# SOEN 6471 - Advanced Software Architectures
# Team D - MATPLOTLIB

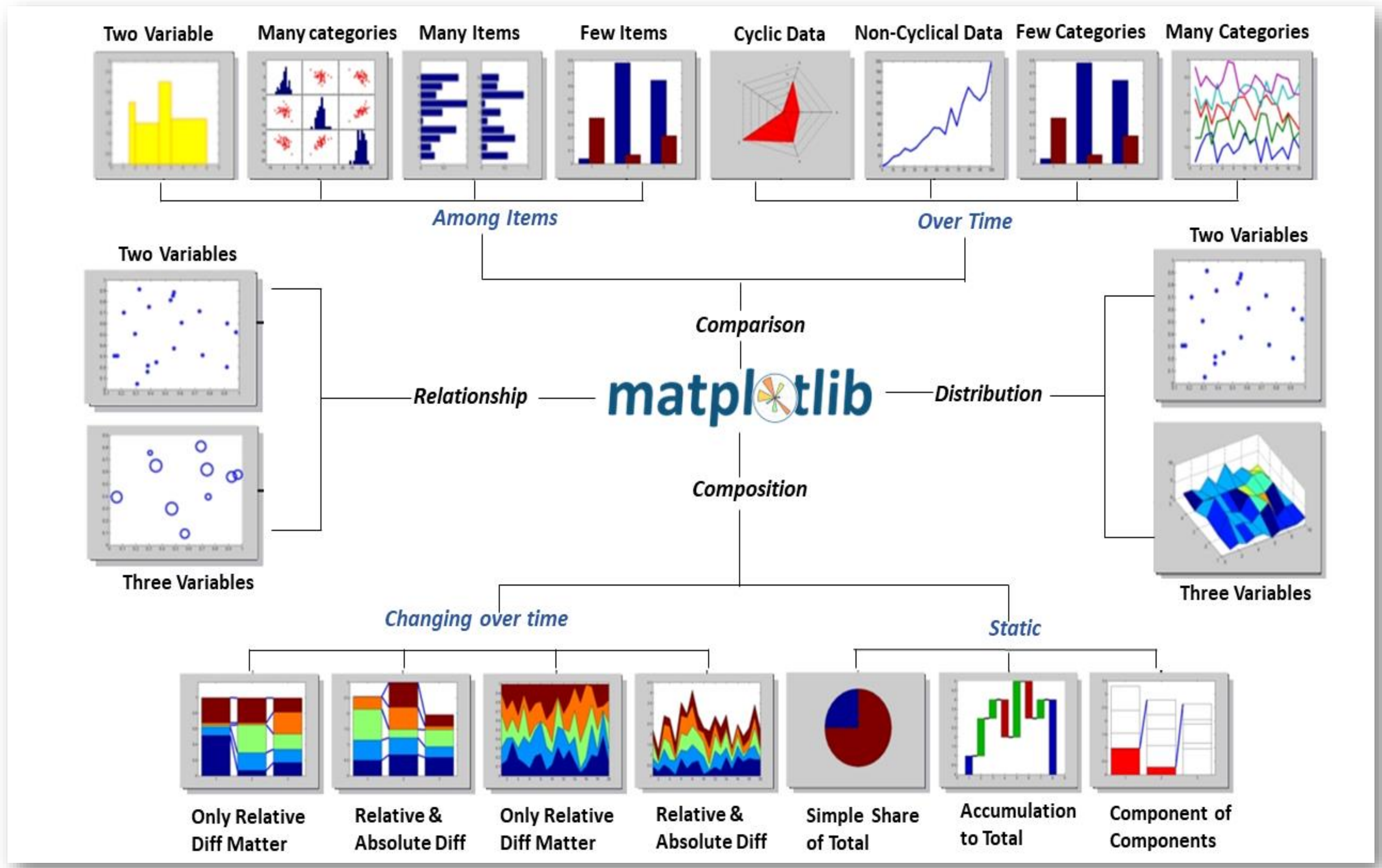
Jigar Maheshbhai Borad | Bhavye Budhiraja | Rancy Chadha | Payal Raj Chaudhary | Raviraj Bhaveshbhai Savaliya

