STUDY OF SCRIPTING LANGUAGES

COMPARISON OF THREE MOST WIDELY USED SCRIPTING LANGUAGE

VIVEK KHAURIA

Lang.		Perl	Python	Ruby
Sr. No.				
1	**INTENDED USE**	Application, Text processing, Scripting, Web	General, Application, Scripting, Web	Application, Scripting, Web
2	**PARADIGM** **Imperative** - describes computation in terms of statements that change a program state. **Reflective** - process by which a computer program can observe (do type introspection) and modify its own structure and behavior at runtime. **Functional** - that treats computation as the evaluation of mathematical functions and avoids state and mutable data. **Procedural** – based upon Procedure call(simply contain a series of computational steps to be carried out). **Aspect-Oriented** - increase modularity by allowing the separation of cross-cutting concerns. **Generic** - in which algorithms are written in terms of to-be-specified-later types that are then instantiated when needed for specific types provided as parameters.	Imperative Procedural Reflective Functional Object-Oriented Generic	Imperative Procedural Reflective Functional Object-Oriented Aspect-Oriented	Imperative Reflective Functional Object-Oriented Aspect-Oriented

3	**STANDARDIZED**	No	No	No
4	**Expression of types**	Implicit	Implicit	Implicit
5	**Type checking** - process of verifying and enforcing the constraints of types	Dynamic	Dynamic	Dynamic
6	**Type safety**		Safe	Safe
7	**Compatibility among composite types** - is how functions are applied to data typed similarly to its intended type.		property-based	property-based
8	**Failsafe I/O**	No	Yes	Yes
9	**Garbage Collection** **Reference Counting** - where each object has a count of the number of references to it. **Mark and Sweep** - maintains a bit (or two) with each object to record whether it is white,black or grey and then sweeps and free the space at their current status.	Reference Counting	Reference Counting(better)	Mark and Sweep
10	**Uniform Access** - services offered by a module should be available through a uniform notation	No	No	Yes
11	**Class Variables / Methods**	No	No	Yes
12	**Access Control** **Name Mangling** - pass more semantic information from the compilers to linkers.	None	Name Mangling(better)	public, protected, private
13	**Multithreading**	No	Yes	Yes
14	**Pointer Arithmetic**	No	No	No

15	**Method Overloading** - allows the creation of several methods with the same name.	No	No	No
16	**Operator Overloading** - where different operators have different implementations depending on their arguments.	Yes	Yes	Yes
17	**Inheritance**	Multiple	Multiple	Multiple
18	**Generic Classes**	N/A	N/A	N/A
19	**Design by Contract**	No	No	Add-on
20	**Language Integration**	C, C++	C, C++, Java	C, C++, Java
21	**Built-In Security**	Yes	No	Yes

Pearl	**Python**	**Ruby**
Pros :	**Pros :**	**Pros :**
Perl is an interpretted language.	Good support for objects, modules, and other reusability mechanisms.	Productivity: Ruby and Ruby on Rails have higher productivity measured as Lines of code per feature size and time to build feature

You have to make a perl file executable on Linux systems - making it a little more difficult to hack.	Easy integration with and extensibility using C and Java.	Ruby tends to be more terse.
It is good at pattern matching, regular expressions, and string manipulation.	Python is Perl without the line-noise.	Total object-oriented; even 1, nil, true and false are objects.
Perl is good for file manipulation.	Enforces good programming style (indentation is meaningful).	Looks very clean because punctuation is minimal.
It is often used for CGI scripts and other web related jobs.	Is cross platform and has a powerful set of libraries	Various, natural ways of expressing flow control decisions.
It's in common use, especially by system administrators.	Is safe - it has dynamic run time type checking and bounds checking on arrays.	No need to declare variables before use.
Perl is very portable - it is available for almost every platform.	Has powerful built-in data types - dictionaries, lists, sequences, functions, sets.	Built-in RSS support in the standard library.
It is better at handling arrays.	Requires less lines of code for any given problem, and is more readable - thus greater productivity.	Advanced flow control structures.

File operations are better - flock command - PHP equivalent is not as good.	Python requires less time, less lines of code to reach a given goal. This allows more time to be spent on the important things. Further more :	
	End of line is end of line (no forgotten semicolons)	
	No type declarations	
Cons :	**Cons :**	**Cons :**
perl has a reputation for being messy which might be considered a detriment for a beginning class	Python is slow. Python is an interpreted language(adds some overhead), dynamic bounds checking, dynamic typing makes it even slow.	Ruby doesn't seem to have quite the library support that Python does. But more terse than Python.
Its object model is undocumented, and not even available most places.	Lack of true multiprocessor support.	Ruby library doesn't supports Exclusive XML Canonicalization.
Compared to languages like Python, its usability is very poor.Needs more code to do the same job		No need to declare variables before use (can lead to hard to spot run-time errors)

CONCLUSION

I chose Python because :

1. Clean and easy Programming language with strong library support. Because of strong library support it can be expanded with the use of different languages it can be integrated with.

2. Since its a widely used language,so the Python community always provides support to Python users.

3. Also in our Project Group python is mosty used therefore easy to be compatible with them.

4. You can write a Python program on a Mac, test it using a Linux environment, and upload it to a Windows server.

5. Python encourages program reusability by implementing modules and packages.