Rexx Vs Python: A Comparative Study

Rohit Sadavarte[1, a)] Dr. Vishalakshi Prabhu[2, b)]

[1]*Computer Science and Engineering, R.V College of Engineering,* Bangalore, India
[2]*Computer Science and Engineering, R.V College of Engineering,* Bangalore, India

[a)]*rohitsachins.cs18@rvce.edu.in*
[b)]*vishalaprabhu@rvce.edu.in*

Abstract: Programming languages have been amedium through which instructions have been given to machines since the the inception of the field. The languages have developed over the ages. These languageshave seen iterations such as the various older languages which started at the basic level of the memory access through instructions, and moved onto better tech where the languages were more powerful, included not onlymore complexity but also more functionality. This func- tionality has come in handy when it comes to various applications of languages as well. The languages can be broadly classified into various categories, including but not limited to low level, high level, object oriented, assembly, hardware language, machine language and many more. Here, we go through two popular scripting languages, Rexx, and python and compare their features.The age of the Rexx language should obviously meanthat the language has lesser features, but an attempt to measure the extent of its effects on scripting has been made here.

Index Terms—Rexx, Python, Language, Comparison

INTRODUCTION

Programmers and computer scientists alike fre- quently have strong opinions about the advantages and disadvantages of various programming languages. Butat the same time, there is a dearth of reliable, accurate information regarding the relative advantages of vari- ous languages. The scientific and engineering literature offers numerous programming language comparisons, done in various ways and under various constraints.

Most of the comparisons thus done are highly opin- ionated and have less amount of data behind it. This work aims at comparing two languages, that is, Rexx and Python, with lesser opinions and relatively more experimentation. There, have been statistical studies before that have done overall averaged studies, such as [1] where a number of programmers writing multiple programs has been considered, and these studies have been taken relatively more seriously with respect tothe same.

Code has been written in both the languages andrun in both of them in this work, in order to reach the conclusions mentioned here. The code has had its ownapplications and works in similar (or in most cases, exactly the same) ways. Thus, the comparison is boundto be more or less uniform across the board.

The logos for the two language projects can be seenin the only two figures of this work.

Fig. 1. Logo of the Rexx language

RELATED WORKS

Previous works in the field have gone over differentconstraints of the works such as in [2] where the entire work has been theoretical. The work has no quanti- tative results and is quite speculative. Many a time, such a comparison has turned into a benchmark contest where the benchmarks were compared as they're the most efficient method of comparison. This, however is wrong due to the fact that ease of use and comparison is thus not compared in such a work, such as [3].

There have been other experiments where a degree of control has been attempted to be introduced, based on certain construct of a language or some style of notation such as in [4] and [5]. These experiments have gone up in the size of programs to have larger programs being executed. Some of these analyses compare data empirically across numerous, larger pro- grams. They talk about things like failure rates and production levels such as [6]. Lack of homogeneity in these comparisons is a problem. It is uncertain what percentage of the variations (or lack of differences) stem from the languages themselves and what per- centage is attributable to differing programmer back- grounds, software processes, application areas, design structures, etc. Each language is represented by a distinct program.

There are other works like [7] where a direct comparison of scripting languages with other languagesis done, this can also form a base for this work, as another method of comparison between Rexx and python is with respect to the fact that python canbe used for whole applications, including front-endin libraries such as Tkinter or PyQt and other back- end services such as Django, while also paving the path in web3 with web3py. These applications canbe seen summarised in [2] and [8]. Here, the obvious application of python where it is used in places such as Machine Learning, Data analytics and others should also be considered as in [9], since the boom has happened, bringing languages such as python and R into the limelight.

Fig. 2. Logo of the Python project

COMPARISON

In this section, the methodology of each type of comparison has been talked about first, and then we goon to see the actual comparison done. Each subsectionhere is a unique comparison in it's own.

