University of North Dakota

Contrast of Two Scripting Languages:

Perl and Python

Derek Trom

Organization of Programming Languages, CSCI 365

Dr. Thomas Stokke

May 8, 2020

As an introduction to this research topic, the author chooses to compare and contrast Perl and Python as scripting languages, and the evolution of the two to meet today's ever-growing industry needs. The author felt this was a good topic because when Perl was developed, its powerful built-in text processing looked promising to be a leading language used for scripting for ages. This was until Python came to popularity within the same realm of capabilities and in some ways surpassed Perl in its uses. This paper will compare some basic syntax differences, some of the built-in features of both or lack thereof in Python's case, and why each has its pros and cons when it comes to scripting projects.

Starting off, Perl was created by a man named Larry Wall. One of Wall's mantras is that "there is more than one way to do something." This phrase, more than anything else, sums up why Perl differs so dramatically from most other programming technologies. Perl isn't just a programming language, it has almost a cult-like following. With the plethora of ways one can accomplish the task this also reinforces the idea Wall injected into this language. Perl was released in 1987[3]. At the end of 1994 Perl 5 was released and this is the language that is still around present day. With the release of Perl 5 came a complete rewrite of the interpreter, which was the biggest development of all because this brought modules into the language of Perl. This meant programmers were now able to add features to the language and reuse functionalities that were already implemented by other people. Another function that helped propel Perl even further into the market, which was released with a language update in 1997, was the inclusion of CGI or Common Gateway Interface module for web-based scripting and the creation of the large Comprehensive Perl Archive Network that contains an expansive number of modules[3]. This is in part a large reason why Perl is still around today in applications everywhere and is known as the “duct tape of the internet”, this meaning comes about from the capability of it to provide quick and easy fixes of data problems[3].

Like Perl, Python is an extremely prevalent language in the web application world. Python started development in the late 1980’s by Guido van Rossum who at the time was working on a project called Amoeba which was a distributed operating system based in the language ABC. Van Rossum used his experience with the ABC language and started developing Python with the pros and cons of ABC in mind[4]. The first version of Python was released in early 1991 and included many features such as numerous core data types like list, dict and string. It was also object oriented and had the ability to use modules. One of the biggest visions in the creation of Python was readability and simplicity which is what Python is so well known for because of the forced symmetrical indentation for the language. Following one of Python’s 13 Rules of Zen and in the release of Python 3 was the removal of many of the duplicate programming constructs and modules, which reinforces rule 13 “There should be one – and preferably one – obvious way to do it.”

Comparing the two languages syntactically they look very different. Perl is based more on the style of C where it uses ‘{}’to delineate sections of code and the ‘;’ to end a line of code. Python includes none of these and relies completely on white space in order to parse blocks and lines of code. Commenting lines of code in both languages are the same by using the ‘#’ symbol. In the below average subroutines it is apparent that the languages are quite different in how they pass and retrieve values from functions. A simple difference is how the subroutines are declared, Perl uses ‘sub’ as a reserved word and Python uses ‘def’. The next key difference between the two is how each pass and retrieve data to be used within a subroutine. Perl’s use of an array to store what values are being passed into a subroutine can become quite complicated if there are many values being passed. Algorithm 1 below this paragraph shows that it will be passing in an array of integers to the subroutine via the function call. Now, in order to access that array being passed from the subroutine call, the user must assign values from the '@_' array for the subroutine in order to access the data. The '@_' array is a parameter array that Perl uses in order to pass information into a subroutine. When passing the parameter list to a Perl function, the order of accessing the data matters as well. In Algorithm 2, the array being passed as a parameter is being passed by reference to the array for Python.

In the author's opinion this makes it simpler to read and decipher what values are being passed and manipulated. The declaration of variables between the two languages are also different. In Perl, when declaring a variable, it is necessary to use 'my' before the variable. In some cases, 'our' is used if the programmer would like to use it across a package. The use of '$' for a scalar, '@' for an array, or '%' for a hash are also necessary in order to instantiate. In Python the object type is decided by the context of what is assigned to the variable. As shown in Algorithm 2 on line 11 the variable dataArray is being assigned as a list object. This compared to line 13 in Algorithm 1 shows that the readability and clarity is enhanced. Taking all of these instances into account a new user may prefer the use of Python over Perl. Nevertheless, an experienced programmer may only require a short learning curve.

