CSC 461 Semester Project Questions

Alex Murtha, Dylan Shaffer, Dustin Cook

Comparison to Python

1. Location it is used

Perl is used for quick fixes and quick scripts kind of like a “glue.”

Python is used in the same situations but also in larger productions as the main application

1. Where it excels

Perl is very good for text processing

Python is easy to pick up for beginners and a sort of jack of all trades

1. Where it fails

Perl fails when it comes to ease of use and maintainability

Python fails because it is slow relatively

Both languages are dynamically typed which can cause some confusion

1. Portability

Perl and Python are both portable by default with a wide array of supported systems

1. Simplicity

Perl syntax is incredibly difficult to follow and can get messy even in smaller applications. There is a steep learning curve with Perl’s syntax.

Python is extremely easy to read and very simple to pick up and start programming.

1. Orthogonality

Perl is not orthogonal by nature even though it is dynamically typed it is still strict on which operations can be performed on which data. Exceptions are common.

Python is built to have orthogonality to achieve its dynamic nature.

1. Reliability

Perl language maintenance is good and is updated frequently without any new bugs introduced.

Python is reliable due to its secure environment and availability of large support community.

Syntax, Parsing and Binding

1. Is the language compiled or interpreted or a hybrid?

Perl is technically compiled. Perl source is compiled into an internal form (parse tree) and then optimized. This intermediate representation is converted to C code and then compiled with the C compiler.

1. Write the EBNF for an expression in the language

<int> -> {1|2...|9}

<string> -> {a|b...|z}

<function> -> <string>()

<expression> -> <int> (\*|/|-|+) <expression> | <empty>

<assign> -> my ($|@)<expression>|<int>|<string>

<line> -> <assign>|<expression>|<function>

1. How does it handle module/namespace/packages, and what is the scope operator?

Packages are declared with the ‘package’ keyword, everything under the package statement belongs to that package until another package statement is used. The scope operator is ‘::’ for example Package::Module.

1. How is memory management handled (garbage collection, explicit, hints, etc) ?

Perl has automatic memory management using the reference counting method.