PHP vs. Python vs. Ruby – The web scripting language shootout

Klaus Purer

Vienna University of Technology
Institute of Computer Languages
Compilers and Languages Group
185.307 Seminar aus Programmiersprachen, 2009S
`klaus.purer@student.tuwien.ac.at`

July 18, 2009

Abstract. Scripting languages became increasingly popular in web application programming in the last years. This paper tries to find differences, advantages and drawbacks of the three most popular languages nowadays: PHP, Python and Ruby. It is obvious that all of them have their lobby and support, so it is a difficult task to state objective facts that satisfy a scientific approach. The languages will be compared concerning history, evolution, popularity, syntax, semantics, features, security and performance in web application environments. In the end a final conclusion recommends a language that looks most promising.

1 Introduction

Dynamic scripting programming languages have arrived in the mainstream market of general purpose programming languages. They have certain properties that distinguish them from classic static languages like C++:

– they are dynamically typed
– they focus on short syntax
– they have an automatic memory management and garbage collection
– they are mostly interpreted (instead of compiled)

These properties are attractive for web application development, because they allow rapid prototyping, fast code changes and fast testing routines. Depending on the selected language there may be also other differences that make the language valuable. However, this paper is not about comparing dynamic languages to static languages, but about comparing three concrete dynamic languages with each other: PHP, Python and Ruby. In this introduction I will give some details about history and popularity of the languages. The main part of the paper will work out comparisons concerning syntax and semantics, language features, security and performance. In the end there will be a conclusion listing all findings and presenting a recommendation based on the need of programmers and users.

The contribution of this paper is an introduction to the three languages, an overview of features and possibilities, a comparison of properties in relation to web development and a decision guidance when evaluating programming languages for a future project.

1.1 History

Web application development has a relatively short history. in the beginning it was all about linking some static documents together. Mehdi Jazayeri pointed out the similar history of web application engineering to software engineering, but on on a much smaller scale[14]. He showed that in the early days there was not much structure and scripts written in Perl generated web pages. The problem was embedding the content and the HTML markup code in the script, which resulted in difficult to manage code. This was the time when PHP entered the market with a contrary approach: script code was inserted into the HTML content.

PHP was started in 1994 by Rasmus Lerdorf and was an acronym standing for Personal Home Page (replaced in 1997 with PHP: Hypertext Preprocessor). The language is developed and implemented by The PHP Group nowadays, which also defines the de facto standard[12] as there is no formal language specification. It is released under terms of on open source license and documentation is freely available online[13].

As web application engineering matured, software engineers realized that a separation of concerns was necessary to implement big and complex programs. A well-known design pattern got new attention in web development: Model-View-Controller[17]. PHP did not have support for object-oriented programming from the beginning, but it was added continuously to the language to catch up with the needs of using a design pattern like Model-View-Controller.

In the meantime the programming language Python has also been some years around. Python was started in 1991 by Guido van Rossum, shortly before the World Wide Web was released to public use. It was designed as a full-featured general purpose language from the beginning and was not specialized to be used as a web scripting language like PHP. There is also no formal specification of the language, but a de facto standard is developed and implemented by the Python Software Foundation[20] (the reference implementation is also often called CPython, because it is written in C). There are several other projects that provide a compatible Python implementation: for example Jython is written in Java and makes use of the underlying Java platform, IronPython is written in C#, utilizing the .NET framework. An implementation called PyPy is written in Python itself. Similar to PHP Python has a strong open source background and also ships with comprehensive online documentation[10].

Another programming language arrived in 1995: Ruby, a scripting language developed by Yukihiro Matsumoto was released and had very similar goals compared to Python. Ruby is completely object-oriented and was strongly influenced by the programming language Smalltalk. There is a standard implementation written in C that also serves as the language de facto standard and is open source (once again similar to PHP and Python). Ruby language development is coordinated by an open community, but there is no formal foundation or organization behind it. Language specification and documentation is available online[9].

1.2 Popularity

Language popularity is important for companies to have a large enough pool of possible employees. Unpopular and unknown languages are a risk for proper planning of maintenance and development of existing and to be written software. As a result, decision making for projects which language should be used also depends on the factor of popularity of the programming language.

At this point I would like to mention that the measurement of popularity is very difficult. Any kind of result can not be seen as true scientific proof, because it is hard to collect representative data.

