Pandas Cheatsheet

Core Data Structures

Pandas is the core library for data manipulation and analysis in Python.

- `Series`: 1D labeled array.
- `DataFrame`: 2D labeled, size-mutable tabular data.

Initialization and Data Loading

- Import: `import pandas as pd`
- Create from Dict: `df = pd.DataFrame({'A': [1, 2], 'B': [3, 4]})`
- From CSV: `df = pd.read_csv('file.csv')`
- From Excel: `df = pd.read_excel('file.xlsx')`
- To CSV: `df.to_csv('output.csv', index=False)`

Inspection and Selection

- View Data: `df.head()`, `df.tail()`, `df.sample(5)`.
- Metadata: `df.info()`, `df.describe()`, `df.shape`, `df.columns`.
- Indexing:
  - `df['col_name']`: Select column as Series.
  - `df[['C1', 'C2']]`: Select multiple columns as DataFrame.
  - `df.loc[row_label, col_label]`: Label-based selection.
  - `df.iloc[row_index, col_index]`: Integer-based selection.
- Filtering: `df[df['A'] > 10]` or `df.query('A > 10')`.

Data Cleaning

- Handle Missing Values:
  - `df.isnull()` / `df.isna()`: Check for NAs.
  - `df.dropna()`: Drop rows with NAs.
  - `df.fillna(value)`: Fill NAs with a constant or mean/median.
- Data Transformation:
  - `df.drop(columns=['A'])`: Drop a column.
  - `df.rename(columns={'old': 'new'})`: Rename columns.
  - `df.astype({'col': 'int32'})`: Change data type.
  - `df.replace(old, new)`: Replace values.
- Duplicates: `df.duplicated()`, `df.drop_duplicates()`.

Aggregation and Grouping

- Statistics: `df.mean()`, `df.sum()`, `df.min()`, `df.max()`, `df.median()`, `df.std()`.
- Grouping:

```
df.groupby('category')['score'].mean()
df.groupby(['A', 'B']).agg(['sum', 'mean'])
```

- Value Counts: `df['col'].value_counts()`.

Merging and Pivoting

- Joins:
  - `pd.concat([df1, df2])`: Combine along axis.
  - `pd.merge(df1, df2, on='key')`: SQL-style join (inner, left, right, outer).
- Reshaping:
  - `df.pivot_table(values='v', index='i', columns='c', aggfunc='mean')`.
  - `df.melt()`: Transform from wide to long format.

Working with Strings and Dates

- Strings: `df['col'].str.upper()`, `df['col'].str.contains('pattern')`.
- Dates:
  - `df['date'] = pd.to_datetime(df['date'])`.
  - `df['date'].dt.year`, `df['date'].dt.month`, `df['date'].dt.day_name()`.

Applying Functions

- `df.apply(func)` / `df.map(func)`: Apply a function to rows/columns or element-wise. `df['A'].apply(lambda x: x * 2)`

Performance Tips

- Use vectorized operations instead of iterating with `for` loops.
- Use `inplace=True` with caution; many methods return a new copy by default.
- Specify `dtypes` during `read_csv` for memory efficiency.
- Use categorical data types for columns with low cardinality. `df['cat'] = df['cat'].astype('category')`