Both C++ and Python's type systems are built on the idea of strong typing. Despite the programmer not being required to specify the variable types in Python, all variables still have types and these types do matter when performing operations on these variables like in C++. However, the difference between the two languages' type systems is that C++ uses static typing while Python uses dynamic typing. This is clearly distinguished when looking at each language's syntax when declaring variables. In C++, the programmer is required to explicitly specify the data type every time they want to declare a variable. For example, the variable declaration char temp; in the readFile function demonstrates this requirement. This allows the program to know the data type, char, which it can check for the variable temp at compile time. In contrast, the variable declaration, original_rows = 2, in the main function doesn't require the programmer to specify that the variable is an int, rather the program checks which type it should be based on the context during run time to make the actual memory address where the data is stored which the variable is referring to into an int.

In terms of type equivalence, C++ uses structural equivalence. This means that two data types are considered equivalent if they have the same underlying storage structure in memory. In contrast, Python uses name equivalence which means that each different variable with a different data type are not considered equivalent even if they have the same underlying structure.

When determining type compatibility Python can convert between types implicitly using coercion. For example, if either of two operands is a floating point, they the other is converted to a floating point. The same can be seen in C++. However, if there is no built in conversion between two different types, it is up to the programmer to explicitly convert types using casting

In both C++ and Python, variable names are identifiers. In C++, it is common practice to use camel case when naming variables, while Python's convention is to use snake case. These

conventions are in place because variable names cannot contain spaces. Camel case starts with a lowercase and combines two words together by making the first letter of the second word into an uppercase. In contrast, snake case keeps all letters lowercase while separating words with an underscore. This is seen in my program with the 2D array in C++ being named originalArray2D versus in Python where it is named array_2D instead.

The variable models that each language uses differ. C++ uses the value model while Python uses the reference model for their variables. This affects the way that variable names are bound to what they are actually referencing. In C++'s value model, the variable name is bound to the memory address of where the variable's data will be stored. For example, the statement, ifstream file;, demonstrates how the name "file" is bound to the actual memory address of where the ifstream object is being stored at. In contrast, with Python's reference model, the variable names are only a reference or pointer which points to the memory address of where the variable the name is referencing is stored at. For example, in the program, the variable name original_array_2D is a name that is bound to a pointer which then points to the actual memory address of where the 2D array is stored at. If we didn't need to use this variable anymore, we could in theory repoint this name to be any other type of variable, regardless of their type, because it is only bound to the reference and not the actual memory address.