## Introduction

Matplotlib is a powerful data visualization library widely used in Python programming. It offers a comprehensive range of tools and functions for creating visually appealing and informative plots, charts, and figures. Users can customize every aspect of the plot, including axes, labels, colors, and styles, providing flexibility in visualizations. This level of customization allows for precise control over the visual representation of data. Integration with NumPy, another popular Python library, enhances Matplotlib's capabilities for data analysis and scientific computing. This integration enables seamless integration and manipulation of numerical data for effective visualizations.

With its flexible and intuitive interface, Matplotlib provides an extensive collection of plot types, such as line plots, scatter plots, bar charts, histograms, and more. This variety of plot types allows users to select the most appropriate visualization for their data Matplotlib serves as an indispensable tool for a wide range of applications, including scientific research, data analysis, business intelligence, and educational purposes. Its versatility makes it suitable for various domains where data visualization is essential.

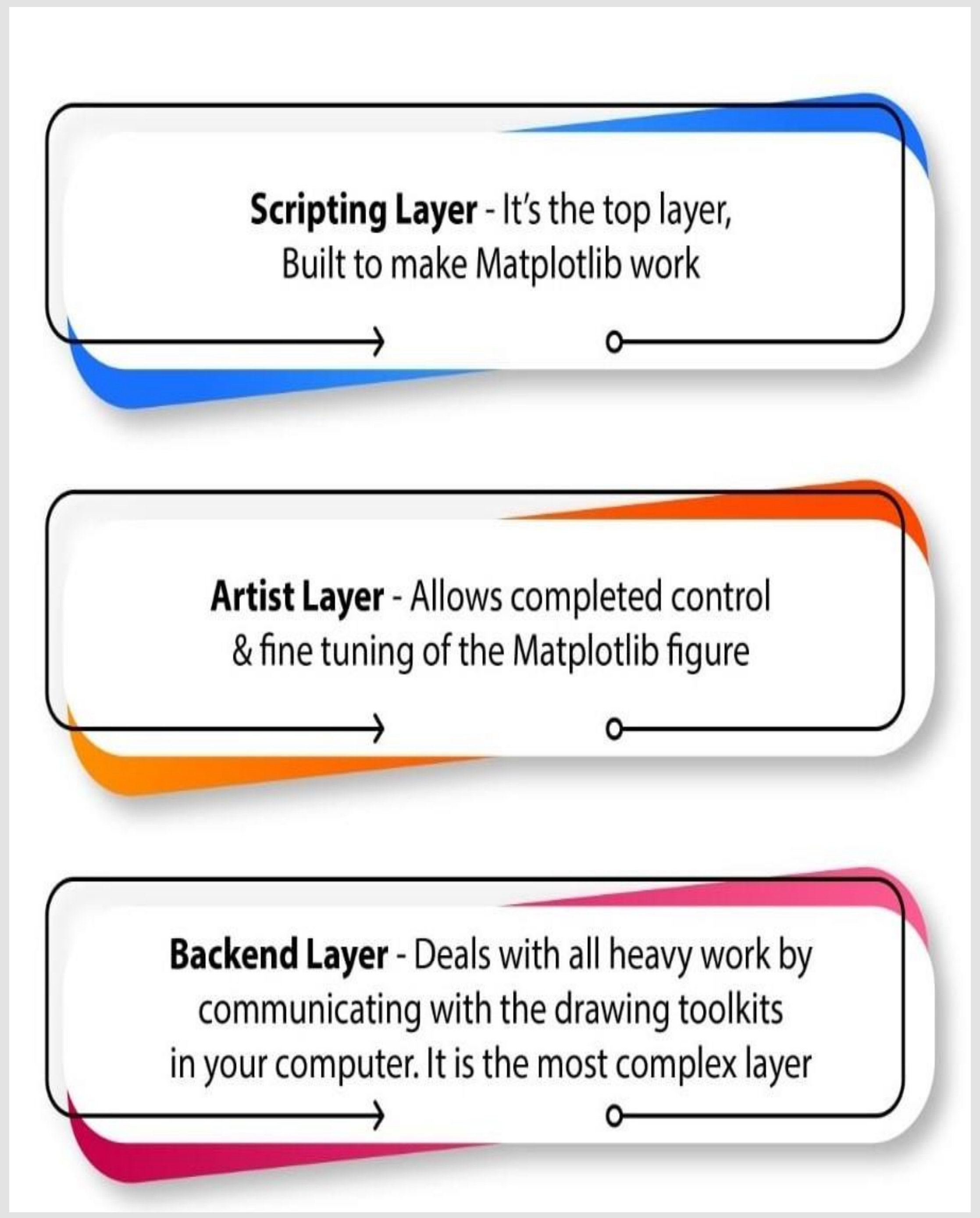
## MatplotLib Capabilities



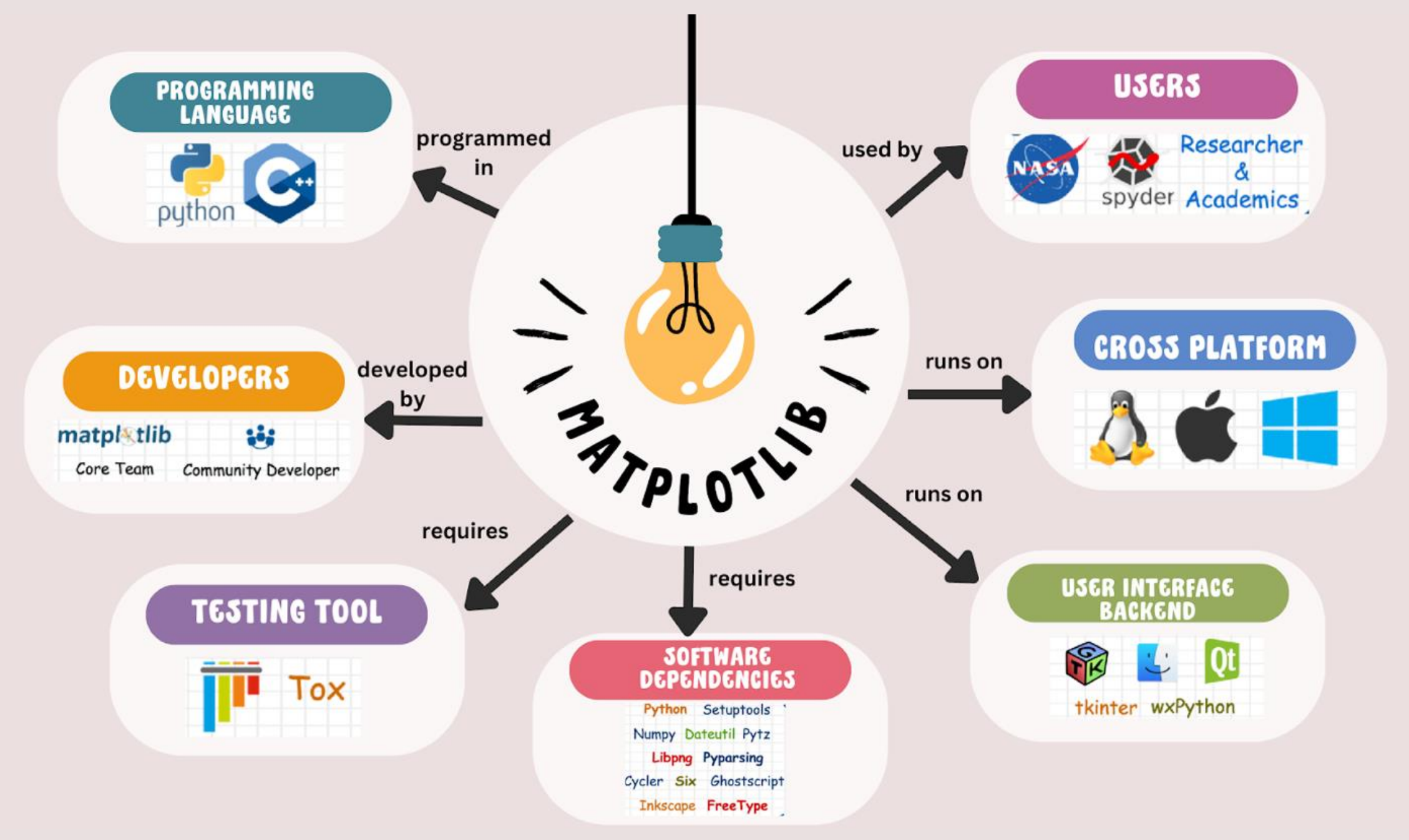
## Learnings

- Discovered about Matplotlib's underlying architecture and design principles such as its modular design, which allows for flexibility and customization
- Comprehended Matplotlib's context within the broader Python ecosystem, particularly its integration with other scientific computing libraries like NumPy and Pandas
- Investigate the software metrics used to measure and evaluate the quality attributes of Matplotlib's architecture
