



Getting Started with Plotly

Last Updated : 31 Aug, 2024

The **Plotly Python** library is an interactive open-source graphing library. It is a very helpful tool for data visualization and understanding the data simply and easily. Plotly graph objects are a high-level interface to plotly which are easy to use. It can plot various types of graphs and charts like scatter plots, line charts, bar charts, box plots, histograms, pie charts, etc.

In this article, we will explore the basics of plotly, how to get started with it, and it's key features that make it preferred choice for creating interactive visualization.

Table of Content

- [What is Plotly?](#)
- [History and Evolution of Plotly](#)
- [Why Use Plotly?](#)
- [Installation and Setup with Plotly](#)
- [Plotly's Architecture in Python](#)
- [Creating the Basic Plots with Plotly](#)
- [Data Structures and Plotly](#)
- [Best Practices for Using Plotly](#)

What is Plotly?

Plotly is a Python library built on top of the Plotly JavaScript library (plotly.js). It supports over 40 unique chart types, including statistical, financial, geographic, scientific, and 3-dimensional visualizations. Plotly is known for its ability to create beautiful, interactive web-based visualizations that can be displayed in Jupyter notebooks, saved as standalone HTML files, or integrated into web applications using Dash.

History and Evolution of Plotly

Plotly was initially released in 2013 and has evolved significantly since then. It was designed to be an open-source library for creating interactive, publication-quality graphs. Over the years, Plotly has added numerous features and chart types, expanded its integrations with other libraries, and improved its performance and ease of use.

Understanding the evolution of Plotly helps appreciate its current capabilities and its role in the data visualization ecosystem.

Why Use Plotly?

Plotly offers several advantages that make it an attractive choice for data visualization:

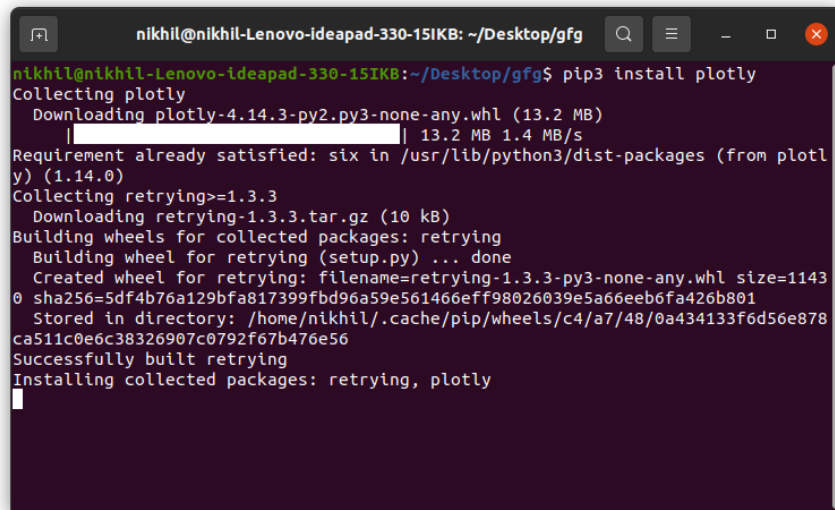
- **Interactivity:** Plotly has hover tool capabilities that allow us to detect any outliers or anomalies in a large number of data points.
- **Ease-of-Use:** It is visually attractive that can be accepted by a wide range of audiences.
- **Customization:** It allows us for the endless customization of our graphs that makes our plot more meaningful and understandable for others.
- **Export Options:** Plotly supports exporting visualizations to various formats, including static images and HTML files.

