

CERDAS MENGUASAI FLASK

Penulis:

Rolly Maulana Awangga

ISBN : 978-602-53897-0-2

Editor:

M. Yusril Helmi Setyawan

Penyunting:

Syafrial Fachrie Pane

Khaera Tunnisia

Diana Asri Wijayanti

Desain sampul dan Tata letak:

Deza Martha Akbar

Penerbit:

Kreatif Industri Nusantara

Redaksi:

Jl. Ligar Nyawang No. 2

Bandung 40191

Tel. 022 2045-8529

Email : awangga@kreatif.co.id

Distributor:

Informatics Research Center

Jl. Sariasisih No. 54

Bandung 40151

Email : irc@poltekpos.ac.id

Cetakan Pertama, 2019

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara
apapun tanpa ijin tertulis dari penerbit

*'Jika Kamu tidak dapat
menahan lelahnya
belajar, Maka kamu harus
sanggup menahan
perihnya Kebodohan.'*

Imam Syafi'i

CONTRIBUTORS

ROLLY MAULANA AWANGGA, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia

CONTENTS IN BRIEF

1 Pemrograman Dasar Python	1
2 Akuisisi Data Menggunakan Sensor EEG	19
3 PYSERIAL MINDWAVE	43
4 PENYAJIAN DATA DALAM PENGGUNAAN MATPLOTLIB	51
5 Pengenalan Flask	95
6 Route dan URL di Flask	137
7 HTTP GET METHOD REQUEST	157
8 HTTP POST, PUT, DELETE	195
9 JSON	219
10 Template	237

DAFTAR ISI

Daftar Gambar	xv
Daftar Tabel	xxvii
Foreword	xxxv
Kata Pengantar	xxxvii
Acknowledgments	xxxix
Acronyms	xli
Glossary	xliii
List of Symbols	xlvi
Introduction	xlvii
<i>Rolly Maulana Awangga, S.T., M.T.</i>	
1 Pemrograman Dasar Python	1
1.1 Variabel Python	1
1.1.1 Contoh Pembuatan Variabel Python	2
1.1.2 Menghapus Variabel	2
1.2 Input Output User	3

1.2.1	Cara mengambil input pada keyboard	3
1.2.2	Menampilkan Variabel dan Teks	3
1.3	Operator Dasar	4
1.3.1	Operator Aritmatika	4
1.3.2	Operator Perbandingan	5
1.3.3	Operator Penugasan	6
1.3.4	Operator Logika	7
1.3.5	Operator Bitwise	8
1.3.6	Prioritas Eksekusi Operator di Python	9
1.4	Perulangan / Loop	9
1.4.1	While Loop	10
1.4.2	For Loop	10
1.4.3	Nested Loop	11
1.5	Kondisi	11
1.5.1	Kondisi IF	11
1.5.2	Kondisi IF Else	12
1.5.3	Kondisi Elif	12
1.6	Pembacaan Error	13
1.6.1	Kesalahan sintak	13
1.6.2	Kesalahan Logika	13
1.7	Try to Except/pengecualian	14
1.7.1	Menangkap Pengecualian dengan Python	14
1.7.2	Menangkap Pengecualian Khusus dengan Python	15
1.7.3	Meningkatkan Pengecualian	16
1.7.4	Try Finally	16
2	Akuisisi Data Menggunakan Sensor EEG	19
2.1	Membuat Fungsi Dalam Satu File dan Cara Menggunakannya	19
2.1.1	Function untuk membuat tampilan awal	20
2.2	Membuat Fungsi Pada File Terpisah Dari Cara Penggunaannya	24
2.3	Membuat Kelas dan Cara Instalasi Kelas Pada Program Utama	29
2.4	Hasil Pengujian	30
2.4.1	Tampilan awal	30
2.4.2	Tampilan Pilih File	30
2.4.3	Tampilan line graph	30
2.4.4	Tampilan Bar Graph	32
2.5	PENANGGANAN ERROR PADA PYTHON	32
2.5.1	Penanganan error	32

2.5.2	Fungsi memilih line graph	34
2.5.3	Fungsi memilih file	35
2.6	Tutorial Penanganan Error Pada Tkinter	37
3	PYSERIAL MINDWAVE	43
3.1	Pyserial Mindwave	43
3.2	Code Program Mindwave	43
3.2.1	Code Program Pyserial Function Pada Mindwave	43
3.3	Tutorial Mengatasi error Pyserial Function pada Mindwave	48
3.3.1	Mengenal error Dalam Program	48
3.3.2	Code error Pada Program Pyserial Mindwave	48
4	PENYAJIAN DATA DALAM PENGGUNAAN MATPLOTLIB	51
4.1	Penjelasan Matplotlib	51
4.2	Pemasangan Matplotlib	51
4.3	Menggambar Plot Dasar	52
4.3.1	Plot Garis	52
4.3.2	Plot Sebaran	53
4.3.3	Histogram	54
4.3.4	Kustomisasi Plot	55
4.4	Pembacaan data dari sebuah file.txt	56
4.4.1	Membuat/memasukkan data melalui notepad	56
4.4.2	Pembacaan data	57
4.4.3	Pemanggilan data ke sebuah plot	58
4.5	Modifikasi Hasil Plot	60
4.5.1	Setting Label	60
4.5.2	Setting Warna	60
4.5.3	Setting Marker	61
4.5.4	Jenis Plot	63
4.6	Plot Ditampilkan Dan Disimpan	68
4.6.1	Plot Ditampilkan	68
4.6.2	Plot Disimpan	69
4.7	Penanganan Error Pada Pemasangan Matplotlib	71
4.8	Error Pada Saat Menggambar Plot Dasar	72
4.8.1	Error Pada Plot Garis	72
4.8.2	Plot Sebaran	74
4.8.3	Error Pada Histogram	76
4.8.4	Error Pada Saat Kustomisasi Plot	77

4.9	Error Pada Pembacaan data dari sebuah file.txt	80
4.9.1	Membuat/memasukkan data melalui notepad	80
4.9.2	Error Pada Pembacaan data	80
4.9.3	Error Pemanggilan data ke sebuah plot	81
4.10	Error Pada Modifikasi Hasil Plot	85
4.10.1	Error Setting Label	85
4.10.2	Error Setting Warna	88
4.10.3	Error Pada Setting Marker	90
5	Pengenalan Flask	95
5.1	Definisi <i>Micro Framework</i>	95
5.2	Jenis-Jenis <i>Framework</i> Python serta Kelebihan dan Kekurangan	96
5.2.1	Django	96
5.2.2	Flask	98
5.2.3	Tornado	99
5.2.4	Falcon	100
5.2.5	Hug	101
5.2.6	Sanic	101
5.2.7	Aiohttp	102
5.2.8	Piramid	103
5.2.9	Web2py	104
5.2.10	TurboGears2	104
5.2.11	Cherrypy	105
5.3	Instalasi dan Hello World di Flask	106
5.3.1	Instalasi Python 2.7.	106
5.3.2	Instalasi Python 3.6	107
5.3.3	Instalasi Framework Flask	109
5.3.4	Instalasi Library Pandas	110
5.3.5	Contoh Penerapan Fungsi Pada Flask Python	111
5.4	Penanganan Error	114
5.4.1	Penanganan Error pada Python	114
5.4.2	Penanganan Error dalam Flask	126
6	Route dan URL di Flask	137
6.1	Aturan Route di Flask	137
6.2	Aturan URL di Flask	139
6.3	Berbagai Macam Jenis Kreasi URL di Flask	140
6.4	Penanganan Error	150

6.4.1	Error 1	150
6.4.2	Error 2	154
7	HTTP GET METHOD REQUEST	157
7.1	HTTP Method Request	157
7.1.1	Pengertian HTTP Method Request	157
7.1.2	Jenis-jenis HTTP Method Request	158
7.1.3	Penjelasan Lengkap HTTP Get Method	158
7.1.4	Pembacaan HTTP Get Method	159
7.1.5	Pembacaan Data pada URL	159
7.2	Mekanisme HTTP Method Request	159
7.2.1	Mekanisme / Alur kerja HTTP Get Method	159
7.3	Contoh URL HTTP Get Method	160
7.3.1	Endpoint 1 (URL API / Dataset)	165
7.3.2	Endpoint 2 (URL API : /dataset/jumlah)	167
7.3.3	Endpoint 3 (URL API : /dataset/<id>)	174
7.3.4	Endpoint 4 (URL API : /info)	176
7.4	Mendapatkan Parameter GET Python Flask	178
7.4.1	Penjelasan Parameter GET Python Flask	178
7.5	Penanganan Error	186
7.5.1	Error 1	186
7.5.2	Error 2	189
7.5.3	Error 3	192
8	HTTP POST, PUT, DELETE	195
8.1	Definisi HTTP Post Method	195
8.2	Mekanisme HTTP Post Method	195
8.3	Contoh URL HTTP Post Method	196
8.4	Definisi HTTP Put	198
8.5	Mekanisme HTTP Put Method	198
8.6	Contoh URL HTTP Put Method	198
8.7	Definisi HTTP Delete	199
8.8	Mekanisme HTTP Delete Method	199
8.9	Contoh URL HTTP Delete Method	199
8.10	Instalasi Dan Tata Cara Penggunaan Postman Untuk HTTP Post, Put Dan Delete	200
8.10.1	Instalasi Postman Di PC	200
8.10.2	Instalasi Ekstensi Postman Pada Web Browser Chrome	205

8.10.3	Cara Penggunaan Postman Untuk Pengujian HTTP POST,PUT, Dan DELETE	206
8.11	Penanganan Error	214
8.11.1	TypeError: 'NoneType' Object Has No Attribute	214
8.11.2	DELETE Error Code 404	216
9	JSON	219
9.1	Membuat Kembalian Web Service Dalam Bentuk JSON Pada HTTP GET, POST, PUT, dan DELETE	219
9.1.1	Definisi JSON	219
9.1.2	Sintaks dan Struktur	220
9.1.3	Perbandingan Dengan XML	221
9.1.4	Sumber-sumber Lain	221
9.1.5	RESTful Web Service	221
9.1.6	Metode HTTP yang biasa digunakan dalam arsitektur REST	222
9.1.7	Cara kerja RESTful Web Service	222
9.1.8	Cara implementasi JSON pada Web Service	222
9.1.9	Sebelum membuat file JSON, buatlah folder js dan img di dalam folder static	226
9.1.10	Dan file keempat, buatlah file HTML yang bernama live.html, kemudian isikan kode seperti pada listing 9.6:	232
10	Template	237
10.1	Cara Mengimplementasikan Jinja pada Flask	237
10.2	Penanganan Error	252
	Daftar Pustaka	259
	Index	261

DAFTAR GAMBAR

1.1	Penulisan Variabel	2
1.2	Hasil Perintah Print	2
1.3	Perintah del	2
1.4	Perintah Print	2
1.5	Hasil perintah print setelah variabel di delete	3
1.6	Hasil input	3
1.7	Hasil menampilkan variabel dan teks	4
1.8	Hasil contoh operator aritmatika	5
1.9	Hasil contoh operator perbandingan	6
1.10	Hasil contoh operator penugasan	7
1.11	Hasil contoh operator logika	8
1.12	Hasil contoh operator bitwise	9
1.13	Hasil menangkap pengecualian khusus	13

1.14	Hasil program kesalahan logika	14
1.15	Hasil setelah di benarkan	14
1.16	Hasil try to except	15
2.1	Tampilan awal	30
2.2	Tampilan pilih file	31
2.3	Data line graph	31
2.4	Data bar graph	32
2.5	Kesalahan sintak	33
2.6	Kesalahan logika	33
2.7	Error plt	34
2.8	Error Choose File	35
2.9	Self Choose File	36
2.10	Error file_name	37
2.11	Error Tkinter	37
2.12	Import Tkinter	38
2.13	Folder Python	39
2.14	Drop Folder	39
2.15	Buka Folder Python	39
2.16	Tambahkan Folder	40
2.17	Install Modul	40
2.18	Modul sudah tersedia	41
3.1	function menampilkan gelombang otak	44
3.2	function Data Plotting pada Mindwave	45
3.3	function Subplot pada Mindwave	46
3.4	Function Memberikan Keterangan Waktu dan Frekuensi	47
3.5	Function Memasukkan Rumus Pada Grafik Gelombang Otak	48
3.6	Error no attribute	49
3.7	Error Indetation	49

3.8	Error Parameter	49
3.9	Error Parameter	49
3.10	Error Penulisan	49
4.1	Validasi Python	52
4.2	Pemasangan Matplotlib	52
4.3	Tampilan Plot Garis	53
4.4	Tampilan Plot Sebaran	54
4.5	Tampilan Histogram	55
4.6	Tampilan Kustomisasi Plot	56
4.7	Data Hasil.txt	57
4.8	Pemanggilan bacadata.py	58
4.9	Pemanggilan pangilda.py	59
4.10	Tampilan pemanggilan data ke sebuah plot	59
4.11	Tampilan setting label	61
4.12	Tampilan setting warna	62
4.13	Tampilan setting tanpa marker	63
4.14	Tampilan setting dengan marker	64
4.15	Tampilan jenis plot garis	64
4.16	Tampilan jenis plot sebaran	65
4.17	Tampilan jenis plot histogram	65
4.18	Tampilan jenis plot bar charts	66
4.19	Tampilan jenis plot pie	66
4.20	Tampilan jenis plot hist	67
4.21	Tampilan jenis plot 3D Line	67
4.22	Tampilan jenis plot 3D Scatter	68
4.23	Tampilan jenis plot 3D Charts	69
4.24	Hasil plot ditampilkan	70
4.25	Menu plot disimpan	70

4.26	Proses plot disimpan	71
4.27	Error Pemasangan Matplotlib	71
4.28	Pemasangan Matplotlib	72
4.29	Skrip Error Plot Garis	72
4.30	Error Plot Garis	73
4.31	Solusi Error Plot Garis	73
4.32	Error Plot Garis	73
4.33	Solusi Error Plot Garis	74
4.34	Error Plot Sebaran	74
4.35	Solusi Error Plot Sebaran	75
4.36	Error Plot Sebaran	75
4.37	Solusi Error Plot Sebaran	75
4.38	Error Histogram	76
4.39	Solusi Error Histogram	76
4.40	Solusi Error Histogram	76
4.41	Tampilan Histogram	77
4.42	Error Kustomisasi Plot	78
4.43	Solusi Error Kustomisasi Plot	78
4.44	Error Kustomisasi Plot	78
4.45	Solusi Error Kustomisasi Plot	78
4.46	Tampilan Kustomisasi Plot	79
4.47	Data Hasil.txt	80
4.48	Panggil File bacadata.py	81
4.49	Error pada pembacaan data	81
4.50	Script Error pada pembacaan data	81
4.51	Solusi Error pada pembacaan data	81
4.52	Tampilan pada pembacaan data	82
4.53	Error pemanggilan data	82

4.54	Solusi Error pemanggilan data	83
4.55	Error pemanggilan data	83
4.56	Solusi Error pemanggilan data	83
4.57	Error tampilan pemanggilan data	84
4.58	Solusi Error pemanggilan data	84
4.59	Error pemanggilan data	85
4.60	Marker Pada Matplotlib	86
4.61	Skrip Marker Pada Matplotlib	86
4.62	Solusi Error Marker	87
4.63	Error Setting Label	87
4.64	Solusi Error Setting Label	87
4.65	Tampilan Solusi Error Setting Label	88
4.66	Tampilan Setting Label	89
4.67	Error Setting Warna	89
4.68	Solusi Error Setting Warna	90
4.69	Color Pada Matplotlib 3	90
4.70	Color Pada Matplotlib 2	91
4.71	Color Pada Matplotlib 1	92
4.72	Solusi Error Color Pada Matplotlib	92
4.73	Tampilan Setting Warna	93
4.74	Error Setting Marker	94
4.75	Solusi Setting MArker	94
4.76	Tampilan Setting Marker	94
5.1	Download Softfile Python 2.7.	106
5.2	Pengecekan Python 2.7.	107
5.3	Pengubahan Environment Python 2.7.	107
5.4	Download Softfile Python 3.6.	108
5.5	Pengecekan Python 3.6.	108

5.6	Pengubahan Environment Python 3.6.	109
5.7	Download Flask	110
5.8	Proses Instalasi Flask	110
5.9	Download Pandas	111
5.10	Proses Instalasi Pandas	112
5.11	Instalasi Pip Flask	112
5.12	Pemanggilan Fungsi Flask Opsi 1	113
5.13	Pemanggilan Fungsi Flask Opsi 2	114
5.14	Output Hello World	114
5.15	Eksekusi Fungsi Percobaan Pertama	115
5.16	Eksekusi Fungsi Percobaan Kedua	115
5.17	Eksekusi Fungsi Percobaan Ketiga	116
5.18	Eksekusi Fungsi Percobaan Keempat	117
5.19	Eksekusi Fungsi Percobaan Kelima	118
5.20	Eksekusi Fungsi Percobaan Keenam	119
5.21	Eksekusi Fungsi Percobaan Ketujuh	119
5.22	Eksekusi Fungsi Percobaan Kedelapan	120
5.23	Eksekusi Fungsi Percobaan Kesembilan	121
5.24	Eksekusi Fungsi Percobaan Kesepuluruh	121
5.25	Eksekusi Fungsi Percobaan Kesebelas	121
5.26	Deskripsi Pengukuran	123
5.27	Error pada program coba.py	124
5.28	Keterangan error pada program coba.py	124
5.29	Eksekusi Program import.py	127
5.30	Eksekusi Program import.py	128
5.31	Eksekusi Program import.py	128
5.32	Hasil Eksekusi Program import.py	129
5.33	Eksekusi Program main.py	130

