**#Language Purpose**

C#: Created in 2000, C# was originally intended as a “simple, modern, general purpose object-oriented language.” C# was Microsoft’s response to the release of Java, and originally looked very similar to Java. It was heavily object-oriented, until the 2.0 version’s release in 2005, which expanded the features to include generics, anonymous methods, and even different iterators, mirroring Java’s recent release of an enhanced for-loop in 2004. In 2007, C# established the LINQ (Language-integrated query), which provided a method to perform operations on collections of data in a much more logical way, and evolved the language in a completely divergent path compared to Java. In recent releases, the language has adapted to incorporate multiple paradigms, such as pattern-matching within the functional programming paradigm.

Python: Originally developed around 1991 by Guido van Rossum, Python was intended to be simple and have high readability, as well as minimalism. It was heavily influenced by Lisp, as well as a language called ABC. The initial release contained exception handling, and was object-oriented primarily.

**Unique Features**

C#: LINQ is a heavy unique feature that C# implemented initially. Interoperability of any language written in a .NET compliancy also enables it to access classes and libraries from code in other languages. The namespace feature of C# enables classes to be accessed independent of their directory. It uses Microsoft’s Dynamic-Link Library system to provide a single universal copy of a library, instead of necessitating multiple copies in every project directory. Another unique feature of C# is its operator overloading, which allows the developer to add additional functionality to basic operators in the language.

Python: The importance of indentation and significant whitespace was unique to python, and assisted in enforcing readability in developers code. Python also includes Generator expressions, which are a different version of list expressions, and enable the run-time generation of iterable structures without consuming as much data.

**Name spaces**

C#: Through the use of the “using” keyword in C#, external namespaces are allowed to be implemented and differentiated from local variables. Namespaces can be nested as desired, to allow for proper closures, and it also enables a hierarchical system to be

Python: Namespaces are implemented as dictionaries, or key-value pairs. The name is used as a key, and an object for the namespace is generated as a value. Every single module, class, function, etc. contains its own namespace, as well as a global namespace. This makes it easy to import other Python projects, even if there are identical names, as each name has an identifier for its scope, and imported modules contain the global namespace scope, which will not have a collision with any local variables.