10. Advanced Pandas Operations

- **Reshaping Data**:
  - **Melt**: Unpivots DataFrame from wide to long format. `pd.melt(df, id_vars, value_vars)`.
  - **Stack and Unstack**: Rearranges data into hierarchical indexes.

```python
# Example of melting
pd.melt(df, id_vars=['Name'], value_vars=['Math', 'Science'])
```

- **Window Functions**:
  - **Rolling**: Computes a moving statistic (e.g., rolling average).
    - `df['

Give numpy notes for freshers interview in detail

Here's a detailed set of notes on **NumPy** for freshers preparing for an interview. These notes cover foundational concepts, functions, and operations in **NumPy**.

---

1. Introduction to NumPy

- **What is NumPy?**
  NumPy (Numerical Python) is a powerful library for numerical and scientific computing in Python. It provides support for arrays, matrices, and a range of mathematical functions to operate on these arrays.
- **Key Features**:
  - Efficiently handles large datasets and high-dimensional data.
  - Offers mathematical, logical, and statistical functions.
  - Integrates well with other libraries like Pandas, Matplotlib, and SciPy.

---

2. NumPy Arrays

- **Creating Arrays**:
  - **1D Array** (Vector): `np.array([1, 2, 3])`
  - **2D Array** (Matrix): `np.array([[1, 2], [3, 4]])`
  - **3D Array** (Tensor): `np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])`
- **Array Attributes**:
  - `ndarray.ndim`: Number of dimensions (axes).
  - `ndarray.shape`: Tuple representing array shape (rows, columns).
  - `ndarray.size`: Total number of elements in the array.
  - `ndarray.dtype`: Data type of array elements.

```python
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr.shape)  # Output: (2, 3)
print(arr.ndim)   # Output: 2
print(arr.size)   # Output: 6
print(arr.dtype)  # Output: dtype('int32') (or another int type)
```

---

3. Array Creation Functions

- **Basic Creation**:
  - `np.array([1, 2, 3])`: Converts a list to an array.
  - `np.zeros((2, 3))`: Creates a 2x3 array filled with zeros.
  - `np.ones((2, 3))`: Creates a 2x3 array filled with ones.
  - `np.full((2, 3), 5)`: Creates a 2x3 array filled with the specified value (5).
- **Special Arrays**:
  - **Identity Matrix**: `np.eye(3)`, creates a 3x3 identity matrix.
  - **Arange**: `np.arange(start, stop, step)`, generates numbers within a specified range.

- **Linspace**: `np.linspace(start, stop, num)`, generates `num` evenly spaced numbers from start to stop.

```python

# Examples
np.zeros((2, 3))    # Output: [[0., 0., 0.], [0., 0., 0.]]
np.ones((2, 3))     # Output: [[1., 1., 1.], [1., 1., 1.]]
np.arange(1, 10, 2) # Output: [1, 3, 5, 7, 9]
np.linspace(0, 1, 5) # Output: [0., 0.25, 0.5, 0.75, 1.]
```

---

4. Array Indexing and Slicing

- **Indexing**:
  - Access elements with `arr[i]` (1D) or `arr[i, j]` (2D).
  - Negative indices to access elements from the end.
- **Slicing**:
  - **1D Array**: `arr[start:stop:step]`
  - **2D Array**: `arr[start_row:end_row, start_col:end_col]`
  - Omitting indices implies start or end of the dimension.

```python

arr = np.array([1, 2, 3, 4, 5])
print(arr[1:4])        # Output: [2 3 4]
arr_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(arr_2d[0:2, 1:3]) # Output: [[2 3], [5 6]]
```

---

5. Reshaping and Flattening Arrays

- **Reshaping**: Changes the shape of an array without modifying its data.
  - `arr.reshape(rows, columns)`: Reshapes array to a new shape.

- **Flattening**:
  - Converts a multi-dimensional array to a 1D array using `arr.flatten()` or `arr.ravel()`.

```python
arr = np.array([[1, 2], [3, 4], [5, 6]])
arr_reshaped = arr.reshape(2, 3) # Output: [[1 2 3], [4 5 6]]
print(arr.flatten())             # Output: [1 2 3 4 5 6]
```

---

6. Array Operations

- **Arithmetic Operations**:
  - Element-wise operations: `+`, `-`, `*`, `/`, `**` (power).
  - Example:

```python
arr = np.array([1, 2, 3])
print(arr + 2)    # Output: [3 4 5]
print(arr * 3)    # Output: [3 6 9]
```

- **Statistical Operations**:
  - `np.sum(arr)`: Sum of all elements.
  - `np.mean(arr)`: Mean of elements.
  - `np.median(arr)`: Median of elements.
  - `np.std(arr)`: Standard deviation.
  - **Axis-Based Operations**: Apply along rows (`axis=0`) or columns (`axis=1`).

```python
arr = np.array([[1, 2], [3, 4]])
print(np.sum(arr, axis=0)) # Output: [4 6] (column-wise sum)
print(np.mean(arr))        # Output: 2.5
```

7. Broadcasting

- **Definition**: Broadcasting allows NumPy to perform element-wise operations on arrays of different shapes.
- **Rules**:
  - If arrays have different dimensions, the smaller array is "stretched" to match the larger array.
  - Compatible shapes for broadcasting have the same size in each dimension or one of them is 1.

```python
arr = np.array([1, 2, 3])
print(arr + 5)            # Broadcasted scalar addition, Output: [6 7 8]
arr_2d = np.array([[1, 2, 3], [4, 5, 6]])
print(arr_2d + arr)       # Broadcasting 1D array to 2D, Output: [[2 4 6], [5 7 9]]
```

---

8. Boolean and Comparison Operations

- **Element-wise Comparison**:
  - `arr1 == arr2`, `arr1 > arr2`, etc., return boolean arrays.
- **Boolean Indexing**:
  - Retrieve elements based on conditions.
  - Example:

```python
arr = np.array([1, 2, 3, 4, 5])
print(arr[arr > 2])    # Output: [3 4 5]
```

- **Logical Operations**:
  - `np.any(arr > value)`: Checks if any element satisfies condition.

- `np.all(arr > value)`: Checks if all elements satisfy condition.

---

9. Array Math Functions

- **Common Math Functions**:
  - `np.exp(arr)`: Exponential.
  - `np.sqrt(arr)`: Square root.
  - `np.log(arr)`: Natural logarithm.
  - `np.sin(arr)`, `np.cos(arr)`: Trigonometric functions.

```python
arr = np.array([1, 4, 9])
print(np.sqrt(arr))  # Output: [1. 2. 3.]
```

- **Aggregate Functions**:
  - `np.min(arr)`, `np.max(arr)`: Min and max values.
  - `np.argmin(arr)`, `np.argmax(arr)`: Indices of min and max values.

---

10. Random Functions

- **Generating Random Numbers**:
  - `np.random.rand(d0, d1, …, dn)`: Generates random floats in [0, 1).
  - `np.random.randint(low, high, size)`: Generates random integers.
  - `np.random.randn(d0, d1, …, dn)`: Generates samples from standard normal distribution.
- **Shuffling**:
  - `np.random.shuffle(arr)`: Shuffles array in-place.
  - `np.random.permutation(arr)`: Returns a new shuffled array.

```python
```

```
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
```