- Learnt about various quality attributes listed and methods to evaluate them to classify which satisfy architecture design and those that do not
- Acquired knowledge on library's structural, behavioral, and deployment aspects, creating architectural views to capture different concerns and perspectives.
- Gained insights on Matplotlib's architectural patterns and styles used in this framework among available patterns such as the Model-View-Controller (MVC) , Observer or composite pattern for managing plot hierarchies

## MatplotLib Architecture


Scripting Layer - It's the top layer, Built to make Matplotlib work
Artist Layer - Allows completed control & fine tuning of the Matplotlib figure
Backend Layer - Deals with all heavy work by communicating with the drawing toolkits in your computer. It is the most complex layer

## Context of use



## Principles

- **Single Responsibility Principle:** Matplotlib applies SRP to improve code maintainability and avoid unexpected side effects.
- **Open-Closed Principle:** Matplotlib follows OCP, enabling easier maintenance through extension rather than modification of existing classes.
- **Liskov Substitution Principle:** It utilizes LSP to ensure seamless use of derived classes without modifying closed classes.
- **Interface Segregation Principle:** Matplotlib favors smaller, user-specific interfaces, promoting composition and decoupling.

## Contribution

Right from project selection till completion of all deliverables we worked as team by ensure that our tasks were equally distributed amongst all of us. We also ensured that for related tasks, individual team members collaborated amongst themselves to strengthen topic understanding and avoid redundant work. Both in-person and zoom meetings were leveraged.

