Pandas - DataFrame and Series Notes

Introduction

Pandas is a powerful data manipulation library in Python, widely used for data analysis and data cleaning. It provides two primary data structures: Series and DataFrame. A Series is a one-dimensional array-like object, while a DataFrame is a two-dimensional, size-mutable, and potentially heterogeneous tabular data structure with labeled axes.

Why Pandas?

Pandas is ideal for handling real-world data: it efficiently handles missing data, supports vectorized operations, and integrates with libraries such as NumPy and Matplotlib. Whether you're performing exploratory analysis or building production-level data pipelines, Pandas offers a rich set of tools.

Creating a Series

A Panda Series is a one-dimensional array-like object that can hold any data type. It is similar to a column in a table.

Creating a DataFrame

A DataFrame is a two-dimensional structure that can store data of different types in columns. It is similar to a table in a relational database or an Excel spreadsheet.

Accessing Data

You can access data in a DataFrame by using column names, row indices, or by using methods like loc, iloc, at, and iat. This flexibility allows you to easily slice and dice your data.

Data Manipulation

Common operations include adding/removing columns, updating values, and dropping rows. Pandas provides robust methods to handle missing data and perform aggregations.

Additional Best Practices

• Always inspect your data with df.info() and df.describe() to understand its structure and summary statistics. • Use vectorized operations for performance gains and avoid chained indexing to prevent unexpected behavior. • When merging data, ensure that the keys are correctly aligned to avoid mismatches.

Real-World Applications

Pandas is widely used in finance, healthcare, marketing, and scientific research. Its capabilities range from simple data cleaning to complex statistical analysis and machine learning data preparation.

References

Pandas Documentation: https://pandas.pydata.org/docs/
ReportLab Documentation: https://www.reportlab.com/documentation/

Creating a Series from a List

```
import pandas as pd
data = [1, 2, 3, 4, 5]
series = pd.Series(data)
print(series)
print(type(series))

# Expected Output:
# 0    1
# 1    2
# 2    3
# 3    4
# 4    5
# dtype: int64
# <class 'pandas.core.series.Series'>
```

Creating a Series from a Dictionary

```
import pandas as pd
data = {'a': 1, 'b': 2, 'c': 3}
series_dict = pd.Series(data)
print(series_dict)

# Expected Output:
# a    1
# b    2
# c    3
# dtype: int64
```

Creating a Series with a Custom Index

```
import pandas as pd
data = [10, 20, 30]
idx = ['a', 'b', 'c']
series_custom = pd.Series(data, index=idx)
print(series_custom)
```

```
# Expected Output:
# a    10
# b    20
# c    30
# dtype: int64
```

Creating a DataFrame from a Dictionary of Lists

```
import pandas as pd
data = {
    'Name': ['Krish', 'Sam', 'Saksham'],
    'Age': [34, 18, 20],
    'City': ['Bangalore', 'Jaunpur', 'New Delhi']
}
df = pd.DataFrame(data)
print(df)
print(type(df))

# Expected Output:
#       Name  Age       City
# 0    Krish   34  Bangalore
# 1      Sam   18    Jaunpur
# 2  Saksham   20  New Delhi
# <class 'pandas.core.frame.DataFrame'>
```

Creating a DataFrame from a List of Dictionaries

```
import pandas as pd
data = [
    {'Name': 'Saksham', 'Age': 18, 'City': 'Jaunpur'},
    {'Name': 'Sam', 'Age': 20, 'City': 'New Delhi'},
    {'Name': 'Krish', 'Age': 34, 'City': 'Bangalore'},
    {'Name': 'John', 'Age': 29, 'City': 'New York'}
]
df = pd.DataFrame(data)
print(df)
print(type(df))

# Expected Output:
#      Name  Age       City
# 0  Saksham   18    Jaunpur
# 1      Sam   20  New Delhi
# 2    Krish   34  Bangalore
# 3     John   29   New York
# <class 'pandas.core.frame.DataFrame'>
```

Reading CSV Data and Displaying Rows

```
import pandas as pd
df = pd.read_csv('sales_data.csv')
print(df.head(5))
print(df.tail(5))

# Expected Output:
# (First 5 rows and last 5 rows of the CSV file will be displayed)
```

Accessing Data from a DataFrame

```
# Assuming df is already defined
```

```
print(df['Name'])      # Accessing the 'Name' column (returns a Series)
print(df.loc[0])       # Accessing the first row (returns a Series)
print(df.loc[0, 'Name'])  # Accessing the element at row 0 and column 'Name'
print(df.iloc[0])      # Accessing the first row using integer-location based indexing

# Expected Output: (Displays corresponding rows/elements from the DataFrame)
```

Data Manipulation: Adding and Removing Columns, Updating Values, Dropping Rows

```
# Adding a new column 'Salary'
df['Salary'] = [50000, 60000, 70000]
print(df)

# Removing the 'Salary' column
df.drop('Salary', axis=1, inplace=True)
print(df)

# Incrementing the 'Age' column by 1
df['Age'] = df['Age'] + 1
print(df)

# Dropping the first row of the DataFrame
df.drop(0, inplace=True)
print(df)

# Expected Output:
# (Displays the DataFrame after each manipulation step)
```

Displaying DataFrame Information and Summary Statistics

```
# Display the data types of each column
print("Data types:\n", df.dtypes)

# Display summary statistics of the DataFrame
print("Statistical summary:\n", df.describe())

# Expected Output:
# Data types of each column and a statistical summary including count, mean, std, min, max, etc.
```