Installation and Setup with Plotly

Installation:

To install this module type the below command in the terminal.

```
pip install plotly
```



```
nikhil@nikhil-Lenovo-Ideapad-330-15IKB: ~/Desktop/gfg$ pip3 install plotly
Collecting plotly
  Downloading plotly-4.14.3-py2.py3-none-any.whl (13.2 MB)
    | 13.2 MB 1.4 MB/s
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from plotly) (1.14.0)
Collecting retrying>=1.3.3
  Downloading retrying-1.3.3.tar.gz (10 kB)
Building wheels for collected packages: retrying
  Building wheel for retrying (setup.py) ... done
  Created wheel for retrying: filename=retrying-1.3.3-py3-none-any.whl size=11430 sha256=5df4b76a129bfa817399fbd96a59e561466eff98026039e5a66eeb6fa426b801
  Stored in directory: /home/nikhil/.cache/pip/wheels/c4/a7/48/0a434133f6d56e878ca511c0e6c38326907c0792f67b476e56
Successfully built retrying
Installing collected packages: retrying, plotly
```

This may take some time as it will install the dependencies as well. For upgrading the version dependency refer to : [How To Upgrade Plotly To Latest Version?](#)

Plotly's Architecture in Python

Plotly is designed with a modular architecture that provides flexibility in creating visualizations. **At the heart of Plotly's architecture are two primary interfaces: [Plotly Express](#) and Plotly Graph Objects.** Each serves a distinct purpose and caters to different user needs, offering varying levels of complexity and customization.

1. Plotly Express

Plotly Express is a high-level, user-friendly interface for creating plots. It is designed to simplify the process of generating common types of visualizations with minimal code. Plotly Express is built on top of Plotly Graph Objects and provides a more streamlined and intuitive API for users who need to quickly produce interactive plots without delving into the intricacies of Plotly's lower-level functionalities.

Features:

- **Simplicity and Ease of Use:** Plotly Express allows users to create plots using a concise and straightforward syntax. This simplicity is achieved by abstracting many of the underlying complexities of Plotly's core

functionalities, making it accessible even for users who are new to data visualization.

- **Quick Visualization:** Plotly Express is ideal for exploratory data analysis and quick visualizations. Users can generate a wide range of common plot types—such as scatter plots, bar charts, and line plots—using just a few lines of code.
- **Automatic Handling:** Plotly Express automatically manages many aspects of the plot, such as axis labels, titles, and legends. This automation helps users focus on data analysis rather than the technical details of plot configuration.

Example:

Here's a simple example of creating a scatter plot using Plotly Express:

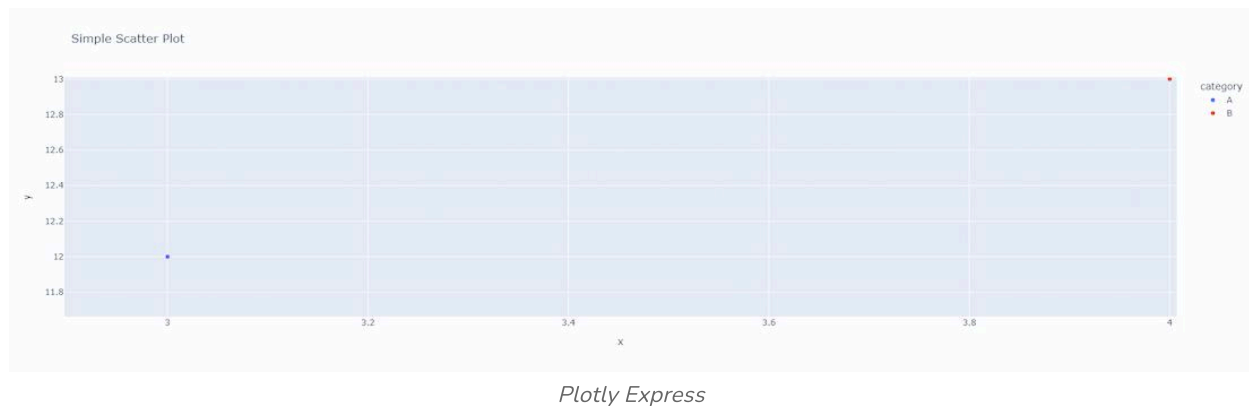
Python



```
1 import plotly.express as px
2 import pandas as pd
3
4 # Sample data
5 df = pd.DataFrame({
6     'x': [1, 2, 3, 4, 5],
7     'y': [10, 11, 12, 13, 14],
8     'category': ['A', 'B', 'A', 'B', 'A']
9 })
10
11 # Create a scatter plot
12 fig = px.scatter(df, x='x', y='y', color='category',
13                 title='Simple Scatter Plot')
14 fig.show()
```

[Python Basics](#)[Interview Questions](#)[Python Quiz](#)[Popular Packages](#)[Python Projects](#)[Practice Python](#)[AI Wit](#)

Output:



In this example, Plotly Express simplifies the process of creating a scatter plot, automatically managing the color differentiation based on the 'category' column.

