DATALEARNS24+7

Pandas Essentials

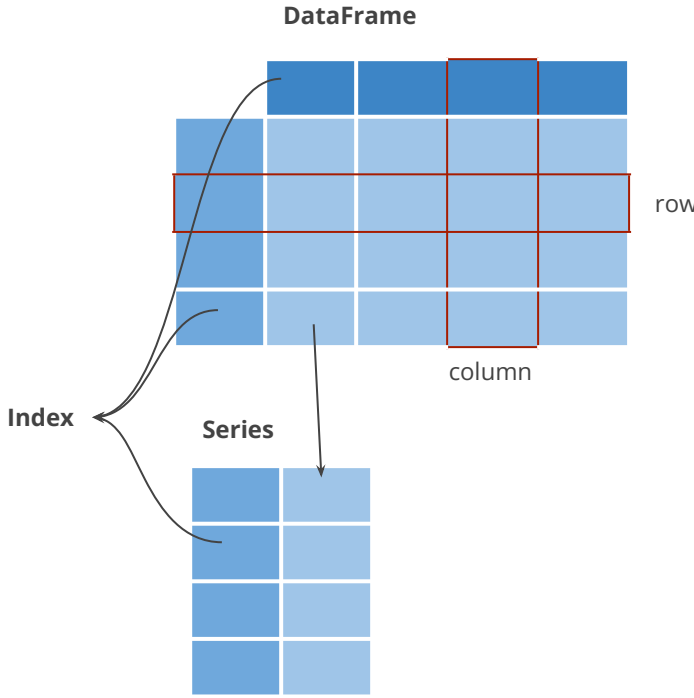
Data Science Fundamental

Chapter 01

Introduction

Apa Itu Pandas ?

- Library Python yang bersifat open-source, untuk analisis dan manipulasi data
- Kelebihan : **kemudahan penggunaan**, **fungsionalitas yang lengkap**, dan **kompatibilitas yang luas**
- Memiliki struktur data untuk memproses dan menganalisis data berbentuk tabular maupun time series, yaitu :
  - Series - data sekuensial satu dimensi
  - DataFrame - data berbentuk tabel dua dimensi
- Setiap kolom DataFrame adalah Series

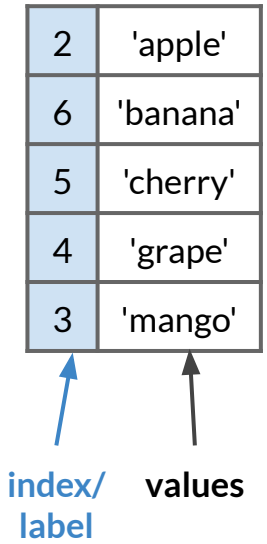


Chapter 02

Pandas Series

Pandas Series

- Seri adalah array 1 dimensi yang berlabel (indeks)
- Dapat menyimpan semua tipe data (number, string, list, dll.)
- Seri bersifat **homogen**; semua elemen harus dalam tipe data yang sama
- Seri bersifat **value-mutable** : nilai dapat diubah



Creating Series

Series dapat dibuat dengan cara:

- Membuat objek Series kosong
- Dibuat dari objek Python (List, numpy arrays, dll.)
- Loading dari file

```
pandas.Series(data=None, index=None, dtype=None,
              name=None, copy=False)
```

- Series index harus **hashable**
- Jika tidak ditentukan tipenya, **dtype** disimpulkan dari data yang di-load
- Tipe default elemen Series adalah objek Python

Create empty Series

```
s = pd.Series(name = "Empty Series")
```

Create from dictionary

```
d = {"apples": 10, "bananas": 20, "cherry": 50}
s = pd.Series(d, name="Fruit Series")
```

Loading from file

```
filename = 'filename.txt'
fs = pd.read_csv(filename, squeeze=True,
                 index_col=0)
```

Series Operations

- Series berperilaku seperti **array NumPy**
- Operasi pada array numpy dapat dilakukan juga pada Series, seperti **sum**, **product**, **min**, **max**, **avg**, dll.
- Operasi antar Series disejajarkan secara otomatis menggunakan **indeks**

