Pandas Beginner Cheat Sheet

1. Import & Core Data Structures

Import the library and understand its two main data structures: `Series` (1D) and `DataFrame` (2D).

Standard Import

```
import pandas as pd
```

Series (1D Labeled Array)

```
# Create a Series from a list
s = pd.Series([1, 3, 5, np.nan, 6, 8])
```

DataFrame (2D Labeled Table)

The most common Pandas object.

```
# Create a DataFrame from a dictionary
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 28],
        'City': ['New York', 'Paris', 'London']}
df = pd.DataFrame(data)
```

2. Reading Data

Load data from external files into a DataFrame.

```
# Read from a CSV file
df_csv = pd.read_csv('my_file.csv')

# Read from an Excel file
# Requires the 'openpyxl' package
df_excel = pd.read_excel('my_file.xlsx')

# Useful parameters for read_csv
df_params = pd.read_csv(
    'data.csv',
    sep=';',           # Specify separator
    header=None,       # If no header row
    names=['a', 'b'] # Provide column names
)
```

3. Inspecting a DataFrame

Quickly view and get information about your data.

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

# Display the last 5 rows
df.tail()

# Get a concise summary of the DataFrame
# (index, dtypes, non-null counts, memory)
df.info()

# Generate descriptive statistics
# (count, mean, std, min, max, quartiles)
df.describe()

# Get the dimensions (rows, columns)
df.shape

# Get the column labels
df.columns

# Get the index (row labels)
df.index
```

4. Selection & Indexing

Select subsets of your data. The most important methods are `.loc` (label-based) and `.iloc` (position-based).

Selecting Columns

```
# Select a single column (returns a Series)
df['Age']
# or using dot notation (less flexible)
df.Age

# Select multiple columns (returns a DataFrame)
df[['Name', 'City']]
```

Selecting Rows with `.loc` (Label-based)

Selects data by index label.

```
# Select row with index label 0
df.loc[0]

# Select rows with index labels 0 and 2
df.loc[[0, 2]]

# Slice rows from index label 0 to 2 (inclusive)
df.loc[0:2]
```

Selecting Rows with `.iloc` (Position-based)

Selects data by integer position (like a Python list).

```
# Select row at integer position 0
df.iloc[0]

# Select rows at integer positions 0 and 2
df.iloc[[0, 2]]

# Slice rows from position 0 up to (not incl.) 3
df.iloc[0:3]
```

Selecting Both Rows and Columns

Use the format `df.loc[rows, columns]`.

```
# Select a single value (scalar)
df.loc[0, 'Name'] # Row 0, column 'Name'

# Select a slice of rows and specific columns
df.loc[0:1, ['Name', 'Age']]
```

5. Filtering Data

Use boolean indexing to select data based on conditions.

```
# Create a boolean mask
mask = df['Age'] > 25

# Apply the mask to the DataFrame
df[mask]

# A more concise way
df[df['Age'] > 25]
```

Combining Conditions

Use `&` for AND, `|` for OR. **Wrap each condition in parentheses!**

Pandas Beginner Cheat Sheet

```
# Filter for Age > 25 AND City == 'New York'
df[(df['Age'] > 25) & (df['City'] == 'New York')]

# Filter for Age < 30 OR City == 'London'
df[(df['Age'] < 30) | (df['City'] == 'London')]
```

6. Basic Operations

Adding a New Column

```
# Create a new column
df['Salary'] = [50000, 80000, 75000]

# Create a new column based on existing ones
df['Age_in_5_Years'] = df['Age'] + 5
```

Dropping Columns or Rows

Use `df.drop()`. Use `axis=1` for columns, `axis=0` for rows.

```
# Drop a column
df.drop('Salary', axis=1)

# Drop a row by index label
df.drop(0, axis=0)

# The 'inplace=True' argument modifies the
# DataFrame directly instead of returning a new one
df.drop('Salary', axis=1, inplace=True)
```

7. Summarizing Data

Perform aggregations and calculate summary statistics.

```
# Mean of a column
df['Age'].mean()

# Sum of a column
df['Age'].sum()

# Count of non-NA values in a column
df['Age'].count()

# Number of unique values
df['City'].nunique()

# Counts of unique values (very useful!)
df['City'].value_counts()
```

8. Handling Missing Data

Find, remove, or replace `NaN` (Not a Number) values.

```
# Create a DataFrame with missing data
data_nan = {'A': [1, 2, np.nan], 'B': [4, np.nan,
    6]}
df_nan = pd.DataFrame(data_nan)

# Check for null values (returns boolean DF)
df_nan.isnull()

# Count null values in each column
df_nan.isnull().sum()

# Drop rows with any missing values
df_nan.dropna()

# Fill missing values with a specific value
df_nan.fillna(value=0)

# Fill with the mean of the column
df_nan['A'].fillna(df_nan['A'].mean())
```