

## **Q1. Create a Pandas Series with data 4, 8, 15, 16, 23, 42**

```
import pandas as pd

# Create Pandas Series
data = [4, 8, 15, 16, 23, 42]
series = pd.Series(data)

# Print the Series
print(series)
```

**Output:**

```
0      4
1      8
2     15
3     16
4     23
5     42
dtype: int64
```

---

## **Q2. Create a list of 10 elements and convert to Series**

```
my_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
series2 = pd.Series(my_list)

print(series2)
```

**Output:**

```
0      10
1      20
2      30
3      40
```

```
4      50
5      60
6      70
7      80
8      90
9     100
dtype: int64
```

---

### Q3. Create a Pandas DataFrame

Suppose the data is like this:

```
Name  Age  Gender
Alice  25   Female
Bob    30   Male
Claire 27   Female
data = {
    "Name": ["Alice", "Bob", "Claire"],
    "Age": [25, 30, 27],
    "Gender": ["Female", "Male", "Female"]
}

df = pd.DataFrame(data)

print(df)
```

#### Output:

```
      Name  Age  Gender
0    Alice  25   Female
1      Bob  30     Male
2  Claire  27   Female
```

---

## Q4. What is a DataFrame? Difference from Series

### DataFrame:

- A 2-dimensional, **tabular data structure** in Pandas.
- Contains rows and columns, each column can have a **different data type**.

### Series:

- 1-dimensional, like a single column.
- Can hold data of any type, but only **one dimension**.

### Example:

```
# Series (1D)
s = pd.Series([10, 20, 30])
print(s)

# DataFrame (2D)
df = pd.DataFrame({
    "Numbers": [10, 20, 30],
    "Letters": ["A", "B", "C"]
})
print(df)
```

### Difference:

- Series: only **one column**, DataFrame: **multiple columns** with labels.

---

## Q5. Common functions to manipulate DataFrame

Function	Purpose
head()	View first n rows

<code>tail()</code>	View last n rows
<code>info()</code>	Summary of DataFrame
<code>describe()</code>	Statistical summary of numeric columns
<code>sort_values</code>	Sort data by a column
<code>s()</code>	
<code>drop()</code>	Remove columns or rows
<code>fillna()</code>	Replace missing values
<code>groupby()</code>	Aggregate data by column

#### Example use:

- Suppose you want the **average age of people in the DataFrame**:

```
avg_age = df[ 'Age' ].mean()  
print("Average Age:", avg_age)
```

---

## Q6. Which is mutable: Series, DataFrame, Panel?

- **Mutable:** Series and DataFrame
- **Panel:** Deprecated in recent versions of Pandas, was 3D structure.

#### Explanation:

- You can modify the contents of a Series or DataFrame after creation.
- 

## Q7. Create a DataFrame using multiple Series

```
# Create Series  
names = pd.Series(["Alice", "Bob", "Claire"])  
ages = pd.Series([25, 30, 27])
```

```
genders = pd.Series(["Female", "Male", "Female"])

# Create DataFrame from Series
df_series = pd.DataFrame({
    "Name": names,
    "Age": ages,
    "Gender": genders
})

print(df_series)
```

**Output:**

```
Name  Age  Gender
0   Alice  25  Female
1     Bob  30    Male
2  Claire  27  Female
```

**Explanation:**

- Each column is created from a **Pandas Series**.
- **DataFrame** automatically aligns **indexes** and combines into a table.