Custom Language Creation: Python-Powered Language Development

By Lillian Jackson, Tori Olender, & Prasiddhi Gyawali

Design and Development Process

When coming up with our language, we began with the idea of creating a syntax that used characters that did not require use of the shift key without sacrificing readability. We started with making basic examples of the language so we could create our grammar with them. Then, when creating the translator we were able to base our parser off of the grammar. We started off with simple, with string and print statements. Then, we moved on to parsing numbers and equations. After that, we did booleans and boolean operations. Finally, we programed the if statements and while loops.

Challenges

There was some struggle when it came to parsing the integers and the print statements initially, as we were learning how write in Regex. For example, in the print statement, we had to figure out how to get the translator to split on the quotation marks for strings as well the plus sign for concatenation. It was also difficult figuring out how to loop the while statement, as our parser reads the input program line-by-line. Later in the project, we found it was also challenging to determine if the conditions where still true when looping the while loop.

Code examples

Accept:

```
str h = ‘hello’.
num x =  -1.
num y = 400.
num z = y.
num a = z - y.
num b = z + y.
num c = z * y.
num c = z / y.
```

```
um d = z mod y.
bool xy = true.
bool zd = y and z.
bool as = z or d.
bool foo = not a.
bool g = x greater y.
bool h = x less y.
bool k = x equal y.
```

```
str age = input[‘how old are you?’].                    // how old are you?,
and stores input in “age” str //
num age = input[‘how old are you?’].                    // can also be used
to get numbers for input //
print[‘x ’, z, true, 40, ‘ number\’\s’].          // x (z’s value)true40
number’s<- //
bool k = x greater y or x equal y
If x = True fi [
        c = true.
]

while x equal true elihw [
        num y = y + 1.
]
```

Grammar Syntax

```
<whileBrnch> ::= while <boolStmt> elihw[<stmt>]
<ifBrnch> ::= if <boolStmt> fi [ <stmt>]  <elseif> <else> [<stmt>] |
              if <boolStmt> fi [ <stmt>] <elseif> <else> [<stmt>]
<elseif> ::= elseif <boolStmt> fi [<stmt>] | | <elseif><elseif>
                                             ↑(nothing is also an option)
<boolStmt> ::= <bool> | <int> less <int> | <int> greater <int> | <int> equal <int> | <boolStmt> and <boolStmt> |
              <boolStmt> or <boolStmt>
<stmt> ::= <op>. | <var_assign> | <printStmt> | <while_brnch> | <if_brnch> |  <stmt> <stmt>
<printStmt> ::= print['<printContent>'].
<printContent> ::= <str> | <var> | <str> + <printContent> | <var> + <printContent>
<op> ::= <op1> + <op> | <op1> - <op> | <op1>
<op1> ::= <op1> * <op2> | <op1> / <op2> | <op1> % <op2> | <op2>
<op2> ::= (<op>) | <int> | -<int>
<var_assign> ::= <var_dec> = <val>.
<var_dec> ::= <typeName> <var>
<type> ::= str | int | bool
<val> ::= <int> | -<int>| <var> | <bool> | <str>
<str> ::= '<char>' | '<char><var>'
<var> ::= <char> | <char><var>
<char> ::= a | b | c | ... | x | y | z | A | B | C | … | X | Y | Z | 1 | 2 | ... | 0 | , | . | … | ~
<int> ::= <digit> | <digit><int>
<digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
<bool> ::= true | false
```

Tutorial

Every line ends with a period, much like a sentence. We used declarative types where booleans are represented by *bool*, integers are represented by *num*, and strings are represented by *str*. We used [] for parenthesis in print statements.

