Pandas for Machine Learning Notes

Lesson 1: Introduction

i) Difference Between DataFrame and Series:

- Series: A one-dimensional labeled array that can hold any data type (similar to a column).

- DataFrame: A two-dimensional labeled data structure with columns of potentially different types (similar to a table or spreadsheet).

ii) Identifying DataFrame and Series:

- Use the type() function to identify whether an object is a Series or a DataFrame.

Example:

```python
import pandas as pd

s = pd.Series([1, 2, 3])
df = pd.DataFrame({'A': [1, 2], 'B': [3, 4]})

print(type(s))  # <class 'pandas.core.series.Series'>
print(type(df))  # <class 'pandas.core.frame.DataFrame'>
```

iii) Important Information:

- Both Series and DataFrames are part of the Pandas library, essential for data manipulation and analysis in Python.

Lesson 2: Reading Data

- Use the read_csv() function to read CSV files:

```python
df = pd.read_csv('fileName.csv')
```

- For other file types, use similar functions:
  - Excel: pd.read_excel('fileName.xlsx')
  - JSON: pd.read_json('fileName.json')

Important Points:

- Ensure the file path is correct.
- Use parameters like header, index_col, and dtype to customize how data is read.

Lesson 3: Important Functions and Attributes

Difference Between Function and Attribute:

- Function: An action that can be performed (e.g., df.head()).
- Attribute: A property of the object (e.g., df.shape).

Key Functions and Attributes:

- head(): Returns the first n rows of the DataFrame (default is 5).

```python
df.head()
```

- tail(): Returns the last n rows of the DataFrame (default is 5).

```python
df.tail()
```

```

- shape: Returns a tuple representing the dimensions of the DataFrame (rows, columns).

```python
df.shape  # Output: (number_of_rows, number_of_columns)
```

- info(): Provides a summary of the DataFrame, including data types and non-null values.

```python
df.info()
```

- describe(): Generates descriptive statistics for numerical columns.

```python
df.describe()
```

Lesson 4: Fetching Rows and Columns

Fetching a Single Row or Column:

- Fetching a single row or column returns a Series.

```python
single_row = df.iloc[0]  # First row
single_column = df['columnName']  # Specific column
```

Fetching Multiple Columns:

- To fetch multiple columns, use double square brackets:
```

```python
multi_columns = df[['columnName1', 'columnName2']]
```

Using iloc() Function:

- iloc(): Allows for integer-location based indexing for selection by position.

Fetching Rows:

```python
rows = df.iloc[1:3]  # Fetch rows at index 1 and 2
```

Fetching Columns:

```python
columns = df.iloc[:, [1, 3, 5]]  # Fetch columns at index 1, 3, and 5
```

Detailed Explanation of iloc():

- Usage: data.iloc[row_indices, column_indices]
  - row_indices: Specify which rows to fetch (single integer, slice, or list of integers).
  - column_indices: Specify which columns to fetch (same as row indices).

Examples:

- Fetching a single row:

```python
first_row = df.iloc[0]  # Fetch the first row
```

- Fetching a range of rows:

```python
middle_rows = df.iloc[2:5]  # Fetch rows 2 to 4
```

- Fetching specific columns:

```python
specific_columns = df.iloc[:, [0, 2]]  # Fetch first and third columns
```