2. Plotly Graph Objects

Plotly Graph Objects (often abbreviated as Plotly GO) is a lower-level interface that provides more granular control over plot creation and customization.

Unlike Plotly Express, which abstracts many details, Plotly Graph Objects allows users to construct plots by manually specifying each component.

This approach is suited for users who need fine-grained control over their visualizations or who are working with more complex plotting requirements.

Features:

- **Detailed Customization:** Plotly Graph Objects offers extensive customization options, allowing users to control nearly every aspect of the plot. This includes setting individual trace properties, customizing layout options, and adding annotations or shapes.
- **Complex Plotting:** Users can create intricate and sophisticated visualizations by combining different types of traces and layers. This capability is essential for building complex plots that require multiple data series or detailed configuration.
- **Manual Configuration:** Plotly Graph Objects requires users to manually define the structure of the plot, including traces, layout, and configuration settings. This approach provides flexibility but requires a deeper understanding of Plotly's syntax and parameters.

Example:

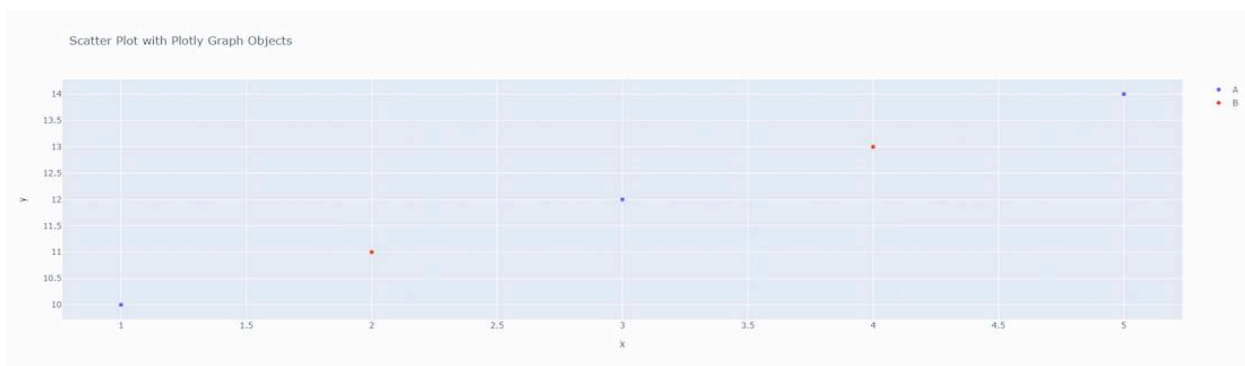
Here's an example of creating a scatter plot using Plotly Graph Objects:

Python



```
1 import plotly.graph_objects as go
2 import pandas as pd
3
4 # Sample data
5 df = pd.DataFrame({
6     'x': [1, 2, 3, 4, 5],
7     'y': [10, 11, 12, 13, 14],
8     'category': ['A', 'B', 'A', 'B', 'A']
9 })
10
11 # Create a scatter plot
12 fig = go.Figure()
13
14 # Add scatter traces
15 for category in df['category'].unique():
16     df_category = df[df['category'] == category]
17     fig.add_trace(go.Scatter(x=df_category['x'],
18                             y=df_category['y'], mode='markers', name=category))
19
20 # Update layout
21 fig.update_layout(title='Scatter Plot with Plotly Graph
22                     Objects', xaxis_title='x', yaxis_title='y')
23
24 fig.show()
```

Output:



Plotly Graph Objects

In this example, Plotly Graph Objects provides detailed control over each trace and the plot's layout. Users manually add each trace and configure the plot's appearance, which allows for a high level of customization.