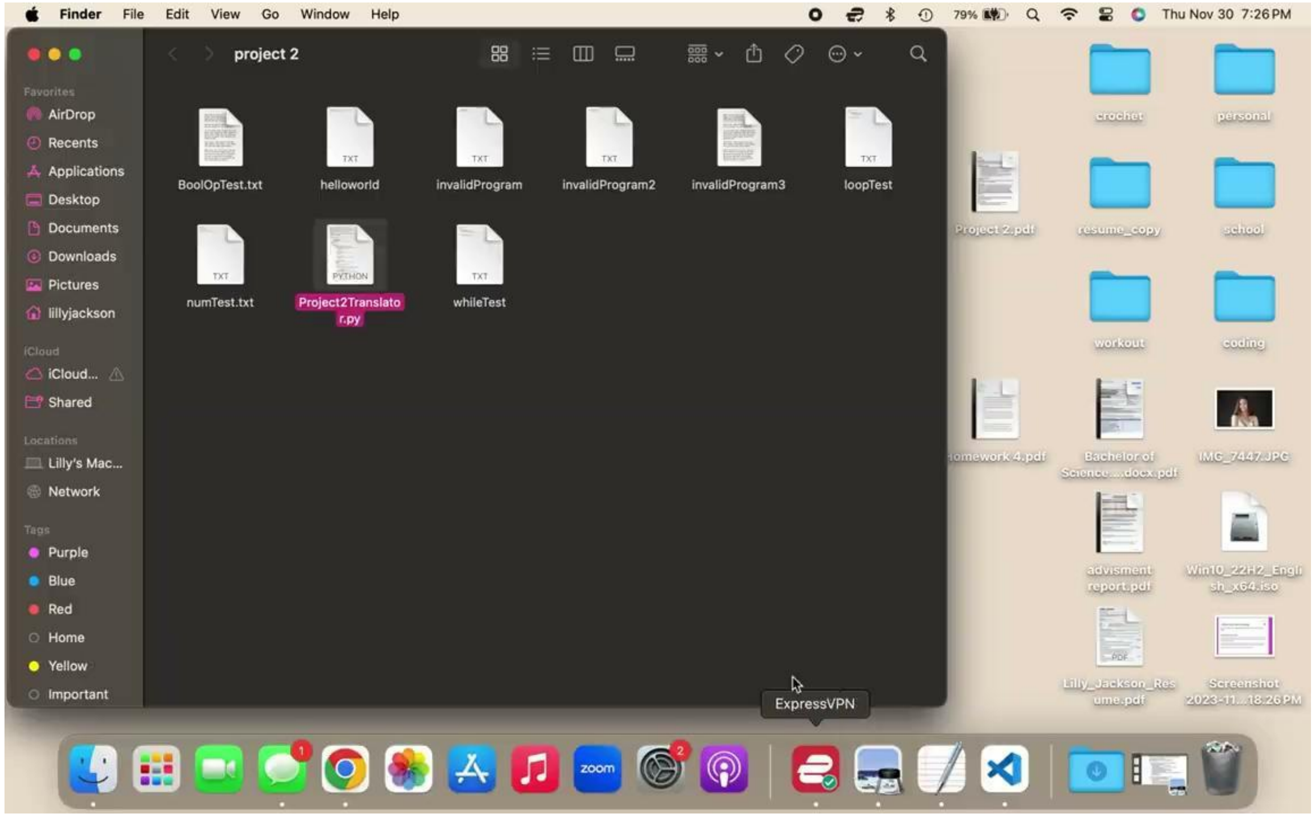
To run the program, execute Project2Translator.py through the terminal or in an IDE. Input the name of the text file that is written in our programming language. Refer to the code examples and the grammar tree for the language syntax.

Continuing Our Language

If we were to continue our language, we would change the structure of the parser so that it is more efficient. For example, the parser read in the text files line-by-line, which led to multiple complications, so finding an improved way to read in the files would benefit the language.

We would also extend the functionality of the language, such as adding functionality for classes or functions, adding more variable types such as floats, or adding structures such as arrays.

Video


Finder
File
Edit
View
Go
Window
Help
Thu Nov 30 7:26 PM
project 2
Favorites
AirDrop
Recents
Applications
Desktop
Documents
Downloads
Pictures
lillyjackson
iCloud
iCloud...
Shared
Locations
Lilly's Mac...
Network
Tags
Purple
Blue
Red
Home
Yellow
Important
BoolOpTest.txt
helloworld
invalidProgram
invalidProgram2
invalidProgram3
loopTest
numTest.txt
Project2Translator.py
whileTest
Project 2.pdf
crochet
personal
resume_copy
school
workout
coding
homework 4.pdf
Bachelor of Science....docx.pdf
IMG_7447.JPG
advisment report.pdf
Win10_22H2_English_x64.iso
Lilly_Jackson_Resume.pdf
Screenshot 2023-11...18.26 PM
ExpressVPN