1. Reading and Writing CSV Files

a. read_csv()

Syntax:

```
pd.read_csv(filepath_or_buffer, sep=',', header='infer', index_col=None, usecols=None, nrows=None)
```

Parameters:

- *filepath_or_buffer* (required): Path or URL of the CSV file.
- *sep* (optional): The delimiter (default is comma ,).
- *header* (optional): Row number(s) to use as column names (default is first row).
- *index_col* (optional): Column(s) to set as index.
- *usecols* (optional): Specify which columns to read.
- *nrows* (optional): Number of rows to read.

Full Code Example:

```
import pandas as pd

# Reading a CSV file into a DataFrame
df = pd.read_csv('employees.csv')

# Display the first 5 rows
print(df.head(),"\n")

# Reading a CSV with a specific separator, custom column names, and limiting rows
df_custom = pd.read_csv('employees.csv', sep=';', names=['Col1', 'Col2', 'Col3'], nrows=10)
print(df_custom)
```

b. to_csv()

Syntax:

```
DataFrame.to_csv(path_or_buf, sep=',', header=True, index=True)
```

Parameters:

- *path_or_buf* (required): File path or object to write to.
- *sep* (optional): The delimiter (default is comma ,).
- *header* (optional): Whether to write column names (default is True).
- *index* (optional): Whether to write row numbers (default is True).

Full Code Example:

```
import pandas as pd

# Creating a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'City': ['New York', 'Los Angeles', 'Chicago']}

df = pd.DataFrame(data)

# Writing a DataFrame to a CSV file
df.to_csv('output_data.csv', index=False)

# Reading back the CSV file to verify
df_read_back = pd.read_csv('output_data.csv')
```

2. Reading and Writing Excel Files

a. read_excel()

Syntax:

```
pd.read_excel(io, sheet_name=0, usecols=None, skiprows=None)
```

Parameters:

- *io* (required): File path or Excel file object.
- *sheet_name* (optional): Name or index of the sheet (default is the first sheet).
- *usecols* (optional): Specify which columns to read.
- *skiprows* (optional): Rows to skip at the beginning.

Full Code Example:

```
import pandas as pd

# Reading an Excel file into a DataFrame
df = pd.read_excel('marks.xlsx')

# Display the first 5 rows
print(df.head())

# Reading specific columns and skipping rows
df_custom = pd.read_excel('marks.xlsx', sheet_name='Sheet1', usecols=['Student', 'Marks'], skiprows=0)
print(df_custom)
```

b. to_excel()

Syntax:

```
DataFrame.to_excel(excel_writer, sheet_name='Sheet1', index=True)
```

Parameters:

- *excel_writer* (required): File path or Excel writer object.
- *sheet_name* (optional): Name of the sheet (default is 'Sheet1').
- *index* (optional): Whether to write row numbers (default is True).

Full Code Example:

```
import pandas as pd

# Creating a sample DataFrame
data = {'Product': ['Laptop', 'Smartphone', 'Tablet'],
        'Price': [1000, 800, 600],
        'Stock': [50, 150, 70]}

df = pd.DataFrame(data)

# Writing a DataFrame to an Excel file
df.to_excel('output_data.xlsx', sheet_name='Products', index=False)

# Reading the Excel file back to verify
df_read_back = pd.read_excel('output_data.xlsx')
print(df_read_back)
```

3. Reading and Writing JSON Files

a. read_json()

Syntax:

```
pd.read_json(path_or_buf, orient=None, lines=False)
```

Parameters:

- *path_or_buf* (required): File path or JSON string.
- *orient* (optional): The format of the JSON string (e.g., 'split', 'records').
- *lines* (optional): Whether to treat each line as a separate JSON object (default is False).

Full Code Example:

```
import pandas as pd

# Reading a JSON file into a DataFrame
df = pd.read_json('cities.json')

# Display the first 5 rows
print(df.head())

# Reading JSON in 'records' orientation
df_custom = pd.read_json('cities.json', orient='records')
print(df_custom)
```

b. to_json()

Syntax:

```
DataFrame.to_json(path_or_buf=None, orient=None, lines=False)
```

Parameters:

- *path_or_buf* (optional): File path or object to write to.
- *orient* (optional): The format for writing the JSON (e.g., 'split', 'records').
- *lines* (optional): Whether to write each row as a separate JSON object (useful for large data).

Full Code Example:

```
import pandas as pd

# Creating a sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Salary': [70000, 85000, 90000],
        'Department': ['HR', 'Finance', 'IT']}

df = pd.DataFrame(data)

# Writing a DataFrame to a JSON file
df.to_json('output_data.json')

# Reading the JSON file back to verify
df_read_back = pd.read_json('output_data.json')
print(df_read_back)

# Writing JSON in 'records' orientation with lines
df.to_json('output_data_records.json', orient='records', lines=True)
```

5. Reading and Writing HTML Data

a. read_html()

Syntax:

```
pd.read_html(io, match=None, flavor=None)
```

Parameters:

- *io* (required): URL, file path, or string containing the HTML content.
- *match* (optional): A string to filter tables.
- *flavor* (optional): The parsing engine ('bs4' or 'lxml').

Full Code Example:

```
import pandas as pd

# Reading tables from an HTML file or webpage
tables = pd.read_html('inputdata.html')

# Display the first table found on the page
print(tables[0])
```

b. to_html()

Syntax:

```
DataFrame.to_html(buf=None, columns=None, index=True)
```

Parameters:

- *buf* (optional): File path or buffer to write to.
- *columns* (optional): Columns to write.
- *index* (optional): Whether to write row indices (default is True).

Full Code Example:

```
import pandas as pd

# Creating a sample DataFrame
data = {'Country': ['USA', 'Canada', 'Germany'],
        'Capital': ['Washington D.C.', 'Ottawa', 'Berlin'],
        'Population': [328, 37, 83]}

df = pd.DataFrame(data)

# Writing a DataFrame to an HTML file
df.to_html('output_data.html')

# Reading back the saved HTML file
with open('output_data.html', 'r') as file:
    html_content = file.read()
    print(html_content)
```