DOMAIN SPECIFIC LANGUAGE FOR GRAPHICAL DATA VISUALISATION

Gheorghe MORARI[1], Nichita PASECINIC[1], Alex CLEFOS[1], Roman BOTEZAT[1], Ana ȘARAPOVA[1]*

[1]Technical University of Moldova, Faculty of Computers, Informatics and Microelectronics, Department of Software Engineering and Automatics, FAF-192, Chișinău, Republic of Moldova

*Corresponding author: Șarapova Ana, sarapova.ana@isa.utm.md

Abstract. *This article is about a Domain Specific Language (DSL) called Python Graphing Language (PGL) that has been developed for the graphing/charting domain as a free, lightweight, and simple alternative to common data visualization software such as MATLAB. This article describes the syntax, the grammar, the implementation, the future plans for this DSL, and the comparison with analogous data visualization software.*

Keywords: *Data Visualisation, Grammar, Matlab, Syntax, Plotting, Python, Python Graphical Language*

Introduction

Graphing, Plotting, and Charting are three words that represent the same idea - graphical representation of data. The domain of charting is a subdomain of the scientific one. Historically charting was done firstly with pen and paper [1], later with the popularisation of computers, there appeared graphing calculators [2], and with the internet revolution, online graphing software and purpose-made software started getting widely used [3].

PGL was made to be easy to use like online graphing software while providing advanced functionality for data manipulation and visualization. These functionalities, unfortunately, come as a compromise between features and simplicity, where PGL is mostly targeting simplicity, and cannot touch the feature set of established scientific grade software.

Solution

PGL is an attempt to solve the following problems that established charting software have, and it is:

- Free-to-use, unlike *MATLAB* [4].
- Novice-friendly, due to simple syntax and grammar.
- Platform-independent, meaning that it can work on any device with a popular operating system [5 - 6].
- Small in size, in comparison to alternatives such as *MATLAB*, which may take up to 29GB for all its licensed product [7].

Nevertheless, this DSL enhances *Python* plotting experience with custom plotting related keywords, *MATLAB*-like syntax, and general simplifications. PGL also provides keywords for quick data manipulation, input, and output, such as interpolation, extrapolation, dimensionality reduction, noise reduction, file input, *CSV* file input, and others.

Syntax

The current syntax of PGL has two instructions: **subplot** and **plot**.

These two instructions contain multiple sub-instructions like **type**, **style**, **legend**, and others.

The sub-instructions themselves have sub-instructions like **simple**, **bar**, **theme**, **color,** and others.

There is an example of code where we can see instructions within instructions:

```
# comment
subplot subplot0 = {
    type = simple,
    x = [0, 5, 10, 15],
    y = [1, 2, 5, 8],
    style = {
        color: red,
        label: "subplot label",
        line_style: "--",
        marker: "*",
        line_width: 1
    }
}
```

Grammar

Notation	Description
<foo>	Means that element is a nonterminal
foo	Means that *foo* is a terminal element
[foo]	Case with *foo* as an optional element
foo+	Show one or more occurrences of *foo*
{}	Used for grouping elements
|	Separates possible alternatives
&	Combine states
#	Shows the comment line
<start>	The starting state of grammar

G = {V_T, V_N, P, S}

S = {<start>}

V_T= {EOF, \n, subplot, plot, colors, letter, digit, logicalOp, style_params, type, save, subplots, :, =,{,},[,],_}

colors ϵ {red, green, blue, yellow, orange, violet, black, white, mint, grey, navy, pink}

letter ϵ { a, …,x, y, z, A, B, C, …, X, Y, Z}

digitϵ{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

logicalOp ϵ { True, False}

style_params ϵ{style, theme, color, line_style, marker, line_width, label, legend, title, loc, shadow, config, tight_layout, grid}

typeϵ {simple, bar, pie, stack}

V_N={start, instruction, plotting_params, color_enum, string, number, types, x_axis, y_axis, plot_styles, legend, config, subplots, type, array}