5.34	Hasil Eksekusi Program main.py	131
5.35	Hasil Eksekusi Program main.py	132
5.36	Hasil Eksekusi Program main.py	132
5.37	Eksekusi Program main.py	134
5.38	Hasil Eksekusi Program main.py	135
6.1	Pengujian helloworld	138
6.2	Hasil Pengujian helloworld	139
6.3	Pengujian url contoh	140
6.4	Hasil Pengujian url contoh	140
6.5	Hasil Pengujian url contoh	140
6.6	Pengujian variabel	141
6.7	Hasil Pengujian Variabel 1	142
6.8	Hasil Pengujian Variabel 2	142
6.9	Hasil Pengujian Variabel 3	142
6.10	Hasil Pengujian Variabel 4	142
6.11	Hasil Pengujian Variabel 5	142
6.12	Hasil Pengujian Variabel 6	143
6.13	Hasil Pengujian Variabel 7	143
6.14	Pengujian simbol	143
6.15	Hasil Pengujian simbol 1	144
6.16	Hasil Pengujian simbol 2	144
6.17	Pengujian heading text	145
6.18	Hasil Pengujian heading text	145
6.19	Pengujian contoh lain	146
6.20	Hasil Pengujian contoh lain 1	146
6.21	Hasil Pengujian contoh lain 2	147
6.22	Hasil Pengujian contoh lain 3	147
6.23	Pengujian cobacoba 1	147

6.24	Hasil Pengujian cobacoba 1	148
6.25	Pengujian cobacoba 2	148
6.26	Hasil Pengujian cobacoba 2	149
6.27	Pengujian sekali lagi	149
6.28	Hasil Pengujian sekali lagi	150
6.29	Pengujian file flasklivechart.py	153
6.30	Hasil Pengujian file flasklivechart.py	154
6.31	CMD file flasklivechart.py	154
6.32	Struktur Projek	155
6.33	Alat yang akan ditampilkan	156
7.1	Hasil Pengujian 1 fungsi dataset (GET)	167
7.2	Hasil Pengujian 2 fungsi dataset (GET)	168
7.3	Hasil Pengujian 3 fungsi dataset (GET)	169
7.4	Hasil Pengujian 4 fungsi dataset (GET)	170
7.5	Hasil Pengujian 5 fungsi dataset (GET)	170
7.6	Hasil Pengujian 6 fungsi dataset (GET)	171
7.7	Hasil Pengujian 7 fungsi dataset (GET)	171
7.8	Hasil Pengujian 8 fungsi dataset (GET)	172
7.9	Hasil Pengujian 1 fungsi dataset/jumlah (GET)	173
7.10	Hasil Pengujian 2 fungsi dataset/jumlah (GET)	173
7.11	Hasil Pengujian 3 fungsi dataset/jumlah (GET)	174
7.12	Hasil Pengujian 1 fungsi dataset/<id> (GET)	175
7.13	Hasil Pengujian 2 fungsi dataset/<id> (GET)	175
7.14	Hasil Pengujian 3 fungsi dataset/<id> (GET)	176
7.15	Hasil Pengujian 1 fungsi dataset/info (GET)	177
7.16	Instalasi Python	178
7.17	Python CMD	179
7.18	Device Manager (Env)	179

7.19	Instalasi Flask	180
7.20	Install Flask di CMD	181
7.21	Instalasi Pandas	182
7.22	Install Pandas di CMD	182
7.23	Aktivasi main.py 2 pada contoh kasus 3	186
7.24	Hasil pengujian 1 untuk error 9 pada contoh kasus 3	187
7.25	Error 1 untuk error 9 pada contoh kasus 3	187
7.26	Hasil pengujian 2 untuk error 9 pada contoh kasus 3	188
7.27	Aktivasi main.py 3 pada contoh kasus 3	189
7.28	Hasil pengujian 1 untuk error 10 pada contoh kasus 3	190
7.29	Error 1 untuk error 10 pada contoh kasus 3	190
7.30	Hasil pengujian 2 untuk error 10 pada contoh kasus 3	191
7.31	Aktivasi main.py 4 pada contoh kasus 3	192
7.32	Hasil pengujian 1 untuk error 11 pada contoh kasus 3	193
7.33	Error 1 untuk error 11 pada contoh kasus 3	193
7.34	Hasil pengujian 2 untuk error 11 pada contoh kasus 3	194
8.1	Terminal Pada Visual Code	196
8.2	Folder Lokasi Codingan	197
8.3	Menjalankan Aplikasi	197
8.4	Contoh HTTP POST	198
8.5	Website Postman	200
8.6	Postman Versi Windows 64-bit	201
8.7	Buka File Instalasi	201
8.8	Instalasi Postman	202
8.9	Halaman Membuat Akun Postman	202
8.10	Login Postman Dengan Akun Google	203
8.11	Password Untuk Login Postman Dengan Akun Google	204
8.12	Tampilan Postman	205

8.13	Pencarian Ekstensi Postman Untuk Chrome	205
8.14	Menambahkan Postman ke Ekstensi Chrome	206
8.15	Ikon Postman Pada Laman App Chrome	206
8.16	Menjalankan Aplikasi Pada CMD	207
8.17	Menambahkan URL Untuk POST Method	207
8.18	Mengubah Request Menjadi POST	207
8.19	Hasil Pengujian Post Method	208
8.20	Preview Pengujian Post Method	208
8.21	Get Request	208
8.22	Hasil Pengujian Get Request	209
8.23	Contoh Put Method	209
8.24	Edit Data Contoh	210
8.25	Data Contoh Terupdate	210
8.26	Data Contoh Yang Terupdate	211
8.27	Edit Data HTTP	211
8.28	Data HTTP Terupdate	211
8.29	Data HTTP Yang Terupdate	212
8.30	Mengubah URL Untuk HTTP Delete	212
8.31	Mengubah Metode Menjadi Delete	213
8.32	Data Inilah Terhapus	213
8.33	Mengubah URL Untuk Delete Data HTTP	213
8.34	Data HTTP Terhapus	214
8.35	TypeError: 'NoneType' Object Has No Attribute	214
8.36	Menjalankan Postman	215
8.37	Mengubah Metode	215
8.38	Memilih Body Pada Postman	215
8.39	Mengubah Object Type	216
8.40	Pengujian	216

8.41	DELETE Error Code 404	216
8.42	Mengubah URI Sesuai Dengan Sistem	217
8.43	Data Sebelum Dihapus	217
8.44	Data Setelah Dihapus	218
9.1	Folder Users	223
9.2	Folder App	223
9.3	Folder Static	227
9.4	EEG	229
9.5	Charts	230
9.6	Halaman Awal Aplikasi	235
9.7	Hasil Data	235
10.1	Pengujian app.py	239
10.2	Hasil Pengujian File app.py	239
10.3	Pengujian File app.py edit	240
10.4	Hasil Pengujian File app.py edit	240
10.5	Struktur Projek	241
10.6	Pengujian file app.py edit 2	242
10.7	Hasil Pengujian file app.py edit 2	243
10.8	Pengujian file app.py edit 2	245
10.9	Hasil Pengujian file app.py edit 2	245
10.10	Pengujian file app.py edit 2	246
10.11	Hasil Pengujian file app.py edit 2	247
10.12	Pengujian file app.py edit 2	248
10.13	Hasil Pengujian file app.py edit 2	249
10.14	Pengujian file app.py edit 2	250
10.15	Hasil Pengujian file app.py edit 2	251
10.16	Pengujian file app.py edit 3	251
10.17	Hasil Pengujian file app.py edit 3	252
10.18	Pengujian file flasklivechart.py	255
10.19	Hasil Pengujian file flasklivechart.py	256
10.20	Pengujian file flasklivechart.py	257
10.21	Hasil Pengujian file flasklivechart.py	257

DAFTAR TABEL

1.1	Operator Aritmatika	4
1.2	Operator Perbandingan	5
1.3	Operator Penugasan	6
1.4	Operator Logika	7
1.5	Operator Bitwise	8
1.6	Prioritas Eksekusi Operator di Python	10
7.1	Penjelasan Schema	160

Listings

1.1	Contoh kode input output	3
1.2	Perintah menampilkan variabel dan teks	3
1.3	Contoh operator aritmatika	4
1.4	Contoh operator perbandingan	5
1.5	Contoh operator penugasan	6
1.6	Contoh operator logika	7
1.7	Contoh operator bitwise	8
1.8	Koding while loop	10
1.9	Koding for loop	10
1.10	Koding nested loop	11
1.11	Koding kondisi if	11
1.12	Koding kondisi if else	12
1.13	Koding kondisi elif	12
1.14	Koding kesalahan sintak	13
1.15	Koding kesalahan logika	13
1.16	Koding exception	14
1.17	Koding special exception	15
1.18	Koding meningkatkan pengecualian	16

1.19 Koding try finnaly	17
2.1 Tampilan awal	20
2.2 Tampilan untuk mencari file	21
2.3 Tampilan untuk memilih file	21
2.4 Tampilan line graph	22
2.5 Tampilan bar graph	22
2.6 Tampilan untuk memilih file	22
2.7 Tampilan function line graph	23
2.8 Tampilan function bar graph	23
2.9 Tampilan function membuka dan membaca file	24
2.10 main.py	24
2.11 Pilih COM	25
2.12 Koneksi	25
2.13 Function Menampilkan Data	25
2.14 Function Menampilkan Gelombang	26
2.15 Function Menampilkan Gelombang Dengan Maker	26
2.16 Function Membuat Title Pada Grafik	27
2.17 xlabel	28
2.18 Warna Scatter	28
2.19 Membuat Kelas	29
2.20 Membuat Objek Kelas	30
2.21 Solusi Kesalahan Sintak	33
2.22 Solusi Kesalahan Logika	33
2.23 Line Graph	34
2.24 Import	35
2.25 Memilih file	35
2.26 Self Choose File	35
2.27 file_name	36
2.28 open_file	36
2.29 Import Tkinter	38
2.30 Pip install "nama modul"	38
3.1 Function Untuk Menampilkan Gelombang Otak pada Mindwave	43
3.2 Function Memasukan Data Plotting Otak pada Mindwave	44
3.3 Function Subplot pada Mindwave	45
3.4 Function Memberikan Keterangan Waktu dan Frekuensi	45
3.5 Function Memasukkan Rumus Pada Grafik Gelombang Otak	47
4.1 Skrip Plot Garis	53
4.2 Skrip Plot Sebaran	54

4.3 Skrip Histogram	55
4.4 Skrip Kustomisasi Plot	56
4.5 Skrip Pembacaan Data	57
4.6 Skrip Pemanggilan data ke sebuah plot	58
4.7 Skrip Setting Label	60
4.8 Skrip Setting Warna	60
4.9 Skrip Setting Tanpa Marker	61
4.10 Skrip Setting Dengan Marker	62
4.11 Skrip Plot ditampilkan	68
4.12 Skrip Plot Garis	72
4.13 Skrip Plot Sebaran	74
4.14 Skrip Histogram	76
4.15 Kustomisasi Plot	77
4.16 Skrip Pembacaan data	80
4.17 Skrip pemanggilan data ke sebuah plot	81
4.18 Contoh Skrip error pemanggilan data ke sebuah plot	83
4.19 Skrip Setting Label	85
4.20 Skrip setting warna	88
4.21 Skrip setting marker	90
5.1 Contoh kode program hello.py	113
5.2 Contoh kode program main.py	113
5.3 File satu.py	114
5.4 File baru satu.py	115
5.5 File baru satu.py	116
5.6 File baru satu.py	117
5.7 File baru satu.py	117
5.8 File baru dua.py	118
5.9 File baru dua.py	119
5.10 File baru dua.py	120
5.11 File coba.py	122
5.12 File coba.py	122
5.13 File coba.py	123
5.14 File try_except.py	124
5.15 File coba_lagi.py	125
5.16 File coba_lagi.py	126
5.17 File import.py	127
5.18 File import.py	128
5.19 File get_top_five.py	129

5.20 File main.py	129
5.21 File get_top_five.py	130
5.22 File delete_column.py	133
5.23 File main.py	133
5.24 File delete_all_data.py	134
6.1 File app.py	137
6.2 Contoh File app.py	139
6.3 Contoh File app.py variabel	140
6.4 Contoh File app.py simbol	143
6.5 Contoh File app.py heading text	144
6.6 Contoh File app.py contoh lain	145
6.7 Contoh File app.py cobacoba 1	147
6.8 Contoh File app.py cobacoba 2	148
6.9 Contoh File app.py sekali lagi	149
6.10 Contoh File app.py cobacoba 3	149
6.11 File flasklivechart.py	150
6.12 File flasklivechart.py edit	154
6.13 File flasklivechart.py edit	155
7.1 Contoh kode untuk schema	159
7.2 Contoh kode untuk membaca file CSV	160
7.3 Contoh kode untuk menampilkan semua data dari file CSV	161
7.4 Contoh kode untuk menampilkan data dari kolom	161
7.5 Contoh kode untuk menampilkan 5 data teratas dan terbawah	161
7.6 Contoh kode untuk mengurutkan data	161
7.7 Contoh kode untuk mengganti data	161
7.8 Contoh kode untuk menghapus kolom tertentu	161
7.9 Contoh kode untuk menghapus semua data	162
7.10 Contoh kode untuk menambah data	162
7.11 Contoh kode untuk Menampilkan data csv	162
7.12 Contoh kode untuk Menampilkan seluruh jumlah baris	162
7.13 Contoh kode untuk Menampilkan data perbaris	162
7.14 Contoh kode untuk Mengubah data berdasarkan index id	163
7.15 Contoh kode untuk Menghapus data berdasarkan index id	163
7.16 Contoh kode untuk Mengubah data menjadi Json	163
7.17 Contoh kode untuk Menampilkan seluruh data	164
7.18 Contoh kode untuk Menghitung jumlah data	164
7.19 Contoh kode untuk Menghitung jumlah kolom	164
7.20 Contoh kode untuk Menghitung jumlah baris	164

7.21 Contoh kode untuk Mengembalikan nilai data	165
7.22 Contoh kode untuk URL API	165
7.23 Method Get dari Endpoint : URL API /dataset/jumlah	167
7.24 Method Get dari Endpoint : URL API /dataset/<id>	174
7.25 Method Get dari Endpoint : URL API /dataset/info	176
7.26 File Main.py Python Flask	183
7.27 Penjelasan I : Main.py	184
7.28 Penjelasan II : Main.py	185
7.29 Penjelasan IV : Main.py	185
8.1 Skrip Fungsi Contoh HTTP POST	196
8.2 Skrip Fungsi Contoh HTTP PUT	198
8.3 Skrip Fungsi Pada HTTP GET	199
8.4 Skrip Fungsi Pada HTTP DELETE	199
9.1 Contoh Objek JSON	220
9.2 Contoh Objek JSON	220
9.3 Flask Live Chart	223
9.4 Highcharts.js	226
9.5 Files.html	228
9.6 live.html	232
9.7 Perintah	234
10.1 File app.py	238
10.2 File app.py edit	239
10.3 File index.html	241
10.4 File app.py edit 2	241
10.5 File index.html	243
10.6 File index.html edit 1	244
10.7 File index.html edit 2	246
10.8 File index.html edit 3	247
10.9 File index.html edit 4	249
10.10 File flasklivechart.py	252

FOREWORD

Sepatah kata dari Kaprodi, Kabag Kemahasiswaan dan Mahasiswa

KATA PENGANTAR

Buku ini diciptakan bagi yang awam dengan flask sekalipun.

R. M. AWANGGA

Bandung, Jawa Barat

Februari, 2019

ACKNOWLEDGMENTS

Terima kasih atas semua masukan dari para mahasiswa agar bisa membuat buku ini lebih baik dan lebih mudah dimengerti.

Terima kasih ini juga ditujukan khusus untuk team IRC yang telah fokus untuk belajar dan memahami bagaimana buku ini mendampingi proses Intership.

R. M. A.

ACRONYMS

ACGIH	American Conference of Governmental Industrial Hygienists
AEC	Atomic Energy Commission
OSHA	Occupational Health and Safety Commission
SAMA	Scientific Apparatus Makers Association

GLOSSARY

git	Merupakan manajemen sumber kode yang dibuat oleh linus torvald.
bash	Merupakan bahasa sistem operasi berbasiskan *NIX.
linux	Sistem operasi berbasis sumber kode terbuka yang dibuat oleh Linus Torvald

SYMBOLS

A Amplitude

& Propositional logic symbol

a Filter Coefficient

B Number of Beats

INTRODUCTION

ROLLY MAULANA AWANGGA, S.T., M.T.

Informatics Research Center
Bandung, Jawa Barat, Indonesia

Pada era disruptif saat ini. git merupakan sebuah kebutuhan dalam sebuah organisasi pengembangan perangkat lunak. Buku ini diharapkan bisa menjadi penghantar para programmer, analis, IT Operation dan Project Manajer. Dalam melakukan implementasi git pada diri dan organisasinya.

Rumusnya cuman sebagai contoh aja biar keren[1].

$$ABC\mathcal{DEF}\alpha\beta\Gamma\Delta \sum_{def}^{abc} \quad (I.1)$$

BAB 1

PEMROGRAMAN DASAR PYTHON

1.1 Variabel Python

Variabel merupakan sebuah ruang kosong untuk menyimpan suatu nilai atau data. Pada saat anda membuat sebuah variabel berarti anda sedang memesan sebuah ruang kosong di memori. Isi dari variabel itu dapat berubah atau mutable sesuai dengan operasi yang diinginkan. Saat program dieksekusi maka variabellah yang bertugas menyimpan data.

Variabel dapat menyimpan berbagai macam tipe data. Dalam pemrograman Python, variabel mempunyai sifat dinamis, yang berarti variabel Python tidak perlu ditentukan tipe data tertentu dan variabel pada Python dapat diubah saat program dieksekusi.

Peraturan penulisan variable Python :

1. Karakter pertama harus berupa huruf atau garis bawah atau underscore (_)

2. Karakter selanjutnya dapat berupa huruf, garis bawah/underscore (_) atau angka

3. Karakter pada nama variabel bersifat sensitif (case-sensitif). Artinya penggunaan uruf besar dan huruf keci sangat berpengaruh. Sebagai contoh, variabel Mahasiswa dan mahasiswa adalah variabel yang berbeda.
4. Nama variabel tidak boleh menggunakan kata kunci yang sudah ada pada Python.

Untuk membuat variabel Python caranya sangat mudah, cukup buat nama variabel lalu diikuti oleh = dan mengisinya dengan nilai yang dinginkan.