Operation on Series

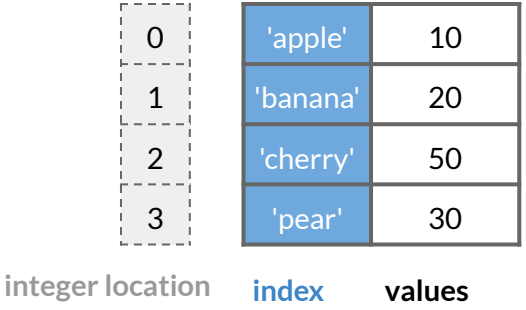
```
fs['cherry']+=5              #updating an item
print('grapes' in fs)       #check membership
print(fs/10)                 #math operation
print(max(fs))               #max, min, avg
```

Operation between Series

```
qty = pd.Series({"apple":10, "banana":20,
                 "cherry":50})
price = pd.Series({"apple":100, "banana":120,
                   "cherry":500})
qty*price
```

Accessing Series : Index vs Location

- 3 metode untuk mengakses series :
  - **`Series.loc[i]`** untuk akses dengan ***index/key/label***
  - **`Series.iloc[i]`** untuk akses dengan ***integer location***
  - **`Series[i]`** untuk akses dengan ***index/key/label*** maupun integer location




```
fs.loc["apple"]
```

```
fs.iloc[0]
```

```
fs.loc["banana":"pear"]
```

```
banana    20
cherry    50
```

```
fs.iloc[1:2]
```

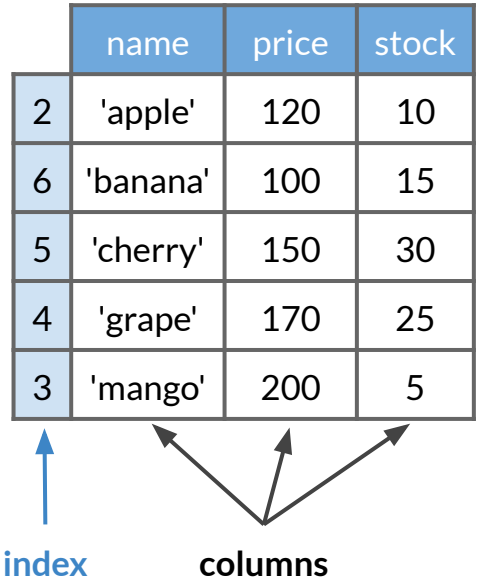
```
banana    20
```

Chapter 03

Pandas DataFrame

Pandas DataFrame

- DataFrame adalah struktur data 2 dimensi yg berlabel
- Dataframe dapat mengandung kolom-kolom dengan tipe data yang berbeda, namun elemen dalam sebuah kolom harus homogen (bertipe data sama).
- DataFrame bersifat value-mutable : nilainya dapat diubah
- DataFrame bersifat size-mutable : ukuran (jumlah baris dan jumlah kolom) dapat diubah



Creating DataFrame

- Membuat empty DataFrame
- Dibuat dari objects lain : numPy array, List, Dictionary, or Series
- Loading dari file : text, csv, JSON, Excel, HDF5

Create empty DataFrame

```
s = pd.DataFrame()
```

Create from Dictionary

```
f = {'name':["apple","banana","cherry","pear"],
     'quantity':[10, 20, 50, 30],
     'price':[1000,500,750,900]}
pd.DataFrame(f)
```

Create from Series

```
s1 = pd.Series({"apple":10,"banana":20,
             "cherry":50,"pear":30})
s2 = pd.Series({"apple":1000,"banana":500,
             "cherry":750,"pear":900})
pd.DataFrame({"price":s1,"quantity":s2})
```

Loading from file

```
filename = 'filename.csv'
df = pd.read_csv(filename)
```

Quick Check & Descriptive Statistics

- Fungsi dan atribut untuk melakukan quick check : **`head(), tail(), info(), shape, columns`**
- Fungsi-fungsi descriptive statistics : **`count`**, **`sum`**, **`mean`**, **`median`**, **`mode`**, **`max`**, **`min`**, etc. Semua fungsi tersebut secara default meng-exclude `NA` / `Null`
- Fungsi summary statistik untuk Series maupun DataFrame : **`describe()`**
- Fungsi statistik deskriptif selengkapnya : https://pandas.pydata.org/docs/user_guide/basics.html#descriptive-statistics

