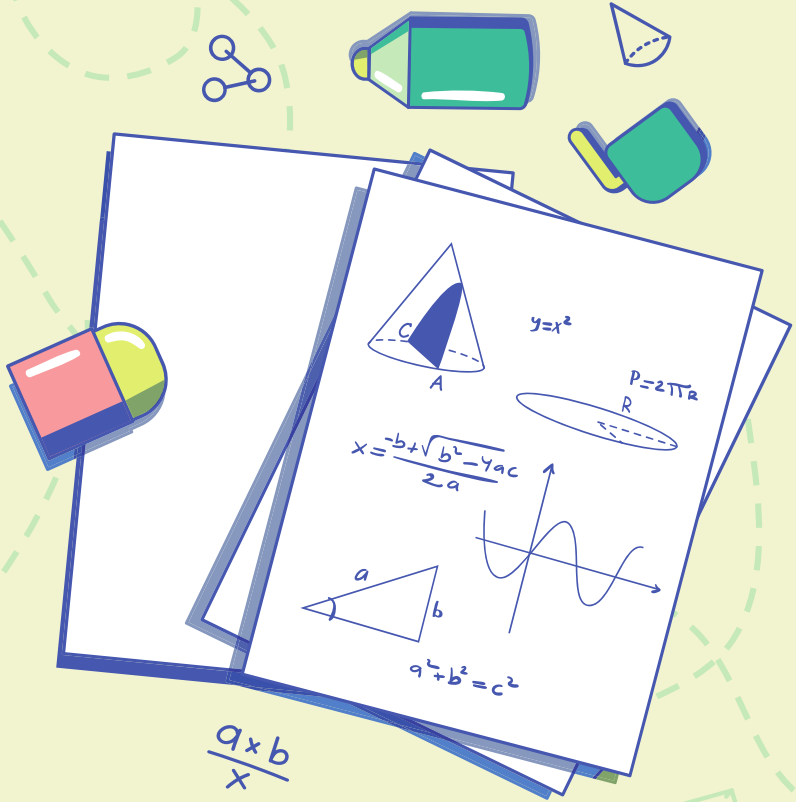
Learning Progres Review Week #7

By Optimistic team

DigitalSkola

Session 19

INTERMEDIATE DATAFRAME



Apa itu dataframe ?

Dataframe adalah Kumpulan atau suatu yang ditampilkan dalam kolom dan baris kemudian berisi kumpulan nilai.

Col1	Col2	Col3	Col4

Sorting in Dataframe

```
>>> df
  col1  col2  col3 col4
0    A     2     0    a
1    A     1     1    B
2    B     9     9    c
3  NaN     8     4    D
4    D     7     2    e
5    C     4     3    F
```

Sort → Mengurutkan
- Berdasarkan abjad
- Berdasarkan angka
- Dsb,

DataFrame.sort_values(by, axis=0,
ascending=True, inplace=False, kind='quicksort',
na_position='last', ignore_index=False, key=None)

Case 1 : Mengurutkan berdasarkan abjad di col1 – part 1

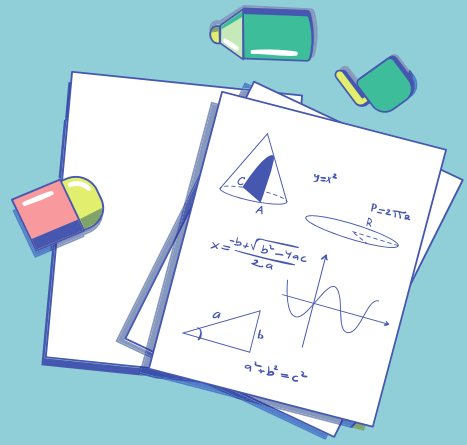
```
>>> df
  col1  col2  col3 col4
0    A     2     0    a
1    A     1     1    B
2    B     9     9    c
3  NaN     8     4    D
4    D     7     2    e
5    C     4     3    F
```

```
>>> df.sort_values(by=['col1'])
  col1  col2  col3 col4
0    A     2     0    a
1    A     1     1    B
2    B     9     9    c
5    C     4     3    F
4    D     7     2    e
3  NaN     8     4    D
```