Together, our efforts resulted in a well-structured and user-friendly Matplotlib documentation, consisting of all the 7 problems, providing users with the necessary information to effectively utilize the library's features and functionalities.

## Quality Attributes

| Satisfy | Not satisfy | |
|---|---|---|
| Accessibility | Affectability | Privacy |
| Customizability | Simplicity | Usability |
| Modularity | Ethicality | |
| Dependability | Legality | |
| Maintainability | Resilience | |
| Performability | Safety | |
| Portability | Scalability | |

## Limitations

- **Challenging Learning Curve:** Understanding plot types, customization, and syntax can be difficult.
- **Limited Default Aesthetics:** Matplotlib's default styles lack visual appeal or suitability for publication-quality plots.
- **Verbosity:** Creating intricate plots often requires writing extensive code.
- **Lack of Interactivity:** It lacks advanced interactive features like tooltips, zooming, or dynamic plot updates.
- **Performance:** Matplotlib may not be optimal for large datasets or real-time visualizations.

## Conclusion

In conclusion, our Matplotlib project demonstrated the power of this versatile data visualization library. We leveraged its advanced customization capabilities to create visually appealing plots. Collaboration and adherence to software design principles played a crucial role in achieving our goals. The integration with other data processing libraries enhanced functionality. This project has deepened our understanding of software architecture and improved our data visualization skills for future projects.

We would like to express our sincere gratitude to the Professor Pankaj Kamthan and TAs for their guidance and support throughout the project on Matplotlib.