

Unlocking Programming Skills In Nigerian Youths Through Software Development In Local Languages

Arthur Ibukunoluwa Arokhamoni,
Chief Solution Architect, ArtMannTech
Solutions, Ilaro Ogun State, Nigeria.
sureandsteadfast@gmail.com
+2348030677646

Dr. Joel Ogunyemi (MNSE)
Department of Electrical/Electronics
Engineering, Federal Polytechnic Ilaro.
joel.ogunyemi@federalpolyilaro.edu.ng
+2348052220569

Isa Hassan Usman (MNSE)
Department of
Electrical/Electronics Engineering,
Federal Polytechnic Bauchi.
usmanih@fptb.edu.ng
+2348062327167

Abstract

The inability of Nigeria to emulate India by leveraging software development as a major driver of economic development resulted in an unprecedented differential GDP of \$280.78 and \$2,987.07 between 1981 and 2021 for the former and latter respectively. Therefore, the objective of this paper is to address the imbalance through the development of a new programming language codenamed “9ja-Reboot”. It will enable Nigerian youths to gain mastery of popular and lucrative programming languages like Python, by teaching them in the following selected Nigerian languages: Pidgin-English, Hausa, Igbo and Yoruba. The research will develop 9ja-Reboot by building a new language lexer and parser for Python 3. It will validate and convert codes written in non-English syntaxes into Python’s English syntax. This ensures codes written in Nigerian languages produces similar runtime-results to those written in English language. 9ja-Reboot is expected to stimulate national interest in programming in Primary, Secondary and Tertiary institutions.

Keywords: *Software Development, GDP, Python, 9ja-Reboot*

I. INTRODUCTION

The ever increasing demand for computer software has consistently driven the global revenue in the software industry, for example, the global revenue from computer software sales and licensing was \$569.34 billion in 2021, and it is projected to hit \$806.24 billion by 2027 (Statista, 2021). Therefore, the yearly impressive revenue numbers from the global computer software industry made Brandcomb, 2019 to declare that computer software is the “crown jewel of the information economy” (Brandcomb, 2019).

Indian youths were quick to tap into the “crown jewel of the information economy”, hence they are currently reaping billions from sales of computer software, while Nigeria youths are spending their hard earned foreign reserve to acquire computer software developed in India and elsewhere.

This paper addresses this imbalance through the creation of a new language parser for Python 3 that will fast-track programming knowledge acquisition amongst Nigerian youths. The parser will

allow youths to interact with Python in a language they can understand, namely English, Hausa, Igbo, Pidgin-English and Yoruba. The paper adopted Python as a base language for building the new parser called 9ja-Reboot because the TIOBE Index for October 2022 ranked Python as the most popular programming language in the world (TIOBE, 2022). Moreover, major companies worldwide are using Python, some of these companies are: Google, Facebook, Quora, Amazon, Stripe, Instagram, Spotify, Netflix, Uber, Reddit, Dropbox, Pinterest, NASA, Instacart, Lyft, Industrial Light and Magic (Cordenne, 2021).

II. RELATED WORKS

Developing a new computer language requires using a lexer to define the lexical rules of the new language and to ensure program input it receives matches the lexical rules. A parser translates lexicon rule tokens received from a lexer into an abstract syntax tree (AST) (De. Andrade, 2017).

A lexer receives inputs supplied to a program and divides it into tokens through lexing or tokenization of the program input using predefined lexical rules. This process can also be described as parsing; however, tokens generated by lexers are often forwarded for additional processing to a parser. Python has a great tool for lexing called PLY (Beazley, 2022). A programmer can use PLY to:

1. Define tokens known to the program.
2. Create regular expression that the tokens should adhere to.
3. Create rules which the regular expressions must follow.