Case 2 : Mengurutkan berdasarkan abjad di col1 – part 2

```
>>> df
  col1  col2  col3 col4
0    A     2     0    a
1    A     1     1    B
2    B     9     9    c
3  NaN     8     4    D
4    D     7     2    e
5    C     4     3    F
```

```
>>> df.sort_values(by='col1', ascending=False)
  col1  col2  col3 col4
4    D     7     2    e
5    C     4     3    F
2    B     9     9    c
0    A     2     0    a
1    A     1     1    B
3  NaN     8     4    D
```



Case 3 : Mengurutkan berdasarkan abjad di col1 – part 3

```
>>> df
  col1  col2  col3 col4
0    A     2     0    a
1    A     1     1    B
2    B     9     9    c
3  NaN     8     4    D
4    D     7     2    e
5    C     4     3    F
```

```
>>> df.sort_values(by='col1', ascending=False)
  col1  col2  col3 col4
4    D     7     2    e
5    C     4     3    F
2    B     9     9    c
0    A     2     0    a
1    A     1     1    B
3  NaN     8     4    D
```

Case 4 : Mengurutkan berdasarkan lebih dari 1 column

```
>>> df
  col1  col2  col3 col4
0    A     2     0    a
1    A     1     1    B
2    B     9     9    c
3  NaN     8     4    D
4    D     7     2    e
5    C     4     3    F
```

```
>>> df.sort_values(by=['col1', 'col2'])
  col1  col2  col3 col4
1    A     1     1    B
0    A     2     0    a
2    B     9     9    c
5    C     4     3    F
4    D     7     2    e
3  NaN     8     4    D
```

FILTERING DATAFRAME

Melakukan seleksi terhadap dataframe untuk mendapatkan hanya informasi yang dibutuhkan/diinginkan. Terdapat beberapa fungsi yaitu,
loc dan iloc.

Get some columns with filter using loc & iloc

- loc : label-based, perlu specify nama column & row
- iloc : integer index-based, perlu specify index dari column & row.

Contoh :

```
[19] df.iloc[[0,2],[1,3]]
```

	sepal_width	petal_width
0	3.5	0.2
2	3.2	0.2

Creating Additional Column

Memberikan column tambahan pada dataframe

- Column tambahan : value dari column yang lain
- Column tambahan : single value
- Column tambahan : other

```
df['sepal_length_v2'] = df['sepal_length'] * 100
df
```

	sepal_length	sepal_width	petal_length	petal_width	flower_class	sepal_length_v2
0	5.1	3.5	1.4	0.2	Iris-setosa	510.0
1	4.9	3.0	1.4	0.2	Iris-setosa	490.0
2	4.7	3.2	1.3	0.2	Iris-setosa	470.0
3	4.6	3.1	1.5	0.2	Iris-setosa	460.0
4	5.0	3.6	1.4	0.2	Iris-setosa	500.0

Grouping & Aggregate in Dataframe

Melakukan grouping dan aggregate, pada dasarnya bertujuan untuk mendapat summary atau rangkuman dari dataframe.

- Menggunakan groupby

```
df.groupby('flower_class').count()
```

flower_class	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	50	50	50	50
Iris-versicolor	50	50	50	50
Iris-virginica	50	50	50	50