Creating the Basic Plots with Plotly

Let's start by creating a simple line plot using Plotly Express, the high-level interface for Plotly. Let's create various plots using this module

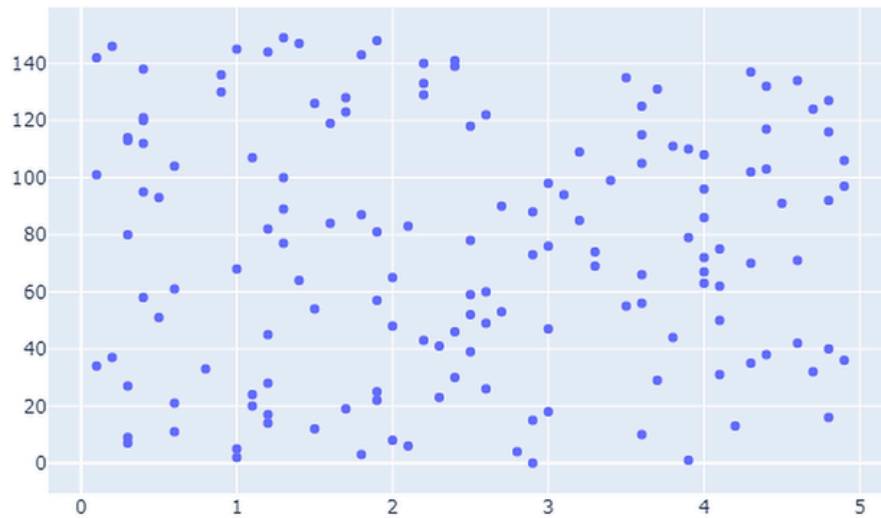
- **Scatter Plot:** Scatter plot represent values for two different numeric variables. They are mainly used for the representation of the relationship between two variables.

Python



```
1 # import all required libraries
2 import numpy as np
3 import plotly
4 import plotly.graph_objects as go
5 import plotly.offline as pyo
6 from plotly.offline import init_notebook_mode
7
8 init_notebook_mode(connected=True)
9
10 # generating 150 random integers
11 # from 1 to 50
12 x = np.random.randint(low=1, high=50, size=150)*0.1
13
14 # generating 150 random integers
15 # from 1 to 50
16 y = np.random.randint(low=1, high=50, size=150)*0.1
17
18 # plotting scatter plot
19 fig = go.Figure(data=go.Scatter(x=x, y=y, mode='markers'))
20
21 fig.show()
```

Output:



- **Bar charts:** Bar charts are used when we want to compare different groups of data and make inferences of which groups are highest and which groups are common and compare how one group is performing compared to others.

Python



```

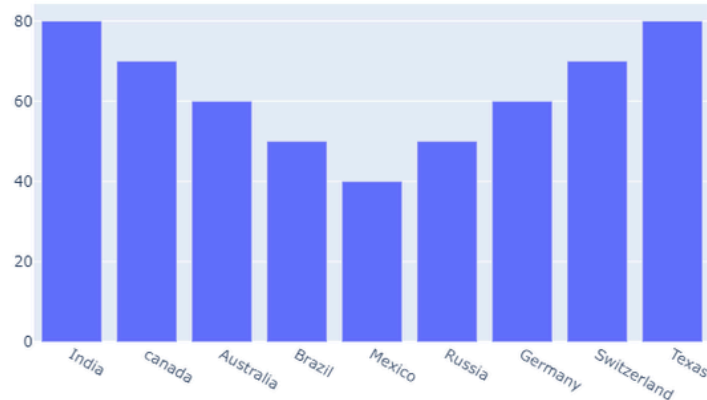
1  # import all required libraries
2  import numpy as np
3  import plotly
4  import plotly.graph_objects as go
5  import plotly.offline as pyo
6  from plotly.offline import init_notebook_mode
7
8  init_notebook_mode(connected = True)
9
10 # countries on x-axis
11 countries=['India', 'canada',
12            'Australia','Brazil',
13            'Mexico','Russia',
14            'Germany','Switzerland',
15            'Texas']
16
17 # plotting corresponding y for each
18 # country in x
19 fig = go.Figure([go.Bar(x=countries,
20                          y=[80,70,60,50,
21                             40,50,60,70,80])])
22