1.1.1 Contoh Pembuatan Variabel Python

Misalnya ada variabel “nama” dengan nilai “Poltekpos”. Maka penulisan variabelnya seperti pada gambar 1.1:

```
>>> nama = "Poltekpos"
```

Gambar 1.1 Penulisan Variabel

Kemudian perintahkan print untuk menampilkan isi dari variabel. Caranya: Print namanya. Contohnya seperti pada gambar 1.2 :

```
>>> print nama
Poltekpos
```

Gambar 1.2 Hasil Perintah Print

1.1.2 Menghapus Variabel

Untuk menghapus variabel caranya sangat mudah. Anda cukup menggunakan perintah del() untuk menghapus variabel yang sudah tidak dibutuhkan lagi. Perintah delete, del(namavariabel) Contohnya seperti pada gambar 1.3 :

```
>>> del(nama)
```

Gambar 1.3 Perintah del

Untuk mengecek apakah variabelnya berhasil dihapus anda bisa menggunakan perintah print seperti pada gambar 1.4.

```
|>>> print nama
```

Gambar 1.4 Perintah Print

Jika hasilnya seperti gambar 1.5 maka variabel telah berhasil dihapus.

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'nama' is not defined
```

Gambar 1.5 Hasil perintah print setelah variabel di delete

1.2 Input Output User

Input adalah masukan yang anda berikan kepada sebuah program dan system akan memproses lalu menampilkan ouputnya.

1.2.1 Cara mengambil input pada keyboard

Di dalam python sudah terdapat fungsi input() dan raw_input(). input() digunakan untuk masukan bernali angka sedangkan raw_input untuk masukan bernali teks.

Cara penggunaannya seperti ini nama_variabel = input ("masukkan teks")

Teks yang anda masukkan akan disimpan ke nama_variabel.

Contohnya seperti pada listing 1.1 :

```
1 # Mengambil input
2 nama = raw_input("nama: ")
3 umur = input("umur: ")
4
5 # Menampilkan output
6 print "Haii",nama,"umur kamu",umur,"tahun"
```

Listing 1.1 Contoh kode input output

Hasilnya seperti pada gambar 1.6:

```
C:\Users\dikasukmap\Documents\Proyek>python iouser.py
nama: Dika
umur: 19
Haii Dika umur kamu 19 tahun
```

Gambar 1.6 Hasil input

1.2.2 Menampilkan Variabel dan Teks

Anda bisa menampilkan variabel dan teks menggunakan perintah print seperti pada listing 1.2

```
1 nama = "Dika"
2 print "Haii",nama
```

Listing 1.2 Perintah menampilkan variabel dan teks

Hasilnya seperti pada gambar 1.7:

Tanda koma pada kode diatas merupakan sebuah spasi apabila kode dieksekusi.

```
C:\Users\dikasukmap\Documents\Proyek>python halo.py
Hai Dika
```

Gambar 1.7 Hasil menampilkan variabel dan teks

Tabel 1.1 Operator Aritmatika

Operator	Contoh	Keterangan
Penjumlahan +	$3 + 1 = 4$	Menjumlahkan masing - masing bilangan
Pengurangan -	$4 - 1 = 3$	Mengurangi nilai operan di sebelah kiri menggunakan operan di sebelah kanan
Perkalian *	$2 * 1 = 2$	Mengalikan dua bilangan
Pembagian /	$4 / 2 = 2$	Untuk membagi operan di sebelah kiri menggunakan operan di sebelah kanan
Sisa Bagi %	$13 \% 4 = 1$	Mendapatkan sisa pembagian dari operan di sebelah kiri operator ketika dibagi oleh operan di sebelah kanan
Pangkat **	$6 ** 2 = 36$	Memangkatkan operan disebelah kiri operator dengan operan di sebelah kanan operator
Pembagian Bulat //	$10 // 3 = 3$	Sama seperti pembagian. Hanya saja angka dibelakang koma dihilangkan

1.3 Operator Dasar

Operator python adalah simbol yang melakukan operasi pada satu atau lebih operan. Operan adalah variabel atau nilai yang digunakan untuk melakukan operasi. Operator pada Python dibagi menjadi beberapa jenis, yaitu :

1.3.1 Operator Aritmatika

Contoh penggunaannya seperti pada listing 1.3 :

```

1 #Penjumlahan
2 print(10 + 2)
3 pensil = 3
4 buku = 5
5 alattulis = pensil + buku
6 print(alattulis)
7
8 #Pengurangan
9 penghapus1 = 10
10 penghapus2 = 5
11 sisaPenghapus = penghapus1 - penghapus2
12 print(sisaPenghapus)
13
14 #Perkalian
15 panjang = 10
16 lebar = 5
17 luas = panjang * lebar
18 print(luas)
19
20 #Pembagian
21 roti = 16
22 anak = 4
23 kuePerAnak = roti / anak
24 print(kuePerAnak)
25
26 #Sisa Bagi / Modulus
27 operan1 = 14
```

Tabel 1.2 Operator Perbandingan

Operator	Contoh	Keterangan
Sama dengan =	2 == 2	bernilai True Jika masing-masing operan memiliki nilai yang sama, maka kondisi bernilai benar atau True.
Tidak sama dengan !=	2 != 2	bernilai False Akan menghasilkan nilai kebalikan dari kondisi sebenarnya.
Tidak sama dengan <>	2 <> 2	bernilai False Akan menghasilkan nilai kebalikan dari kondisi sebenarnya
Lebih besar dari >	3 > 2	bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, maka kondisi menjadi benar.
Lebih kecil dari <	2 < 3	bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, maka kondisi menjadi benar.
Lebih besar / sama dengan >=	3 >= 2	bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, atau sama, maka kondisi menjadi benar.
Lebih kecil / sama dengan =	2 <= 3	bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, atau sama, maka kondisi menjadi benar.

```

28 operan2 = 5
29 hasil = operan1 % operan2
30 print("Sisa bagi dari bilangan ", operan1, " dan ", operan2, " adalah "
      , hasil)
31
32 #Pangkat
33 operan3 = 6
34 operan4 = 2
35 hasilPangkat = operan3 ** operan4
36 print(hasilPangkat)
37
38 #Pembagian Bulat
39 print(10//3)
40 #10 dibagi 3 adalah 3.3333. Karena dibulatkan maka akan menghasilkan
    nilai 3

```

Listing 1.3 Contoh operator aritmatika

Hasilnya seperti pada gambar 1.8:

```
C:\Users\dikasukmap\Documents\Proyek>python operator_aritmatika.py
12
8
('Sisa penghapus Anda adalah ', 5)
50
('Setiap anak akan mendapatkan bagian kue sebanyak ', 4)
('Sisa bagi dari bilangan ', 14, ' dan ', 5, ' adalah ', 4)
64
3
```

Gambar 1.8 Hasil contoh operator aritmatika

1.3.2 Operator Perbandingan

Contoh penggunaannya seperti pada listing 1.4 :

```

1 # file: operator_perbandingan.py
2 j = input("Inputkan nilai j: ")
3 k = input("Inputkan nilai k: ")
4
5 # apakah j sama dengan k?
6 l = j == k
7 print "Apakah %d == %d: %r" % (j ,k ,l)
8
9 # apakah j < k?
10 l = j < k

```

Tabel 1.3 Operator Penugasan

Operator	Contoh	Penjelasan
Sama dengan =	A = 3	Memberikan nilai di kanan ke dalam variabel yang berada di sebelah kiri.
Tambah sama dengan +=	A += 3	Memberikan nilai variabel dengan nilai variabel itu sendiri ditambah dengan nilai di sebelah kanan.
Kurang sama dengan -=	A -= 3	Memberikan nilai variabel dengan nilai variabel itu sendiri dikurangi dengan nilai di sebelah kanan.
Kali sama dengan *=	A *= 3	Memberikan nilai variabel dengan nilai variabel itu sendiri dikalikan dengan nilai di sebelah kanan.
Bagi sama dengan /=	A /= 6	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan.
Sisa bagi sama dengan %=%	A %= 3	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan. Yang diambil nantinya adalah sisa baginya.
Pangkat sama dengan **=%	A **= 2	Memberikan nilai variabel dengan nilai variabel itu sendiri dipangkatkan dengan nilai di sebelah kanan.
Pembagian bulat sama dengan //=%	A //= 4	Membagi bulat operan sebelah kiri operator dengan operan sebelah kanan operator kemudian hasilnya disisakan ke operan sebelah kiri.

```

11 print "Apakah %d < %d: %r" % (j ,k ,l)
12
13 # apakah j > k?
14 l = j > k
15 print "Apakah %d > %d: %r" % (j ,k ,l)
16
17 # apakah j <= k?
18 l = j <= k
19 print "Apakah %d <= %d: %r" % (j ,k ,l)
20
21 # apakah j >= k?
22 l = j >= k
23 print "Apakah %d >= %d: %r" % (j ,k ,l)
24
25 # apakah j != k?
26 l = j != k
27 print "Apakah %d != %d: %r" % (j ,k ,l)

```

Listing 1.4 Contoh operator perbandingan

Hasilnya seperti pada gambar 1.9:

```
C:\Users\dikasukmap\Documents\Proyek>python operator_perbandingan.py
Inputkan nilai j: 2
Inputkan nilai k: 3
Apakah 2 == 3: False
Apakah 2 < 3: True
Apakah 2 > 3: False
Apakah 2 <= 3: True
Apakah 2 >= 3: False
Apakah 2 != 3: True
```

Gambar 1.9 Hasil contoh operator perbandingan

1.3.3 Operator Penugasan

Operator Penugasan digunakan untuk memodifikasi sebuah nilai yang ada pada variabel tertentu.

Contoh penggunaannya seperti pada listing 1.5 :

```

1 # file : operator_perbandingan.py
2 j = input("Inputkan nilai j: ")
3 k = input("Inputkan nilai k: ")
4
5 # apakah j sama dengan k?
6 l = j == k

```

Tabel 1.4 Operator Logika

Nama	Simbol
Logika And	and
Logika Or	or
Negasi/kebalikan	not

```

7 print "Apakah %d == %d: %r" % (j ,k ,1)
8
9 # apakah j < k?
10 l = j < k
11 print "Apakah %d < %d: %r" % (j ,k ,1)
12
13 # apakah j > k?
14 l = j > k
15 print "Apakah %d > %d: %r" % (j ,k ,1)
16
17 # apakah j <= k?
18 l = j <= k
19 print "Apakah %d <= %d: %r" % (j ,k ,1)
20
21 # apakah j >= k?
22 l = j >= k
23 print "Apakah %d >= %d: %r" % (j ,k ,1)
24
25 # apakah j != k?
26 l = j != k
27 print "Apakah %d != %d: %r" % (j ,k ,1)

```

Listing 1.5 Contoh operator penugasan

Hasilnya seperti pada gambar 1.10:

```
C:\Users\dikasukmap\Documents\Proyek>python operator_penugasan.py
masukkan nilai k: ?
Nilai j = 7
Nilai setelah ditambah 5:
j = 14
```

Gambar 1.10 Hasil contoh operator penugasan

1.3.4 Operator Logika

Operator logika digunakan untuk membuat operasi logika, seperti logika AND, OR, dan NOT.

Contoh penggunaannya seperti pada listing 1.6 :

```

1 #file operator_logika.py
2 d = True
3 i = False
4
5 # Logika AND
```

Tabel 1.5 Operator Bitwise

Nama	Simbol
AND	&
OR	—
XOR	^
Negasi	~
Left Shift	<<
Right Shift	>>

```

6 a = d and i
7 print "%r and %r = %r" % (d,i,a)
8
9 # Logika OR
10 a = d or i
11 print "%r or %r = %r" % (d,i,a)
12
13 # Logika Not
14 a = not d
15 print "not %r = %r" % (d,a)

```

Listing 1.6 Contoh operator logika

Hasilnya seperti pada gambar 1.11:

```
C:\Users\dikasukmap\Documents\Proyek>python operator_logika.py
True and False = False
True or False = True
not True = False
```

Gambar 1.11 Hasil contoh operator logika

1.3.5 Operator Bitwise

Operator bitwise digunakan untuk melakukan operasi dalam bentuk bit atau biner.

Contoh penggunaannya seperti pada listing 1.7 :

```

1 d = input("Masukan nilai a: ")
2 i = input("Masukan nilai b: ")
3
4 # Operasi AND
5 a = d & i
6 print "d & i = %s" % a
7
8 # Operasi OR
9 a = d | i
10 print "d | i = %s" % a
11
12 # Operasi XOR
13 a = d ^ i

```

```

14 print "d ^ i = %s" % a
15
16 # Operasi Not
17 a = ~d
18 print "~d = %s" % a
19
20 # Operasi shift left (tukar posisi biner)
21 a = d << i
22 print "d << i = %s" % a
23
24 # Operasi shift right (tukar posisi biner)
25 a = d >> i
26 print "d >> i = %s" % a

```

Listing 1.7 Contoh operator bitwise

Hasilnya seperti pada gambar 1.12:

```

C:\Users\dikasukmap\Documents\Proyek>python bitwise.py
Masukan nilai a: 3
Masukan nilai b: 4
d & i = 0
d | i = 7
d ^ i = 7
~d = -4
d << i = 48
d >> i = 0

```

Gambar 1.12 Hasil contoh operator bitwise

1.3.6 Prioritas Eksekusi Operator di Python

Berikut adalah prioritas yang dieksekusi di dalam perintah python. Maksudnya adalah di dalam python terdapat beberapa operator dan prioritasnya masing - masing. Tabel ?? merupakan prioritas python dari mulai yang pertama sampai dengan yang terakhir dalam proses pengeksekusian :

1.4 Perulangan / Loop

Secara umum, perintah pada bahasa pemrograman Python akan dijalankan secara berurutan. Pernyataan pertama dalam sebuah fungsi dijalankan pertama, lalu diikuti oleh yang kedua, dan seterusnya. Tetapi akan ada situasi dimana Anda harus menulis banyak kode, dimana kode tersebut sangat banyak. Jika dilakukan secara manual maka Anda hanya akan membuang-buang waktu dan tenaga. Untuk itu Anda perlu menggunakan pengulangan atau loop di dalam bahasa pemrograman Python.

Di dalam bahasa pemrograman Python pengulangan dibagi menjadi 3 bagian, yaitu :

Tabel 1.6 Prioritas Eksekusi Operator di Python

Operator	Keterangan
**	Aritmatika
~,+, -	Bitwise
*,/,,%	Arimatika
+, -	Aritmatika
>>, <<	Bitwise
&	Bitwise
^, —	Bitwise
<=,<,>,>=	Perbandingan
<>, ==, !=	Perbandingan
Err:509	Penugasan
Is, is not	Identitas
In, not in	Keanggotaan
Not, or, and	Logika

1.4.1 While Loop

Pada while loop pernyataan akan dieksekusi berkali-kali selama kondisi bernilai benar atau true.

Contoh penggunaannya seperti pada listing 1.8 :

```

1 count = 1
2 while (count < 9):
3     print ('The count is:', count)
4     count = count + 2
5
6 print ("Hai")

```

Listing 1.8 Koding while loop

1.4.2 For Loop

For Loop memiliki kemampuan untuk mengulangi item dari urutan apapun,misalnya string atau list.

Contoh penggunaannya seperti pada listing 1.9 :

```

1 #Contoh pengulangan for
2 buah = ["nanas", "apel", "jeruk"]
3 for makanan in buah:
4     print("Saya suka makan", makanan)

```

Listing 1.9 Koding for loop

1.4.3 Nested Loop

Nested loop memungkinkan terjadi sebuah pengulangan di dalam pengulangan lainnya.

Contoh penggunaannya seperti pada listing 1.10 :

```

1 i = 2
2 while(i < 100):
3     j = 2
4     while(j <= (i/j)):
5         if not(i%j): break
6         j = j + 1
7     if (j > i/j) : print(i, " is prime")
8 i = i + 1 print
9 print "haii"

```

Listing 1.10 Koding nested loop

1.5 Kondisi

1.5.1 Kondisi IF

Pada saat pengambilan keputusan (kondisi if) digunakan untuk mengantisipasi kondisi yang akan terjadi saat program dijalankan. Dan akan menentukan tindakan apa yang akan diambil sesuai dengan kondisi program pada saat dieksekusi.

Pada python ada beberapa kondisi diantaranya adalah if, else, dan elif. Kondisi if hanya bisa digunakan pada saat kondisi benar saja.

Contoh penggunaannya seperti pada listing 1.11 :

```

1 #Kondisi if adalah kondisi yang akan dieksekusi oleh program jika
  bernilai benar atau TRUE
2
3 nilai = 9
4
5 #jika kondisi benar/TRUE maka program akan mengeksekusi perintah
  dibawahnya
6 if(nilai > 6):
7     print("Selamat Ulang Tahun")
8
9 #jika kondisi salah/FALSE maka program tidak akan mengeksekusi
  perintah dibawahnya
10 if(nilai > 11):
11     print("Selamat Ulang Tahun")

```

Listing 1.11 Koding kondisi if

Dari contoh tersebut, jika if pertama dijalankan maka akan menampilkan “Selamat Ulang Tahun” sebanyak 1 kali, di statement ke 2 perintah print(“Selamat Ulang Tahun”) tidak akan dieksekusi karena statement bernilai salah.

1.5.2 Kondisi IF Else

Pengambilan kondisi IF Else tidak hanya mengeksekusi jika program yang telah dite-tapkan bernilai benar, tetapi kondisi IF Else juga menentukan tindakan apa yang diambil jika sebuah program bernilai salah atau tidak sesuai. Di dalam kondisi IF Else, perintah IF akan dijalankan jika pernyataan yang dieksekusi benar. Sedangkan jika pernyataan yang dieksekusi salah maka akan dijalankan adalah perintah Else.

Contoh penggunaannya seperti pada listing 1.12 :