- Menggunakan pivot_table

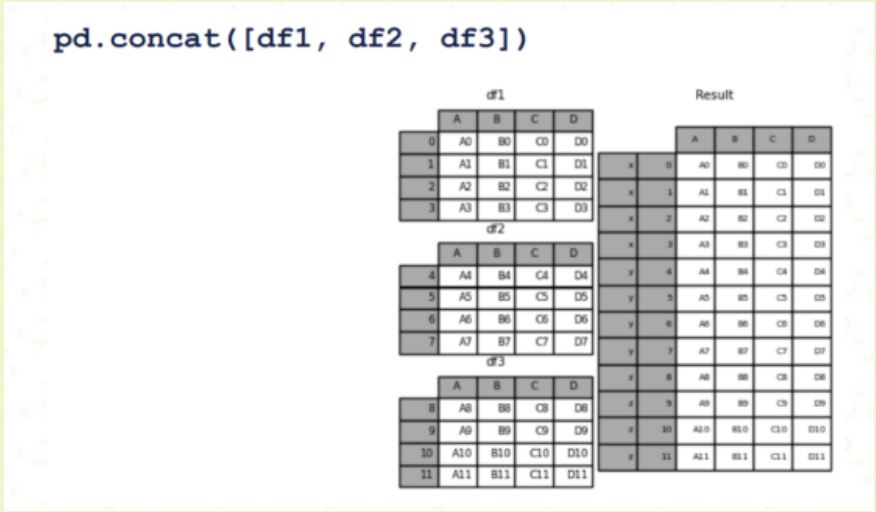
```
pd.pivot_table(df, values = 'sepal_length', index = 'flower_class', aggfunc = ['mean', 'median'])
```

	mean	median
	sepal_length	sepal_length
flower_class		
Iris-setosa	5.006	5.0
Iris-versicolor	5.936	5.9
Iris-virginica	6.588	6.5

COMBINING DATAFRAME

Melakukan proses kombinasi dataframe, tujuan utamanya untuk melengkapi atau memperkaya data yang ada.

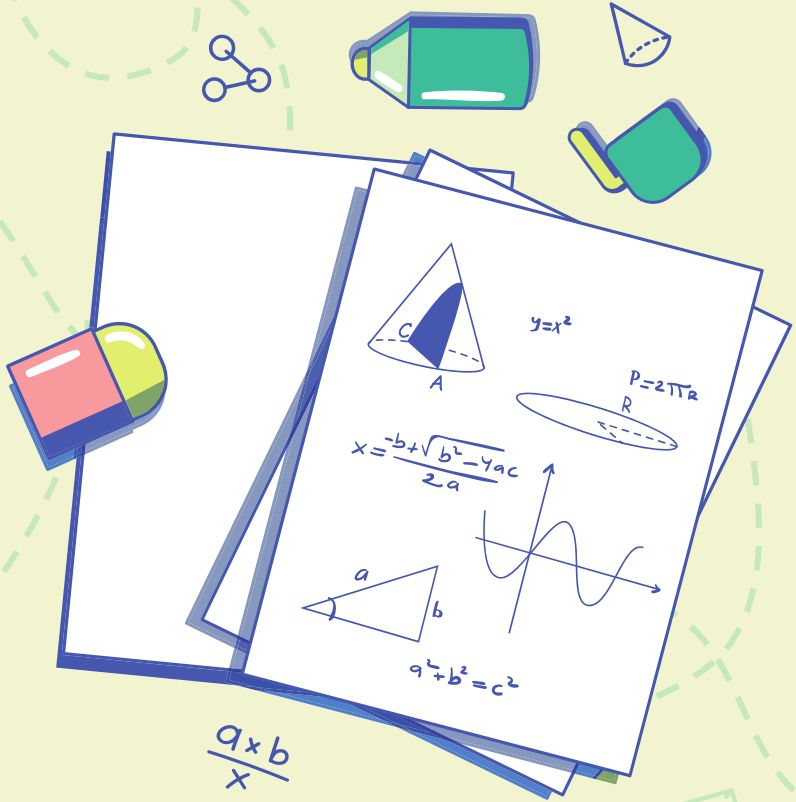
- Melakukan concatenate
- Melakukan merge
- Melakukan concatenate



DigitalSkola

Session 20

Advanced Dataframe



Join Dataframe

Join Dataframe adalah cara menggabungkan tabel berdasarkan konsep query. Ada 4 cara untuk melakukan combine pada suatu Dataframe, yaitu dengan Menggunakan Merge, Join, Concatenate.

Metode Join Dataframe

Left Join

Menggabungkankan 2 table dataframe dengan mengambil semua elemen primary key yang unik dari tabel pertama/kiri.

Right Join

Sama seperti left join, namun berkebalikan.