One indicator of popularity is the TIOBE Programming Community Index[6]. It is a programming language ranking based on skilled engineers, courses, third party vendors and search engine ratings. The website says "The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system." As of May 2009 PHP is ranked on the fourth place, Python on the sixth place and Ruby on the tenth place.

Another approach is to measure discussions on online media like IRC, newsgroups etc. Anton Ertl provides a statistics page of postings in the various comp.lang newsgroups[4], where as of March 2009 Python is ranked first, Ruby is ranked fourth and PHP ninth. Similar to the TIOBE index there is the Programming Language Popularity website that has also ratings based on IRC and discussion sites like Slashdot[5]. The normalized discussion site results rank Python on the third place, Ruby on the sixth place and PHP on the ninth place.

Considering the languages in the web programming context, we can also take the existing software ecosystem into account. As mentioned before, all three languages have a close relationship to free and open source software, so there are many web frameworks and content management systems available. PHP is used by the most popular content management systems like MediaWiki, Drupal, Joomla and Wordpress, just to name a few. Python's most popular web frameworks are the Zope application server, that is used by content management systems like Plone, and Django. Ruby offers the widely known Ruby on Rails web framework, but has no popular ready to use content management system available.

PHP has one major advantage for smaller businesses and private customers: it is available on almost every shared hosting provider and is the most established runtime environment on servers today.

All in all PHP, Python and Ruby are always among the top ten popular languages and are well-known world wide. PHP has advantages on search engine ratings and availability on hosting providers, Python has advantages on discussion media ratings. PHP offers most popular full-featured content management systems, that also allow site building without programming skills, while Python and Ruby have elaborated web frameworks available, that provide more flexibility, but also need more programming effort.

2 Syntax and Semantics comparison

This chapter will focus on code snippets and programming language constructs that can be used in one of the three languages to compare: PHP, Python and Ruby.

2.1 PHP

As stated in the introduction, PHP invented the technique of embedding code directly into a content document. A simple example in listing 1.1 shows a built-in PHP echo command that is executed when the script is triggered by an incoming request.

Listing 1.1. Embedded PHP code in a HTML document

```
<html>
  <head>
    <title>Hello World Example</title>
  </head>
  <body>
    <?php
      echo "Hello World!";
    ?>
  </body>
</html>
```

This was useful for small and static web pages, but as the web evolved and applications got more and more complex, embedded code moved to template files (template engines) that are responsible for the view, design and appearance only. Business logic code was separated in other files to improve clarity and maintainability. An example of a typical PHP code snippet is given in listing 1.2 where basic language constructs are shown. It is taken from Armin Ronacher's work on security in web applications[19].

Listing 1.2. A simple PHP example for checking login data

```
<?php
function checklogin($username, $password) {
  global $db;
  $username = mysql_realescape_string($username, $db);
  $hash = md5($password);
  $sql = "select userid from users
          where username = '$username' and
```

```
                    password = '$hash';";
  if ($result = mysql_query($sql, $db)) {
    if (($row = mysql_fetch_assoc($result)) !== null) {
      return $row['userid'];
    }
  }
  return   1;
}
?>
```

As you can see, PHP syntax is derived from C syntax and functions are the central element in the language. There are no namespaces now (to be introduced in future PHP 5.3), all functions need to have unique names. Variables have to start with the $ character and need no initialization, global variables can be accessed with a special keyword ("global"). PHP has a weak type system and uses implicit type conversion, e.g. an integer and a string can be compared in a boolean expression. This often causes confusion and uncertainty which operator should be used in which case, but there are comparison tables and hints in the PHP documentation.

In the example above there are MySQL database statements directly integrated in the code, which was a disadvantage in PHP for a long time, because database systems were tightly coupled to PHP by specific functions. Since PHP 5.1 there is support for PDO (PHP Data Objects), a way to abstract the database layer from manufacturer-based functions.

The paradigm of object-oriented programming is relatively new in the PHP language and has been introduced in PHP 4 and was enhanced in PHP 5. However, PHP's procedural roots are obvious in the code of most PHP web frameworks and content management systems nowadays, but there is a process of migrating functionality to object-oriented approaches.

