

**UNIVERSIDAD DE PUERTO RICO**

**RECINTO UNIVERSITARIO DE MAYAGÜEZ**

***Final Report:***

***Poinsly***

**Edmanuel Ayala**

**Jorge Huertas**

**14 de mayo de 2021**

**COMP 4036-070**

**Prof. Alcibidades Bustillo**

***Introduction:***

Poinsly is a programming language made in Windows that helps to solve linear or quadratic equations similarly to a graphing calculator. It utilizes the SLY premade parser, lexer and interpreter. How it works is that you enter either a linear or quadratic equation like this 1x^2+-5x+4 or 2x+24 and it would give you the result of said equation and a corresponding graph. Our motivation with Poinsly is to gain a better understanding on how to make a programming language.

***Language tutorial :***

Python 3.6 or higher is required to run the programming language. We used Windows, Anaconda and VScode to build and test the language through a virtual environment, so you may use those. To create the virtual environment in VScode you have to type “python -m venv C:\Directories\venv”, in later you install the respective libraries using “pip install Library” Run the program through a terminal and enter either a linear or quadratic equation for it to solve and generate a graph depending on the input. The graph should appear in the taskbar after entering the equation. Negatives are used like this -2x^2+-1x+-13. If you attempt to enter a power higher than 2 like 3x^6, it will return an error. If you attempt a variable other than x you would be met with an error. There are 2 Shift/Reduce and 1 Reduce/Reduce warnings when running the language and 1 complex number warning when generating the graph, however they do not appear to interfere with the main function of the language.

***Language reference manual :***

*(Best used when viewing alongside code)*

From the top we import from SLY the Lexer and Parser. Cmath is imported because it is used for the quadratic formulas. And lastly we import both numpy as np and matplotlib.pyplot as plt to generate the graphs for both linear and quadratic equations.

*Lexer:*

The Language uses 4 tokens in the Lexer: NUMBER, ADD, EXP, and TOKEN\_X.

NUMBER uses the regular expression ‘-?\d+’ to take both positive and negative numbers.

ADD uses ‘\+’ to take in Plus signs +.

EXP uses ‘\^’ to take in Circumflexes ^.

TOKEN\_X uses ‘[xX]’ to find x or X

@\_(r’\d+’) is used to converts NUMBER into a numeric value

@\_(r’\n+’) is used for newlines

and lastly in the lexer is def error to identify illegal characters in the language.

*Parser:*

the debugfile is used to print the grammar rules into a txt file named ‘Pparser.txt’

(note: a txt file of the same name must already exist in the same directory in order to print them out)

To resolve ambiguity, especially in expression grammars, SLY allows individual tokens to be assigned a precedence level and associativity.

Precedence in this language goes like this:

Level 1 left associative precedence - TOKEN\_X , ADD

Level 2 left associative precedence - EXP

To clarify the next few rules we’ll be going slightly out of order

@\_(’number TOKEN\_X’) is used to obtain the a linear term such as 1x

@\_(’linear\_term’) is used to return the coefficient of the linear term found in @\_(’number TOKEN\_X’)

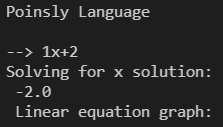
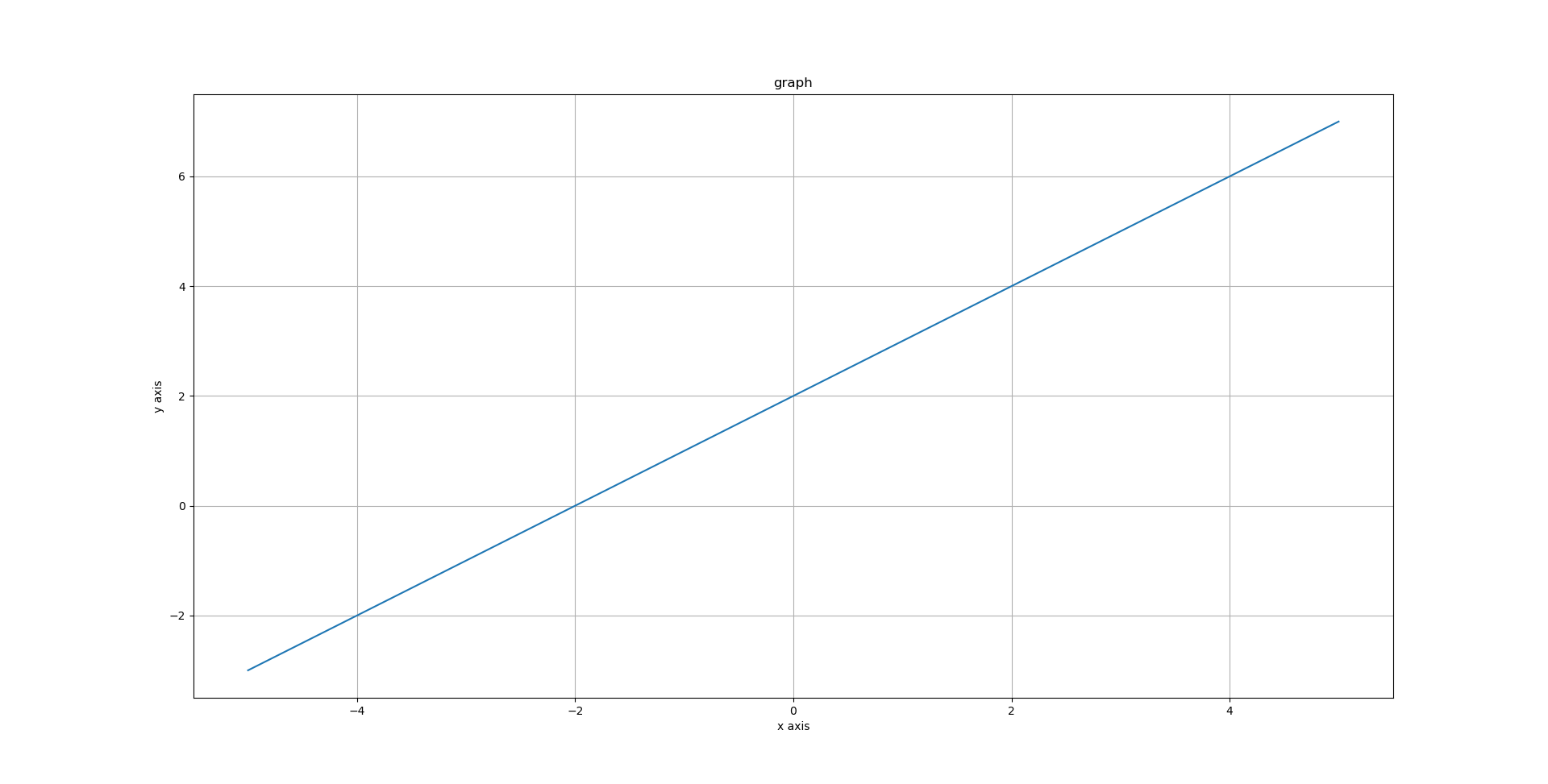
The same applies to the @\_('number TOKEN\_X EXP number') and @\_('quadratic\_term')

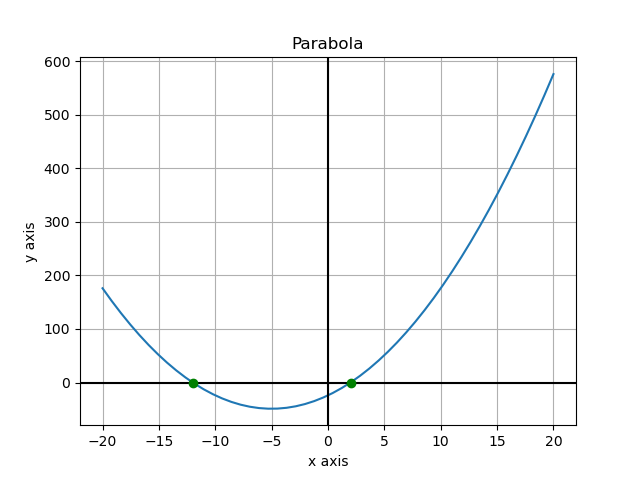
except in @\_('number TOKEN\_X EXP number') is used to obtain the quadratic term such as 1x^2 (note: you will receive an error if you use anything but the number 2 for the second number)

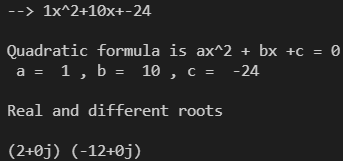
@\_('linear\_term ADD number') is used when a linear equation is introduced such as 1x+2. it then solves for x and prints out a graph in a separate window thanks to numpy and pyplot.

Example:

@\_('quadratic\_term ADD linear\_term ADD number') is used when a quadratic equation is introduced such as 1x^2+2x+3. It then solves the discriminant and prints the solution depending if the discriminant is higher or equal to 0 then prints out a graph with numpy and pyplot in a separate window. If the discriminant is less than 0 it will print out “no real solutions”.

Examples:





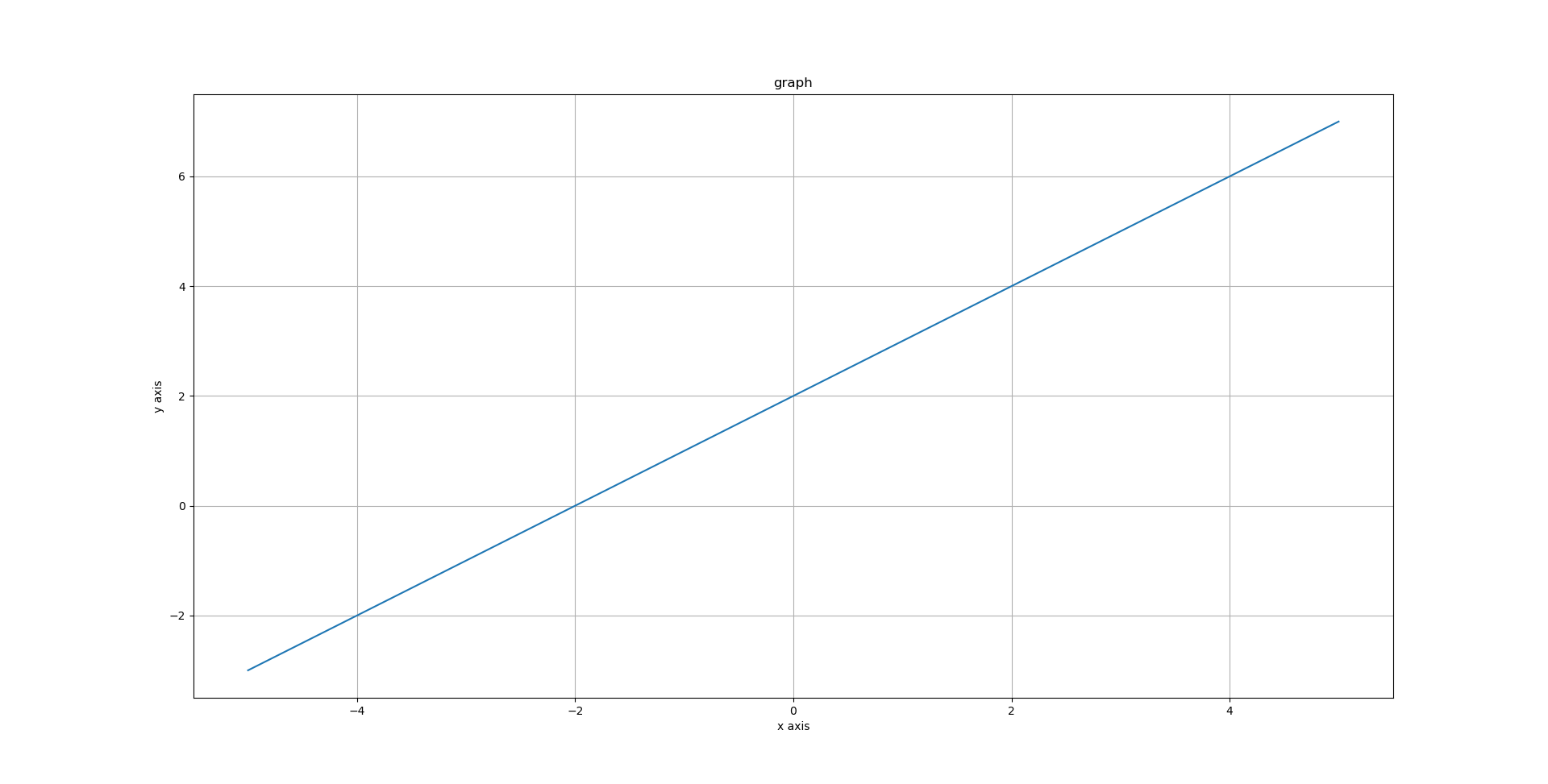
And lastly in the parser there's @\_(‘NUMBER’) that allows the use of the NUMBER token with just ‘number’.

*Interpreter:*

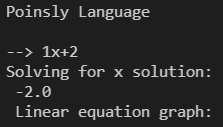
The interpreter calls both lexer and parser and checks with ‘while True:’ that as long as there are no End Of File errors(EOFError) you can enter text that will be separated into tokens so that the parser may parse the tokens into the result variable then it simply prints result.

***Language development:***

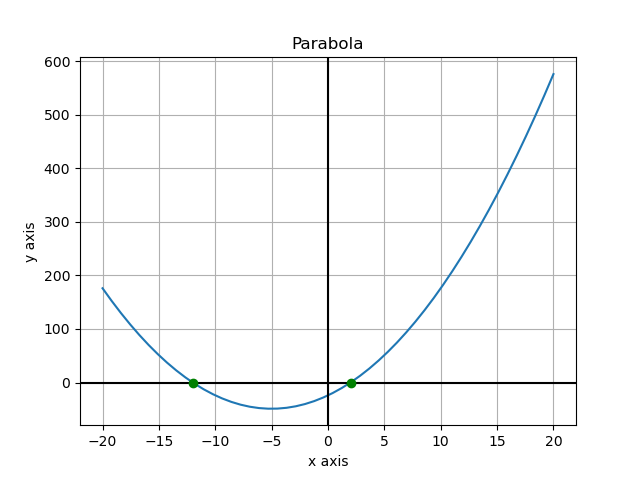
The translator architecture that we utilize is the one provided by SLY since it has a premade Lexer, Parser and Interpreter. We use cmath to solve the square roots for the quadratic equations, then we use numpy and matplotlib to generate the graph and use the result of the quadratic equation to plot the 2 or 1 point(s). The SDE we used to make this language was Python since it was required in order to use SLY, Python at the very least is simple enough for us to understand. The methodology we used was a combination of unit testing, system testing and acceptance testing. We ensured that every addition be it module, formula or the generation of graphs at the very least worked together correctly.

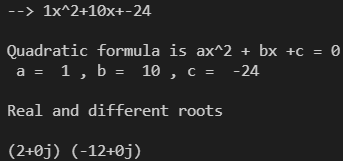
*Examples*

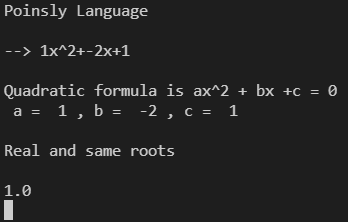
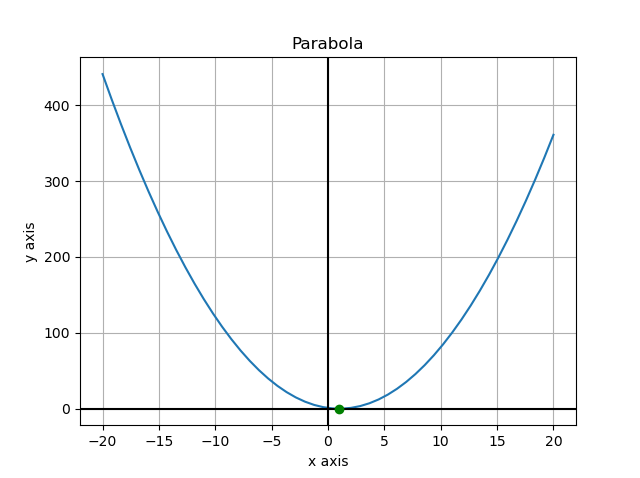
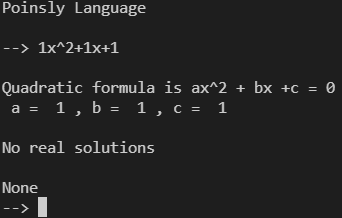
Linear equation:



Quadratic equations:

1. Roots are real and different



1. Roots are real and the same
2. No real solution

***Conclusions***

Therefore, poinsly is a great programming language to solve linear and quadratic equations. It is very simple to use and to understand. Using one of the best beginner friendly programming language(Python) we were able to fully utilize SLY. With the help of SLY the program was much easier to manage, we were able to move and play with the lexer and parser. This was key to build poinsly. However, building the equation was a bit of a problem, using the tokens to gather positive and negative integers made it hard to apply. The program accepting two types of equations was another challenge. Implementing the ways of solving both linear and quadratic is something that a human can do but to give the instruction to a computer is complex. With both equations up and running, we added the python library matplotlib to help us graph the solution. With this library our programming language Poinsly was created. A programming language that can graph the solution of linear and quadratic equations.