Some useful commands for quick check

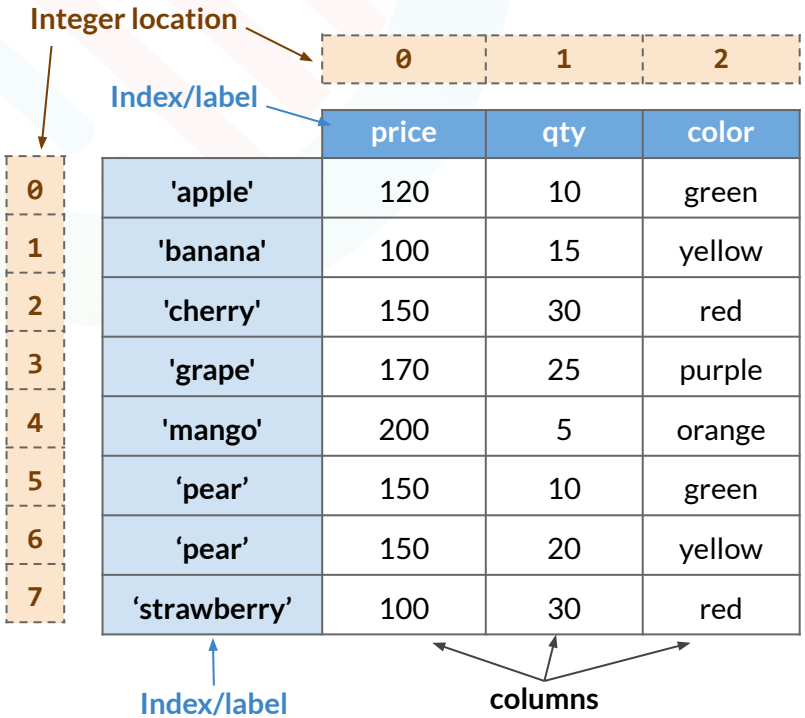
```
df.head()
df.tail()
df.info()
df.shape
df.columns
```

Descriptive Statistics

```
df.sum(col1)
df.max(col2)
df.describe()
```

Accessing DataFrame

Akses DataFrame dapat menggunakan **`df.loc[]`**, **`df.iloc[]`**, serta **`df[]`**



`df.loc[<row indexer>, <column indexer>]`

- Single label, 'apple'
- Slice of labels, 'apple':'pear'
- List of labels, ['apple','cherry']
- Boolean array, df['price']<10

- Single column, 'price'
- Slice of columns['qty':]
- List of columns ['qty','price']
- Boolean array

`df.iloc[<row location>, <column location>]`

- Single integer, 2
- Slice of rows, 2:9
- Integer list [0,1,2...]
- Boolean array

- Single integer, 3
- Slice of columns, 3:9
- Integer list [0,1,2...]
- Boolean array

***Catatan** : perhatikan tipe data yang dikembalikan dari setiap metode akses!*

Select By Conditions

- Kita dapat memilih baris berdasarkan kondisi berdasarkan nilai kolom menggunakan **`df[]`** atau **`df.loc[]`**
- Gunakan operator boolean **`&`** atau **`|`** untuk menggabungkan beberapa kondisi
- Kita juga dapat memilih berdasarkan daftar nilai dengan menggunakan fungsi **`isin(list of value)`**.

Select by condition

```
df[df['quantity'] > 20]
df.loc[df['quantity'] > 20]
```

Select by multiple conditions

```
df[(df1.color == 'green') | (df.price > 150)]
df.loc[(df.color == 'green') | (df.price > 150)]
```

Select based on list

```
df.loc[df['color'].isin(['green','yellow'])]
df.loc[df['color'].isin(['green','yellow']) \
       | (df['quantity'] < 10)]
```

Lab 02

DataFrame Basics

DataFrame Basics

We will use AirBnB data

Skenario untuk pengolahan dataframe : loading, checking, and cleansing.

- Load
- Quick check
- Akses dengan berbagai metode

Pertanyaan untuk latihan :

1. Berapa row dan kolom?
2. Apa saja kolomnya?
3. Data dari baris ke x1 sampai x2
4. Data dari kolom ke y1 sampai y2
5. Berapa nilai rata-rata, maksimum dan minimum untuk kolom2 tertentu?