```



```
fig.show()
```

Output:



- **Pie chart:** A pie chart represents the distribution of different variables among total. In the pie chart each slice shows its contribution to the total amount.

Python



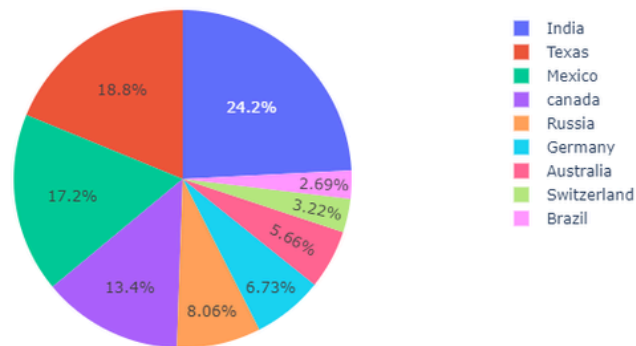
```
1 # import all required libraries
2 import numpy as np
3 import plotly
4 import plotly.graph_objects as go
5 import plotly.offline as pyo
6 from plotly.offline import init_notebook_mode
7
8 init_notebook_mode(connected = True)
9
10 # different individual parts in
11 # total chart
12 countries=['India', 'canada',
13           'Australia','Brazil',
14           'Mexico','Russia',
15           'Germany','Switzerland',
16           'Texas']
17
18 # values corresponding to each
19 # individual country present in
```

```

# countries
21 values = [4500, 2500, 1053, 500,
22           3200, 1500, 1253, 600, 3500]
23
24 # plotting pie chart
25 fig = go.Figure(data=[go.Pie(labels=countries,
26                               values=values)])
27
28 fig.show()

```

Output:



- **Histogram:** A histogram plots the continuous distribution of variable as series of bars and each bar indicates the frequency of the occurring value in a variable. In order to use a histogram, we simply require a variable that takes continuous numeric values

Python



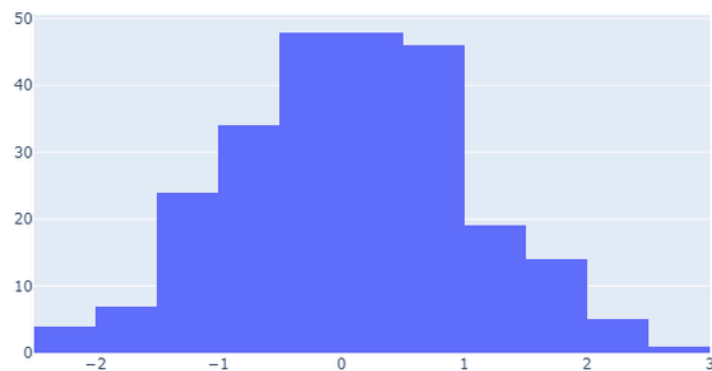
```

1 # import all required libraries
2 import numpy as np
3 import plotly
4 import plotly.graph_objects as go
5 import plotly.offline as pyo
6 from plotly.offline import init_notebook_mode
7
8 init_notebook_mode(connected = True)
9

```

```
# save the state of random
11 np.random.seed(42)
12
13 # generating 250 random numbers
14 x = np.random.randn(250)
15
16 # plotting histogram for x
17 fig = go.Figure(data=[go.Histogram(x=x)])
18
19 fig.show()
```

Output:



- **Box plot:** A box plot is the representation of a statistical summary. Minimum, First Quartile, Median, Third Quartile, Maximum.

