Pandas Cheat Sheet

Pandas Basics

1. Creating DataFrames

In [ ]:
```
import pandas as pd

# From a dictionary
data = {'Name': ['John', 'Anna', 'Peter'], 'Age': [28, 24, 35]}
df = pd.DataFrame(data)

# From a list of dictionaries
data = [{'Name': 'John', 'Age': 28},
        {'Name': 'Anna', 'Age': 24},
        {'Name': 'Peter', 'Age': 35}]
df = pd.DataFrame(data)

# From a CSV file
df = pd.read_csv('file.csv')

# From an Excel file
df = pd.read_excel('file.xlsx')
```

2. Viewing Data

In [ ]:
```
df.head()          # First 5 rows
df.tail()          # Last 5 rows
df.sample(n=5)     # Random sample of 5 rows
df.shape           # Dimensions of the DataFrame
df.info()          # Information about the DataFrame
df.describe()      # Statistical summary
df.columns         # Column names
df.index           # Index of the DataFrame
```

3. Selecting Data

In [ ]:
```
df['column_name']              # Select a single column
df[['column1', 'column2']]     # Select multiple columns

df.iloc[0]                     # Select first row (by position)
df.loc[0]                      # Select first row (by label)

df.iloc[0:5]                   # Select first 5 rows (by position)
df.loc[0:5]                    # Select first 6 rows (by label)

df.iloc[:, 0]                  # Select first column (by position)
df.loc[:, 'column_name']       # Select first column (by label)
```

4. Filtering Data

In [ ]:
```
df[df['column_name'] > 20]     # Filter rows by column value
df[df['column_name'] == 20]

# Filter with OR condition
df[(df['column_name'] == 20) | (df['column_name'] == 30)]
# Filter with AND condition
df[(df['column_name_1'] == 20) & (df['column_name_2'] == 30)]
```

5. Adding/Removing Data

In [ ]:
```
# Adding columns
df['New_Column'] = df['column1'] + df['column2']

# Removing columns
df.drop('column_name', axis=1, inplace=True)
df.drop(columns=['column1', 'column2'], inplace=True)

# Adding rows
new_row = {'Name': 'Linda', 'Age': 30}
df = df.append(new_row, ignore_index=True)

# Removing rows
df.drop(0, axis=0, inplace=True)   # Drop first row
df.drop([0, 1], axis=0, inplace=True)  # Drop multiple rows
```

6. Removing Duplicates

In [ ]:
```
df.drop_duplicates()
```

7. Handling Missing Data

In [ ]:
```
df.isnull()                    # Check for missing values
df.isnull().sum()              # Sum of missing values per column
df.dropna()                    # Drop rows with missing values
df.dropna(axis=1)              # Drop columns with missing values
df.fillna(0)                   # Replace missing values with 0
df.fillna(df.mean())           # Replace missing values with column mean
```

8. Plotting Data

In [ ]:
```
import matplotlib.pyplot as plt

df.plot(kind='line')           # Line plot
df.plot(kind='bar')            # Bar plot
df.plot(kind='hist')           # Histogram
df.plot(kind='box')            # Box plot
df.plot(kind='scatter', x='column1', y='column2')  # Scatter plot

plt.show()
```

Data Operations

1. Data Type Conversion

In [ ]:
```
df['column'] = df['column'].astype('int')
df['column'] = df['column'].astype('str')
```

2. Sorting Data

In [ ]:
```
# Sort by column
df.sort_values('column_name')

# Sort by multiple columns
df.sort_values(['column1', 'column2'], ascending=[True, False])

# Sort by index
df.sort_index()

# Sort just a single column
df["column_name"].sort_values()
```

3. Grouping Data

In [ ]:
```
# Group by column and calculate mean
df.groupby('column_name').mean()

# Group by multiple columns and calculate sum
df.groupby(['column1', 'column2']).sum()
```

4. Aggregations

In [ ]:
```
# Custom aggregation
df.groupby('column_name').agg({'column1': 'mean', 'column2': 'sum'})
```

5. Merging Data

In [ ]:
```
# Inner join
pd.merge(df1, df2, on='key')

# Left join
pd.merge(df1, df2, on='key', how='left')

# Right join
pd.merge(df1, df2, on='key', how='right')

# Outer join
pd.merge(df1, df2, on='key', how='outer')

# Concatenation
pd.concat([df1, df2])                    # Concatenate along rows
pd.concat([df1, df2], axis=1)            # Concatenate along columns
```

6. Pivot Tables

In [ ]:
```
pd.pivot_table(df, values='value_column', index='index_column',
               columns='columns_column', aggfunc='mean')
```

Advanced Operations

1. Applying Functions

In [ ]:
```
df.apply(np.sum)          # Apply function to each column
df.apply(np.sum, axis=1)  # Apply function to each row
df['column'].apply(np.sqrt)  # Apply function to a column
```

2. Lambda Functions

In [ ]:
```
df['column'] = df['column'].apply(lambda x: x + 1)
df.apply(lambda row: row['col1'] + row['col2'], axis=1)
```

3. Working with Dates

In [ ]:
```
df['Date'] = pd.to_datetime(df['Date'])  # Convert to datetime
df['Year'] = df['Date'].dt.year          # Extract year
df['Month'] = df['Date'].dt.month        # Extract month
df['Day'] = df['Date'].dt.day            # Extract day
df.set_index('Date', inplace=True)       # Set datetime column as index
```

4. String Operations

In [ ]:
```
df['column'].str.lower()                # Convert to lower case
df['column'].str.upper()                # Convert to upper case
df['column'].str.contains('substring') # Check for substring
df['column'].str.replace('old', 'new') # Replace substring
df['column'].str.split('delimiter')    # Split strings
```

5. File Operations

In [ ]:
```
# CSV
df.to_csv('file.csv', index=False)     # Write to CSV
df = pd.read_csv('file.csv')           # Read from CSV

# Excel
df.to_excel('file.xlsx', index=False)  # Write to Excel
df = pd.read_excel('file.xlsx')        # Read from Excel

# JSON
df.to_json('file.json')                # Write to JSON
df = pd.read_json('file.json')         # Read from JSON
```