Working With Missing Values

- Default untuk missing value atau Null di Pandas adalah **NaN**, tetapi null dapat juga berupa **None** (null Python)
- Null untuk datetime di Pandas adalah **NaT**
- Untuk memeriksa apakah suatu elemen null, gunakan **isna(), isnull(), notnull(),** or **notna()**
- Untuk mengisi value null, gunakan **fillna()**
- Untuk menghapus data yang mengandung missing value, gunakan **dropna()**

Find Missing Values

```
df.isna()
df.notna()
df.col1.isna()
df[df.col1.isna()] #select rows where col1 is null
```

Replace Missing Values

```
df.fillna(value='NA')     #with default value
df.fillna(method='bfill') #with previous row value
```

Drop rows with missing value

```
df.dropna()
df.dropna(how='all')
df.dropna(thresh=2)
```

Edit DataFrame : Add & Delete Rows

- Untuk menambahkan baris baru, kita bisa langsung meng-assign sebuah dictionary ke lokasi indeks yang baru, atau
- Menggunakan fungsi **`pandas.concat(list of dataframes)`**
- Assignment ke indeks baru mengupdate DataFrame langsung (in place), sedangkan concat mengembalikan dataframe baru
- Menghapus baris dilakukan menggunakan fungsi **`df.drop()`**

Adding new row

```
dict = {'name':'mango','qty':10,'color':'orange'}
df.loc[6] = dict
```

Adding new rows with concat

```
df.concat([df1, df2, df3])
```

Drop rows

```
#drop rows by index
print(df.drop([0,3]))

#drop rows by conditions
print(df.drop(df[df['col1'] == 10].index))
```

Edit DataFrame : Add & Delete Columns

- Untuk menambahkan kolom baru, assign list or series ke kolom yang baru menggunakan **`df[‘nama kolom baru’]`**, atau menggunakan **`insert()`**
- Penambahan kolom dilakukan secara inplace (mengubah DataFrame aslinya)
- Menghapus kolom dilakukan menggunakan fungsi **`df.drop()`** dengan parameter *axis = 1* atau menggunakan parameter *columns*

Adding new column

```
df['new_col'] = [10,11,12,13]
df.loc[:,'new_col'] = [10,11,12,13]

df.insert(2, 'new_col', [10,11,12,13])
```

Drop columns

```
df.drop(['col1', 'col2'], axis=1)
df.drop(columns=['col1', 'col2'])
```

Edit DataFrame : Rename Columns

- Mengubah nama kolom dilakukan menggunakan fungsi **`df.rename()`**
- Kita dapat mengubah satu atau beberapa nama kolom sekaligus dengan meng-assign *List* atau *Series* ke dalam variabel **`df.columns`**

Rename column

```
df.rename(columns={"oldname1": "newname1",
                   "oldname2": "newname2"})
```

Assign List object into df.columns

```
df.columns=["colname1","colname2"]
```

Duplicates & Unique Value

- Secara default, **`DataFrame.duplicated()`** mempertimbangkan semua kolom untuk cek duplikasi
- Untuk melakukan cek duplikasi berdasar satu atau beberapa kolom saja, gunakan parameter *subset*
- Secara default, semua baris duplikat akan ditandai sebagai **`True`**, kecuali baris pertama. Untuk mengubahnya, gunakan parameter *keep*
- Untuk menghapus duplikat, gunakan fungsi **`drop_duplicates()`**
- Untuk menampilkan nilai unik suatu kolom, gunakan **`unique()`**. Untuk mendapatkan jumlah nilai unik, gunakan **`nunique()`**

Find duplicates

```
df = pd.DataFrame({
       'name': ['apple','apple','pear','pear'],
       'color': ['red','green','green','yellow'],
       'qty': [4, 4, 5, 15]}

df.duplicated()
df.duplicated(subset=['name','qty'])
```

Drop duplicates

```
df.drop_duplicates(subset=['name','qty'])
df.drop_duplicates(subset=['name','color'],
                   keep='last')
```

Show unique value

```
df.color.unique()
df[‘color’].drop_duplicates()
```

Lab 03

Handling Missing & Duplicate Values

DataFrame Basic

Skenario untuk pengolahan dataframe : loading, checking, and cleansing

- Check missing value
- Check unique value
- Clean data as needed

Pertanyaan untuk latihan :

1. Berapa data yang null?
2. Adakah data duplikat (row yang persis sama)
3. Berapa jumlah unique value untuk kolom x, y, z?
4. Apa yang perlu dilakukan untuk kasus2 tersebut?

Working With Text Data

- Fungsi pengolahan string sangat bermanfaat terutama untuk cleaning dan transformasi
- Untuk mengakses sebuah kolom sebagai string, gunakan **`Series.str`**
- Misalnya untuk mengubah nilai string sebuah kolom menjadi lowercase : **`df[column_name].str.lower()`**
- Fungsi string beserta contohnya dapat dilihat di https://pandas.pydata.org/docs/user_guide/text.html#method-summary

Working with text

```
df['name'].str.lower()
df['name'].str.lower().str.strip()
```


```
df['name'].str.lower()
df['name'].str.lower().str.strip()
```

```
df['name'].str.lower()
df['name'].str.lower().str.strip()
```

Menggabungkan DataFrame

- Menggabungkan 2 DataFrame by kolom (join) dilakukan dengan fungsi **`pd.merge()`**
- Secara default, join yang dilakukan adalah inner join. Untuk memilih jenis join yang lain, gunakan parameter *how* : ***left, right, outer***

Joining DataFrames

```
df = pd.DataFrame({
                'name':["apples", "bananas", "pears"],
                'qty':[10, 20, 30]
        })
colors = pd.DataFrame({
                'name':["apples", "bananas", "pears"],
                'color':['red', 'yellow', 'green']
        })

pd.merge(df, colors, on='name', how='inner')
```

Lab 03

More Cleansing and Resolving

Cleansing & Resolve Data

Skenario untuk pengolahan dataframe

- Load reference
- Quick check, rename column name
- Join
- Check result and handling unresolved data

Pertanyaan untuk latihan :

1. Berapa baris hasil join data a dan data b?
2. Ada berapa yang tidak ketemu hasil combine-nya?
3. Apa value kolom col1 untuk row tertentu? (hasil combine)

Chapter 04

Analyze & Visualize

Aggregations

- Fungsi agregasi dalam pandas adalah **`DataFrame.aggregate()`** or **`DataFrame.agg()`**
- Agregasi menggunakan parameter satu fungsi menghasilkan **`Series`**
- Agregasi menggunakan parameter list of function(s) menghasilkan **`DataFrame`**

Passing a single function

```
df.agg(np.sum)
df.agg("sum")
```

(Multiple) functions as a list

```
df.agg(["sum","mean"])
df.agg(["sum"])
```

Grouping A DataFrame

- Pengelompokan data dilakukan dengan fungsi **`DataFrame.groupby()`**
- Kita dapat menggunakan parameter berupa single column/index atau multiple columns/indexes dalam list
- Fungsi groupby mengembalikan ***GroupBy object***, yaitu sekumpulan DataFrame yang dikelompokkan berdasar kriteria tertentu
- Objek GroupBy dapat diiterasi

Group by a single column

```
df.groupby("col1")
df.groupby(["col1"])
```

Multiple columns as a list

```
df.groupby(["col1","col2"])
```

Iterate a GroupBy object

```
for name, group in grouped:
   print(name)
   print(group)
```

Apply A Function on GroupBy Object

- Secara default, kolom yang digunakan untuk pengelompokan akan dijadikan indeks pada DataFrame yang dihasilkan
- Pengelompokan beberapa kolom akan menghasilkan DataFrame multi-indeks
- Untuk mengubahnya, kita dapat mengatur parameter as_index pada fungsi groupby menjadi **False**
- Kita juga bisa menggunakan **`DataFrame.reset_index()`** pada DataFrame hasil agregasi

Group by multiple columns

```
grouped = df.groupby(["col1", "col2"])
grouped.sum()
```

To get a single indexed DataFrame

```
grouped = df.groupby(["col1", "col2"],
                     as_index = False)
grouped.agg("sum")
```

Using reset_index

```
grouped = df.groupby(["col1", "col2"])
grouped.agg("sum").reset_index()
```

Named Result

- Secara default, nama kolom agregasi sama dengan nama fungsi, misalnya sum, mean, dll.
- Untuk mengganti nama kolom hasil, panggil fungsi tersebut sebagai tupel dan assign hasil fungsi tersebut ke nama yang diinginkan

```
DataFrame.agg(
     result_name=
          (“aggregated-column”,”function-name”)
     )
```

Set name to the result

```
grouped = df.groupby(["name"])
grouped.agg(
      min_qty=('qty','min'),
      max_price=('price','max'),
      mean_price=('price','mean')
   )
```

Chart Visualization

- Pandas menyediakan fungsi ‘pembungkus’ untuk ***matplotlib***
- Gunakan parameter ***kind*** dengan fungsi **`df.plot`**, atau gunakan fungsi **`df.plot.<kind>`**
  - *‘bar’* or *‘barh’* : bar plots
  - *‘hist’* : histogram
  - *‘box’* : boxplot
  - *‘kde’* atau *‘density’* : density plots
  - *‘area’* : area plots
  - *‘scatter’* : scatter plots
  - *‘hexbin’* : hexagonal bin plots
  - *‘pie’* : pie plots
- Pandas plot menggunakan index sebagai default ***x*** value