Instruction ϵ{subplot_assign, plot_assign, string, subplot_params, plot_params, subplot_param, plot_param, types, x_axis, y_axis, plot_styles, legend, config, subplots, type, array}

plotting_params ϵ {subplot_params, plot_params, subplot_param, plot_param}

P={<start> → <instruction>+
<instruction> → <subplot_assign> | <plot_assign> | **\n** | **EOF**
<subplot_assign> → **subplot** <string> **=** <subplot_params>
<subplot_assign> → **plot** <string> **=** <plot_params>
<subplot_params> → **{** <subplot_param> [**,** <subplot_param>]+ **}**
<subplot_param> → <types> | <x_axis> | <y_axis> | <plot_styles>
<plot_params> → **{** <plot_param> [**,** <plot_param>]+ **}**
<plot_param> → <types> | <x_axis> | <y_axis> | <plot_styles>
| <legend> | <config> | <subplots>
<types> → **type =** <type>
<type> → **simple** | **bar** | **pie** | **stack**
<x_axis> → **x =** <array>
<y_axis> → **y =** <array>
<plot_styles> → **style = {** <plot_style> [**,** <plot_style>]+ **}**
<plot_style> → <theme> | <color> | <line_style> | <marker> | <line_width> | <label>
<theme> → **theme :** <string>
<color> → **color :** <color_enum>
<line_style> → **line_style :** <string>
<marker> → **marker :** <string>
<line_width> → **line_width :** <number>
<label> → **label :** <string>
<legend> → **legend = {** <legend_param> [**,** <legend_param>]+ **}**
<legend_param> →<x_label> | <y_label> | <title> | <loc> | <shadow>
<x_label> → **x :** <string>
<y_label> → **y :** <string>
<title> → **title :** <string>
<loc> → **loc :** <string>
<shadow> → **shadow :** <bool>
<config> → **config = {** <config_param> [**,** <config_param>]+ **}**
<config_param> → <grid> | <tight_layout> | <save>
<tight_layout> → **tight_layout :** <bool>
<grid> → **grid :** <bool>
<save> → **save :** <string>
<subplots> → **subplots = {** <string> [**,** <string>]+ **}**
<color_enum> → **red** | **green** | **blue** | **yellow** | **orange** | **violet** | **black**
| **white** | **mint** | **grey** | **navy** | **pink**
<array> → **[** <number> [**,** <number>]+ **]**
<number> → <digit>+
<digit> → **0** | **1** | … | **9**
<bool> → **True** | **False**
<string> → <char>+
<char> → **a** | **b** | . | **z** | **A** | **B** | . | **Z** | **0** | **1** | … | **9** | **_** | }

Implementation

PGL in its current form is implemented in python using the *Lark* library for parsing and grammar definition. Most of the features implemented are using *Matplotlib*, *Pandas,* and *NumPy.*

Matplotlib is a library that displays the plots, and it is the heart of the implementation. The rest of the libraries are used for convenience and performance.

PGL is still in its infancy where most of the core features still need implementation. We are currently planning to implement multiple plot figures, file input and output, and array management.

It is planned to introduce other trivial features like array data management, and data science tools like interpolation, extrapolation, random value generation, and others.

The current state of our implementation: https://github.com/nikitaal/dsl

Conclusion

PGL is a viable alternative to established software by providing an enhanced plotting experience via user-friendly syntax and simplified data and plotting manipulation processes while being open-source and free to use.

References:

1. *History of data visualisation* [online, accessed 28 Feb 2021 14:30], available on: https://en.wikipedia.org/wiki/Data_visualization#History
2. *History of Graphing Calculators*, Vernon Morris [online, accessed 1 Mar 2021 16:17], available on: https://www.meta-calculator.com/blog/history-of-graphing-calculators-and-tools/ [online, accessed 1 Mar 2021 16:02]
3. *The 10 Best Graphing Calculators (Physical and Online)*, Carrie Cabral [online, accessed 1 Mar 2021 16:39], available on: https://blog.prepscholar.com/graphing-calculator
4. *MathWorks Pricing & Licensing* [online, accessed 28 Feb 2021 14:34], available on: https://www.mathworks.com/pricing-licensing.html
5. *Supported platforms and architectures* [online, accessed 28 Feb 2021 14:43] https://pythondev.readthedocs.io/platforms.html
6. *Operating System Market Share Worldwide*, GlobalStats, [online, accessed 28 Feb 2021 15:41], available on: https://gs.statcounter.com/os-market-share
7. *MATLAB Licensing, Software Licensing* [online, accessed 28 Feb 2021 15:48], available on: https://it.cornell.edu/software-licensing/matlab-licensing