PHP always had the image of being easy to learn for beginners. Example Code is available online on very many places and the number of lines of code is small for accomplishing common tasks. The complete software technology for running a web application, the famous LAMP stack (Linux Apache MySQL PHP), was always freely available with good documentation and tutorials. The syntax is familiar to many programmers, because it is similar to widely known languages like C and Java. The behavior of PHP was always very robust, using implicit type conversion, keeping warnings silent and failing with errors just in fatal cases (an advantage for beginners, but a pitfall for developers of complex code). These were all factors that gave PHP the strong user base and the distribution on many systems.

2.2 Python

Python is a language that is not particularly focused on web application development in the first place, so it took some time until web scripting in Python got common. First approaches used CGI (Common Gateway Interface) in the way shown in listing 1.3.

Listing 1.3. A simple Python script returning a Hello World document

```
#!/usr/bin/env python
print "Content-Type: text/html"
print
print """\
<html>
  <head>
    <title>Hello World Example</title>
  </head>
  <body>
    <h2>Hello World!</h2>
  </body>
</html>
"""
```

Web frameworks were written in Python, but they used different techniques like CGI, mod_python etc. to communicate with the web server gateway. Therefore the frameworks were incompatible to different web servers or different gateway interfaces. The solution to this problem was WSGI: "WSGI is the Web Server Gateway Interface. It is a specification for web servers and application servers to communicate with web applications (though it can also be used for more than that). It is a Python standard, described in detail in PEP 333."[7] WSGI allows to change the environment or the gateway system of the web application without needing to touch to source code of it, so it has the advantage of being more portable.

According to WSGI a Python web application can be implemented in its simplest form shown in listing 1.4.

Listing 1.4. Hello World through WSGI

```
def simple_app(environ, start_response):
    """Simplest possible application object"""
    status = '200 OK'
    response_headers = [('Content-type','text/plain')]
    start_response(status, response_headers)
    return ['Hello world!\n']
```

There are benefits in this approach by having a unified interface, but it is also a little bit more complicated for beginners. More details have to be defined explicit instead of being implicit as in PHP or the CGI solution above.

Listing 1.5 shows the same login check example from the PHP section, but now implemented in Python. Again, it is taken from Armin Ronacher's work on security in web applications[19].

Listing 1.5. Checking login data in a Python function

```
import md5

def checklogin(username, password):
    hash = md5.new(password).hexdigest()
    cursor = db.cursor()
    cursor.execute("""
         select userid from users
         where username = %s and password = %s;
    """ , (username, hash))
    row = cursor.fetchone()
    if row:
        return row[0]
```

Contrary to PHP, Python was an object-oriented language from the beginning, but it is not limited to this paradigm and also supports procedural programming and some functional features. It is claimed that its syntax is easy to read and understand, as there is just a small set of keywords. Python syntax differs from PHP and Ruby by one major syntax property: indentation is mandatory within code blocks (e.g. function bodies). That means that the source code is always correctly structured, which is not enforced by most other programming languages. Statements in Python do not need to be separated by semi colons (the newline character defines the end of one), but it is allowed to use them.

The type system of Python is strong. On runtime errors there is a traceback of recent calls that give debug information to the developer.

2.3 Ruby

Ruby is a language that has only one major web framework in the market: Ruby on Rails. It makes use of CGI as gateway but also provides its own web server, which is recommended for development and testing only. I will skip a hello world example here and continue with listing 1.6, the check login function in Ruby.

Listing 1.6. Checking login data in a Ruby

```
require 'digest/md5'

def checklogin(username, password)
  hash = Digest::MD5.hexdigest("#{password}")
  username = db.escape_string("#{username}")
  res = db.query("
    select userid from users
    where username = '" + username +"'
    and password = '" + password +"';")
  row = res.fetch_row
  unless row.nil?
    return row
  end
end
```

The syntax looks similar to Python, but the semantics are little bit different: Ruby is purely object-oriented and is structured in a Smalltalk language syntax fashion. This, means that primitive types are not different from complex types and objects communicate with each other by sending messages. Every object can receive any message and starts exception handling on unknown messages. All language constructs, also primitive ones like assignments and if clauses are treated as objects that receive messages with certain parameters.

