Getting Started with Pandas Cheatsheet

Visit KDnuggets.com for more cheatsheets and additional Data Science, Machine Learning, AI & Analytics learning resources

pandas is open-source and the most popular Python tool for data wrangling and analytics. It is fast, intuitive, and can handle multiple data formats such as CSV, Excel, JSON, HTML, and SQL.

Creating DataFrames

Change the layout, rename the column names, append rows and Create a pandas dataframe object by specifying the columns name and index.

From dictionary:

```
df = pd.DataFrame( {"A" : [1, 4, 7], "B" : [2, 5, 8],
    "C" : [3, 6, 9]}, index=[101, 102, 103])
```

From multi-dimensional list:

```
df = pd.DataFrame( [[1, 2, 3], [4, 5, 6],[7, 8, 9]],
   index=[101, 102, 103], columns=['A', 'B', 'C'])
```

	A	B	C
101	1	2	3
102	4	5	6
103	7	8	9

Importing Data

Import the data from text, Excel, website, database, or nested JSON file.

```
pd.read_csv(file_location)     # import tabular CSV file

pd.read_table(file_location)   # import delimited text file

pd.read_excel(file_location)   # import Excel file

# connect and extract the data from SQL database
pd.read_sql(query, connection_object)

# import from JSON string, file, URL
pd.read_json(json_string)

# extract tables from HTML file, URL
pd.read_html(url)
```

Exporting Data

These commands are commonly used to export files in various formats but you can also export the dataframe into binary Feather, HDF5, BigQuery table, and Pickle file.

```
df.to_csv(filename)        # export CSV tabular file

df.to_excel(filename)      # export Excel file

# apply modifications to SQL database
df.to_sql(table_name, connection_object)

df.to_json(filename)       # export JSON format file
```

Inspecting Data

Understand the data and the distribution by using these commands.

```
# view first n rows or use df.tail(n) for last n rows
df.head(n)

# display and ordered first n values or use df.nsmallest(n,
'value') for ordered last n rows
df.nlargest(n, 'value')

df.sample(n=10)  # randomly select and display n rows

Df.shape         # view number of rows and columns

# view the index, datatype and memory information
df.info()

# view statistical summary of numerical columns
df.describe()

# view unique values and counts of the city column
df.city.value_counts()
```

Subsetting

Select a single row or column and multiple rows or columns using these commands.

```
df['sale']                     # select a single column

df[['sale', 'profit']]         # select two selected columns

df.iloc[10 : 20]               # select rows from 10 to 20

# select all rows with columns at position 2, 4, and 5
df.iloc[ : , [2, 4, 5]]

# select all rows with columns from sale to profit
df.loc[ : , 'sale' : 'profit']

# filter the dataframe using logical condition and select sale
and profit columns
df.loc[df['sale'] > 10, ['sale', 'profit']]

df.iat[1, 2]       # select a single value using positioning

df.at[4, 'sale']   # select single value using label
```

Querying

Filter out the rows using logical conditions. The `query()` returns a boolean for filtering rows.

```
df.query('sale > 20')  # filters rows using logical conditions

df.query('sale > 20 and profit < 30')   # combining conditions

# string logical condition
df.query('company.str.startswith("ab")', engine="python")
```

Reshaping Data

Change the layout, rename the column names, append rows and columns, and sort values and index.

```
pd.melt(df)                    # combine columns into rows

# convert rows into columns
df.pivot(columns='var', values='val')

pd.concat([df1,df2], axis = 0) # appending rows

pd.concat([df1,df2], axis = 1) # appending columns

# sort values by sale column from high to low
df.sort_values('sale', ascending=False)

df.sort_index()                # sort the index

df.reset_index()               # move the index to columns

# rename a column using dictionary
df.rename(columns = {'sale':'sales'})

# removing sales and profit columns from dataframe
df.drop(columns=['sales', 'profit'])
```

Abid Ali Awan, 2022