Algorithm 1

```
1  #a simple function in Perl
2  sub average{
3   my @array = @_; #retrieve array
4   my $sum = 0; #declare sum
5   my $count = 0; #declare counter
6   foreach (@array){#iterate array
7     $sum += $_; #add item value
8     $count += 1;#increment count
9  }
10    return $sum / $count;#return
11 }
12 #initialize array
13 my @dataArray = (1,2,3,4,5);
14 #call function and assign value
15 #to variable
16 my $averageArray = average(dataArray);
```

Algorithm 2

```
1  #a simple function in Python
2  def average(list):
3   sum = 0 #initialize list
4   count = 0 #initialize counter
5   for item in list: #iterate list
6     sum += item #add item value
7     count += 1 #increment count
8   return sum/count #return value
9
10 #initialize list
11 dataArray = [1,2,3,4,5]
12 #call function and assign value
13 #to variable
14 listAverage = average(dataArray)
15
16
```

While researching these languages the most parroted theme that separates the two visually is readability. However, it is the author's opinion that while Python may be easier to learn in the beginning, Perl is also quite easy to pick up once all the nuances of the language are learned. Down on a deeper level, as mentioned before in the history of the languages is the difference in philosophies of the two languages. First, Perl is designed for the programmer to be able to do the same functions in many different ways. Python on the other hand leans heavily toward the fact that there should be one and only one obvious way to do it, which increases simplicity and conciseness of code. Both languages in one way or another rely heavily on being open source with Perl and its CPAN and Python also being an open source language and are similar in those respects.

When it comes to online web uses Perl has been around for a long time and because of its flexibility has been used in numerous occasions and easily integrates with many server and web applications, hence its widespread use for years. Python, just as Perl does, has an enormous following leading to many open-source modules and libraries available for use which is also what makes it flexible in its own right while still maintaining its "Pythonic" ways. But, the flexibility also comes with a memory hit which can lead to Python being larger for certain tasks than a Perl file would be. Perl was a major go to for many companies because of its flexibility in scripting and database interface coming in forms such as CGI and DBI, as well as its very powerful built in regular expression functionalities. CGI makes scripting for the web extremely easy especially on Apache web servers. In recent years however, this module is slowly being phased out in modern web interfaces, making way for other modules such as Dancer and Catalyst. DBI or Database Interface Module is the go to for Perl in order to maintain consistent interface independent of the database being used. A simple Perl one-liner makes connecting to databases very simple and queries into the database are very similar to SQL statements. This along with Perl's built in regex capabilities make it very powerful and efficient.

For Python to perform the same functions external libraries must be imported and may be hard to find via third party sources depending on the task. The library Python uses for its regex capabilities is re. The syntax and uses are similar to many other regex libraries but again, this requires the importing of a library, where Perl does not. A popular Python framework to achieve the same functionalities of CGI is Flask which has been shown to be a lightweight microframework and is comparable, if not better. As far as Python's equivalent to DBI it seems like DBI may win in this battle because of its flexibility to work in many different databases in one script only changing a couple words working between multiple database platforms. A popular import for working with databases in Python is DB-API. This API makes it bit more complicated to write portable code, because writing the database driver is up to the user compared to Perl's DBI.

Overall, the author is unable to delve into a larger algorithm of full-scale implementation and compare the two languages side by side on a real application. But to the author, one thing that has become apparent is that Perl was and still is an extremely powerful scripting language for database and web interfaces and it will never go away completely because of its powerful text processing and built-in libraries. However, Python has become a fan favorite in all aspects of the computer world and has done so because of how powerful, clean, and concise its code can be. As far as which is better the choice, it is up to the programmer. If they are worried about size of library imports and want to be able to complete the task in many ways, the obvious choice is Perl. If program size does not matter, and they want one obvious way to do it, Zen with Python.

Works Cited

[1]“Comprehensive Perl Archive Network.” *The Comprehensive Perl Archive Network*, www.cpan.org/.

[2]Hartigan, Matt. “The Fall of Perl, The Web's Most Promising Language.” *Fast Company*, Fast Company, 2 Apr. 2015, www.fastcompany.com/3026446/the-fall-of-perl-the-webs-most-promising-language.

[3]“History of Perl.” *Perl*, yapc.tv/history-of-perl/.

[4]“Python Course.” *Python Tutorial: History and Philosophy of Python*, www.python-course.eu/python3_history_and_philosophy.php.