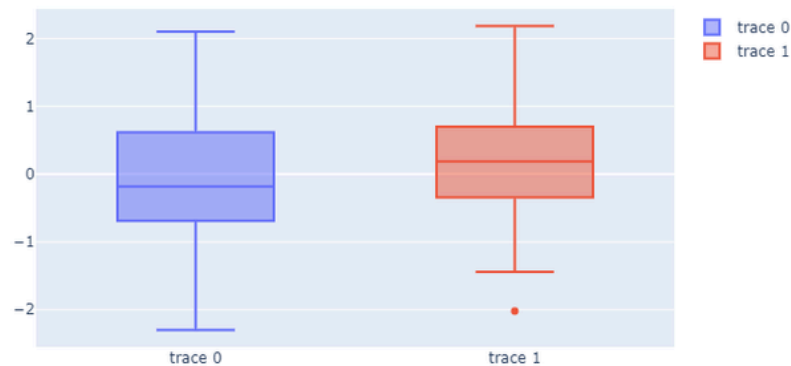
Python



```
1 # import all required libraries
2 import numpy as np
3 import plotly
4 import plotly.graph_objects as go
5 import plotly.offline as pyo
6 from plotly.offline import init_notebook_mode
7
8 init_notebook_mode(connected = True)
9
10 np.random.seed(42)
11
```

```
# generating 50 random numbers
13 y = np.random.randn(50)
14
15 # generating 50 random numbers
16 y1 = np.random.randn(50)
17 fig = go.Figure()
18
19 # updating the figure with y
20 fig.add_trace(go.Box(y=y))
21
22 # updating the figure with y1
23 fig.add_trace(go.Box(y=y1))
24
25 fig.show()
```

Output:



Data Structures and Plotly

Plotly is designed to be flexible and versatile, supporting a range of data structures for creating visualizations. This flexibility is essential for accommodating different types of data and ensuring that users can effectively represent their information in visual formats.

- The primary data structures used with Plotly include Pandas DataFrames, dictionaries, lists, and NumPy arrays.
- Understanding how to work with these structures is crucial for effective data visualization.

Each data structure has its own strengths and use cases:

- **Pandas DataFrames:** Ideal for structured, tabular data and easy integration with Plotly Express.
- **Dictionaries:** Useful for custom data representation and detailed plot configuration.
- **Lists:** Effective for representing data series and managing multiple traces.
- **NumPy Arrays:** Suitable for efficient numerical handling and large datasets.

Best Practices for Using Plotly

Plotly is a powerful tool for creating interactive and informative visualizations, but to make the most of it, it's important to follow best practices that ensure clarity, usability, and performance. Here are some key best practices to consider:

1. Choose the Right Plot Type

When to Use Plotly Express:

- **Quick and Simple Visualizations:** Use Plotly Express for rapid creation of common plot types (e.g., scatter plots, line charts, bar charts) with minimal code.
- **Exploratory Data Analysis:** Ideal for initial data exploration where ease of use and quick feedback are critical.

When to Use Plotly Graph Objects:

- **Complex Visualizations:** Opt for Plotly Graph Objects when you need to create intricate plots that require detailed customization.
- **Fine-Grained Control:** Use Graph Objects for advanced features like multiple traces, custom shapes, annotations, or intricate layout adjustments.

2. Keep Your Data Organized

Use Pandas DataFrames:

- **Structured Data:** Organize data in DataFrames for easy integration with Plotly Express, leveraging built-in support for DataFrames.
- **Data Cleaning:** Ensure data is clean and well-structured to avoid issues with plotting. This includes handling missing values and ensuring consistent

data types.

Alternative Structures:

- **Dictionaries and Lists:** Use dictionaries and lists for more customized data setups or when working with non-tabular data. Be mindful of the data format and ensure it aligns with the plot requirements.

3. Optimize for Performance

- **Large Datasets:** For large datasets, consider sampling or aggregating data to avoid performance issues. Plotly can handle large datasets, but performance may degrade with very large amounts of data.
- **Balance Interactivity and Performance:** While interactive features are powerful, too many interactive elements can slow down performance. Use interactivity judiciously to balance usability and performance.