Inner Join

Menggabungkan dua table dataframe yang memiliki primary key yang sama, jika berbeda maka akan dihilangkan.

Outer Join

Menggabungkan semua data yang ada di dua tabel dataframe.

Merge dengan Pandas:

Dua dataframe

```
import pandas as pd

kiri = pd.DataFrame({
    'Id': [1,2,3],
    'Nama': ['Budi', 'Joko', 'Maya'],
    'Alamat' : ['Jakarta', 'Surabaya', 'Medan']
})

kanan = kiri = pd.DataFrame({
    'Id': [1,2,3],
    'Nama': ['Jane', 'mike', 'dave'],
    'Alamat' : ['Semarang', 'Yogya', 'Solo']
})
```

Merge dua dataframe

```
gabung = pd.merge(kiri, kanan, on='Id')
gabung
```

	Id	Nama_x	Alamat_x	Nama_y	Alamat_y
0	1	Jane	Semarang	Jane	Semarang
1	2	mike	Yogya	mike	Yogya
2	3	dave	Solo	dave	Solo

Join dengan Pandas:

Dua dataframe

```
import pandas as pd
karyawan = pd.DataFrame({
    'Nama': ['Budi', 'Joko', 'Maya'],
    'Alamat': ['Jakarta', 'Surabaya', 'Medan'],
    'Id' : [1,7,2]
})
pekerjaan = pd.DataFrame({
    'Id': [2,1,4,3],
    'Nama': ['Programmer', 'Data Engineer',
             'Manager', 'Web Developer'],
    'Gaji' : [5000, 4000, 3000, 6000]
})
print(karyawan)
print(pekerjaan)
```

Merge dua dataframe

```
gabung_join = karyawan.join(pekerjaan, how='left', rsuffix='_pekerjaan')
gabung_join
```

	Nama	Alamat	Id	Id_pekerjaan	Nama_pekerjaan	Gaji
0	Budi	Jakarta	1	2	Programmer	5000
1	Joko	Surabaya	7	1	Data Engineer	4000
2	Maya	Medan	2	4	Manager	3000

Concatenate Dataframe

Concatenate adalah menggabungkan Dataframe secara vertikal/ horizontal.

- Penggabungan secara horizontal jika axis = 1, yaitu penggabungan berdasarkan nama row yang sama dan menambah jumlah kolom.
- Penggabungan secara vertikal jika axis = 0, yaitu penggabungan berdasarkan nama kolom yang sama dan menambah jumlah baris.

Contoh Penggunaan Concatenate Dataframe

```
kiri = pd.DataFrame({
    'Nama': ['Budi', 'Joko', 'Maya'],
    'Alamat' : ['Jakarta', 'Surabay', 'Medan']
})

kanan = kiri = pd.DataFrame({
    'Nama': ['Jane', 'mike', 'dave'],
    'Alamat' : ['Semarang', 'Yogya', 'Solo']
})

gabung_concat = pd.concat([kiri, kanan], axis=0, ignore_index=True)
gabung_concat
```

	Nama	Alamat
0	Jane	Semarang
1	mike	Yogya
2	dave	Solo
3	Jane	Semarang
4	mike	Yogya
5	dave	Solo

Append Dataframe

Append adalah fungsi untuk menyatukan 2 data yang berbeda dengan ketentuan harus ada salah satu kolom yang sama,dan apabila tidak ada maka akan dibuatkan kolom baru.

Contoh Penggunaan Append Dataframe

```
kiri = pd.DataFrame({
    'Nama': ['Budi', 'Joko', 'Maya'],
    'Alamat' : ['Jakarta', 'Surabaya', 'Medan']
})

kanan = pd.DataFrame({
    'Nama': ['Jane', 'mike', 'dave'],
    'Alamat' : ['Semarang', 'Yogya', 'Solo']
})

gabung_append = kiri.append(kanan, ignore_index=True)
gabung_append
```

	Nama	Alamat
0	Budi	Jakarta
1	Joko	Surabaya
2	Maya	Medan
3	Jane	Semarang
4	mike	Yogya
5	dave	Solo

Indexing DataFrame

Indexing Dataframe dengan reset_index

Ketika kita menggabungkan antara dua table menggunakan fungsi join, concat, ataupun append indeks yang awalnya disusun untuk tiap tabel setelah digabungkan tidak rapi. Maka untuk merapikan index menggunakan reset_index. Contoh:

```
gabung_append = kiri.append(kanan)
gabung_append
```

	Nama	Alamat
0	Budi	Jakarta
1	Joko	Surabaya
2	Maya	Medan
0	Jane	Semarang
1	mike	Yogya
2	dave	Solo

```
new = gabung_append.reset_index(drop=True)
new
```

	Nama	Alamat
0	Budi	Jakarta
1	Joko	Surabay
2	Maya	Medan
3	Jane	Semarang
4	mike	Yogya
5	dave	Solo

Indexing Dataframe dengan set_index
Set_index digunakan untuk mengganti index dengan yang sudah ada. Dengan set_index juga bisa menggunakan tipe string.
Contoh:

```
kiri = pd.DataFrame({
    'ID' : ['001', '002', '003'],
    'Nama': ['Budi', 'Joko', 'Maya'],
    'Alamat' : ['Jakarta', 'Surabay', 'Medan']
})
```

```
kiri.set_index('ID', inplace=True)
kiri
```

ID	Nama	Alamat
001	Budi	Jakarta
002	Joko	Surabay
003	Maya	Medan

Pivoting Table

Membuat tabel pivot dari proses agregasi tabel yang dapat dilakukan menggunakan library pandas. Melakukan perubahan pada bentuk data dari yang tadinya panjang menjadi lebar.

Pivot

df

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

```
df.pivot(index='foo',
         columns='bar',
         values='baz')
```

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

Melting Table

Merupakan kebalikan dari proses pivot, mengubah suatu data dengan memutar dari yang tadinya berada di posisi column menjadi di posisi row. Proses melting melakukan perubahan pada bentuk data dari yang tadinya lebar menjadi panjang (menyingkat jumlah kolom).

	ID	col1	col2	col3
0	0	22	25	10
1	1	19	23	31
2	2	18	15	12
3	3	20	32	18
4	4	7	21	6

df.melt (id_vars="ID")

	ID	variable	value
0	0	col1	22
1	1	col1	19
2	2	col1	18
3	3	col1	20
4	4	col1	7
5	0	col2	25
6	1	col2	23
7	2	col2	15
8	3	col2	32
9	4	col2	21
10	0	col3	10
11	1	col3	31
12	2	col3	12

Lambda Function

Lambda Function adalah sebuah ekspresi untuk membuat satu baris fungsi.

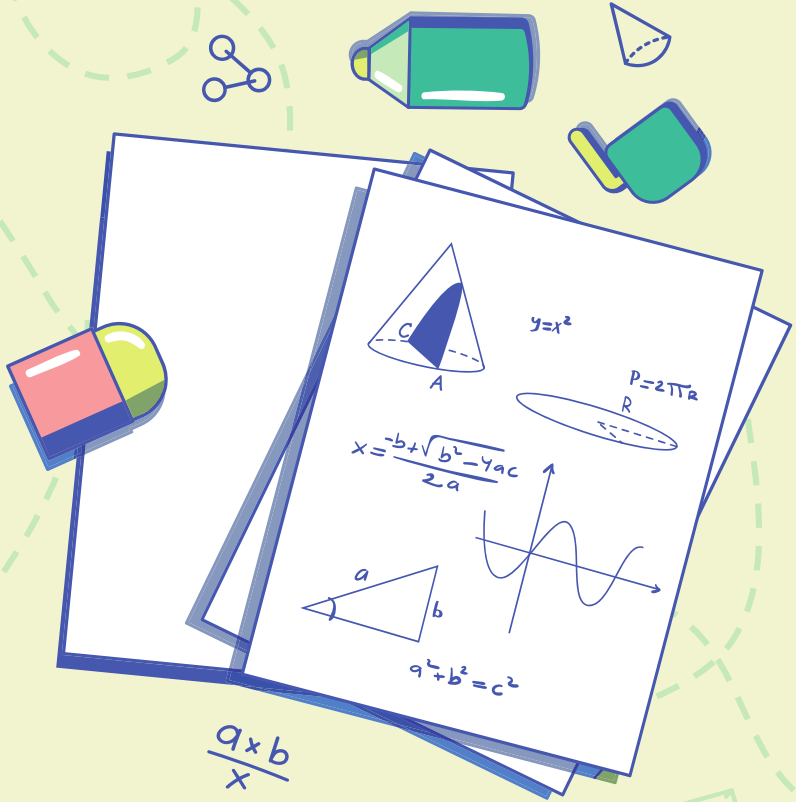
```
import pandas as pd
values_list = [[15, 2.5, 100], [20, 4.5, 50], [25, 5.2, 80],
               [45, 5.8, 48], [40, 6.3, 70], [41, 6.4, 90],
               [51, 2.3, 111]]

df = pd.DataFrame(values_list, columns=['Field_1', 'Field_2', 'Field_3'])
df = df.assign(Product=lambda x: (x['Field_1'] * x['Field_2'] * x['Field_3']))
df
```