Ruby also separates statements by newline characters, and it is also possible to append a semicolon at the end of a statement. Ruby does not enforce indentation like Python and uses the "do" and "end" keyword (or curly braces in an abbreviated form) to indicate blocks. It also has support for procedural programming paradigms, some functional language constructs and meta-programming.

Ruby follows the principle of least surprise, meaning that the language is designed to be intuitive to use and that expected behavior should match actual behavior of Ruby programs. Ruby is a very dynamic language, it is possible to change the behavior of every object at runtime, including the ability to mutate the semantics of built-in types. This can lead to completely unexpected results, for example reversing the meaning of "true" and "false" in if-clauses would break most programs. Therefore it is necessary that all program parts can be trusted and that there are rules or guidelines what can be changed and what must not be changed.

2.4 Readability and Usability

It is difficult to define what readability and usability means to programming language users. PHP follows a very classical approach, is extensively documented and will probably be the most familiar to former C-programmers. Python with its strict indentation enforcements and the small set of keywords will probably be the best choice for programming beginners. Finally Ruby will probably be attractive for Smalltalk-enthusiasts and experienced programmers, that look for elegant and powerful programming expressiveness.

While Python seems to have the most readable syntax of the three languages (because of the enforced program strucuture), Ruby seems to be the most usable one (because of its principle of least surprise). Of course PHP is a readable language too, because most programmers are familiar with C-based syntax.

3 Language feature comparison

This chapter will pick up some programming language features, that are useful in common cases and some that are very helpful in web application development.

3.1 Exception handling

Exception handling is a feature that all three languages offer. Python and Ruby supported it from the beginning, PHP added it in version 5. Therefore some PHP content management systems and frameworks lack proper exception support in their code – this feature is not facilitated widely.

The differences between the languages are only syntactic by using other keywords: PHP has "try/catch/finally" for handling and "throw" for triggering new exceptions, Python uses "try/except/finally" and "raise", Ruby provides "begin/rescue/ensure" and "raise". The system works in all languages very similar and behaves like most programmers would expect it. Python version 2.5 introduced a new additional feature that allows objects to define standard cleanup actions (e.g. file objects that close the file regardless whether the operation on it failed or succeeded). The programmer does not have to care about "finally" blocks anymore, which reduces source code length and ensures proper cleanup, that can have an impact on performance in large programs. For reference see PEP 343[1].

3.2 Relational database abstraction

Nearly all modern web applications need an underlying database layer to store and retrieve data. To have an easy to use, safe and vendor-independent interface to relational databases, a good way to abstract from plain old SQL statements is necessary.

PHP has a long history with the MySQL database system and PHP web applications were and are often tightly coupled to this specific database. Since PHP version 5.1 a concept called PHP Data Objects (PDO) was introduced as abstraction layer for SQL-based database systems. By using PDO and standard SQL statements it is now possible to change the database system without changing much of the source code of the web application. There are also object-relational mapping (ORM) libraries from third parties available, but they are not used by most of the existing content management systems, PHP web frameworks use various of them.

Python has specified a database API in PEP 249[2] to encourage similarity between the different database modules for accessing different database systems. When using popular Python web frameworks there are very often ORM tools included that add a further abstraction layer to object-oriented database access.

Ruby makes use of the ActiveRecord system to provide an ORM pattern in the only popular web framework Ruby on Rails. It accomplishes similar tasks as ORM tools in the other languages' frameworks.

All in all there is support for database abstraction in all three languages and ORM is widely used in all web frameworks. PHP is a little bit behind the other two languages, as the abstraction has not been adapted by long-living PHP projects.

3.3 Functional language features

Python and Ruby have both support for functional language features. There are mechanisms for list comprehension used on lists and collection types; functions are treated as first-class citizens and there is a “lambda” keyword available to write anonymous functions.

Ruby implements closures - a concept to hand over code blocks as arguments to function calls (Python can achieve the same behavior). This is a functional feature using first-class functions with free variables that keep their scope and lifetime in the closure itself.

PHP has no special focus on functional programming, although it is possible to store a function name as string in a variable and then use it to call a referenced function. This is just a very basic and incomplete functional behavior, therefore PHP cannot be classified to provide functional paradigms.