Conclusion

Plotly is a powerful tool for creating interactive and engaging visualizations in Python. Its flexibility, ease of use, and integration with other data science tools make it a valuable asset for data analysts, scientists, and engineers. By understanding Plotly's features, architecture, and best practices, users can effectively leverage its capabilities to enhance their data visualization and analysis efforts.

Looking to dive into the world of programming or sharpen your Python skills? Our [Master Python: Complete Beginner to Advanced Course](https://www.geeksforgeeks.org/master-python-complete-beginner-to-advanced-course/) is your ultimate guide to becoming proficient in Python. This course covers everything you need to build a solid foundation from fundamental programming concepts to advanced techniques. With **hands-on projects**, real-world examples, and expert guidance, you'll gain the confidence to tackle complex **coding challenges**. Whether you're starting from scratch or aiming to enhance your skills, this course is the perfect fit. Enroll now and master Python, the language of the future!

Previous Article

Introduction to Plotly-online using Python

Next Article

Plotly for Data Visualization in Python

Similar Reads

How to hide legend with Plotly Express and Plotly in Python?

In this article, we will learn How to hide legend with Plotly Express and Plotly. Here we will discuss two different methods for hiding legend in plotly and plotl...

2 min read

Plotly - How to show legend in single-trace scatterplot with plotly express?

In this article let's see how to show legend in single-trace scatterplot with plotly express. A 'trace' is the name given to each plot inside the chart. Generally in...

1 min read

Define Colors in a Figure Using Plotly Graph Objects and Plotly Express

Whether you use Plotly Graph Objects or Plotly Express, defining colors in your figures allows you to create visually appealing and informative visualizations....

3 min read

Python | Getting started with SymPy module

SymPy is a Python library for symbolic mathematics. It aims to become a full-featured computer algebra system (CAS) while keeping the code as simple as...

4 min read

Python | Getting started with psycopg2-PostgreSQL

PostgreSQL is a powerful, open source object-relational database system. PostgreSQL runs on all major operating systems. PostgreSQL follows ACID...

1 min read

Getting started with Python for Automated Trading

Automated Trading is the terminology given to trade entries and exits that are processed and executed via a computer. Automated trading has certain...

3 min read

Getting Started with Pygame

Pygame is a free-to-use and open-source set of Python Modules. And as the name suggests, it can be used to build games. You can code the games and then...

3 min read

Getting Started with Python OpenCV

Computer Vision is one of the techniques from which we can understand images and videos and can extract information from them. It is a subset of artificial...

15+ min read

Getting started with Django

Python Django is a web framework that is used to create web applications very efficiently and quickly. Django is called a battery included framework because it...

15+ min read

Getting Started on Heroku with Python

Heroku is a cloud platform as a service supporting several programming languages where a user can deploy, manage and scale their applications. It is...

3 min read

Article Tags : [Python](#) [Python-Library](#) [python-modules](#)

Practice Tags : [python](#)



Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305) | Registered Address:- K 061,
Tower K, Gulshan Vivante Apartment,
Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305



Company

About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science

- Operating Systems
- Computer Network
- Database Management System
- Software Engineering
- Digital Logic Design
- Engineering Maths
- Software Development
- Software Testing

System Design

- High Level Design
- Low Level Design
- UML Diagrams
- Interview Guide
- Design Patterns
- OOAD
- System Design Bootcamp
- Interview Questions

School Subjects

- Mathematics
- Physics
- Chemistry
- Biology
- Social Science
- English Grammar
- Commerce
- World GK

DevOps

- Git
- Linux
- AWS
- Docker
- Kubernetes
- Azure
- GCP
- DevOps Roadmap

Interview Preparation

- Competitive Programming
- Top DS or Algo for CP
- Company-Wise Recruitment Process
- Company-Wise Preparation
- Aptitude Preparation
- Puzzles

GeeksforGeeks Videos

- DSA
- Python
- Java
- C++
- Web Development
- Data Science
- CS Subjects

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved