```
np.random.rand(3, 2)        # Output: 2D array with random floats
np.random.randint(1, 10, 5) # Output: Array of random integers from 1 to 10
```

These **NumPy** notes should give freshers a strong understanding of fundamental concepts and functions frequently asked in interviews.

Give matplotlib notes for freshers interview in detail

Here's a detailed guide to **Matplotlib** for freshers, covering key topics, syntax, and examples that are essential for interview preparation.

---

1. Introduction to Matplotlib

- **What is Matplotlib?**
  Matplotlib is a powerful Python library used for data visualization. It is particularly useful for creating static, interactive, and animated plots.
- **Why Use Matplotlib?**
  - Highly customizable.
  - Integrates well with libraries like Pandas, NumPy, and SciPy.
  - Used extensively for data analysis and data science projects.

---

2. Getting Started with Matplotlib

- **Importing Matplotlib**:
  - The `pyplot` module is commonly used for plotting, typically imported as `plt`.

```
python
```

```
import matplotlib.pyplot as plt
```

- **Basic Plot Structure**:
  - **Figure**: The entire canvas or plotting area.
  - **Axes**: The area where the data is plotted (a figure can have multiple axes).
  - **Axis**: X and Y (or Z) dimensions in the plot.

---

3. Basic Plotting Commands

- **Line Plot**:
  - Used to visualize data trends over time or ordered data points.

```python
plt.plot([1, 2, 3, 4], [10, 20, 25, 30], label='Line 1')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Plot')
plt.legend()
plt.show()
```

- **Scatter Plot**:
  - Displays individual data points and is ideal for showing relationships between variables.

```python
x = [5, 7, 8, 7, 2, 17, 2, 9]
y = [99, 86, 87, 88, 100, 86, 103, 87]
plt.scatter(x, y, color='blue')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')
plt.show()
```

- **Bar Plot**:

- Used to compare values across different categories.

```python
categories = ['A', 'B', 'C']
values = [10, 15, 7]
plt.bar(categories, values, color='green')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Plot')
plt.show()
```

- **Histogram**:
  - Useful for showing data distribution over intervals.

```python
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]
plt.hist(data, bins=5, color='purple')
plt.xlabel('Bins')
plt.ylabel('Frequency')
plt.title('Histogram')
plt.show()
```

- **Pie Chart**:
  - Visualizes part-to-whole relationships.

```python
sizes = [20, 30, 25, 25]
labels = ['Category A', 'Category B', 'Category C', 'Category D']
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Pie Chart')
plt.show()
```

---

4. Customizing Plots

- **Titles and Labels**:

- Use `plt.title()`, `plt.xlabel()`, and `plt.ylabel()` to add titles and axis labels.

- **Legend**:
  - Add legends with `plt.legend()`. You can specify location with `loc`.

```python
plt.plot([1, 2, 3], label='Line 1')
plt.plot([3, 2, 1], label='Line 2')
plt.legend(loc='upper right')
plt.show()
```

- **Colors and Line Styles**:
  - Customize lines with color (e.g., `'blue'`, `'r'`) and line styles (`'-'`, `'--'`, `':'`).

```python
plt.plot([1, 2, 3], color='red', linestyle='--', linewidth=2)
plt.show()
```

- **Markers**:
  - Add markers to highlight data points. Common markers are `'o'`, `'*'`, `'s'`.

```python
plt.plot([1, 2, 3], marker='o')
plt.show()
```

- **Grid**:
  - Add grids with `plt.grid()`, which enhances plot readability.

```python
plt.plot([1, 2, 3])
plt.grid(True)
plt.show()
```

---

5. Advanced Plotting Techniques

- **Subplots**:
  - Allows multiple plots in a single figure.

```python
fig, (ax1, ax2) = plt.subplots(1, 2)
ax1.plot([1, 2, 3], [4, 5, 6])
ax2.plot([1, 2, 3], [7, 8, 9])
plt.show()
```

- **Adding Annotations**:
  - Adds text or shapes to highlight specific parts of a plot.

```python
plt.plot([1, 2, 3, 4], [10, 20, 25, 30])
plt.annotate('Highest Point', xy=(4, 30), xytext=(3, 27),
             arrowprops=dict(facecolor='black', arrowstyle='->'))
plt.show()
```

- **Twin Axes**:
  - Used when plotting two datasets with different y-axes on the same plot.

```python
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.plot([1, 2, 3], [4, 5, 6], 'g-')
ax2.plot([1, 2, 3], [10, 15, 25], 'b-')
plt.show()
```

---

6. Working with Matplotlib and Pandas

- **Plotting Pandas DataFrame**:
  - Pandas integrates seamlessly with Matplotlib for easy data visualization.

```python
import pandas as pd
data = {'Category': ['A', 'B', 'C'], 'Values': [4, 3, 5]}
df = pd.DataFrame(data)
df.plot(kind='bar', x='Category', y='Values', color='orange')
plt.show()
```

---

7. Plot Styling

- **Styles**:
  - Use predefined styles to give plots a professional look. Available styles can be listed with `plt.style.available`.

```python
plt.style.use('ggplot')  # Apply 'ggplot' style
plt.plot([1, 2, 3], [4, 5, 6])
plt.show()
```

- **Saving Figures**:
  - Save plots as image files using `plt.savefig()`.

```python
plt.plot([1, 2, 3], [4, 5, 6])
plt.savefig('plot.png', dpi=300, bbox_inches='tight')
plt.show()
```

---

8. Common Interview Questions on Matplotlib

- **What is the difference between `plt.plot()` and `plt.scatter()`?**
  - `plt.plot()` is used for line plots and connects data points with a line, while `plt.scatter()` is used for scatter plots, displaying individual data points without

connecting them.

- **How do you create subplots in Matplotlib?**
  - Using `plt.subplot()` for a single subplot or `plt.subplots()` to create a grid of multiple subplots in a figure.
- **How can you customize line properties in a plot?**
  - Use parameters like `color`, `linestyle`, and `linewidth` within `plt.plot()` to customize line appearance.
- **How do you add a legend to a plot?**
  - Add labels to each plot using `label` within the plotting command, then call `plt.legend()` to display the legend.
- **How can you make a plot interactive?**
  - Use `%matplotlib inline` in Jupyter notebooks or `%matplotlib notebook` for interactive plots. Additionally, libraries like **Plotly** can add interactivity on top of Matplotlib.

---

9. Summary of Commonly Used Functions

Function	Description
`plt.plot()`	Creates line plots.
`plt.scatter()`	Creates scatter plots.
`plt.bar()`	Creates bar plots.
`plt.hist()`	Creates histograms.
`plt.pie()`	Creates pie charts.
`plt.xlabel()` / `plt.ylabel()`	Sets labels for x and y axes.
`plt.title()`	Sets title for the plot.
`plt.legend()`	Adds legend to the plot.
`plt.grid()`	Adds grid lines to the plot.
`plt.subplot()` / `plt.subplots()`	Adds subplots to the figure.
`plt.savefig()`	Saves plot as an image file.