	Field_1	Field_2	Field_3	Product
0	15	2.5	100	3750.0
1	20	4.5	50	4500.0
2	25	5.2	80	10400.0
3	45	5.8	48	12528.0
4	40	6.3	70	17640.0
5	41	6.4	90	23616.0
6	51	2.3	111	13020.3

DigitalSkola

Session 21

Application Programming Interface (API)

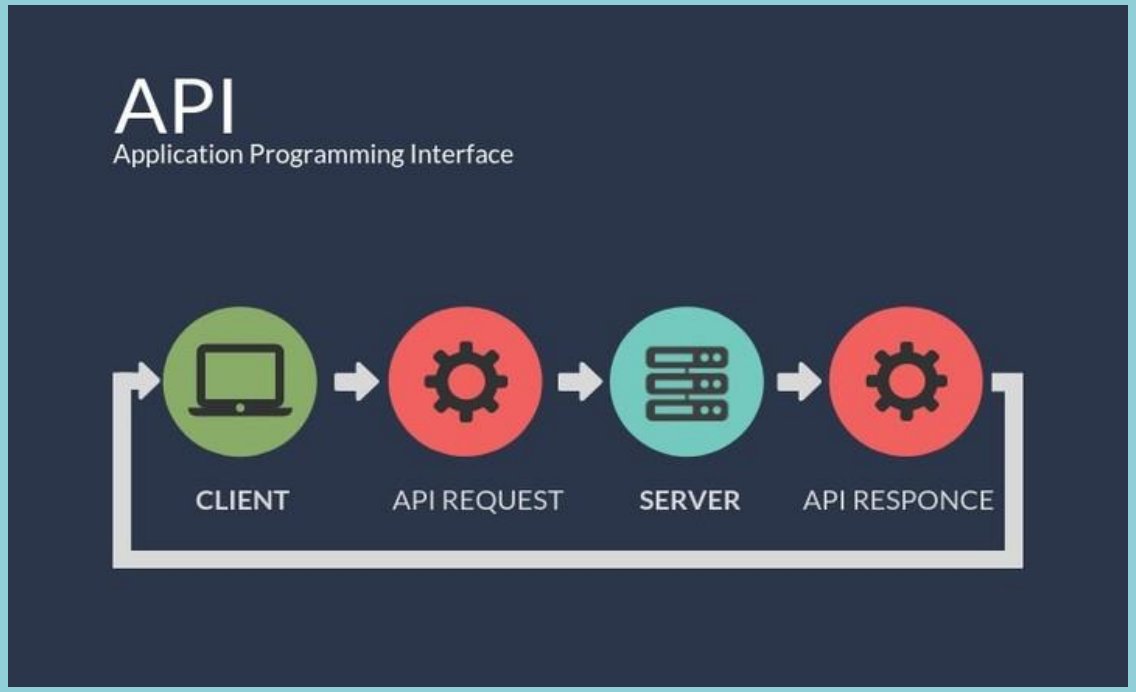


APPLICATION PROGRAMMING INTERFACE (API)