The process of tokenizing the program input below is captured in Figure 1:

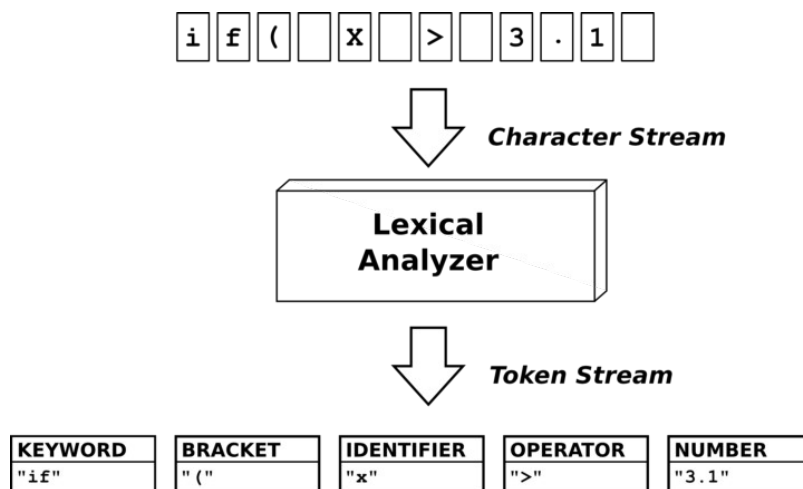


Figure 1: Lexer process of transforming a string into a list of tokens (De. Andrade, 2017).

Parsers execute a syntax check of the tokens generated by the lexer. It takes the list of tokens as input and creates abstract syntax tree (AST) as output, this process is captured in Figure 2 below (De Andrade, 2017).

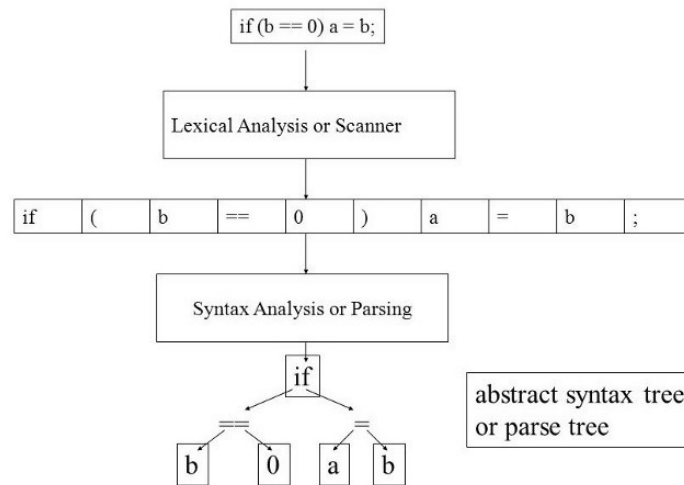


Figure 2: Lexical analysis followed by parsing to create the AST (De. Andrade, 2017).

Several libraries exist in Python which enable programmers to use lexer and parser functions in their applications, two popular libraries are enumerated below:

PLY: PLY was designed in 2001 for use in an “Introduction to Compilers” course where students used it to build a compiler for a simple Pascal-like language. It was eventually ported to Python by the author thereafter. However, the author announced on October 11, 2022 that PLY will no longer receive official updates. PLY consists of two files, lex.py and yacc.py that can be included in any Python code (Beazley, 2022).

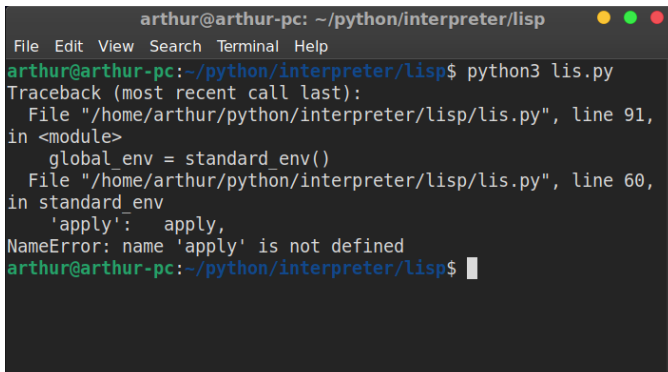
SLY: Beazley (2022) explained that SLY provides two separate classes Lexer and Parser. The Lexer class is used to break input text into a collection of tokens specified by a collection of regular expression rules. The Parser class is used to recognize language syntax that has been specified in the form of a context free grammar. Beazley discontinued official updates to SLY on October 11, 2022.

