespace has its own scope	Not available	Not available, must use modules
Closures	Limited; lambdas can capture by value/reference	Full closures that extend the lifetime of the outer scopes	Full closures using the LEGB rule

Name order resolution	Inner block → function → class → namespace → global	Inner block → function → outer lexical scopes → global → built-in	The LEGB rule: Local → Enclosing → Global → Built-in
Shadowing	Inner scope can shadow outer names	Inner block/function can shadow, variables are hoisted	Inner scope shadows outer scope; assignment creates local unless they are global or nonlocal
Constant behavior	Makes variable read-only	Binding is read-only but object can mutate	Not available, you must use conventions or dataclasses/froze

Visual Diagrams of scope chains:







Analysis of design trade-offs:

The binding times of C++, Javascript, and Python largely affect code flexibility, execution time, memory, writability and performance of their programs. As the most static of the 3 languages, C++ uses early binding at compile time to increase performance and execution time at the sacrifice of code flexibility. Data must be type bound to their addresses in memory which reduces run-time errors and optimizes performance and memory allocation. Using early binding, each data type in C++ must be defined correctly and specifically by the programmer and cannot be as easily manipulated in the code.

Javascript uses late binding at compile time to increase flexibility and moderate execution time at the sacrifice of performance and moderate memory allocation. Data is dynamically bound

and the memory addresses can be manipulated easily by the programmer, making code flexible and writable, with a relatively moderate execution time. However, performance is greatly affected by the changing data types of dynamic binding, as type-errors must be performed at run-time. Memory allocation requirements in Javascript are less than Python but more than C++.

Python uses late binding at run-time to increase flexibility and writability at the sacrifice of performance, execution time and reliability. Data is dynamically bound and can remain undefined until run-time, and dynamic scoping can be simulated to allow a programmer to very easily manipulate and write their Python code. However, this means type-errors and type-checks must be performed at run-time, decreasing the performance and execution time and increasing the run-time errors and memory allocation requirements.

Works Cited

"7. Using Bind Variables — Python-Oracledb 2.0.0b1 Documentation." Python-Oracledb.readthedocs.io, python-oracledb.readthedocs.io/en/latest/user_guide/bind.html.

"Comparing Python to Other Languages." Python.org, Python.org, 2019, www.python.org/doc/essays/comparisons/.

KathleenDollard. "Early and Late Binding - Visual Basic." Learn.microsoft.com, learn.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/early-late-binding/.

"Program Structure :: Eloquent JavaScript." Eloquentjavascript.net, eloquentjavascript.net/02_program_structure.html.

"Python Closures: Comprehensive Guide with Practical Examples." W3resource, 24 Aug. 2024, www.w3resource.com/python/python-closures-with-examples.php. Accessed 12 Dec. 2025.

"Structured Binding Declaration (since C++17) - Cppreference.com." Cppreference.com, 2024, en.cppreference.com/w/cpp/language/structured_binding.html.