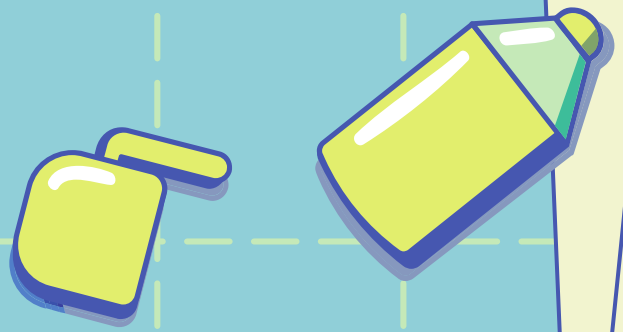
API adalah sebuah software sebagai perantara antara dua aplikasi atau beberapa bahasa pemrograman untuk saling berbicara satu sama lain.

Fungsinya sebagai pembawa pesan atau permintaan dari client kepada server, kemudian membawakan kembali jawaban yang diminta dari server kepada client.

API terdapat pada web, aplikasi mobile, dan desktop.



Jenis API



1. Public (dapat diakses secara umum)
   - Social Media
   - Website
   - Wikipedia
   - Aplikasi-aplikasi lain

2. Privat (dibuat oleh internal perusahaan dan hanya dapat diakses oleh pegawai saja).

Kelebihan API

- Otomatisasi

  Dengan API, komputer lebih efisien dalam mengatur pekerjaan serta update alur kerja dengan lebih cepat dan produktif.

- Integrasi

  Dengan API, konten dapat diakses dengan lebih mudah dan lebih efisien dalam berkomunikasi dengan beberapa aplikasi antar negara sehingga menjamin penyampaian informasi dengan lebih lancar.

- Data baru

  API menyediakan data dan informasi yang di-generate pada level yang dapat diatur agar dapat diakses oleh semua orang.

- Personalia

  Dengan API, konten dan layanan yang paling sering digunakan dapat disesuaikan serta memudahkan dalam membuat aplikasi yang kompleks.

FORMAT DATA API

JSON {...}

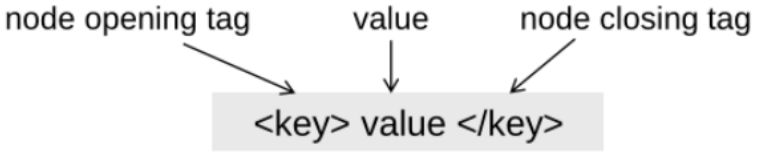
- Terbentuk dari *Javascript*
- *Simple* karena hanya terdiri dari *key & values* saja
- Kunci mewakili atribut tentang objek yang dideskripsikan
- Paling sering digunakan

{“key” : “value”}

Contoh :
{“toping” : “coklat”}

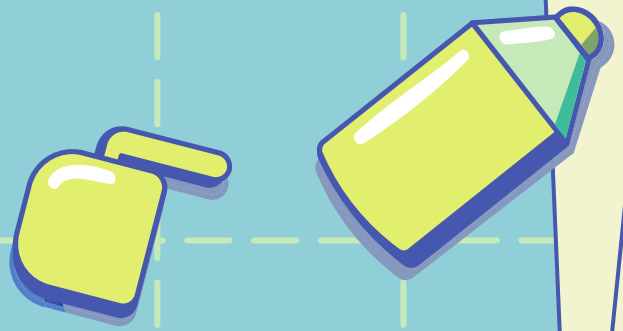
XML <...>

- Mudah dipakai
- Sangat powerful
- Terdapat beberapa node



API PADA PYTHON

Terdapat 3 packages yang dapat digunakan



1. Requests, untuk mengambil API yang sudah jadi.
2. Flask, untuk membuat API dari input sampai output.
3. Django, untuk membuat API dari input sampai output juga.

KOMPONEN API

1. Endpoint

Merupakan URL yang menggambarkan data dalam berinteraksi dengan client. Sistemnya mirip dengan bagaimana web-URL terikat ke halaman tertentu.

