Introduction

The objective of this project is to create an innovative programming language that could, hopefully, change the way its users view the otherwise rigid structure that most already existing programming languages follow.

In this case this is done by making language reactive.

This means that while in traditional languages variables do not update on their own, in this one they do.

For example, if I were to write the following piece of pseudo code:

‘’’
var A = 5;
var B = A + 2;
var A = 2;
print(B);
‘’’

Then a traditional language would print 7 in the end while this one will print 4 (because A has been updated and 2 + 2 = 4).

This, while a relatively small change, fundamentally changes the way the language functions and the way the programmer must think to achieve the desired results.
On the other hand, if done correctly, this has the potential to save a large amount of development time resources by forcing the user to create a minimalistic codebase.

The syntax

For the sake of simplicity as this technology is still in its infancy, the syntax will be light.

Every variable declaration is done strictly, meaning the type of the variable must be written in front of its name, followed by “ = expression” where expression is the code that defines the variable.

Variables must be declared only once, changes to the variable do not need the type.

Variable names are case sensitive and can only contain letters.

Currently we only work with integers (int), floating point numbers (float), strings (str) and boolean (bool) values.

Statements such as if, elif, else and so on, as well as loops are currently not functional.
For those, stay tuned for a future update.

String interpolation is automatic and does not require any extra characters.
Only the “ character is used to delimit string values. The \” key can be used to print it within a string.

Printing can be done by calling “Print(youVariable)”

‘’’
Str A = “abc “ 4 “ cba”
Print(A)
‘’’

Will print “abc 4 cba”

Scoping

Every variable is in a global scope, as mitigated scopes do not make sense.
To explain this let’s take the following code as an example:

‘’’
Int A = 2;
if(true){
	Int B = A + 5;
}

Int C = 5;

A = C;
‘’’

In this instance, in most other languages, C is not declared for B to interact with and B is out of scope for C, however, for us B = C + 5 .

Variable Call loops

One of the main challenges I found while playing around with this concept is the dangers of accidental infinite loop around within variables.
If a variable ends up calling itself, or a variable that down the line will call it, that will create an infinite loop.
These loops are sometimes useful however, as such they are currently legal and do not throw any errors.

Compiler VS Interpreter

First of all, the difference between the two is that while a compiler translates the code directly to a computer language, an interpreter uses a third party language as an inbetween

to do the same.
This makes compilers more efficient, it also makes them infinitely more complicated.
Thereby making the creation of an interpreted the correct choice for this project.

Why python?

The project is created in python.
Python's flexibility makes the development of this interpreter much easier than other languages.
The downsides of Python are its slowness and its inability to create exe files, therefore forcing the user to have python installed to run the project.
However, as we are at the phase where we use an interpreter instead of a compiler, speed is not yet of utmost importance.
And the ability to run the project directly from the source code makes it much easier to present it in a flexible manner.

Competition

While a complete reactive programming language do not exists, there are a few similar projects in circulation:

- Red Is an open source programming language with reactive elements.
- Javascript has a multitude of reactive libraries for automated UI updates.
    - RxJs, LemonadeJs, SolidJs, ArrowJs ..

The market

As the small list of projects that attempted to create something similar suggests, there is a definitive need for this form of programming.
However, the market is limited by the number of developers willing to learn a completely different form of programming. Breaking the base concepts of a programming language means the developers need to retrain their own way of thinking to use the new language, which is hard to do in a fast paced and large scale working environment.

Conclusion

In conclusion, This is very much an ambitious project, one that has the potential to change a programmers landscape if developed to a high enough degree.
However, this technology is still, very much in its infancy.