3.4 Interactive development with the interpreter

The PHP, Python and Ruby standard implementations all offer the possibility for interactive development with the interpreter. This means that the interpreter can be started in an interactive mode, where some sort of command line accepts statements and expressions in the language line by line and prints out immediate results. It can be used for example to test small code snippets directly and observe the outcome, or to experiment with the language behavior without needing to edit, save and run files containing code. This features is valuable especially for beginners, but also for experienced programmers for testing purposes.

Interactive development and testing is existent and usable in all three languages very well.

3.5 Duck typing

Duck typing is a paradigm for polymorphism in dynamically typed object-oriented languages, where properties and methods of an object determine the applicability and not the type definition of the object. Python and Ruby promote and encourage the use of duck typing and offer appropriate exceptions to catch errors, while PHP is not specified as a duck typed language and recommends the use of polymorphism via inheritance. PHP introduced a concept called type hinting, where arguments of functions or methods are annotated with a specific type and throw errors if wrong types are handed over.

Programmers that design software with the duck typing approach should consider using Python or Ruby instead of PHP.

3.6 Major drawbacks of PHP

There are some major drawbacks of the PHP platform, which are described by Nikolaj Cholakov[8] and will be summarized here. PHP allows the use of uninitialized variables, often has inconsistent naming conventions and similar functions use a different ordering of parameters. There are no namespaces (yet), so functions need to have unique names; The number of available functions is really high, some of them performing the same tasks and therefore confuse programmers. Error handling is solved twice: procedural with the function "set_error_handler()" and object-oriented with exceptions.

PHP also had notorious features like "register_globals" and "magic quotes", but they have been removed from the language and will not be discussed here anymore.

3.7 Other differences

There are also other detailed differences between the languages, that are not particulary important for web applications. PHP has no support for threads (concurrent programming) and Python and Ruby offer a "yield" statement for generator functions (there is a semenatic difference between Python and Ruby how yield is used), just to name a few details that are not covered here.

4 Security comparison

The most important security-critical part in web applications is user-generated input. To avoid attacks it is crucial to sanitize and secure all data that is handed over to web applications from the outside, like user and password credentials, comments, forum posts or other content. In this section I will discuss some popular attack techniques.

4.1 SQL injection

SQL injection is an attack that makes use of characters that have a special meaning in SQL statements that are used to query databases (for example the string "−−" is used as comment escape sequence). In PHP it is possible to use functions to get rid of this special characters (see listing 1.2 for example), but this is database-dependent and not portable. It is recommended to use prepared SQL statements with PDO (PHP Data Objects), where the query is build in a safe manner (malicious data is encoded or escaped automatically). Python accomplishes the same with the database API (see listing 1.5) and Ruby also provides this automatic support in the Rails framework. If these APIs are used, then SQL injection is not a problem for all three languages.

4.2 Cross Site Scripting (XSS)

Cross site scripting is an attack that inserts a prepared, malicious script fragment into the content of a web application, which is then executed in the browser of a victim. The countermeasure is to rigorously sanitize user provided (in this case attacker provided) input. PHP, Python and Ruby offer functions to escape potentially dangerous input and many template engines in web frameworks provide escaping or even do it automatically themselves. All three languages are prepared well to handle it.

4.3 Server Side Code Execution

Scripting languages often have a very dangerous feature that undermines the security of a web application: “eval()” or “exec()”. The purpose of this function is to execute code that is stored inside a string variable. If the content of the string variable is user generated, the usage of eval leads to a huge security hole. All three languages provide this function, but the usage is discouraged.

4.4 PHP security flaws

PHP can be used in a secure manner, however there are also some core security problems that result from a poor language design. There are projects like Suhosin[1] that provide a protection system and a PHP hardening patch, which fixes issues concerning buffer overflows or format string vulnerabilities. PHP allows poor programming practices which result in many security related bugs in web applications. About one third of stored vulnerabilities by the National Vulnerability Database is PHP-related. Considering this, PHP might not be the best choice for security-critical applications.

5 Performance comparison

Performance, speed and responsiveness are important factors for web applications, although such characteristics are very difficult to measure objectively. The whole system environment influences performance: network speed and response time, database connectivity, web server performance, server hardware power, amount of simultaneous requests etc.