Plot using DataFrame

```
df.plot.bar()
df.plot(x="qty", y="price", kind="scatter")
```

Plot using Series

```
df['color'].value_counts().plot.pie()
```

Plot using Seaborn

```
sns.countplot(df['color'], palette="plasma")
```

Visualization Packages

- Untuk mendapatkan lebih banyak pilihan formatting, kita dapat menggunakan matplotlib, atau paket visualisasi lainnya seperti Seaborn, Bokeh, dll.
- More on Pandas plot : https://pandas.pydata.org/docs/user_guide/visualization.html
- More on matplotlib : https://matplotlib.org/stable/tutorials/index.html
- More on Seaborn : https://seaborn.pydata.org/tutorial.html

Lab 05

Analyze & Visualize

Aggregate & Visualize

- Grouping data
- Agregat
- Visualisasi

Pertanyaan untuk latihan :

- Ada berapa neighbourhood group?
- Ada berapa host dalam tiap neighbourhood group?
- Ada berapa room type? Berapa jumlah host untuk masing2 room type dalam tiap2 neighbourhood group?

Chapter 07

Input/Output Handling

Pandas I/O

- The pandas I/O API is a set of top level reader & writers functions, in the form of **`pandas.read_`*`file format`*`()`** and **`pandas.to_`*`file format`*`()`**
- Some of the popular readers functions are **`pandas.read_csv()`**, **`pandas.read_json()`**, **`pandas.read_sql()`**, and **`pandas.read_pickles()`**
- For complete list of available reader & writers see https://pandas.pydata.org/docs/user_guide/io.html

CSV Format

- Some aspects to be considered :
  - Header, separator
  - Data types
  - Missing values, ‘bad’ lines
  - Quotes, escape characters, encoding
- For complete guide see https://pandas.pydata.org/docs/user_guide/io.html#csv-text-files

Read csv file

```
data = pd.read_csv("filename.csv, sep="|")
```

Read csv with options

```
data = pd.read_csv("filename.csv,
    sep="|",
    quotechar="'",
    dtype={"price": int})
```

...

```
df.to_csv('filename.csv', header=False,
index=False)
```

Reading JSON (JavaScript Object Notation)

- A text format for data interchange
- "self-describing" and easy to understand
- There are 5 format options:
  - ***split*** : `{index -> [index], columns -> [columns], data -> [values]}`
  - ***records*** : `[{column -> value}, … , {column -> value}]`
  - ***index*** : `{index -> {column -> value}}`
  - ***columns*** : `{column -> {index -> value}}`
  - ***values*** : just the values array
  - ***table*** : adhering to the JSON Table Schema

Read json

```
df = pd.read_json("filename.json")
```

Read JSON with options

```
df = pd.read_json("filename.json",
 orient="records",
 dtype={"price": "float32", "qty": "int64"})
```

Write to JSON

```
df.to_json('filename.json', orient='columns')
```

Database Connection

- Database connection is done using **SQLAlchemy** package.
- **Note** : all opened db connections must be closed. Leaving a connection open may result in locking the database or other breaking behaviour.
- For more info, go to https://docs.sqlalchemy.org/en/14/tutorial/index.html

Import package

```
import sqlalchemy
```

Create engine

```
engine =
 sqlalchemy.create_engine(connection_string)
```

Read table/query

```
df = pandas.read_sql(tablename, engine)
df = pandas.read_sql(query, engine)
```

Close connection

```
engine.dispose()
```

References

- **Python Data Science Handbook** by *Jake VanderPlas*
  https://jakevdp.github.io/PythonDataScienceHandbook/
- **Python Documentation** https://docs.python.org/3/
- **Pandas Documentation** https://pandas.pydata.org

“THE REAL TRAINING BEGINS WHEN THE CLASS ENDS”

DR. Billy Kueek

Development Team:

M. Urfah
Sigit Prasetyo

About DATALearns247

DATALearns247 is part of PT Dua Empat Tujuh that has long experience in Big Data implementation and research in Indonesia. DATALearns247 focus on developing expertise through training and certification programs especially for Big Data and Artificial Intelligence.