2. Data

Jika client menggunakan metode yang melibatkan perubahan data di REST API, maka harus disertakan payload data dengan permintaan yang mencakup semua data yang akan dibuat atau diubah.

3. Headers

Berisi metadata apapun yang perlu disertakan dengan permintaan, seperti token otentikasi, tipe konten yang harus dikembalikan, dan kebijakan caching apapun.

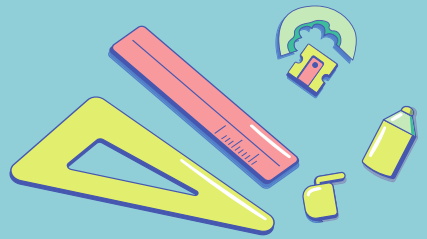
4. Method

Meneruskan bagaimana client berinterkasi dengan sumber daya yang disediakan. REST API dapat menyediakan metode seperti create menggunakan POST, read menggunakan GET, update menggunakan PUT, dan delete menggunakan DELETE.

STATUS API

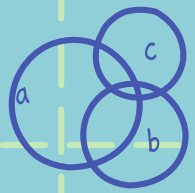
Kode status respons HTTP menunjukkan apakah permintaan HTTP tertentu telah berhasil diselesaikan atau belum. Tanggapan dikelompokkan berdasarkan angka yang dihasilkan, seperti:

1. Respons informasional (‘100’ - ‘199’)
2. Respons yang berhasil (‘200’ - ‘299’)
3. Pengalihan (‘300’ - ‘399’)
4. Kesalahan client (‘400’ - ‘499’)
5. Kesalahan server (‘500’ - ‘599’)

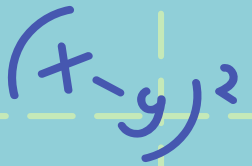


CONTOH PENGGUNAAN API

Contoh API yang paling populer dan bisa kamu temui setiap saat adalah login universal atau kemudahan untuk masuk ke situs menggunakan detail profil Facebook, Twitter, atau Google. Fitur ini sangat praktis karena membuat situs mana pun memanfaatkan API untuk mengautentikasi pengguna dengan cepat. Fitur ini juga menghemat waktu karena kita tidak perlu repot mengisi profil baru untuk setiap layanan situs atau keanggotaan baru.







CARA REQUEST DARI API

Untuk meminta data dari server API perlu memakai library requests yang nantinya akan menggunakan perintah method yang terdiri dari GET, POST, CREATE, DELETE. Untuk mengambil data seperti JSON dari API menggunakan REQUESTS.GET

```
[ ]  import pandas as pd
     import requests
```

```
url = 'https://api.covidtracking.com/v1/states/ca/daily.json'

response = requests.get(url)

data = response.json()
data
```

Hasilnya :


```
[{'checkTimeEt': '03/06 21:59',
  'commercialScore': 0,
  'dataQualityGrade': None,
  'date': 20210307,
  'dateChecked': '2021-03-07T02:59:00Z',
  'dateModified': '2021-03-07T02:59:00Z',
  'death': 54124,
  'deathConfirmed': None,
  'deathIncrease': 258,
  'deathProbable': None,
  'fips': '06',
  'grade': '',
```


Dataframe :

```
api = pd.DataFrame(data)
api
```

	date	state	positive	probableCases	negative	pending	totalTestResultsSource	totalTestResults	hospitalizedCurrently
0	20210307	CA	3501394	None	NaN	NaN	totalTestsViral	49646014	4291.0
1	20210306	CA	3497578	None	NaN	NaN	totalTestsViral	49512828	4513.0
2	20210305	CA	3493126	None	NaN	NaN	totalTestsViral	49294503	4714.0
3	20210304	CA	3488467	None	NaN	NaN	totalTestsViral	49147685	4967.0
4	20210303	CA	3484963	None	NaN	NaN	totalTestsViral	49028048	5110.0

DigitalSkola

Thank you!

Our Team

1. **Aldiva Wibowo**
2. **Asprizal Rizky**
3. **Gilang Rahmat R**
4. **Lutfia Humairosi**
5. **Millenia Winadya P**