However, there is also the impact of the programming language used, i. e. the implementation of the specific language. The computer language benchmark game[3] is one attempt to compare different programming languages regarding performance. The website says “How can we benchmark language implementations? We can’t - we measure particular programs.” There are programming tasks solved in commonly known languages and they are run on the same hardware and operating system. The results rank the standard implementations of PHP, Python and Ruby at the end of the list, mostly because these are scripting languages that are not compiled to machine code, but interpreted at runtime. All three languages are relatively close to each other, in the overall ranking not more than 10% different. None of these three languages has a significantly better performance than any other. Interestingly Ruby got a huge performance improvement comparing version 1.8 to 1.9, where the median results of 1.8 are nearly twice as bad (this seems to be related to the use of a byte code compiler and virtual machine execution in 1.9). There is also the JRuby implementation listed, that performs very similar to the reference implementation.

A common way to improve performance is the use of caching systems for speed optimization. PHP extensions are very common and widely used that

[1] http://www.hardened-php.net/

cache compiled bytecode to avoid parsing and compiling the source code on each request. Python also offers a caching system for web applications, namely memcached[2], a more generic memory caching system originally developed for the Django web framework. Ruby is also capable of caching mechanisms like using the mentioned memcached.

Overall it is difficult to nominate a performance winner, as the languages all rank nearly equally.

6 Related work

There are many papers that deal with web application engineering: Mehdi Jazayeri wrote an interesting article about trends and status quo of web applications[14], Vosloo and Kourie presented an overview of web frameworks and concepts[22] and Armin Ronacher documented very practical security-related issues[19]. This is just a small selection that is explicitly used in this paper, it is expected that there will be even more publications in th future, as web development is constantly growing.

Scripting languages are also covered by many papers, there is the work of Luis Rei et al. that takes a look at dynamic languages in general[18] and Ronald Loui describes programming pragmatism when using scripting languages[16].

Additionally there is material on each of the compared languages in this paper. Nikolaj Cholakov analyzed PHP and summarized some drawbacks[8] and Xavier Spriet presented some tips for PHP security and configuration[21]. Python programming is introduced and explained by Greg Lindstrom[15], David Geer took a look at Ruby on Rails[11].

As PHP, Python and Ruby are free and open source languages, there is much related work and help available online, see the provided web resources[13][10][9].

7 Conclusion

This paper presented various aspects of the three dynamic programming languages PHP, Python and Ruby from a web application engineering point of view. Properties, features and risks were discussed and analyzed in order to give a comparison between the languages. The benefits and drawbacks for programmers (users) of the languages have been pointed out.

The decision which language to choose should base on the use case at first: what is the web application about to accomplish? If it is used for common tasks like a web shop or a news site, than maybe a PHP content management system is the best choice. It can be setup very fast and extended for personal needs. The disadvantage is flexibility: extensions are tied to the system's data model and the internal program flow. If the software should be used for more specialized purposes, then it is surely better to write it from scratch with a web framework. In this case consider table 1, where criterions that are important for your use case are listed with the recommended languages in an order from best to worst.

[2] http://danga.com/memcached/

Table 1. Comparing PHP, Python and Ruby based on several criterions

criterion	detailed description	best	middle	worst
Popularity	job market	PHP	Python	Ruby
Availability	on most systems existent	PHP	Python	Ruby
Readability	maintainability and personnel changes	Python	PHP Ruby	-
Usability	rapid prototyping and development	Ruby	Python	PHP
Security	for critical use cases	Python Ruby	PHP	-
Performance	speed and execution time	PHP Python Ruby	-	-
Database abstraction	vendor-independence and object-relational mapping	Python Ruby	PHP	-
Exception handling	error control and recovery	Python Ruby	PHP	-
Functional features	possibility of functional programming techniques	Python Ruby	-	PHP

To have a better understanding how much the languages differ from each other, have a look at figure 1.

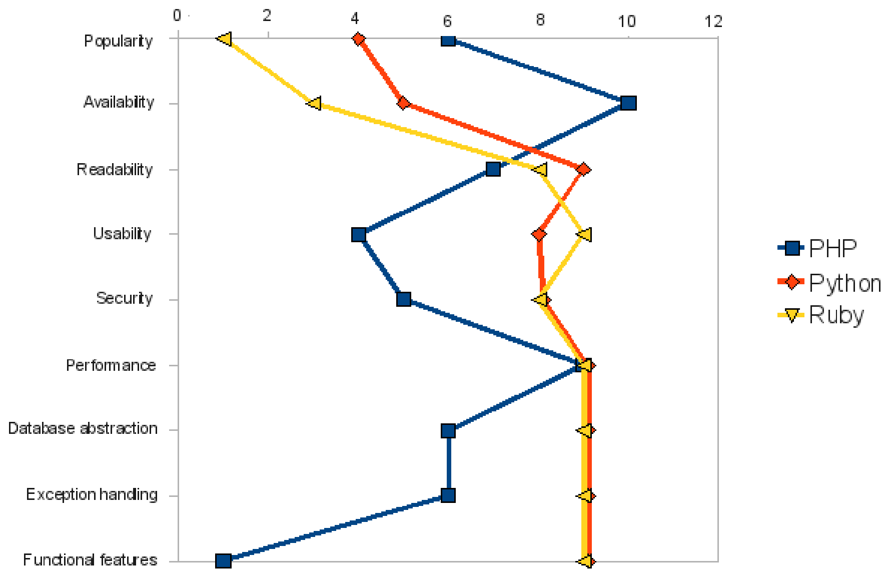

Fig. 1. Relative comparison of the languages to each other, rating from 0 (worst) to 10 (best)

Disclaimer: The languages PHP, Python and Ruby are under permanent development, so this table will change probably in the next months or years. The analysis of the languages tried to derive the outcome from objective facts – however, choosing a programming language is always connected to personal experiences, therefore discussions about languages are often emotional and irrational.

As a final recommendation I would suggest to use Python for the general use case of a web application, Ruby is closely behind but has not yet catched up in popularity. PHP should only be used with the popular content management systems where not much programming and customization is necessary.

References

1. *Python enhancement proposal 343*, http://www.python.org/dev/peps/pep-0343/, 2006.
2. *Python enhancement proposal 249*, http://www.python.org/dev/peps/pep-0249/, 2008.
3. *The computer language benchmarks game*, http://shootout.alioth.debian.org/, 2009.
4. *How popular are various programming languages?*, http://www.complang.tuwien.ac.at/anton/comp.lang-statistics/, 2009.
5. *Programming language popularity*, http://www.langpop.com/, 2009.
6. *TIOBE programming community index*, http://www.tiobe.com/index.php/tiobe_index, 2009.
7. *What is WSGI*, http://www.wsgi.org/wsgi/What_is_WSGI, 2009.
8. Nikolaj Cholakov, *On some drawbacks of the php platform*, CompSysTech '08: Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing (New York, NY, USA), ACM, 2008, pp. II.7–2.
9. The Ruby Community, *Ruby documentation*, http://www.ruby-lang.org/en/documentation/, 2009.
10. Python Software Foundation, *Python documentation*, http://www.python.org/doc/, 2009.
11. David Geer, *Will software developers ride ruby on rails to success?*, Computer **39** (2006), no. 2, 18–20.
12. The PHP Group, *PHP language reference*, http://php.net/langref, 2009.
13. ———, *PHP manual*, http://php.net/manual, 2009.
14. Mehdi Jazayeri, *Some trends in web application development*, FOSE '07: 2007 Future of Software Engineering (Washington, DC, USA), IEEE Computer Society, 2007, pp. 199–213.
15. Greg Lindstrom, *Programming with python*, IT Professional **7** (2005), no. 5, 10–16.
16. Ronald P. Loui, *In praise of scripting: Real programming pragmatism*, Computer **41** (2008), no. 7, 22–26.
17. Trygve Reenskaug, *Models - views - controllers*, Tech. report, Technical Note, Xerox Parc, 1979.
18. Luis Rei, Sara Carvalho, Marco Alves, and João Brito, *A look at dynamic languages*, Tech. report, Faculty of Engineering University of Porto, 2007.
19. Armin Ronacher, *Sicherheit in Webanwendungen*, http://dev.pocoo.org/ blackbird/fachbereichsarbeit.pdf, 2006.

20. Python software foundation, *Python language reference*, http://docs.python.org/reference/, 2009.
21. Xavier Spriet, *Real-world php security*, Linux J. **2004** (2004), no. 120, 1.
22. Iwan Vosloo and Derrick G. Kourie, *Server-centric web frameworks: An overview*, ACM Comput. Surv. **40** (2008), no. 2, 1–33.