Syntactic Differences

Since python is more than just a scripting language, the syntactic complexity is higher in python than in Rexx. Although things like variables are data typesare similar in the two languages, such as

```
variable = value
```

being the exact same in both of the languages. Both ofthese languages don't have a method for declaring the data type, unlike in languages like Java, C, C++ and others. Here, sticking to these languages, we see that the operations are also lesser in number in Rexx, such as, the fact that multiple assignment is not possible in Rexx.

```
x = y
y = y +1
```

whereas, in python, we can have

```
x,y = y, y+1
```

These differences do not make the big differences to the coder, so moving onto the differences that make a difference to the developer, we have the difference in importance for indentation among the two languages. The usage of indentation importantly is an unique feature of the python language. This can be seen as both, a boon and a bane. The fact that code is already indented makes it easier to read by default, whereas in other languages, readability of code becomes a huge issue unless written by a coder with good practices. That being said, python is helpful for beginners in coding as the complexity is lower, and it creates good coding practices. Thus, in python , we have indented blocks that form the code, while in Rexx, indentation is not necessary, but simply a tool to make code readable.

Data Structures

Python can be a classful language, so it can have essentially infinite data structures such as [10], while also having many data structures predefined, such as dictionaries, lists and has strong libraries for graphs and other data types. Rexx on the other hand, es- sentially has two data types, strings and integers. This however, has been extended to create lists with creation of functions like

```
word(x,#) pos(string, x)
parse var x ele1 ele2reverse(x)
```

where list like operations can be done. The word() function for example, can be used as an indexing method, on the string, as it finds the nth word in the string. Here, the string looks like :

```
string = "element1 element2element3"
```

and so on.

Thus, we can concatenate multiple strings to add elements to the list as well. The pos() function can then be used to reverse map the index to the numberof letters, so that other functions that are not mentioned here can be used.

In python on the other hand, we have lists as adatatype, where we can use indexing, appends, deletes, and other functions directly. The operations on this listis more natural and thus can be said to be better.

```
list = ["element1","element2","element3"]
```

The above are simply one dimensional lists, butsince python allows multiple dimensional lists andmore, we can use those for particular applications,especially in fields such as data science, where the number of dimensions in the can go in the hundreds. The libraries like sci-kit-learn [11], tensorflow [12], and keras [13] also have a lot of their own classes, objects and data types in general that are usually basedoff numpy arrays. These data structures are never used in python as a scripting language, but are worth mentioning here.

There are also other data structures such as dic- tionaries and arrays, and multidimensional lists in python [14]. Libraries like numpy [15] and pandas [16] have their own data structures that are much more powerful, in their own regard. The operationson these multidimensional arrays as described by theirrespective libraries are also immensely powerful.

Control and flow of the program

One of the biggest paradigms of modern program- ming is control flow. Control flow is the concept in modern programming where the control flows from one stub to another. The flow thus forms the entire program. This is usually limited to functions and func-tional programming, but here we include conditional statements and also loops. Thus, programs can be described with flowcharts that essentially describe this,but pictorially.

Starting off with functions, although Rexx has the capability to write functions, most of the functions are essentially go-to statements, where a bunch of lines areredone based on the calls. Thus, these functions have no parameters and no return values. It is a control flowand only a control flow. Nothing beyond. Example, a typical function would look like :

```
functionname :
code goes herereturn
```

and the call to the function just uses a keyword call,that is

```
call functionname
```

This is useful only when it comes to scripts where the exact same lines need to be executed, with barely any scope for parameters. The functions are also file specific, that is, a function from one file cannot be runin another file, moreover the absence of classes also makes the entire application less powerful.

Moving onto python, the functions can be put in classes, the classes have default functions and can be parameterized. Overloading is not supported by defaultunlike in languages like Java, but instead overwriting takes place when newer functions are written with the same name. This is especially true since python is an interpreted language. The functions here can be writtenas :

```
def functionname(parameters):code goes here
    return var
```

The calls here do not require any keyword, and can be directly called as in

```
returnvalue = functionname(parameters)
```

The return statement in Rexx is to transfer control back to the calling line, without which, the flow would simply go on to the end of the file from the functionin case it is written after the call in the file, and an infinite loop otherwise. Whereas in python, the return statement is optional and does not need to be written always, the control is returned to the calling line right after the block of the function ends.

The return statement also returns a value back tothe call, which is why we have a returnvalue variable that stores the value returned by the function in the example above.

Moving onto if conditions and loops, there exist for and while loops in python, where as in Rexx, all the loops are do loops. Although for loops can be emulatedby usage of :

```
do <number>
```

The other loops in Rexx are :

```
do while ... enddo until ... end
```

The usage of these loops would be similar in both the languages, that is, having to script the similar functions. Although it is true that the iterators in python are much more useful when compared to the loops in Rexx, since the iterators can be used for moreversatile applications.

Finally, in this section if conditions can be dis-cussed. The working of if conditions is trivial and the only difference in the two languages here is syntactic.

In Rexx, one would write an if condition as

```
if condition then dostatements
else do statements
```

A noticeable feature is the absence of indents. They are not mandatory to be used, whereas in python we have

```
if condition :statements
elif condition:statements
else:
    statements
```

The difference is purely syntactic and the workingis exactly the same, since an if condition is the most basic building block of any programming language.

Strings

Both python and Rexx have string handling capa- bilities. The major difference however, is the way in which string handling functions work. The functions are bound to be more in number in an open source language which was contributed to, by people all over the world, than in a proprietary language like Rexx created by IBM. For the sake of this comparison,

```
string[start:end]
```

The common functions in python that are used for string manipulations are

```
split lower upper endswithindex replace
```

Speed of Execution

The speed of execution in the case of python is expected to be slower than in the case of Rexx due to the fact that scripting is only a feature of the language,while Rexx was made only for scripting, thus making

`translate('abc','xv','ab') == 'xvc'` the language as a whole very lightweight and quick Some of these functions maybe more useful in scripting, which is why they are provided by defaultin Rexx. Whereas, in the case of python, while these functions can be written with relative ease, due to the presence of data structures like dictionaries, they do not come packaged with the language.

It is however worth mentioning, that this is oneof the few functions in Rexx that cannot be found elsewhere, and most other functions are more or elsestandard across the board for most modern languages. Moving to the other functions, both of these lan- guages have pretty much the same functions in termsof string handling. An exception to this is due to thefact that lists or arrays are simulated in Rexx through these strings. Thus, we have string functions that countthe number of words, and a few basic functions to manipulate those words built into the language. Thisunorthodox way of dealing with lists has its own pitfalls, but it provides for a few string functions thatsome scripts would benefit from. This is because, a lot of scripting involves reading from log files, andlog files are strings, which would be needed to be converted into lists for them to be able to be used inpython scripts, but instead, here, we simply use the inbuilt functions to achieve the same.

Some of these functions are mentioned below

```
DELWORD SUBWORD WORD WORDINDEXWORDPOS WORDS
```

Most of these functions are self explanatory, where the DELWORD function deletes a word from a string after some n words, the WORDS function unpacksthe "words" in the string into the variables, and so on. Thus, the functioning of strings in the two languages can be said to be stronger in Rexx, but python being more adaptable to newer changes.

For the sake of completeness, the common ways in which python strings are worked with are slicing and only similar functions that exist for both languagesare going to be compared.

The replace function for example in the case ofpython replaces a certain string with another string. The corresponding function in Rexx, which is the translate function, can be used to have a reference tablein and out, where the characters in the in table and those in the out table are mapped and those characters are replaced. A function that is mostly absent in other languages. The syntax of this function is

```
TRANSLATE(string [,[tableout][,[tablein] [,pad]]])
```

A better understanding can be obtained from the example below.
to run on servers and otherwise. On an average, the triggers by Rexx can be said to be faster than Rexx. However, when it comes to scripts where meaningful work is being done, the python script prevails, this also depends on other factors such as the programmer, the expertise of the programmer, the data structures that need to be used in the course of the program, but on anaverage, similar scripts that perform similar functions should take the same amount of time, as long as no complex logic or algorithmic coding is done in these scripts, as is common for scripts.

Program writing

A major part of comparison of languages is thelearning curve, the availability of information and the ease with which programs can be written. This has been made into a section, since there is statistical evidence on the same, to look at whether coding inone language is faster than the other. Learning of the language on the other hand is a very subjective term and is relatively harder to compare. But to put it fairly, the Rexx programming language has lesser resources when compared to python due to he difference in theirpopularity, and the usage of the two languages is also comparatively in favour of python.

However, from a dev-ops perspective, where script- ing is the main way in which programming is done, Rexx is more similar to batch files and command files, thus making it the easier choice. Python, although comparatively harder in this regard, does not have as steep a learning curve as some other languages.

Coming to the statistical analysis, as done in [17], the amount of time taken by the programmers who wrote the same program in Rexx and python, wascomparable, with python having the lower averagetime. It can be seen that statistically, there is more con- sistency among a decent number of python program- mers, which is more or less absent in the programmers of other languages, including Rexx. This, however is a statistical study, but since all the people involved were volunteers, we can say with a fair degree of certainty that the standard deviation in case of python is much lower, with the mean time also being lower when compared to the other languages in the same comparison.

Other comparisons have also been made in the above cited work, but those are trivial, and somewhat subjective in a lot of ways.

Usage and applications

As mentioned previously, python is used immenselyin programs where large scale development needs to be done and it needs to be done fast. Python is versatileand is not limited by functionality and with the help of modern libraries, can practically perform any other task. Rexx on the other hand is limited by the number of publicly available libraries.

Python has been used traditionally for everything from beginner level programming to large scale cloud and Artificial Intelligence applications. Rexx, due to itslimitations has been a scripting language and nothing else. This limitation of has been mentioned over and over again in this work, but that is the only reason why Rexx has the disadvantages it does. Python being as popular is due to its ease of use, versatility and the contributions of thousands of people all over the world.

CONCLUSION

While python is more versatile, has more applica- tions and can be used to create full stack applications, Rexx has its own advantages in terms of ease of coding and speed of execution when it comes to smaller scripts. Rexx also happens to be more lightweight, as it was designed for this very purpose. Python having object oriented concepts, the ability to write scripts and other small and large scale programs, has advantages with respect to exception handling, support for modernfull stack applications, REST and otherwise.

To pass a final verdict, everything that can be done by Rexx can now be done in python, and rightly so, asit has reached its end of life, but the minor advantages of Rexx may make it worthwhile to use in a nicheof applications, where a large amount of scripting is involved. Nonetheless, it provides an upgrade over thebatch file scripts written in windows to do certaintasks.

REFERENCES

[1] L. Prechelt, *An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program,* ser. Interner Bericht. Fakultät für Informatik, Universität Karl- sruhe. Universität Karlsruhe (TH), 2000, vol. 2000, no. 5.

[2] E. W. Dijkstra, “Letters to the editor: Go to statement considered harmful,” *Commun. ACM*, vol. 11, no. 3, p. 147–148, mar 1968. [Online]. Available:https://doi.org/10.1145/362929.362947

[3] P. Hudak and M. A. Jones, in *An Experiment in Software Prototyping Productivity*, 1994.

[4] Lewis, J. A. Henry, S. M. Kafura, D. G. Schulman, and R. S., “On the relationship between the object-oriented paradigm and software reuse: An empirical investigation,” 1992.

[5] G. Stevens, G. Quaisser, and M. Klann, *Breaking It Up: An Industrial Case Study of Component-Based Tailorable Software Design*, 11 2006, vol. 9, pp. 269–294.

[6] L. Hatton, “Does oo sync with how we think?” *Software,IEEE*, vol. 15, pp. 46 – 54, 06 1998.

[7] L. Prechelt, “Are scripting languages any good? a validation of perl, python, rexx, and tcl against c, c++, and java,” *Advances in Computers*, vol. 57, pp. 207–271, 2003.

[8] A. Nagpal and G. Gabrani, “Python for data analytics, scien- tific and technical applications,” in *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 140– 145.

[9] S. Raschka, *Python Machine Learning.* Packt Publishing - ebooks Account, 2015. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1783555130

[10] W. McKinney, in *Data Structures for Statistical Computing inPython*, 01 2010, pp. 56–61.

[11] O. Kramer, *Scikit-Learn*. Cham: Springer International Publishing, 2016, pp. 45–53. [Online]. Available: https://doi.org/10.1007/978-3-319-33383-0 5

[12] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo,
S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. D. Hoffman, and R. A. Saurous, “Tensorflow distributions,” *CoRR*, vol. abs/1711.10604, 2017. [Online]. Available: http://arxiv.org/abs/1711.10604

[13] N. Ketkar, *Introduction to Keras*. Berkeley, CA: Apress, 2017, pp. 97–111. [Online]. Available: https://doi.org/10.1007/978- 1-4842-2766-4 7

[14] M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, *Data Structures and Algorithms in Python*, 1st ed. Wiley Publish- ing, 2013.

[15] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers,
P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg,
N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. vanKerkwijk, M. Brett, A. Haldane, J. F. del R´ıo, M. Wiebe,
P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy,
W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[16] W. Mckinney, “pandas: a foundational python library for data analysis and statistics,” *Python High Performance ScienceComputer*, 01 2011.

[17] L. Prechelt, “An empirical comparison of seven programming languages,” *Computer*, vol. 33, no. 10, pp. 23–29, 2000.