Pandas Basics - Teaching Guide for College Students

# 1. Introduction to Pandas

Pandas is an open-source Python library used for data analysis and manipulation. It provides data structures like Series (1D) and DataFrame (2D), which are powerful tools for handling structured data.

# 2. Why Use Pandas?

- Easy handling of missing data  
- Powerful group by functionality  
- Easy data filtering and selection  
- Support for various file formats (CSV, Excel, JSON, etc.)  
- Data alignment and integrated handling of time series data

# 3. Creating Series and DataFrame

Example: Creating a Series

import pandas as pd  
  
data = [10, 20, 30, 40]  
s = pd.Series(data)  
print(s)

output:

0 10

1 20

2 30

3 40

dtype: int64

Example: Creating a DataFrame

import pandas as pd  
  
data = {  
  "calories": [420, 380, 390],  
  "duration": [50, 40, 45]  
}  
  
#load data into a DataFrame object:  
df = pd.DataFrame(data)  
  
print(df)

Result:

calories duration

0 420 50

1 380 40

2 390 45

**Locate Row**

As you can see from the result above, the DataFrame is like a table with rows and columns.

Pandas use the loc attribute to return one or more specified row(s)

Example

Return row 0:

#refer to the row index:  
print(df.loc[0])

Example

**Return row 0 and 1:**

#use a list of indexes:  
print(df.loc[[0, 1]])

Result

calories duration

0 420 50

1 380 40

**Named Indexes**

With the index argument, you can name your own indexes.

Example

Add a list of names to give each row a name:

import pandas as pd  
  
data = {  
  "calories": [420, 380, 390],  
  "duration": [50, 40, 45]  
}  
  
df = pd.DataFrame(data, index = ["day1", "day2", "day3"])  
  
print(df)

Result

calories duration

day1 420 50

day2 380 40

day3 390 45

**Locate Named Indexes**

Use the named index in the loc attribute to return the specified row(s).

Example

Return "day2":

#refer to the named index:  
print(df.loc["day2"])

**Display By Column name**

print(df['calories'])

**Display By Column name without index**

print(df['calories'].to\_string(index=False))

# 5. Basic Operations

# Accessing columns  
print(df['Name'])  
  
# Basic statistics  
print(df.describe())  
  
# Filtering data

print(df[df['duration'] > 40])

# 6. Modifying Data

# Adding a new column

df['Vitamin'] = [50, 600, 70]

print(df)

# Deleting a column

del df['duration']

print(df)

# 7. Tips for Students

- Always look at the data using `head()` and `info()`.  
- Use `describe()` to understand summary statistics.  
- Filter data using conditions in square brackets.  
- Pandas is best learned by practicing with real-world datasets.