Several researchers embarked on the challenging task of using Python to either build a new programming language or to emulate an existing programming language. Some notable efforts are enumerated below.

Lisp Programming Language with Python 3.x

Lisp or LISP was developed a year after FORTRAN in 1958, Lisp uses parenthesized prefix notation for computer codes and was one of the first computer languages that was used by researchers to study artificial intelligence in computers (Reilly, 2003). Norvig (2012) developed a

simple LISP interpreter in Python 3. He later extended his interpreter by adding new LISP data types such as string, boolean, complex, and port using Python. Although Norvig claimed on his website that his lis.py interpreter is updated for Python 3, the program displayed error messages and refused to execute on Python 3 as shown in Figure 3. The researchers discovered the existence of Python 2.x codes in his program on lines 60 and 98. The researchers fixed the aforementioned depreciated codes and tried re-executing lis.py, the program ran and exited without showing any output as show in Figure 4, the researchers fixed this by adding a `__main__` module that called function repl(), this produced the desired result as show in Figure 5.

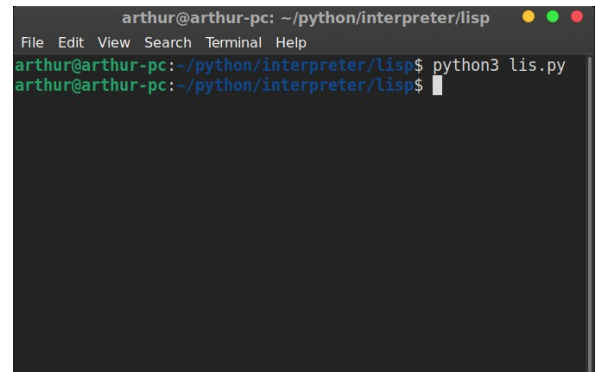


```

arthur@arthur-pc: ~/python/interpreter/lisp
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/lisp$ python3 lis.py
Traceback (most recent call last):
  File "/home/arthur/python/interpreter/lisp/lis.py", line 91,
in <module>
    global env = standard_env()
  File "/home/arthur/python/interpreter/lisp/lis.py", line 60,
in standard_env
    'apply': apply,
NameError: name 'apply' is not defined
arthur@arthur-pc:~/python/interpreter/lisp$

```

Figure 3: lis.py errors while executing due to legacy codes from Python 2.7

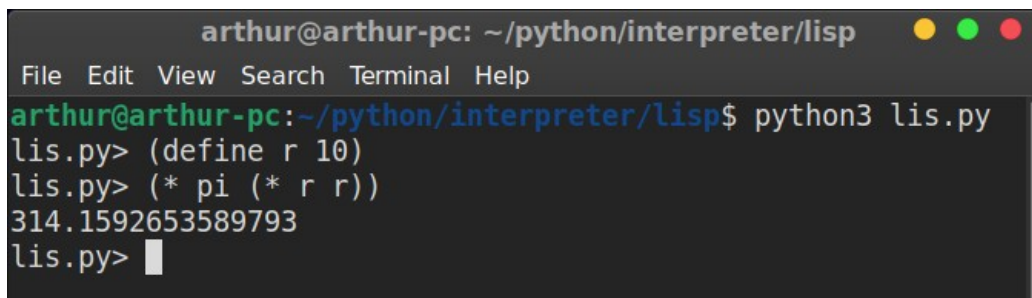


```

arthur@arthur-pc: ~/python/interpreter/lisp
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/lisp$ python3 lis.py
arthur@arthur-pc:~/python/interpreter/lisp$

```

Figure 4: No output after fixing Python 2.x legacy codes



```

arthur@arthur-pc: ~/python/interpreter/lisp
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/lisp$ python3 lis.py
lis.py> (define r 10)
lis.py> (* pi (* r r))
314.1592653589793
lis.py>

```

Figure 5: Using Lisp to compute the Pi of 10 x 10

Cell Programming Language with Python 3.x

Balaam (2016) developed a new programming language called Cell using Python 3.x, unfortunately Cell language syntaxes and coding styles are totally different from Python's, hence Cell is not suitable for teaching programming skills to beginners. Cell's design principle is "... to be as complete a programming language as possible, while also being as small to implement as possible. The implementation is designed to be easy to read, since the purpose is to demonstrate how to write a programming language" (Balaam, 2016). The researchers downloaded a copy of Cell programming language from its repository (Balaam, 2016), the program worked perfectly, however it failed to extend Python's rich libraries/modules, it is more difficult for beginners to learn and it produces longer codes compared to Python – this is show in Figures 6 and 7.

```

square = {(x) x * x};

num1 = 3;
num2 = square( num1 );

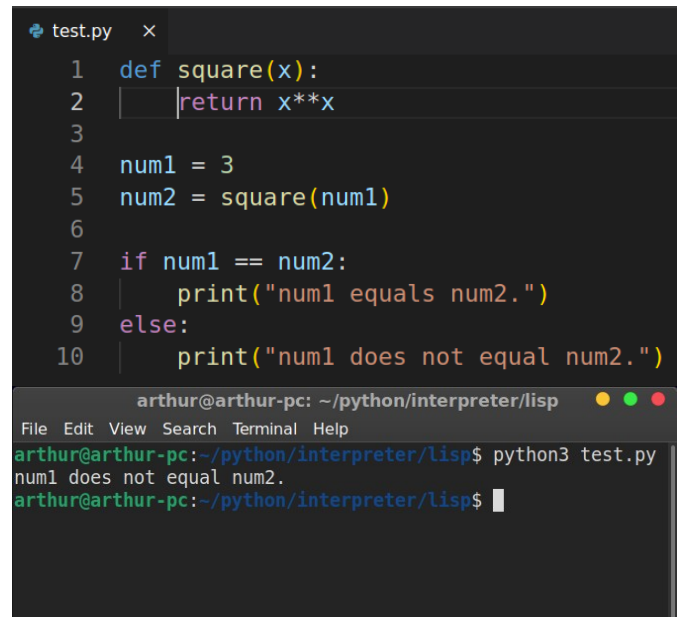
if( equals( num1, num2 ),
    {
        print( "num1 equals num2." );
    },
    {
        print( "num1 does not equal num2." );
    }
);

```

This prints:

```
num1 does not equal num2.
```

Figure 6: Cell program code and output. Code is 14 lines long.



```

test.py x
1 def square(x):
2     return x**x
3
4 num1 = 3
5 num2 = square(num1)
6
7 if num1 == num2:
8     print("num1 equals num2.")
9 else:
10    print("num1 does not equal num2.")

```

```

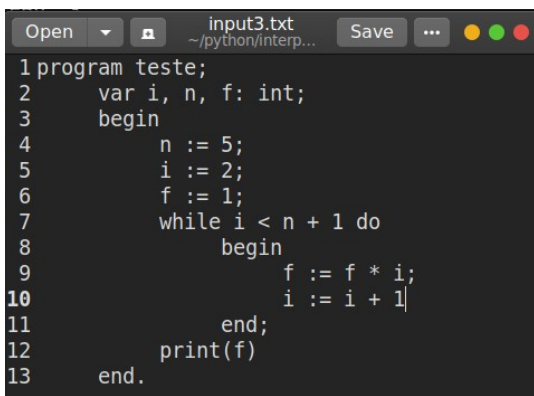
arthur@arthur-pc: ~/python/interpreter/lisp
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/lisp$ python3 test.py
num1 does not equal num2.
arthur@arthur-pc:~/python/interpreter/lisp$

```

Figure 7: Python program code and output. Code is 10 lines long for similar output.

Simple Pascal Compiler with Python 3.x

A compiler is a computer program that translates human-readable computer codes (source codes) into a lower level machine readable codes that can be executed by a computer, such as assembly language codes, object code, or machine code (Aho et al, 2007). Therefore for De Andrade (2017) to successfully develop a simple Pascal programming language with a compiler using Python is a testament to the versatility of Python programming language. The researchers downloaded and tested his Pascal compiler on Python 3.9 (De Andrade, 2017), it worked seamlessly by producing Pascal compiled codes. Figure 8 represents the screen captured image of the source text (in Pascal), while Figure 9 represents the compiled codes (in machine language) from PascalToyCompiler.

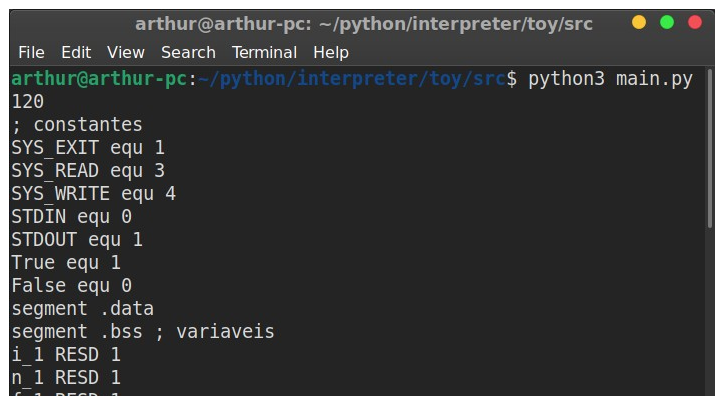


```

Open input3.txt Save ...
1 program teste;
2   var i, n, f: int;
3   begin
4     n := 5;
5     i := 2;
6     f := 1;
7     while i < n + 1 do
8     begin
9       f := f * i;
10      i := i + 1;
11    end;
12    print(f);
13  end.

```

Figure 8: Input Pascal code



```

arthur@arthur-pc: ~/python/interpreter/toy/src
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/toy/src$ python3 main.py
120
; constantes
SYS_EXIT equ 1
SYS_READ equ 3
SYS_WRITE equ 4
STDIN equ 0
STDOUT equ 1
True equ 1
False equ 0
segment .data
segment .bss ; variaveis
i_1 RESD 1
n_1 RESD 1
f_1 RESD 1

```

Figure 9: Output Machine language code

Having seen the limitations of Cell programming language, the researchers decided to extend Python by adding a simple parser and additional lexicons in English, Hausa, Igbo, Pidgin-English and Yoruba to Python. Attempts to add Fulfulde wasn't successful due to the difficulty of getting timely translations. Nonetheless we believe the supported languages will spark programming interest in Nigerian youths, especially when fully extended.

III. PROPOSED WORK

Although Python is easy to learn, however, this research aim to make it easier for Nigerian youths to learn and use Python in any language they understand. Therefore the task of extending Python 3.9 to support popular Nigerian languages commences with the modification of the Python custom interpreter file called **code.py** (GitHub, Python, 2022).

Python Docs described the custom Python interpreter modules as modules that "...allow writing interfaces similar to Python's interactive interpreter." It supports adding special feature in addition to the Python language (Python, 2022).

The researchers renamed **code.py** as **9ja-reboot.py** and made ample changes to the file. Aside from renaming and modifying **code.py**, the researchers wrote additional Python programs to that extends Python with additional language tokens. The additional files are: **wazobia.py**, **eng.py**, **hausa.py**, **igbo.py**, **pidgin.py** and **yoruba.py**.

9ja-Reboot

The test edition of 9ja-Reboot contains six tokens which are stored as Python list. The list is made up of the following items ["add", "sub", "mul", "div", "print", "quit"], which represents Addition, Subtraction, Multiplication, Division, Print, and Quit respectively. These tokens are equally available in Hausa as ["tarawa", "debewa", "sau", "rabawa", "buga", "daina"]. In Igbo as ["ngbako", "nwepu", "mubaa", "div", "ike", "kwusi"] – note that the researchers were unable get the actual word for division in Igbo. In Pidgin-English as ["add", "comot", "times", "divide", "print", "quit"], and in Yourba as ["ropo", "kuro", "dipupo", "dinku", "jade", "kuro"].

The main parser file for 9ja-Reboot is called **wazobia.py**. The file contains a single class called **WaZoBia** which is responsible for lexing or converting program inputs into tokens. Tokenized inputs are forwarded to Python for execution. WaZoBia handles unknown inputs by simply returning the input to Python for processing and execution.

The code listing for **eng.py** (which is the default language) is shown in Figure 10 below. Note that each of the remaining language file are translated from **eng.py**.

```

eng.py - python - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
PYTHON
> __pycache__
__future__.py
9ja-reboot.py
ast.py
code.py
codeop.py
custom.py
eng.py
hausapy
igbo.py
interpreter.py
pidgin.py
traceback.py
warnings.py
wazobia.py
yoruba.py

1 banner = ""
2 #cppt = 'Type "help", "copyright", "credits" or
  "license" for more information.'
3 intro = "\n\n9ja-Reboot Version 0.01 on Python "
4 loaded = "\n\nEnglish Loaded (Type: quit - to close)...\n"
5 customshell = "9ja-Reboot>>> "
6 exitmsg = "\n\nThanks for using 9ja-Reboot\n\n"
7 KnowCommands = ["add", "sub", "mul", "div", "print",
  "quit"]
8 PrintText = "print"
9 CommandText = "Commands:"
10 YouCanText = "You can also use any Python commands\n\n"
11 dLangauge = "eng"

```

Figure 10: English language file listing

Enumerated in Table 1 are test parameters used to test each 9ja-Reboot's language file:

Table 1: Test parameter

S/N	Description	9ja-Reboot Commands	Python 3 Commands
1.	<p>Civil Engineering: Compute the bending moment at a distance of x for a uniformly distributed load:</p> <p><i>Where x is 2</i></p>	<p><u>English and Pidgin:</u> x = 2 print f'Bending distance where x = 2 is {10 x 2 – x**2}'</p> <p><u>Hausa:</u> x = 2 buga f'Bending distance where x = 2 is {10 x 2 – x**2}'</p> <p><u>Igbo:</u> x = 2 ike f'Bending distance where x = 2 is {10 x 2 – x**2}'</p> <p><u>Yoruba:</u> x = 2 jade f'Bending distance where x = 2 is {10 x 2 – x**2}'</p>	<p>x = 2 print (f'Bending distance where x = 2 is {10 x 2 – x**2}')</p>

Table 2: Sundry Tests

S/N	Description	9ja-Reboot Commands	Python 3 Commands
1.	Additions	<u>English/Pidgin-English:</u> add 200 300 4000 or add 200+300+4000 or add 200,300,4000 <u>Hausa:</u> Replace add with tarawa <u>Igbo:</u> Replace add with ngbako <u>Yoruba:</u> Replace add with ropo	200+300+4000
2.	Subtraction	<u>English:</u> sub 700 200 100 or sub 700-200-100 or sub 700,200,100 <u>Hausa:</u> Replace sub with debewa <u>Igbo:</u> Replace sub with nwepu <u>Pidgin:</u> Replace sub with comot <u>Yoruba:</u> Replace sub with kuro	700-200-100
3.	Multiplication	<u>English:</u> mul 500 200 2 or mul 500*200*2 or mul 500,200,2 <u>Hausa:</u> Replace mul with sau <u>Igbo:</u> Replace mul with mubaa <u>Pidgin:</u> Replace mul with times <u>Yoruba:</u> Replace mul with dipupo	500*200*2
5.	Division	<u>English/Igbo*:</u> div 1000 2 10 or div 1000/2/10 or div 1000,2,10 <u>Hausa:</u> Replace div with rabawa <u>Pidgin:</u> Replace div with divide <u>Yoruba:</u> Replace div with dinku	1000/2/10
6.	Exit 9ja-Rebot	<u>English/Pidgin:</u> quit <u>Hausa:</u> daina <u>Igbo:</u> kwusi <u>Yoruba:</u> kuro	quit()

IV. RESULTS

Collated results of testing computation of bending distance of x:

```
arthur@arthur-pc: ~/python/interpreter/python
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/python$ python3 9ja-reboot.py

9ja-Reboot Version 0.01 on Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110]

English Loaded (Type: quit - to close)...

Commands:
*****
add = (Addition)
sub = (Subtraction)
mul = (Multiplication)
div = (Division)
print = (Print)
quit = (Quit)

You can also use any Python commands

9ja-Reboot>>> x=2
9ja-Reboot>>> f"Bending distance where x = 2 is 10x-x^2 = {10*x-x**x}"
'Bending distance where x = 2 is 10x-x^2 = 16'
9ja-Reboot>>> █
```

Figure 11: English and Pidgin

```
arthur@arthur-pc: ~/python/interpreter/python
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/python$ python3 9ja-reboot.py igbo

9ja-Reboot Udi 0.01 na Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110]

Igbo Ibu (Dee: kwusi - imechi)...

Iwu (Commands):
*****
ngbako = (Addition)
nwepu = (Subtraction)
mubaa = (Multiplication)
div = (Division)
lke = (Print)
kwusi = (Quit)

I nwekwara ike iji iwu Python o byla

9ja-Reboot>>> x=2
9ja-Reboot>>> ike f"Bending distance where x = 2 is 10x-x^2 = {10*x-x**x}"
Bending distance where x = 2 is 10x-x^2 = 16
9ja-Reboot>>> █
```

Figure 13: Igbo

```
arthur@arthur-pc: ~/python/interpreter/python
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/python$ python3 9ja-reboot.py hausa

9ja-Reboot Sigar 0.01 kan Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110]

Lodi Hausa (Shiga: daina - don rufewa)...

Umunni (Commands):
*****
tarawa = (Addition)
debewa = (Subtraction)
sau = (Multiplication)
rabawa = (Division)
buga = (Print)
daina = (Quit)

Hakanan zaka iya amfani da kowane umarnin Python
9ja-Reboot>>> buga f"Bending distance where x = 2 is 10x-x^2 = {10*x-x**x}"
Bending distance where x = 2 is 10x-x^2 = 16
9ja-Reboot>>> █
```

Figure 12: Hausa

```
arthur@arthur-pc: ~/python/interpreter/python
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/python$ python3 9ja-reboot.py yoruba

9ja-Reboot ti Ikede 0.01 lori Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110]

Ati kojopọ Yoruba (Te: kuro - lati pa)...

Ofin (Commands):
*****
ropo = (Addition)
kuro = (Subtraction)
dipupo = (Multiplication)
dinku = (Division)
jade = (Print)
kuro = (Quit)

0 tun le lo eyikeyi awon ofin Python

9ja-Reboot>>> x=2
9ja-Reboot>>> jade f"Bending distance where x = 2 is 10x-x^2 = {10*x-x**x}"
Bending distance where x = 2 is 10x-x^2 = 16
9ja-Reboot>>> █
```

Figure 14: Yoruba

Collated results of testing addition x:

```
arthur@arthur-pc: ~/python/interpreter/python
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/python$ python3 9ja-reboot.py

9ja-Reboot Version 0.01 on Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110]

English Loaded (Type: quit - to close)...

Commands:
*****
add = (Addition)
sub = (Subtraction)
mul = (Multiplication)
div = (Division)
print = (Print)
quit = (Quit)

You can also use any Python commands

9ja-Reboot>>> add 200 300 4000
4500
9ja-Reboot>>> █
```

Figure 15: English and Pidgin

```
arthur@arthur-pc: ~/python/interpreter/python
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/python$ python3 9ja-reboot.py hausa

9ja-Reboot Sigar 0.01 kan Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110]

Lodi Hausa (Shiga: daina - don rufewa)...

Umunni (Commands):
*****
tarawa = (Addition)
debewa = (Subtraction)
sau = (Multiplication)
rabawa = (Division)
buga = (Print)
daina = (Quit)

Hakanan zaka iya amfani da kowane umarnin Python

9ja-Reboot>>> tarawa 200 300 4000
4500
9ja-Reboot>>> █
```

Figure 16: Hausa

```

arthur@arthur-pc: ~/python/interpreter/python
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/python$ python3 9ja-reboot.py igbo

9ja-Reboot Udi 0.01 na Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110]

Igbo Ibu (Dee: kwusi - imechi)...

Iwu (Commands):
*****
ngbako = (Addition)
nwepu = (Subtraction)
mubaa = (Multiplication)
div = (Division)
ike = (Print)
kwusi = (Quit)

I nwekwara ike iji iwu Python o bula

9ja-Reboot>>> ngbako 200 300 4000
4500
9ja-Reboot>>> █

```

Figure 17: Igbo

```

arthur@arthur-pc: ~/python/interpreter/python
File Edit View Search Terminal Help
arthur@arthur-pc:~/python/interpreter/python$ python3 9ja-reboot.py yoruba

9ja-Reboot ti Ikede 0.01 lori Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110]

Ati kojopọ Yoruba (Te: kuro - lati pa)...

Ofin (Commands):
*****
ropo = (Addition)
kuro = (Subtraction)
dipupo = (Multiplication)
dinku = (Division)
jade = (Print)
kuro = (Quit)

O tun le lo eyikeyi awọn ofin Python

9ja-Reboot>>> ropo 200 300 4000
4500
9ja-Reboot>>> █

```

Figure 18: Yoruba

The researchers are willing to share the source codes of this project with fellow researchers who wish to run the remaining tests on their own. The twelve pages limitation imposed on each paper prevent us from posting additional test results.

V. CONCLUSION AND RECOMMENDATIONS

9ja-Reboot is a research prototype with amazing possibilities for Nigeria and our oil-based economy, 9ja-Reboot has the potentials of stimulating language-based research in engineering, science, technology, medicine and finance.

It is a known fact that knowledge impartation in a local dialect lead to rapid understanding and assimilation. Therefore, extending 9ja-Reboot to support additional commands will be of immense benefits to Nigerian youths who are interested in unlocking their programming skills.

We recommend that relevant stakeholders come up with plans that will fast-track the development of a full-blown programming language which is built on the principles of 9ja-Reboot. It is our hope as researchers that 9ja-Reboot will reboot our digital economy.

References

1. Statista (2021). Software market revenue worldwide 2016-2027, by segment. In <https://www.statista.com/forecasts/954176/global-software-revenue-by-segment>. Retrieved on September 9, 2022.
2. Branscomb, A. W. (2019). Computer software: protecting the crown jewels of the information economy. In Intellectual Property Rights in Science, Technology, and Economic

- Performance (pp. 47-60). Routledge. International Journal of Smart Education and Urban Society (IJSEUS), 11(2), 16-27.
3. TIOBE, (2022) "TIOBE Index for October 2022". In <https://www.tiobe.com/tiobe-index/> Retrieved on October 11, 2022
 4. Cordenne, B., (2021) "16 Examples of Global Companies Using Python in 2022". In <https://www.trio.dev/blog/companies-using-python/> Retrieved on October 11, 2022.
 5. Reilly, E. D. (2003). Milestones in computer science and information technology. Greenwood Publishing Group. pp. 156–157. ISBN 978-1-57356-521-9.
 6. Norvig, B., (2012) "(How to Write a (Lisp) Interpreter (in Python))". In <http://norvig.com/lispy.html/> Retrieved on October 20, 2022.
 7. Norvig, B., (2016) "(An ((Even Better) Lisp) Interpreter (in Python))". In <http://norvig.com/lispy.html/> Retrieved on October 20, 2022.
 8. Balaam, A., (2016) "Cell". In https://gitlab.com/cell_lang/cell/ Retrieved on October 20, 2022.
 9. De Andrade, M., (2017) "PascalToyCompiler". In <https://github.com/marcelogdeandrade/PascalToyCompiler/> Retrieved on October 20, 2022.
 10. Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2007). Compilers: principles, techniques, & tools. Pearson Education India.
 11. Beazley, D, M., (2022) "PLY - Python Lex-Yacc". In <https://github.com/dabeaz/ply/> Retrieved on October 20, 2022.
 12. Beazley, D, M., (2022) "SLY (Sly Lex Yacc)". In <https://github.com/dabeaz/sly/> Retrieved on October 20, 2022.
 13. Python, D, (2022) "Custom Python Interpreters". In <https://docs.python.org/3.9/library/custominterp.html/> Retrieved on October 21, 2022.
 14. GitHub, Python., (2022) "cpython/Lib/code.py". In <https://github.com/python/cpython/blob/3.9/Lib/code.py> / Retrieved on October 21, 2022.