```

1 #Koneksi if else adalah jika kondisibernilai benar maka akan
   dieksekusi pada IF, jika kondisi bernilai salah maka akan
   dieksekusi pada Else.
2 nilai=3
3 #Jika pernyataan pada if bernilai benar maka akan dieksekusi , tetapi
   jika salah maka else yang akan dieksekusi .
4
5 If(nilai > 7):
6   print("Selamat Ulang Tahun")
7 else:
8   print("Maaf anda tidak ulang tahun hari ini")
```

Listing 1.12 Koding kondisi if else

Pada program diatas pernyataan bernilai salah, maka yang akan dieksekusi adalah perintah else dan akan menampilkan “Maaf anda tidak ulang tahun hari ini”.

1.5.3 Kondisi Elif

Kondisi if elif merupakan percabangan logikan dari “kondisi if”. Dengan Elif anda bisa membuat kondisi program menjadi banyak pilihan. Dan kondisi ini akan menyeleksi kemungkinan yang akan terjadi. Kondisi ini memiliki banyak pilihan, ini merupakan perbedaan dari kondisi elif dan kondisi if else.

Contoh penggunaannya seperti pada listing 1.13 :

```

1 #Contoh penggunaan kondisi elif
2
3 hari_ini = "Minggu"
4
5 if(hari_ini == "Senin"):
6   print("Saya akan bekerja")
7 elif(hari_ini == "Selasa"):
8   print("Saya akan bekerja")
9 elif(hari_ini == "Rabu"):
10  print("Saya akan bekerja")
11 elif(hari_ini == "Kamis"):
12  print("Saya akan bekerja")
13 elif(hari_ini == "Jumat"):
14  print("Saya akan bekerja")
15 elif(hari_ini == "Sabtu"):
16  print("Saya akan bekerja")
17 elif(hari_ini == "Minggu"):
18  print("Saya akan libur")
```

Listing 1.13 Koding kondisi elif

Karena pernyataan diatas adalah hari minggu maka akan dieksekusi pada saat hari minggu dan akan menampilkan “ Saya akan libur”.

1.6 Pembacaan Error

Pada saat membuat suatu program, terkadang program tidak berjalan seperti yang diinginkan. Anda harus mengetahui error apa yang terjadi dengan program yang telah dibuat. Berikut adalah 2 macam error yang terjadi pada python.

1.6.1 Kesalahan sintak

Biasanya yang paling mudah dikenali, kesalahan sintaksis terjadi ketika Anda membuat kesalahan ketik. Tidak mengakhiri pernyataan if dengan titik dua adalah contoh kesalahan sintak, seperti salah mengeja kata kunci Python (mis. Menggunakan whille alih-alih sementara). Kesalahan sintak biasanya muncul pada waktu kompliasi dan dilaporkan oleh interpreter. Contoh penggunaannya seperti pada listing 1.14 :

```

1 x = int(input('Enter a number: '))
2
3 whille x%2 == 0:
4     print('You have entered an even number.')
5 else:
6     print ('You have entered an odd number.')

```

Listing 1.14 Koding kesalahan sintak

Hasilnya seperti pada gambar 1.13: Kesalahan terdapat pada “whille”, perintah

```

C:\Users\dikasukmap\Documents\Proyek>python error.py
      File "error.py", line 3
        whille x%2 == 0:
               ^
SyntaxError: invalid syntax

```

Gambar 1.13 Hasil menangkap pengecualian khusus

tersebut harusnya tidak menggunakan doubel l tetapi “while”.

1.6.2 Kesalahan Logika

Juga disebut kesalahan semantik, kesalahan logis menyebabkan program berperilaku tidak benar, tetapi mereka biasanya tidak crash program. Tidak seperti program dengan kesalahan sintaks, program dengan kesalahan logika dapat dijalankan, tetapi tidak beroperasi sebagaimana dimaksud.

Pertimbangkan contoh kesalahan logis seperti pada listing 1.15:

```

1 x = float(input('Enter a number: '))
2 y = float(input('Enter a number: '))

```

```

3
4 z = x+y/2
5 print ('The average of the two numbers you have entered is:',z)

```

Listing 1.15 Koding kesalahan logika

Contoh di atas harus menghitung rata-rata dari dua angka yang dimasukkan pengguna. Tetapi, karena urutan operasi dalam aritmatika (pembagian dievaluasi sebelum penambahan) program tidak akan memberikan jawaban yang benar seperti pada gambar 2.6:

```
C:\Users\dikasukmap\Documents\Proyek>python operator_logika.py
True and False = False
True or False = True
not True = False
```

Gambar 1.14 Hasil program kesalahan logika

Untuk memperbaiki masalah ini, cukup menambahkan tanda kurung: $z = (x + y) / 2$ Ini adalah hasil yang benar seperti pada gambar 1.15:

```
C:\Users\dikasukmap\Documents\Proyek>python logicalerror.py
Enter a number: 4
Enter a number: 3
('The average of the two numbers you have entered is:', 3.5)
```

Gambar 1.15 Hasil setelah di benarkan

1.7 Try to Except/pengecualian

Python memiliki banyak exceptions bawaan yang memaksa program Anda untuk menghasilkan kesalahan ketika ada sesuatu yang salah di dalamnya. Ketika exceptions ini terjadi, itu menyebabkan proses saat ini berhenti dan meneruskannya ke proses panggilan sampai ditangani. Jika tidak ditangani, program ini akan macet.

Misalnya, jika fungsi A memanggil fungsi B yang pada gilirannya memanggil fungsi C dan pengecualian terjadi di fungsi C. Jika tidak ditangani dalam C, pengecualian beralih ke B dan kemudian ke A. Jika tidak pernah ditangani, pesan kesalahan dilemparkan dan program ini terhenti secara tiba-tiba.

1.7.1 Menangkap Pengecualian dengan Python

Dalam Python, pengecualian dapat ditangani menggunakan pernyataan coba. Operasi kritis yang dapat meningkatkan pengecualian ditempatkan di dalam klausula coba dan kode yang menangani pengecualian ditulis dalam kecuali klausula.

Contoh penggunaannya seperti pada listing 1.16 :

```

1 # import module sys to get the type of exception
2 import sys

```

```

3 randomList = [ 'a' , 0 , 2]
4
5 for entry in randomList:
6     try :
7         print("The entry is", entry)
8         r = 1/int(entry)
9         break
10    except:
11        print("Oops!",sys.exc_info()[0],"occured.")
12        print("Next entry .")
13        print()
14
15 print("The reciprocal of",entry,"is",r)

```

Listing 1.16 Koding exception

Hasilnya seperti pada gambar 1.16:

```

C:\Users\dikasukmap\Documents\Proyek>python exception.py
('The entry is', 'a')
('Oops!', <type 'exceptions.ValueError'>, 'occured.')
Next entry .
()
('The entry is', 0)
('Oops!', <type 'exceptions.ZeroDivisionError'>, 'occured.')
Next entry .
()
('The entry is', 2)
('The reciprocal of', 2, 'is', 0)

```

Gambar 1.16 Hasil try to except

Dalam program ini, anda harus mengulang sampai pengguna memasukkan bilangan bulat yang memiliki timbal balik yang valid. Bagian yang dapat menyebabkan pengecualian ditempatkan di dalam blok try.

Jika tidak ada pengecualian terjadi, kecuali blok dilewati dan aliran normal berlanjut. Tetapi jika ada pengecualian terjadi, itu ditangkap oleh blok kecuali. System mencetak nama pengecualian menggunakan fungsi ex.info () di dalam modul sys dan meminta pengguna untuk mencoba lagi. Anda dapat melihat bahwa nilai 'a' dan '1,3' menyebabkan ValueError dan '0' menyebabkan ZeroDivisionError.

1.7.2 Menangkap Pengecualian Khusus dengan Python

Dalam contoh di atas, pengecualian dalam klausanya kecuali tidak disebutkan. Ini bukan praktik pemrograman yang baik karena akan menangkap semua pengecualian dan menangani setiap kasus dengan cara yang sama. Anda dapat menentukan pengecualian mana yang dikecualikan klausanya kecuali.

Klausanya coba dapat memiliki sejumlah kecuali klausanya untuk menanganiannya secara berbeda tetapi hanya satu yang akan dieksekusi jika terjadi pengecualian. Anda bisa menggunakan tupel nilai untuk menentukan beberapa pengecualian dalam klausanya kecuali.

Contoh penggunaannya seperti pada listing 1.17 :

```

1 try :
2     # do something
3     pass
4
5 except ValueError:
6     # handle ValueError exception
7     pass
8
9 except (TypeError, ZeroDivisionError):
10    # handle multiple exceptions
11    # TypeError and ZeroDivisionError
12    pass
13
14 except:
15     # handle all other exceptions
16     pass

```

Listing 1.17 Koding special exception

1.7.3 Meningkatkan Pengecualian

Dalam pemrograman Python, pengecualian dimunculkan ketika kesalahan yang sesuai terjadi pada waktu berjalan, tetapi dapat dengan paksa menaikkannya menggunakan peningkatan kata kunci. Anda juga bisa memberikan nilai secara opsional pada pengecualian untuk mengklarifikasi mengapa pengecualian itu dinaikkan.

Contoh penggunaannya seperti pada listing 1.18 :

```

1 >>> raise KeyboardInterrupt
2 Traceback (most recent call last):
3 ...
4 KeyboardInterrupt
5
6 >>> raise MemoryError("This is an argument")
7 Traceback (most recent call last):
8 ...
9 MemoryError: This is an argument
10
11 >>> try :
12 ...     a = int(input("Enter a positive integer: "))
13 ...     if a <= 0:
14 ...         raise ValueError("That is not a positive number!")
15 ... except ValueError as ve:
16 ...     print(ve)
17 ...
18 Enter a positive integer: -2
19 That is not a positive number!

```

Listing 1.18 Koding meningkatkan pengecualian

1.7.4 Try Finally

Pernyataan coba dengan Python dapat memiliki klausa finally opsional. Klausa ini dieksekusi apa pun yang terjadi, dan umumnya digunakan untuk melepaskan

sumber daya eksternal. Misalnya, anda dapat terhubung ke pusat data jarak jauh melalui jaringan atau bekerja dengan file atau bekerja dengan Graphical User Interface (GUI).

Dalam semua keadaan ini, anda harus membersihkan sumber daya yang pernah digunakan, apakah itu berhasil atau tidak. Tindakan-tindakan ini (menutup file, GUI atau memutuskan sambungan dari jaringan) dilakukan pada klausula terakhir untuk menjamin eksekusi.

Contoh penggunaannya seperti pada listing 1.19 :

```
1 try :  
2     f = open("test.txt",encoding = 'utf-8')  
3     # perform file operations  
4 finally:  
5     f.close()
```

Listing 1.19 Koding try finnaly

Tipe konstruksi ini memastikan file ditutup bahkan jika pengecualian terjadi.

BAB 2

AKUISISI DATA MENGGUNAKAN SENSOR EEG

2.1 Membuat Fungsi Dalam Satu File dan Cara Menggunakannya

Pada bagian ini terdapat tutorial cara membuat function yang berhubungan dengan “Akuisisi Data Menggunakan Sensor EEG”. Di bagian awal ini terdapat beberapa function yang disipan dalam satu file diantaranya adalah function untuk membuat tampilan, tampilan yang ada di tutorial ini terbagi ke dalam :

1. Membuat tampilan untuk mencari file.
2. Membuat tampilan untuk memilih file.
3. Membuat tampilan untuk mengubah ke line graph.
4. Membuat tampilan untuk mengubah ke bar graph.
5. Membuat tampilan “file tidak ada” jika file belum dipilih.

Setelah itu ada function untuk :

1. memilih file.

2. mengconvert file ke line graph.

3. Mengkonversi file ke bar graph.

Pada bagian ini function akan dieksekusi oleh satu fungsi dalam file yang sama.

2.1.1 Function untuk membuat tampilan awal

Seperti pada listing 2.1:

```

1 #====Modul yang perlu diimport====#
2 from Tkinter import *
3 import matplotlib.pyplot as plt
4 import tkFileDialog as filedialog
5
6 x, y = [], []
7
8 class Plot:
9     def __init__(self, master):
10         self.master=master
11         self.master.title("File-to-Graph Converter")
12         self.master.minsize(200,100)
13
14         self.judul = Label(self.master, text = "File-to-Graph
15             Converter")
16         self.judul.grid(row=0,column=1)
17         self.label1 = Label(self.master, text = "Pilih File: ", anchor
18 =E, justify=RIGHT)
19         self.chooseFile = Button(self.master, text="Browse",command=
20             self.choose_file)
21         self.label1.grid(row=1,column=0,ipadx=25)
22         self.chooseFile.grid(row=1,column=1,ipadx=20)
23         self.txt=Label(self.master, text="*.txt",anchor=W, justify=
24             LEFT)
25         self.txt.grid(row=1,column=2,ipadx=25)
26
27         self.keterangan = Label(self.master, text = "Ubah ke:")
28         self.keterangan.grid(row=2, column=1)
29
30         self.lineGraph = Button(self.master, text = "Line Graph",
31             command=self.line_Graph)
32         self.lineGraph.grid(row=3,column=0,ipadx=15)
33         self.barGraph = Button(self.master, text = "Bar Graph",
34             command=self.bar_Graph)
35         self.barGraph.grid(row=3,column=1,ipadx=15)
36
37         self.adaFile = Label(self.master, text="File belum ada")
38         self.adaFile.grid(row=4,column=1)
39
40     def choose_file(self):
41         file_name = filedialog.askopenfilename()
42         if not file_name:
43             return
44         for line in open(file_name, 'r'):
45             values = [float(s) for s in line.split()]
46             x.append(values[0])

```

```

41     y.append(values[0])
42     self.adaFile["text"] = "File sudah tersedia!"
43
44 def line_Graph(self):
45     plt.figure(num='Line Graph | File-to-Graph Converter')
46     plt.plot(x, marker="o")
47     plt.xlabel("Banyak")
48     plt.ylabel("Panjang Gelombang")
49     plt.show()
50
51 def bar_Graph(self):
52     plt.figure(num='Bar Graph | File-to-Graph Converter')
53     plt.bar(x,y)
54     plt.xlabel("Banyak")
55     plt.ylabel("Panjang Gelombang")
56     plt.show()
57
58
59 def main():
60     root_window = Tk()
61     program = Plot(root_window)
62     root_window.mainloop()
63
64
65 if __name__ == '__main__':
66     main()

```

Listing 2.1 Tampilan awal

Langkah pertama membuat tampilan

1. Membuat tampilan untuk mencari file. Seperti pada listing 2.2:

```

1 self.judul = Label(self.master, text = "File-to-Graph Converter")
2         self.judul.grid(row=0,column=1)
3         self.label1 = Label(self.master, text = "Pilih File: ", anchor=E, justify=RIGHT)
4         self.chooseFile = Button(self.master, text="Browse", command=self.choose_file)
5             self.label1.grid(row=1,column=0,ipadx=25)
6             self.chooseFile.grid(row=1,column=1,ipadx=20)
7             self.txt=Label(self.master, text="*.txt", anchor=W, justify=LEFT)
8                 self.txt.grid(row=1,column=2,ipadx=25)

```

Listing 2.2 Tampilan untuk mencari file

2. Membuat tampilan untuk memilih file. Seperti pada listing 2.3:

```

1 self.judul = Label(self.master, text = "File-to-Graph Converter")
2         self.judul.grid(row=0,column=1)
3         self.label1 = Label(self.master, text = "Pilih File: ", anchor=E, justify=RIGHT)
4         self.chooseFile = Button(self.master, text="Browse", command=self.choose_file)
5             self.label1.grid(row=1,column=0,ipadx=25)

```

```

6     self.chooseFile.grid(row=1,column=1,ipadx=20)
7     self.txt=Label(self.master, text="*.txt", anchor=W,
8         justify=LEFT)
      self.txt.grid(row=1,column=2,ipadx=25)

```

Listing 2.3 Tampilan untuk memilih file

3. Membuat tampilan untuk mengubah ke line graph. Seperti pada listing 2.4:

```

1 self.keterangan = Label(self.master, text = "Ubah ke:")
2     self.keterangan.grid(row=2, column=1)
3
4     self.lineGraph = Button(self.master, text = "Line Graph",
5         command=self.line_Graph)
      self.lineGraph.grid(row=3,column=0,ipadx=15)

```

Listing 2.4 Tampilan line graph

4. Membuat tampilan untuk mengubah ke bar graph. Seperti pada listing 2.5:

```

1 self.keterangan = Label(self.master, text = "Ubah ke:")
2     self.keterangan.grid(row=2, column=1)
3
4     self.barGraph = Button(self.master, text = "Bar Graph",command=
5         self.bar_Graph)
      self.barGraph.grid(row=3,column=1,ipadx=15)

```

Listing 2.5 Tampilan bar graph

5. Kemudian function untuk memilih file. Seperti pada listing 2.6:

```

1 def choose_file(self):
2     file_name = filedialog.askopenfilename()
3     if not file_name:
4         return
5     for line in open(file_name, 'r'):
6         values = [float(s) for s in line.split()]
7         x.append(values[0])
8         y.append(values[0])
9     self.addData["text"]="File sudah tersedia!"

```

Listing 2.6 Tampilan untuk memilih file

- Buat askopenfilename() yang bertujuan untuk memberikan pertanyaan file mana yang akan dipilih.
- Value nya dibuat float dengan menggunakan method split agar dapat mengembalikan lagi string line yang dipanggil.
- Membuat function open agar bisa menampilkan menu open file.
- Didefinisikan menjadi return jika file tidak ada dan bermaksud agar perintah tersebut bisa diulang-ulang.
- Membuat x.append dan y.append untuk value di sumbu x dan sumbu y.

- Membuat print file sudah tersedia dengan self dengan value “file sudah tersedia”.

6. Membuat function Line Graph. Seperti pada listing 2.7:

```

1 def line_Graph(self):
2     plt.figure(num='Line Graph | File-to-Graph Converter')
3     plt.plot(x, marker="o")
4     plt.xlabel("Banyak")
5     plt.ylabel("Panjang Gelombang")
6     plt.show()

```

Listing 2.7 Tampilan function line graph

- Definisikan line_graph menjadi self.
- Membuat plt.figure dan variable num dengan value line graph dan file graph to converter untuk mengconvert data ke graph.
- Buat plt.plot dengan value maker sebagai tanda sumbu x dan sumbu y.
- Buat plt.xlabel sebagai title yang memberitahu panjang gelombang.
- Buat plt.ylabel sebagai title yang memberitahu banyak gelombang.
- Pada bagian ini data dibuat ke plt atau plot karena data akan disajikan ke dalam bentuk grafik.
- Masing-masing plot didefinisikan sehingga tampilan grafik akan sesuai dan dapat dimengerti.
- Buat plt.show untuk menampilkan hasil.

7. Kemudian mendefinisikan bar graph. Seperti pada listing 2.8:

```

1 def bar_Graph(self):
2     plt.figure(num='Bar Graph | File-to-Graph Converter')
3     plt.bar(x,y)
4     plt.xlabel("Banyak")
5     plt.ylabel("Panjang Gelombang")
6     plt.show()

```

Listing 2.8 Tampilan function bar graph

- Definisikan bar_graph menjadi self.
- Membuat plt.figure dan variable num dengan value line graph dan file graph to converter untuk mengconvert data ke graph.
- Buat plt.plot dengan value maker sebagai tanda sumbu x dan sumbu y.
- Buat plt.xlabel sebagai title yang memberitahu panjang gelombang.
- Buat plt.ylabel sebagai title yang memberitahu banyak gelombang.
- Pada bagian ini data dibuat ke plt atau plot karena data akan disajikan ke dalam bentuk grafik.
- Masing-masing plot didefinisikan sehingga tampilan grafik akan sesuai dan dapat dimengerti.
- Buat plt.show untuk menampilkan hasil.

2.2 Membuat Fungsi Pada File Terpisah Dari Cara Penggunaannya

Pada bagian ini akan diberitahu bagaimana cara membuat function secara terpisah dalam file yang berbeda-beda tetapi dapat dijalankan oleh satu function. Function tersebut diantaranya:

1. Function membuka dan membaca file.
2. Function Untuk Menjalankan.
3. Function Menampilkan Data
4. Function Untuk Menampilkan Gelombang.
5. Function Menampilkan Gelombang Dengan Marker.
6. Function Membuat Titile Pada Grafik.
7. Function Membuat Keterangan Pada Sumbu X.
8. Function Membuat Keterangan Pada Sumbu Y.
9. Membuat fungsi untuk menampilkan row di sumbu x dan sumbu y.
10. Function Membuat warna pada row.
11. Membuat scatter

Pada bagian dua, fungsi dibuat secara terpisah pada file yang berbeda-beda dan hanya dapat menjalankan fungsi itu saja tidak bisa berbarengan dengan fungsi yang lain. Dan perintah untuk menjalankannya ada di dalam fungsi tersebut.

1. Function membuka dan membaca file. Seperti pada listing 2.9:

```
1 f = open("hasil.txt", "r")
2 print(f.read())
```

Listing 2.9 Tampilan function membuka dan membaca file

- Buat f sebagai objek/variable yang menampung isi file.
- Kemudian pilih file yang akan diambil sebagai data.
- Buat print dan f.read agar data kemudian dibaca dan dicetak.

2. Function Untuk Menjalankan. Seperti pada listing 5.2:

```
1 import mindwave, time
2
3 headset = mindwave.Headset('COM4', '1425')
4 time.sleep(1)
5
6 headset.connect()
7 print("Connecting...")
```

```

8
9 while headset.status != 'connected':
10    time.sleep(1)
11    if headset.status == 'standby':
12        headset.connect()
13        print "Retrying connect..."
14    print "Connected."
15
16 while True:
17     print headset.raw_value
18     time.sleep(1)

```

Listing 2.10 main.py

- Pada bagian ini awalnya kita mengimport mindwave, maksudnya adalah agar dapat mengambil data di alat pembaca sinyal.
- Setelah itu membuat variable headset yang berisikan data yang ada di computer, headset tadi kita isi dengan COM4 sesuai dengan computer yang anda gunakan.

Seperti pada listing 2.11:

```

1 headset = mindwave.Headset('COM4', '1425')
2 time.sleep(1)

```

Listing 2.11 Pilih COM

- Setelah itu membuat fungsi koneksi atau menghubungkan Seperti pada listing 2.12:

```

1 headset.connect()
2 print "Connecting ..."
3
4 while headset.status != 'connected':
5     time.sleep(1)
6     if headset.status == 'standby':
7         headset.connect()
8         print "Retrying connect..."
9 print "Connected."

```

Listing 2.12 Koneksi

- Kita definisikanJika data di mindwave belum terhubung maka laptop akan mencoba menghubungkan kembali, jika sudah ada maka akan muncul “connected”.

3. Function Menampilkan Data Seperti pada listing 2.13:

```

1 from PIL import Image, ImageDraw
2 img = Image.open('blank.png')
3 draw_img = ImageDraw.Draw(img)
4
5 data = ['4', '5', '87', '1', '44', '83', '93', '2', '54', '84', '100', '64']
6 x = 0
7

```

```

8 for i in data:
9     x = x + 30
10    y = 200 - int(i)
11    draw_img.line((x,200,x,y), width=10, fill=(255,0,0,255))
12
13 img.show()

```

Listing 2.13 Function Menampilkan Data

- Membuat img = img.open sebagai function untuk memilih gambar, kemudian dipilih gambar yang akan ditampilkan.
- Membuat variable draw.
- Data diambil dan dimasukan.
- Data yang ada akan diimport ke dalam bentuk gambar dan ditampilkan.

4. Function Untuk Menampilkan Gelombang. Seperti pada listing 2.14:

```

1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12
13 plt.plot(x)
14
15
16 plt.show()

```

Listing 2.14 Function Menampilkan Gelombang

- Pertama-tama import dulu data matplotlib menjadi plot agar mudah mendefinisikan data menjadi sebuah plot.
- Setelah itu buatlah variable plot agar dapat membaca file csv.
- Csv.reader agar python dapat membaca format csv.
- Kemudian buat plt.show agar data gelombang dalam bentuk grafik dapat ditampilkan.

5. Function Menampilkan Gelombang Dengan Marker. Seperti pada listing 2.15:

```

1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6

```

```

7 with open('hasil.txt', 'r') as csvfile:
8 plots= csv.reader(csvfile , delimiter=',')
9 for row in plots:
10 x.append(int(row[0]))
11
12
13 plt.plot(x, marker='o')
14
15
16 plt.show()

```

Listing 2.15 Function Menampilkan Gelombang Dengan Maker

- Pertama-tama import dulu data matplotlib menjadi plot agar mudah mendefinisikan data menjadi sebuah plot.
- Setelah itu buatlah variable plot agar dapat membaca file csv.
- Kemudian buat fungsi plot dan isi valuenya dengan (x, marker='o') untuk membuat sumbu x dan o.
- Kemudian buat plt.show agar data gelombang dalam bentuk grafik dapat ditampilkan.

6. Function Membuat Title Pada Grafik. Seperti pada listing 2.16:

```

1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile , delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12
13 plt.plot(x, marker='o')
14
15 plt.title('Data Gelombang Otak')
16
17
18 plt.show()

```

Listing 2.16 Function Membuat Title Pada Grafik

- Pertama-tama import dulu data matplotlib menjadi plot agar mudah mendefinisikan data menjadi sebuah plot.
- Setelah itu buatlah variable plot agar dapat membaca file csv.
- Kemudian buat fungsi plot da nisi valuenya dengan (x, marker='o') untuk membuat sumbu x dan o.
- Kemudian buat fungsi plt.title da nisi valuenya sesuai yang diinginkan.

7. Function Membuat Keterangan Pada Sumbu X. Seperti pada listing 2.17:

```

1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12
13 plt.plot(x, marker='o')
14
15 plt.title('Data Gelombang Otak')
16
17 plt.xlabel('Banyak Data')
18
19 plt.show()

```

Listing 2.17 xlabel

- Pertama-tama import dulu data matplotlib menjadi plot agar mudah mendefinisikan data menjadi sebuah plot.
- Setelah itu buatlah variable plot agar dapat membaca file csv.
- Kemudian buat fungsi plot da nisi valuenya dengan (x, marker='o') untuk membuat sumbu x dan o.
- Kemudian buat fungsi plt.title da nisi valuenya sesuai yang diinginkan.
- Buat plt.xlabel untuk keterangan di sumbu x.

8. Function Pembuatan Rows dan Scatter Seperti pada listing 2.18:

```

1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11         y.append(int(row[0]))
12
13
14 plt.scatter(x,y, facecolor='red')
15
16 plt.title('Data Gelombang Otak')
17
18 plt.xlabel('Banyak Data')
19 plt.ylabel('Gelombang otak')
20

```

21 plt.show()

Listing 2.18 Warna Scatter

- Pertama-tama import dulu data matplotlib menjadi plot agar mudah mendefinisikan data menjadi sebuah plot.
- Setelah itu buatlah variable plot agar dapat membaca file csv.
- Kemudian buat fungsi plot da nisi valuenya dengan (x, marker='o') untuk membuat sumbu x dan o.
- Setelah itu buat fungsi plt.title untuk judul di atas.
- Buat plt.xlabel untuk keterangan di sumbu x.
- Buat plt.ylabel untuk keterangan di sumbu y.
- Setelah itu buat fungsi x.append untuk sumbu x.
- Buat fungsi y.append untuk sumbu y agar di setiap sumbu terdapat row tingkatan jumlah data.
- Buat plt.hist dengan value facecolor dan definisikan warnanya.
- Setelah itu membuat fungsi scatter untuk sumbu x.
- Setelah itu membuat fungsi scatter untuk sumbu y agar gelombang dan row dapat dibaca dengan pas.
- Tambahkan scatter facecolor pada x dan y.

2.3 Membuat Kelas dan Cara Instalasi Kelas Pada Program Utama

Pada bagian ini akan dibahas cara membuat kelas dan cara memanggil kelas tersebut pada program utama. Pertama-tama buatlah kelas terlebih dahulu.

Seperti pada listing 2.19:

```

1 class Product:
2     __vendor_message = "Ini adalah rahasia"
3     name = ""
4     price = ""
5     size = ""
6     unit = ""
7
8     def __init__(self, name):
9         print "Ini adalah constructor"
10        self.name = name
11        self.unit = "ml"
12        self.size = 250
13
14     def get_vendor_message(self):
15         print self.__vendor_message
16
17     def set_price(self, price):
18         self.price = price

```

Listing 2.19 Membuat Kelas

Langkah selanjutnya buatlah dua objek yang sama-sama memanggil class tersebut. Seperti pada listing 2.20:

```

1 p = Product("Ultora Milek")
2 p.set_price(5500)
3
4 print "%s dengan ukuran %s %s harganya Rp. %d" % (p.name, p.size, p.
   unit, p.price)
5 # print p.__vendor_message
6
7 p.get_vendor_message()
8
9 p1 = Product("Indomilek")
10 p1.set_price(5000)
11
12 print "%s dengan ukuran %s %s harganya Rp. %d" % (p.name, p.size, p.
   unit, p.price)

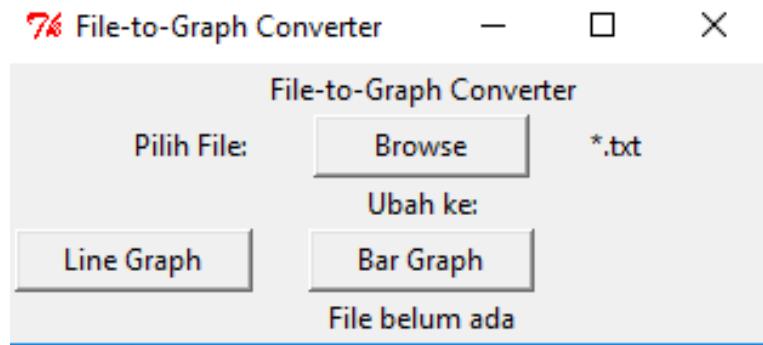
```

Listing 2.20 Membuat Objek Kelas

2.4 Hasil Pengujian

2.4.1 Tampilan awal

Hasilnya seperti pada gambar 2.1:



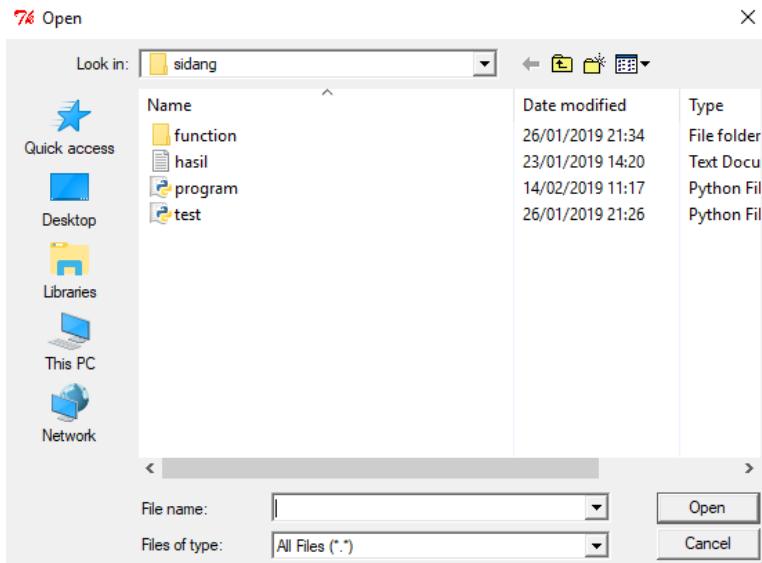
Gambar 2.1 Tampilan awal

2.4.2 Tampilan Pilih File

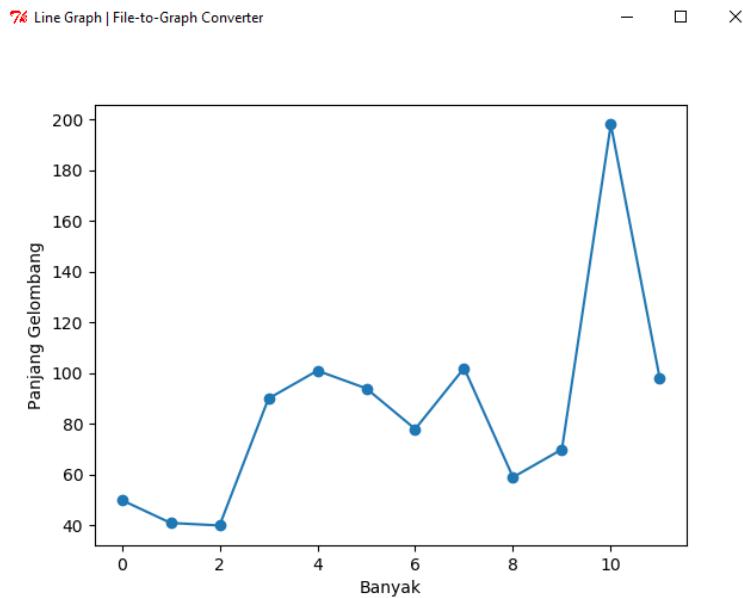
Hasilnya seperti pada gambar 2.2.

2.4.3 Tampilan line graph

Hasilnya seperti pada gambar 2.3:



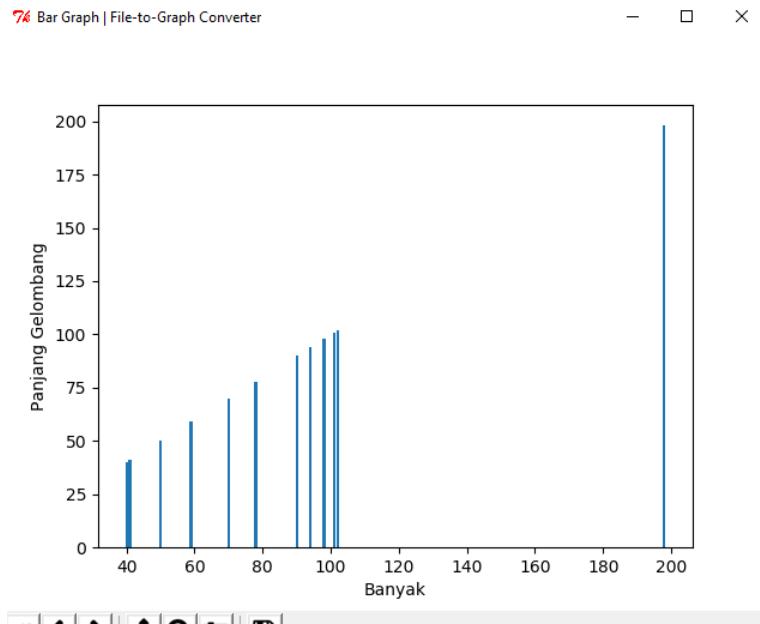
Gambar 2.2 Tampilan pilih file



Gambar 2.3 Data line graph

2.4.4 Tampilan Bar Graph

Hasilnya seperti pada gambar 2.4:



Gambar 2.4 Data bar graph

2.5 PENANGANAN ERROR PADA PYTHON

2.5.1 Penanganan error

Saat menjalankan program terkadang program tidak jalan/tidak dapat dieksekusi, untuk menangani hal tersebut harus diketahui terlebih dahulu errornya, setelah error diketahui maka program dapat dijalankan kembali. Error yang terjadi pada saat menjalankan program di python terbagi menjadi dua yaitu error pada penulisan syntax dan error pada logika yang diperintahkan dalam program. Karena suatu python tidak akan mengerti perintah yang dimasukkan jika program tersebut terdapat kesalahan dan tidak dapat dimengerti oleh computer.

1. Kesalahan syntax

Kesalahan syntax biasanya terjadi dalam kesalahan penulisan, jika syntax tidak ada dalam python dan tidak bisa dipahami maka python akan menunjukkan adanya kesalahan dalam program tersebut. Untuk mengatasi error dalam penulisan syntax maka pembuat program harus lebih teliti lagi dalam pengetikan saat memberikan perintah di python.

Berikut adalah contoh error yang terjadi karena kesalahan syntax seperti pada gambar 2.5

```
File "C:/Users/dzihan/Desktop/sidang/untitled1.py", line 10
    whille x%2 == 0:
           ^
SyntaxError: invalid syntax
```

Gambar 2.5 Kesalahan sintak

Gambar 2.5 adalah contoh error yang terjadi yang disebabkan oleh kesalahan dalam penulisan syntax. Hal tersebut terjadi karena dalam program tersebut terdapat ‘kata’ atau perintah yang tidak dapat dipahami oleh komputer.

Untuk menangani hal tersebut maka periksa lagi lagi atau klik errornya seperti pada listing 2.21:

```
1 x = int(input('Enter a number: '))
2
3 while x%2 == 0:
4     print('You have entered an even number.')
5 else:
6     print ('You have entered an odd number.')
```

Listing 2.21 Solusi Kesalahan Sintak

2. Kesalahan logika Selain kesalahan syntax ada juga kesalahan logika. Kesalahan logika terjadi akibat terdapat logika yang tidak dapat dipahami dalam program yang dibuat seperti pada gambar 2.6

```
Enter a number: 1
Enter a number: 2
The average of the two numbers you have entered is: 1.5
In [11]: runfile('C:/Users/dzihan/Desktop/sidang/untitled1.py', wdir='C:/Users/dzihan/Desktop/sidang')
Enter a number: 2
Enter a number: 3
The average of the two numbers you have entered is: 3.5
In [12]: |
```

Gambar 2.6 Kesalahan logika

Untuk memperbaiki error tersebut cukup dengan menambahkan tanda kurung dalam rumusnya seperti pada listing 2.22:

```
1 x = float(input('Enter a number: '))
2 y = float(input('Enter a number: '))
3
4 z = x+y/2
5 print ('The average of the two numbers you have entered is:',z)
6
```

```

7
8 x = float(input('Enter a number: '))
9 y = float(input('Enter a number: '))
10
11 z = (x+y)/2
12 print ('The average of the two numbers you have entered is:',z)
:
```

Listing 2.22 Solusi Kesalahan Logika

2.5.2 Fungsi memilih line graph

Seperti pada listing 2.23:

```

1 from Tkinter import *
2 import tkFileDialog as filedialog
3
4 def line_Graph(self):
5     plt.figure(num='Line Graph | File-to-Graph Converter')
6     plt.plot(x, marker="o")
7     plt.xlabel("Banyak")
8     plt.ylabel("Panjang Gelombang")
9     plt.show()
```

Listing 2.23 Line Graph

Seperti pada gambar 2.7

```
D:\Kuliah\Semester 3\sidang>python program.py
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Python27\lib\lib-tk\Tkinter.py", line 1541, in __call__
    return self.func(*args)
  File "program.py", line 44, in line_Graph
    plt.figure(num='Line Graph | File-to-Graph Converter')
NameError: global name 'plt' is not defined
```

Gambar 2.7 Error plt

Pada bagian awal terdapat exception pada bagian tkinter, karena untuk menjalankan fungsi plot diperlukan adanya sebuah import terlebih dahulu dari matplotlib agar data dapat dibaca dan dikonversi ke plot.

Pada bagian ini terdapat error yang menunjukkan bahwa global name dengan nama plt tidak terdefinisikan. Dalam fungsi pengambilan data, data diharuskan diimport dulu ke plot dengan menggunakan matplotlib agar data dapat disajikan dalam bentuk plot. Agar lebih mudah didefinisikan, maka matplotlib menggunakan “as” atau alias sebagai plt.

Untuk menangani error tersebut ada beberapa langkah yang harus dikerjakan :

1. Import matplotlib.pyplot
2. buatlah “as” atau alias pada bagian import matplotlib agar fungsi yang ada di line_graph dapat mendefinisikan plt tersebut sebagai matplotlib.

3. Pada bagian fungsi line_graph harus didefinsikan sama seperti alias yang telah ditentukan Seperti pada listing 5.17:

```

1 from Tkinter import *
2 import matplotlib.pyplot as plt
3 import tkFileDialog as filedialog
4
5 def line_Graph(self):
6     plt.figure(num='Line Graph | File-to-Graph Converter')
7     plt.plot(x, marker="o")
8     plt.xlabel("Banyak")
9     plt.ylabel("Panjang Gelombang")
10    plt.show()

```

Listing 2.24 Import

4. Setelah dibuat langkah-langkah tersebut maka function dapat dijalankan kembali dan function line graph dapat mendefinisikan plt sebagai matplotlib.

2.5.3 Fungsi memilih file

Seperti pada listing 2.25:

```

1 def choose_file():
2     file_name = filedialog.askopenfilename()
3     if not file_name:
4         return
5     for line in open(file_name, 'r'):
6         values = [float(s) for s in line.split()]
7         x.append(values[0])
8         y.append(values[0])
9     self.addData["text"] = "File sudah tersedia!"

```

Listing 2.25 Memilih file

```

D:\Kuliah\Semester 3\sidang>python program.py
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Python27\lib\lib-tk\Tkinter.py", line 1541, in __call__
    return self.func(*args)
TypeError: choose_file() takes no arguments (1 given)

```

Gambar 2.8 Error Choose File

Pada gambar 2.8 terdapat error exception bagian type error, yang menunjukkan bahwa ada error di bagian choose file dan adanya sebuah fungsi yang dieksekusi dengan type objek yang tidak sesuai.

Untuk menangani hal tersebut perlu dibuat parameter self, jika sebuah method yang ada pada dirinya sendiri harus diawali dengan self. Seperti pada listing 2.26:

```

1 def choose_file(self):
2     file_name = filedialog.askopenfilename()
3     if not file_name:

```

```

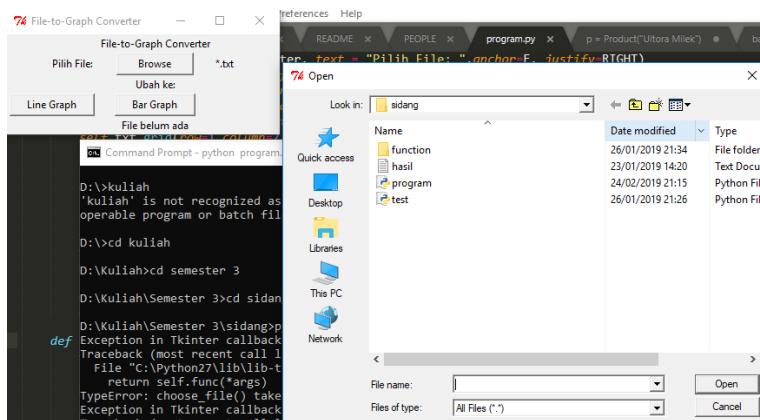
4         return
5     for line in open(file_name , 'r'):
6         values = [ float(s) for s in line .split()]
7         x.append(values[0])
8         y.append(values[0])
9         self .adaFile["text"] ="File sudah tersedia!"

```

Listing 2.26 Self Choose File

self.adaFile["text"] = "File sudah tersedia!" Pada bagian ini sebelum menjalankan perintah "adaFile" dan memunculkan print "file sudah tersedia, self akan memanggil dirinya sendiri terlebih dahulu kemudia menjalankan perintah "adaFile". Itulah kegunaan self dan hasil dari penanganan error menggunakan self membuat fungsi choose file dapat dijalankan kembali.

Seperti pada gambar 2.9

**Gambar 2.9** Self Choose File

Seperti pada listing 2.27:

```

1 def choose_file(self):
2     file_name = filedialog
3     if not file_name:
4         return
5     for line in open(file_name , 'r'):
6         values = [ float(s) for s in line .split()]
7         x.append(values[0])
8         y.append(values[0])
9         self .adaFile["text"] ="File sudah tersedia!"

```

Listing 2.27 file_name

Pada gambar 2.10 memberitahu exception yang terjadi pada fungsi memilih file, tepatnya pada bagian file_name. Exception yang tertera di gambar adalah TypeError yang memberitahu bahwa method tersebut membutuhkan string.

Seperti pada listing 2.28:

```
D:\Kuliah\Semester 3\sidang>python program.py
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Python27\lib\lib-tk\Tkinter.py", line 1541, in __call__
    return self.func(*args)
  File "program.py", line 38, in choose_file
    for line in open(file_name, 'r'):
TypeError: coercing to Unicode: need string or buffer, module found
```

Gambar 2.10 Error file_name

```
| for line in open(file_name, 'r'):
```

Listing 2.28 open_file

Pada bagian ini terdapat perintah open yang memanggil file_name dan r, namun r disini tidak dapat ditemukan. Maka penaganan error untuk bagian ini adalah sebagai berikut :

1. Definisikan r menjadi askopenfile agar r dapat menjalankan sebuah perintah.
file_name = filedialog.askopenfilename().

2.6 Tutorial Penanganan Error Pada Tkinter

Pada pembuatan sebuah tampilan menggunakan GUI, maka harus menggunakan tkinter untuk dapat menampilkan interface berbentuk menu GUI. Dalam penggunaan tkinter sering terdapat kesalahan/error saat proses import tkinter ke python. Seperti pada gambar 2.11

```
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\dzihan>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

C:\Users\dzihan>cd desktop
C:\Users\dzihan\Desktop>cd sidang

C:\Users\dzihan\Desktop\sidang>python program.py
Traceback (most recent call last):
  File "program.py", line 2, in <module>
    from Tkinter import *
ModuleNotFoundError: No module named 'Tkinter'

C:\Users\dzihan\Desktop\sidang>
```

Gambar 2.11 Error Tkinter

Dalam kasus error tersebut python tidak dapat menampilkan program.py karena tidak terdapat modul tkinter. Dalam program.py tersebut ada source code yang diperintahkan untuk menampilkan menu GUI, dalam menu GUI tersebut terdapat fungsi-

fungsi. Jika menu tidak bisa diakses maka fungsi pun tidak akan dapat dijalankan. Untuk menangani error tersebut perlu dilakukan import tkinter.

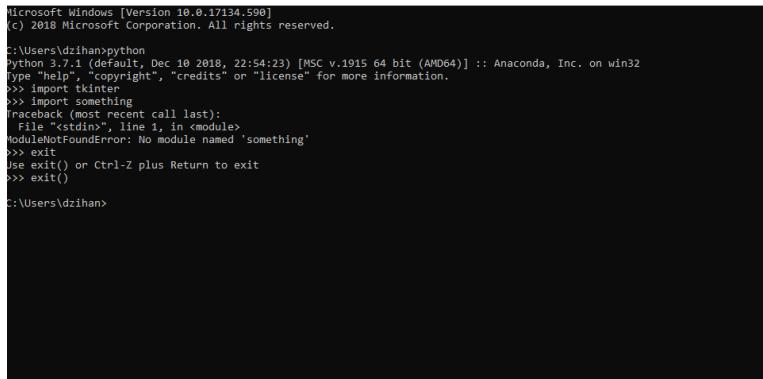
1. Buka cmd
2. Buka python
3. Ketik Import tkinter

Seperti pada listing 2.29:

```
| Import tkinter
```

Listing 2.29 Import Tkinter

Seperti pada gambar 2.12



```
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\dzihan>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> import tkinter
>>> import something
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'something'
>>> exit
Use exit() or Ctrl-Z plus Return to exit
>>> exit()

C:\Users\dzihan>
```

Gambar 2.12 Import Tkinter

4. Kemudian buka instalasi python yang telah didownload. Seperti pada gambar 2.13
5. Arahkan atau drop ke cmd Seperti pada gambar 2.14
6. Setelah itu import kembali tkinter

Jika cara tersebut tidak berhasil, cobalah cara lain seperti tutorial di bawah ini :

1. Buka folder python. Seperti pada gambar 2.15
2. Buka cmd
3. Tambahkan folder python tadi ke cmd. Seperti pada gambar 2.16
4. Ketik pip install “nama modul” Seperti pada listing 2.30:

```
| Pip install "nama_modul"
```

Listing 2.30 Pip install "nama modul"

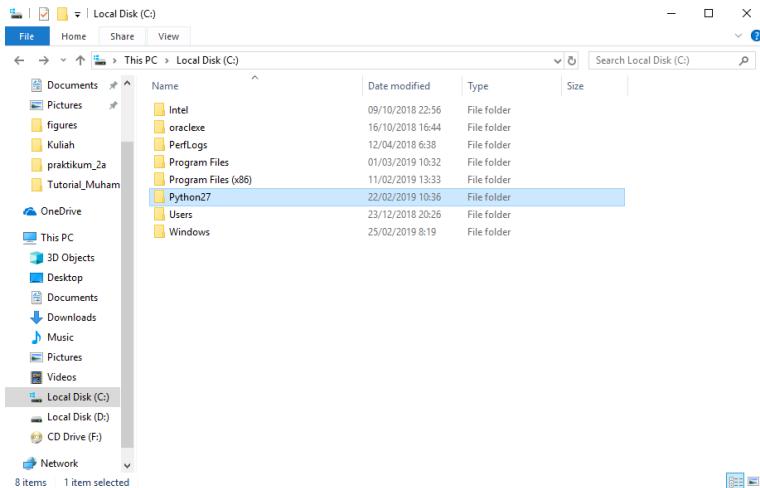
Name	Date modified	Type	Size
README	14/04/2018 22:47	Text Document	56 KB
pythonw	30/04/2018 16:31	Application	28 KB
python	30/04/2018 16:31	Application	28 KB
NEWS	30/04/2018 16:37	Text Document	482 KB
LICENSE	30/04/2018 16:41	Text Document	38 KB
Tools	22/02/2019 10:36	File folder	
tcl	22/02/2019 10:36	File folder	
Scripts	22/02/2019 11:00	File folder	
libs	22/02/2019 10:35	File folder	
Lib	22/02/2019 11:44	File folder	
include	22/02/2019 10:35	File folder	
Doc	22/02/2019 10:36	File folder	
DLLs	22/02/2019 10:35	File folder	

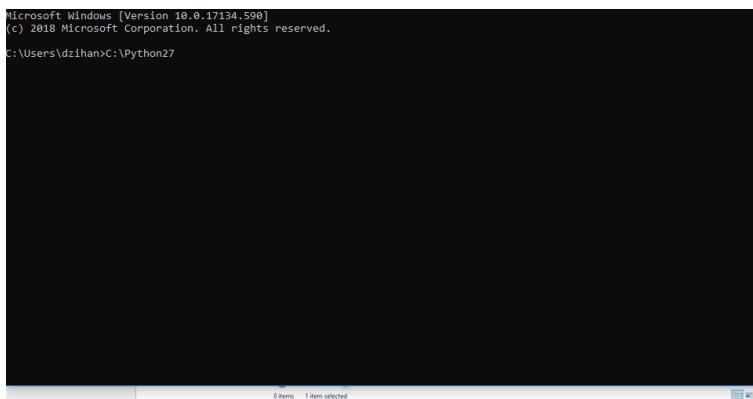
Gambar 2.13 Folder Python

```
microsoft windows [version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

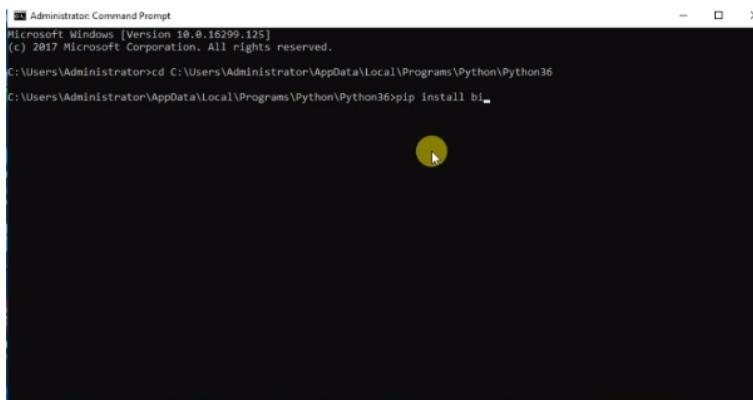
C:\Users\dzihan>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tkinter
>>> import something
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'something'
>>> exit
Use exit() or Ctrl-Z plus Return to exit
>>> exit()

C:\Users\dzihan>C:\Python27\python.exe
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:30:26) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Gambar 2.14 Drop Folder**Gambar 2.15** Buka Folder Python



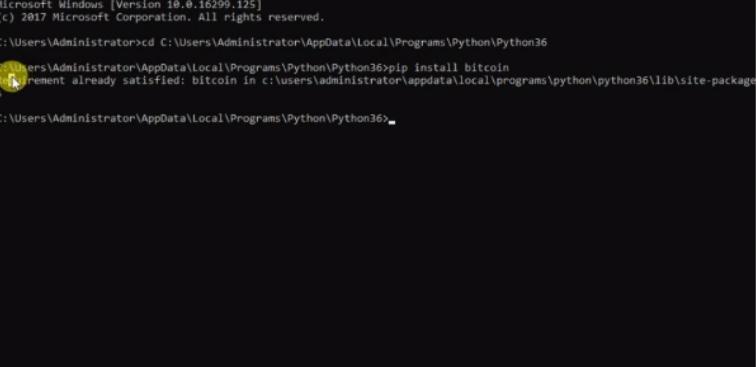
Gambar 2.16 Tambahkan Folder



Gambar 2.17 Install Modul

Seperti pada gambar 2.17

5. Modul sudah tersedia Seperti pada gambar 2.18



```
Microsoft Windows [Version 10.0.16299.125]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd C:\Users\Administrator\AppData\Local\Programs\Python\Python36

C:\Users\Administrator>AppData\Local\Programs\Python\Python36>pip install bitcoin
Requirement already satisfied: bitcoin in c:\users\administrator\appdata\local\programs\python\python36\lib\site-packages

C:\Users\Administrator>AppData\Local\Programs\Python\Python36>
```

Gambar 2.18 Modul sudah tersedia

BAB 3

PYSERIAL MINDWAVE

3.1 Pyserial Mindwave

NeuroSky Mindwave EEG adalah alat yang digunakan dalam penelitian ini dan alat terhubung ke Kode Python sebagai bahasa pemrograman yang digunakan sehingga menampilkan sinyal mentah nyata. Serta menggunakan pyserial merupakan modul Python siap-pakai dan gratis yang dibuat untuk memudahkan kita dalam membuat program komunikasi data serial RS232 dalam bahasa Python.

3.2 Code Program Mindwave

3.2.1 Code Program Pyserial Function Pada Mindwave

1. Berikut Codingan Function Untuk Menampilkan Gelombang Otak pada Mindwave seperti pada listing 3.1 :

```
1 def kesatu(ukurangmbr , f , h , i , namafile) :  
2     ax = plt . subplot(ukurangmbr)  
3     t = np . arange(f , h , i)  
4     s = np . cos(2 * np . pi * t)
```

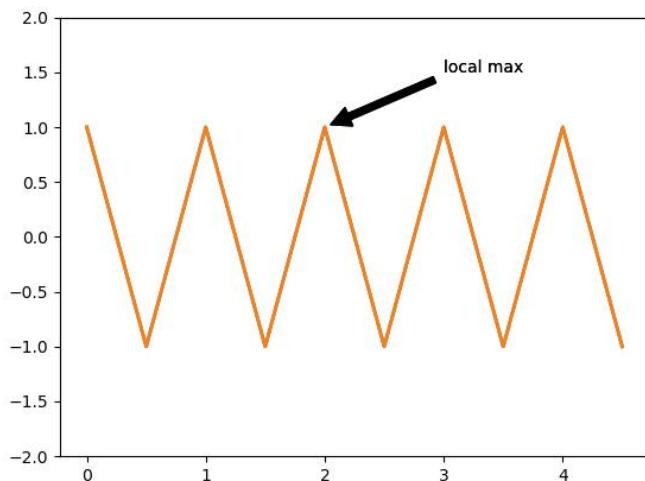
```

5 line , = plt.plot(t , s , lw=2)
6 plt.annotate("local max" , xy=(2, 1) , xytext=(3, 1.5) , arrowprops=
7 dict(facecolor="black" , shrink=0.05) ,
8 )
9 plt.ylim(-2, 2)
9 plt.savefig(namafile)

```

Listing 3.1 Function Untuk Menampilkan Gelombang Otak pada Mindwave

Pada listing 3.1: menunjukkan adanya pemberian ukuran pada grafik gelombang otak, dan menampilkan gelombang otak dalam grafik. Berikut gambar 3.1 hasil dari function menampilkan gelombang otak.



Gambar 3.1 function menampilkan gelombang otak

2. Codingan Function Memasukan Data Plotting Pada Mindwave seperti pada listing 3.2 :

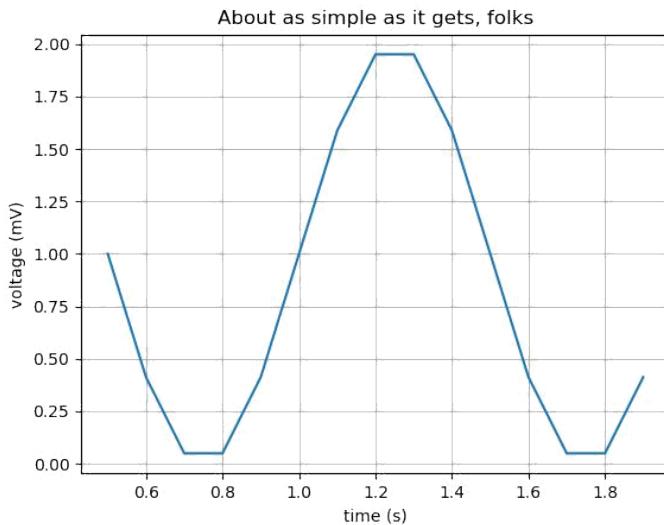
```

1 def kedua(ukurangmbr , f , g , h , namafile) :
2 ax = plt.subplot(ukurangmbr)
3 # Data for plotting
4 t = np.arange(f , g , h)
5 s = 1 + np.sin(2 * np.pi * t)
6 fig , ax = plt.subplots()
7 ax.plot(t , s)
8 ax.set(xlabel="time (s)" , ylabel="voltage (mV)" ,
9 title="About as simple as it gets , folks")
10 ax.grid()
11 plt.savefig(namafile)

```

Listing 3.2 Function Memasukan Data Plotting Otak pada Mindwave

Pada listing 3.2 menunjukan bahwa adanya penambahan pada data plotting dengan memberikan judul pada grafik gelombang otak serta memberikan keterangan waktu dan voltage. Berikut hasil dari codingan pada function memasukkan data plotting seperti pada gambar 3.2 :



Gambar 3.2 function Data Plotting pada Mindwave

3. Code Program Function Sub.plot Pada Mindwave seperti pada listing 3.3 :

```

1 def ketiga(ukurangmbr,f,g,h,namafile) :
2     ax = plt.subplot(ukurangmbr)
3     t1 = np.arange(f,g,h)
4     for n in [1, 2, 3, 4]:
5         plt.plot(t1, t1**n, label="n=%d"%(n,))
6     leg = plt.legend(loc="best", ncol=2, mode="expand", shadow=True,
7                       fancybox=True)
8     leg.get_frame().set_alpha(0.5)
9     plt.savefig(namafile)

```

Listing 3.3 Function Subplot pada Mindwave

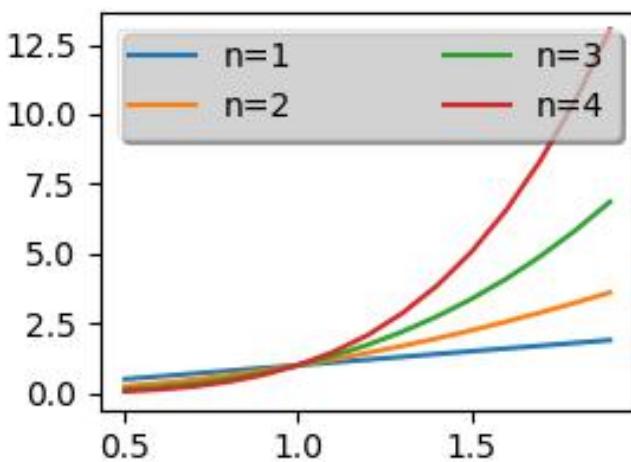
Pada listing 3.3 menjelaskan bahwa adanya perubahan pada sub.plot mindwave ini yang menunjukkan tampilan grafik gelombang otak dengan ukuran gambar yang kecil. Berikut hasil dari codingannya seperti pada gambar 3.3 :

4. Code Program Function Memberikan Keterangan Waktu dan Frekuensi seperti pada listing 3.4 :

```

1 def ketiga(ukurangmbr,f,g,h,namafile) :
2     evenly sampled time at 200ms intervals t = np.arange(f,g,h)

```



Gambar 3.3 function Subplot pada Mindwave

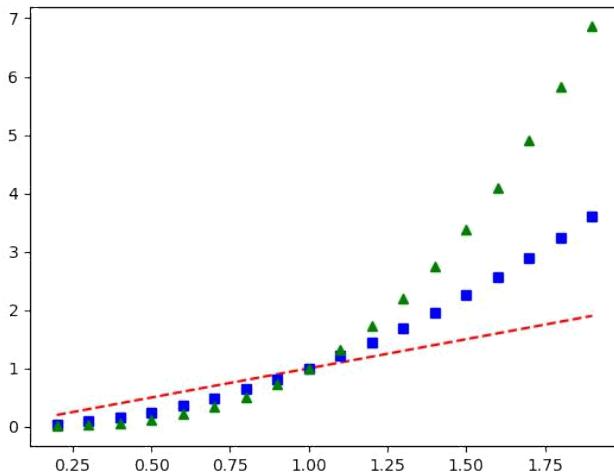
```

3 red dashes , blue squares and green triangles plt.plot(t, t, "r--"
, t, t**2, "bs", t, t**3, "g"))
plt.savefig(namafile)

```

Listing 3.4 Function Memberikan Keterangan Waktu dan Frekuensi

Pada listing 3.4 menjelaskan adanya penambahan function keterangan waktu dan frekuensi pada pyserial mindwave. Berikut hasil dari codingannya seperti pada gambar 3.4 :



Gambar 3.4 Function Memberikan Keterangan Waktu dan Frekuensi

5. Code Program Function Memasukkan Rumus Pada Grafik Gelombang Otak seperti pada listing 3.5 :

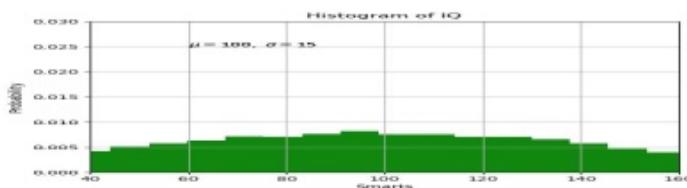
```

1 def kelima(d,i,xlabel,ylabel,title,namafile) :
2 #Fixing random state for reproducibility
3 np.random.seed(19680801)
4 mu, sigma = d,i
5 x = mu + sigma * np.random.randn(10000)
6 # the histogram of the data
7 n, bins, patches = plt.hist(x, 50, normed=1, facecolor="g", alpha
=0.75)
8 plt.xlabel(xlabel)
9 plt.ylabel(ylabel)
10 plt.title(title)
11 plt.text(60, .025, r"\mu=100,\ \sigma=15")
12 plt.axis([40, 160, 0, 0.03])
13 plt.grid(True)
14 plt.savefig(namafile)

```

Listing 3.5 Function Memasukkan Rumus Pada Grafik Gelombang Otak

Pada listing 3.5 menjelaskan adanya penambahan function memasukkan rumus gelombang otak pada pyserial mindwave. Berikut hasil dari codingannya seperti pada gambar 3.5 :



Gambar 3.5 Function Memasukkan Rumus Pada Grafik Gelombang Otak

3.3 Tutorial Mengatasi error Pyserial Function pada Mindwave

3.3.1 Mengenal error Dalam Program

Pada bahasa pemrograman, terdapat bahasa yang berbasis dekstop, web, dan lain-lain. Misalnya di dalam bahasa C, setiap pernyataan harus diakhiri dengan titik koma, namun pada visual basic tidak perlu digunakan.

Saat kita menjalankan suatu program dalam bahasa pemrograman menemukan kesalahan pada sintaks atau codingan di dalam suatu prorgam sehingga menyebabkan program tidak berjalan dengan baik.

Dilihat dari jenis kesulitan dalam memperbaiki suatu program, terdapat 3 kesalahan yaitu :

1. error Tata Bahasa (Sintaks)

Merupakan jenis error yang sering terjadi dalam pembuatan program. Namun error ini sangat mudah terdeteksi karena umumnya compier dari masing-masing bahasa program sebelum program di jalankan.

2. Error Runtime

Merupakan tingkatan dimana error ini dapat terdeteksi saat program dijalankan. Penyebabnya beragam, karena terjadinya kesalahan dalam proses input, serta perhitungan saat proses output.

3. Error Logika (Logical Error)

Merupakan jenis error yang sangat sulit terdeteksi karena terjadinya bukan karena kesalahan pada penulisan sintaks, melainkan dari kesalahan programmer dalam menggunakan algoritma.

3.3.2 Code error Pada Program Pyserial Mindwave

1. Error no attribute Hasilnya seperti pada gambar 3.6 : Jika Masih ada yang error

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'module' object has no attribute 'fungsi2'
```

Gambar 3.6 Error no attribute

di kodingnya. Perhatikan lagi kodingnya.

2. Error Indentation Hasilnya seperti pada gambar 3.7 : Berarti masih ada yang

```
>>> import seblak
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "seblak.py", line 48
      plt.plot(t1, t1**n, label="n=%d"%n,>
IndentationError: unindent does not match any outer indentation level
```

Gambar 3.7 Error Indetation

error dalam jarak spasi coba di sesuaikan dengan baris Yang lain.

3. Error Parameter Hasilnya seperti pada gambar 3.8 : Jika terjadi error seperti di

```
>>> seblak.keempat(211,0.2,2.0,1,'oi')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: keempat() takes exactly 4 arguments (5 given)
```

Gambar 3.8 Error Parameter

atas berarti kita memasukkan parameternya Kelebihan atau kekurangan. Jadi sesuaikan parameter yang anda isi dengan Kodingan.

4. Error parameter Hasilnya seperti pada gambar 3.9 : Jika terjadi seperti di-

```
>>> seblak.kelima(100,50,'x','y','Histogram','dia')
C:\Python27\lib\site-packages\matplotlib\axes.py:6571: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been"
```

Gambar 3.9 Error Parameter

atas maka cek dulu nama parameternya, kemungkinan masih terjadi kesalahan penulisan.

5. Error Penulisan Hasilnya seperti pada gambar 3.10 : Jika terjadi error seperti

```
>>> seblak.keenam(0.1,2,10,'cinta)
  File "<stdin>", line 1
    seblak.keenam(0.1,2,10,'cinta'
SyntaxError: EOL while scanning string literal
```

Gambar 3.10 Error Penulisan

diantas, maka perhatikan penggunaan tanda baca.

BAB 4

PENYAJIAN DATA DALAM PENGGUNAAN MATPLOTLIB

4.1 Penjelasan Matplotlib

Matplotlib adalah sebuah modul/librari Python yang menghasilkan gambar publikasi berguna untuk keperluan dalam penyajian pada data science. Matplotlib juga bermutu di dalam berbagai format hardcopy dan lingkungan interaktif sepanjang platform. Matplotlib dapat digunakan di dalam script Python, shell Python dan ipython. Dengan matplotlib, kita dapat membuat sebuah plots, histograms, spectra, bar charts, errorcharts, scatterplots, dan sebagainya. Pembuat matplotlib ialah seseorang yang bernama John D. Hunter yang pada 28 Agustus 2012 lalu meninggal dunia setelah bergelut dengan komplikasi kanker yang diidap beliau. Jasa beliau untuk Python Community sungguh sangat luar biasa (khususnya python untuk science [1]).

4.2 Pemasangan Matplotlib

Disini saya menggunakan sistem operasi windows, sebelum anda menginstall matplotlib kita harus memasang python terlebih dahulu. Disini saya menggunakan python 2,7.

Jika kita sudah memasang python maka selanjutnya ialah kita memvalidasi python menggunakan CMD sampai tampilan seperti pada gambar 4.1:

```
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\user>python
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:22:17) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Gambar 4.1 Validasi Python

Jika tampilan sudah seperti gambar 4.1, maka selanjutnya kita akan memasang modul matplotlib. Matplotlib dapat diinstall dengan menjalankan perintah berikut di dalam CMD, disini saya akan menggunakan pip, namun anda dapat menggunakan tool lainnya.

```
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\user>pip install matplotlib
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.7.
Requirement already satisfied: matplotlib in c:\python27\lib\site-packages (2.2.3)
Requirement already satisfied: cycler>=0.10 in c:\python27\lib\site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\python27\lib\site-packages (from matplotlib) (2.7.5)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\python27\lib\site-packages (from matplotlib) (1.0.1)
Requirement already satisfied: numpy>=1.7.1 in c:\python27\lib\site-packages (from matplotlib) (1.15.4)
Requirement already satisfied: pyparsing!=2.0.4,!!=2.1.2,!!=2.1.6,>=2.0.1 in c:\python27\lib\site-packages (from matplotlib) (2.3.0)
Requirement already satisfied: pytz in c:\python27\lib\site-packages (from matplotlib) (2018.7)
Requirement already satisfied: backports.functools-lru-cache in c:\python27\lib\site-packages (from matplotlib) (1.5)
Requirement already satisfied: six>=1.10 in c:\python27\lib\site-packages (from matplotlib) (1.12.0)
Requirement already satisfied: setuptools in c:\python27\lib\site-packages (from kiwisolver>=1.0.1->matplotlib) (39.0.1)

You are using pip version 19.0, however version 19.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\user>
```

Gambar 4.2 Pemasangan Matplotlib

Jika tampilan sudah seperti gambar 4.2, maka matplotlib telah terpasang dan siap digunakan [2].

4.3 Menggambar Plot Dasar

Sekarang kita akan membahas tentang membuat sebuah plot dasar, atau membaca sebuah data yang berupa angka menjadi sebuah plot.

4.3.1 Plot Garis

Kita akan membahas sebuah contoh sederhana dalam menggambar sebuah plot garis menggunakan matplotlib. Dalam contoh ini, kita akan membuat sebuah garis dari data angka yang memiliki sumbu x dan y Untuk data yang akan kita buat sebagai plot adalah sebagai berikut :

$$x = (4, 8, 13, 17, 20)$$

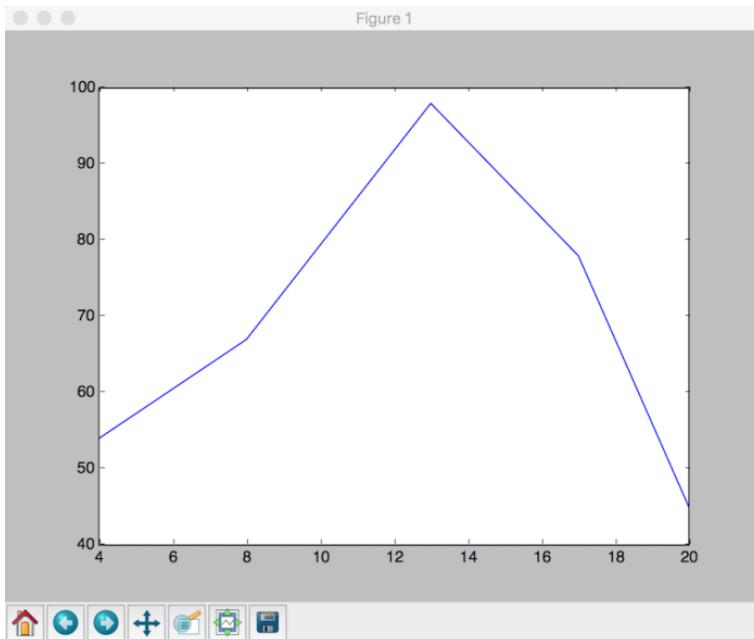
```
y = (54, 67, 98, 78, 45)
```

Ini dapat dilakukan dengan menggunakan seperti pada listing 4.1 :

```
1 import matplotlib.pyplot as plt  
2 plt.plot([4,8,13,17,20],[54, 67, 98, 78, 45])  
3 plt.show()
```

Listing 4.1 Skrip Plot Garis

Perhatikan bahwa kita menyajikan titik x dan y sebagai daftar. Dalam contoh ini, hasilnya akan menjadi seperti pada gambar 4.3:



Gambar 4.3 Tampilan Plot Garis

Garis pada gambar di atas adalah garis default yang digambarkan untuk sebuah plot dasar, baik bentuk dan warnanya. Kita dapat memodifikasi itu dengan mengubah bentuk dan warna garis menggunakan beberapa fungsi dari dokumentasi matplotlib.

4.3.2 Plot Sebaran

Sebuah plot sebaran adalah sebuah grafik yang menunjukkan hubungan antara dua set data, di plot sebaran ini digambarkan melalui titik-titik yang saling terhubung ke angka-angka yang ada di sumbu x dan y. Di dalam contoh ini, kita akan membahas menggambar sebuah plot sebaran menggunakan matplotlib.

Untuk data yang akan kita buat sebagai plot adalah sebagai berikut :

```
x = [2,4,6,7,9,13,19,26,29,31,36,40,48,51,57,67,69,71,78,88]
```

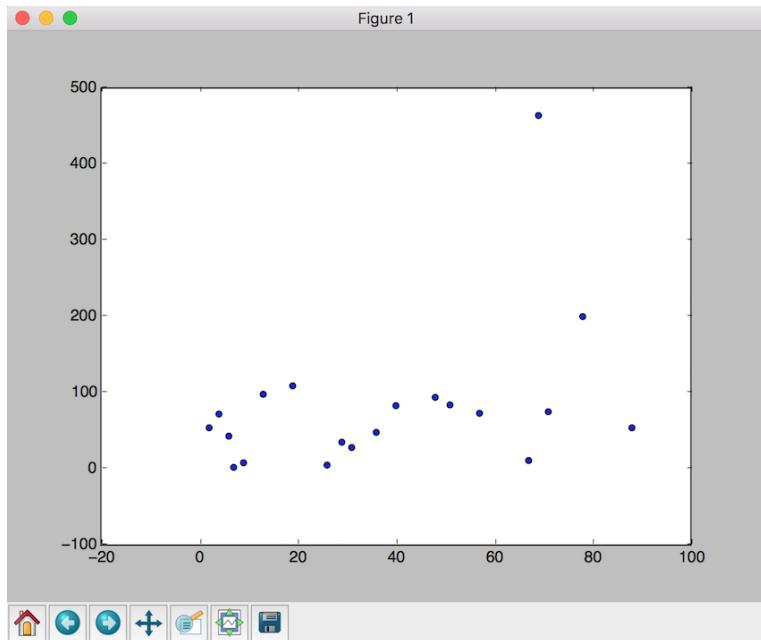
`y = [54, 72, 43, 2, 8, 98, 109, 5, 35, 28, 48, 83, 94, 84, 73, 11, 464, 75, 200, 54]`

Plot sebaran dapat digambarkan menggunakan seperti pada listing 4.2 :

```
1 import matplotlib.pyplot as plt
2 x =[2,4,6,7,9,13,19,26,29,31,36,40,48,51,57,67,69,71,78,88]
3 y =[54,72,43,2,8,98,109,5,35,28,48,83,94,84,73,11,464,75,200,54]
4 plt.scatter(x,y)
5 plt.show()
```

Listing 4.2 Skrip Plot Sebaran

Dalam contoh ini, hasilnya akan menjadi seperti pada gambar 4.4:



Gambar 4.4 Tampilan Plot Sebaran

Tentu saja, kita dapat mengubah warna marker sebagai tambahan untuk pengaturan lainnya, seperti yang ditunjukkan di dalam dokumentasi.

4.3.3 Histogram

Plot histogram adalah sebuah grafik yang menampilkan frekuensi data menggunakan sebuah batang, dimana angka dikelompokkan dalam rentang yang tertentu. Dengan kata lain, frekuensi setiap elemen data di dalam daftar ditunjukkan menggunakan histogram. Angka yang dikelompokkan dalam bentuk rentang tertentu. Mari lihat contoh untuk lebih mengerti ini.

Contoh data yang ingin kita tentukan histogramnya adalah sebagai berikut:

`x = [2, 4, 6, 5, 42, 543, 5, 3, 73, 64, 43, 97, 63, 76, 63, 8, 73, 97, 23, 45, 56, 8]`

Script Python yang dapat kita gunakan untuk menampilkan histogram seperti pada listing 4.3 :

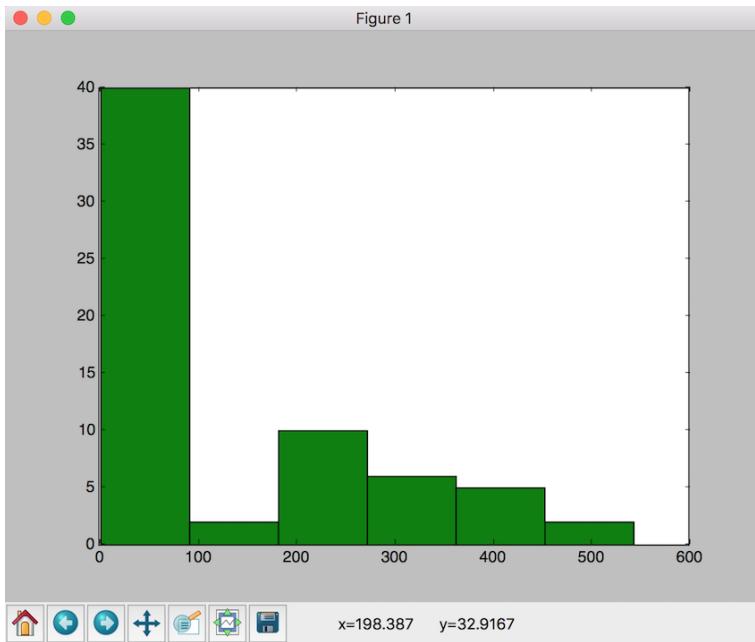
```

1 import matplotlib.pyplot as plt
2 x =
3     [2,4,6,5,42,543,5,3,73,64,43,97,63,76,63,8,73,97,23,45,56,...,309]
4 num_bins = 6
5 n, bins, patches = plt.hist(x, num_bins, facecolor = 'green')
6 plt.show()

```

Listing 4.3 Skrip Histogram

Ketika kamu menjalankan script, kamu harusnya mendapatkan sesuatu serupa dengan grafik berikut (histogram) seperti pada gambar 4.5:



Gambar 4.5 Tampilan Histogram

Tentu saja ada lebih banyak parameter untuk function hist, seperti yang ditunjukkan di dokumentasi.

4.3.4 Kustomisasi Plot

Sekarang kita dapat menambahkan judul, label untuk garis horizontal dan vertikal dengan memanggil fungsi `xtitle`, `xlabel`, dan `ylabel` seperti pada seperti pada listing 4.4 :

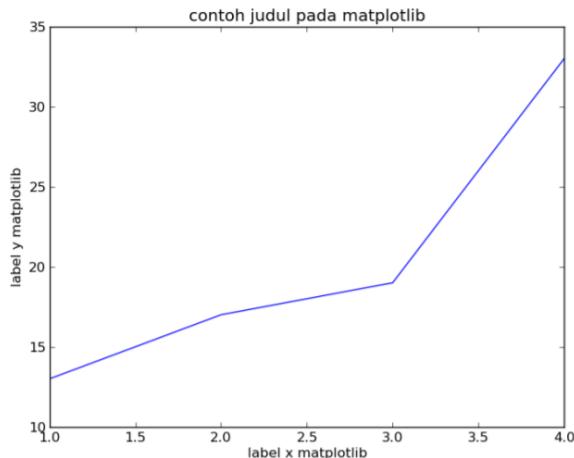
```

1 import matplotlib.pyplot as plt
2 plt.plot([1, 2, 3, 4], [13, 17, 19, 33]) []
3 plt.title("contoh judul pada matplotlib")
4 plt.xlabel("label x matplotlib")
5 plt.ylabel("label y matplotlib")
6 plt.show()

```

Listing 4.4 Skrip Kustomisasi Plot

Seperti pada gambar 4.6:



Tabel ringkasan fungsi:

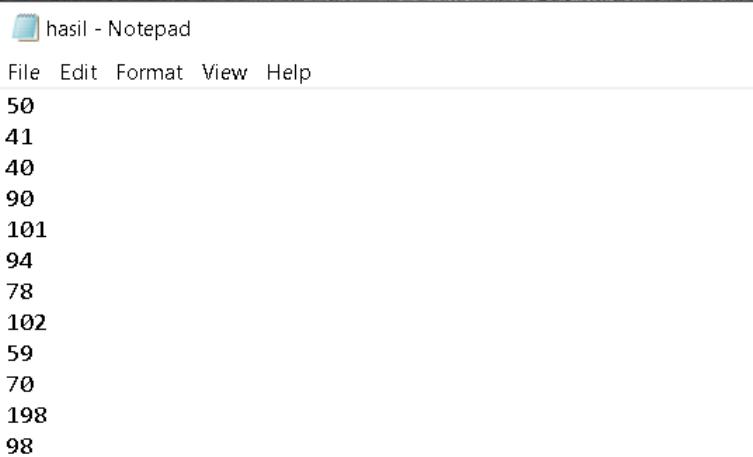
Fungsi	Keterangan
plot	melakukan plot
title	memberi judul pada gambar plot
xlabel	memberi nama label untuk garis x
ylabel	memberi nama label untuk garis y
show	menampilkan gambar plot

Gambar 4.6 Tampilan Kustomisasi Plot

4.4 Pembacaan data dari sebuah file.txt

4.4.1 Membuat/memasukkan data melalui notepad

Cara yang pertama ialah dengan memasukkan seluruh data berupa angka ke sebuah halaman notepad dan kemudian disimpan sebagai file.txt (disini saya membuat nama file hasil.txt) Seperti pada gambar 4.7:



```
50
41
40
90
101
94
78
102
59
70
198
98
```

Gambar 4.7 Data Hasil.txt

4.4.2 Pembacaan data

Setelah kita berhasil membuat file dengan nama hasil.txt, kemudian kita akan memanggil file tersebut kesebuah kode program, sehingga source code seperti pada listing 4.5 :

```
1 f = open("hasil.txt", "r")
2 print(f.read())
```

Listing 4.5 Skrip Pembacaan Data

Source code diatas bernama ‘ bacadata.py ‘ lalu dengan menggunakan python kita akan memanggil bacadata.py tersebut.

Cara pemanggilan sebuah file python adalah sebagai berikut :

1. Buka CMD
2. Arahkan ke direktori tempat penyimpanan file python yang kamu buat (disini saya meletakkan file python bacadata.py tersebut di desktop)
3. Lalu panggil dengan perintah seperti pada gambar 4.8:
4. Perlu diketahui bahwa file hasil.txt harus terletak di direktori dan folder yang sama (disini saya meletakkan hasil.txt di dekstop juga)
5. Sehingga jika python bacadata.py maka akan membaca keseluruhan isi angka pada data hasil.txt tersebut.
6. Perlu diketahui lagi bahwa ini hanya membaca dan menampilkan data berupa angka juga, sehingga hasilnya akan seperti gambar diatas.

```
C:\Users\user\Desktop>python bacadata.py
50
41
40
90
101
94
78
102
59
70
198
98
```

Gambar 4.8 Pemanggilan bacadata.py

4.4.3 Pemanggilan data ke sebuah plot

Seperti pada listing 4.6 :

```
1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12 plt.plot(x, marker='o')
13 plt.title('Data Gelombang Otak')
14 plt.xlabel('Banyak Data')
15 plt.ylabel('Gelombang otak')
16 plt.show()
```

Listing 4.6 Skrip Pemanggilan data ke sebuah plot

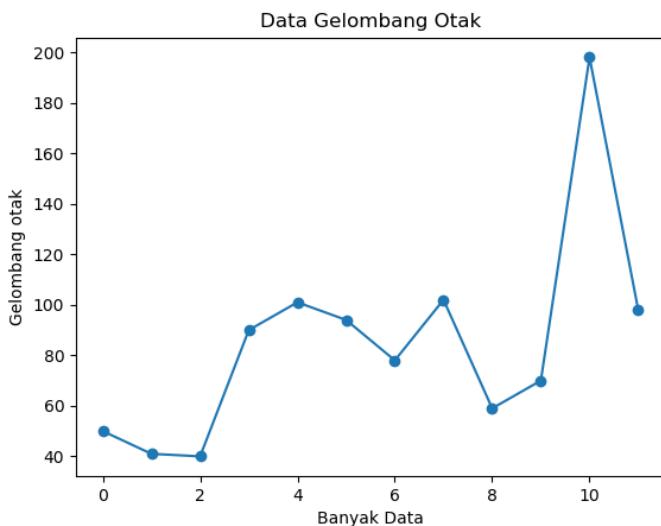
Dengan menggunakan source code diatas kita bisa memanggil hasil.txt yang telah kita buat tadi ke sebuah fungsi plot, sehingga kita tidak perlu lagi menulis seluruh isi data angka di source code plot, cukup menggunakan ‘with open’ kita dapat memanggil file yang kemudian akan dibaca dan ditampilkan berupa plot.

Disini saya membuat file python yang bernama panggildata.py .Sehingga jika kita panggil melalui CMD akan seperti pada gambar 4.9:

Jika sudah kita panggil maka secara otomatis akan muncul figure baru seperti pada gambar 4.10:

```
C:\Users\user\Desktop>python panggildata.py
```

Gambar 4.9 Pemanggilan panggildata.py



Gambar 4.10 Tampilan pemanggilan data ke sebuah plot

4.5 Modifikasi Hasil Plot

4.5.1 Setting Label

Kita dapat Kustomisasi Tampilan pada plot, pertama kita akan membahas tentang label. Anda dapat menambahkan judul, label untuk garis horisontal dan vertikal dengan menambahkan nama label untuk garis x dan y Disini ada contoh source code untuk menambahkan nama pada label, Seperti pada listing 4.7 :

```

1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12
13 plt.plot(x)
14 plt.xlabel('Banyak Data')
15 plt.ylabel('Gelombang otak')
16 plt.show()

```

Listing 4.7 Skrip Setting Label

Adapun hasil yang ditampilkan pada source code diatas adalah seperti pada gambar 4.11:

Pada gambar 4.11 adalah sebuah plot dengan menambahkan nama pada label x yaitu banyak data dan label y yaitu gelombang otak.

4.5.2 Setting Warna

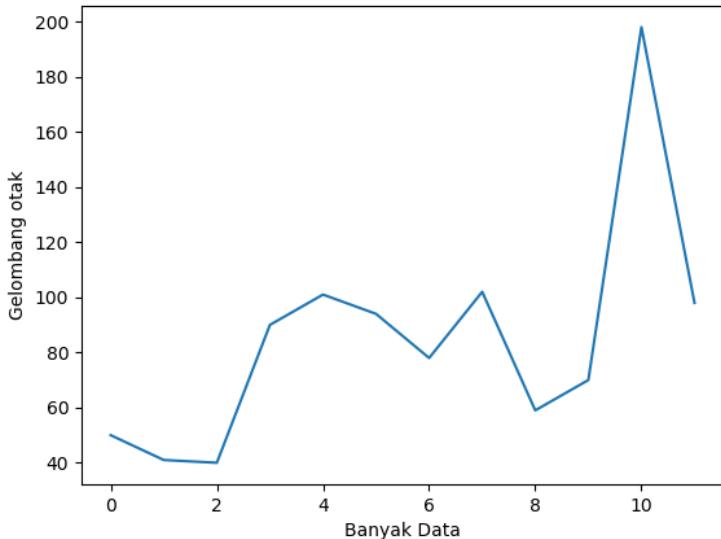
Selanjutnya kita akan membahas bagaimana kustomisasi warna pada plot, dengan menggunakan facecolor = 'red' kita akan mengganti warna plot menjadi merah.

Disini ada contoh source code untuk mengganti warna pada plot, tetapi disini saya coba menggunakan plt.hist, Seperti pada listing 4.8 :

```

1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12
13 plt.hist(x, facecolor = 'red')

```



Gambar 4.11 Tampilan setting label

```

14 plt.xlabel('Banyak Data')
15 plt.ylabel('Gelombang otak')
16 plt.show()

```

Listing 4.8 Skrip Setting Warna

Jika anda membuat source seperti diatas maka akan menampilkan hasil seperti pada gambar 4.12:

Dengan menggunakan perintah facecolor maka kita bisa mengganti warna pada plot dengan warna yang kita inginkan.

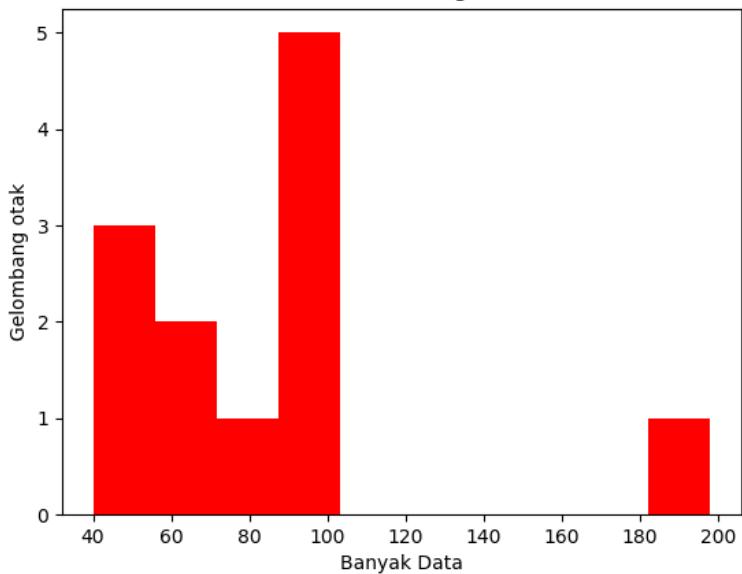
4.5.3 Setting Marker

Selanjutnya kita akan kustomisasi marker pada plot, disini kita bisa mengganti penanda label pada x dan y Ikuti contoh source code Seperti pada listing 4.9 :

```

1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12

```



Gambar 4.12 Tampilan setting warna

```

13 plt.plot(x)
14
15 plt.xlabel('Banyak Data')
16 plt.ylabel('Gelombang otak')
17
18 plt.show()

```

Listing 4.9 Skrip Setting Tanpa Marker

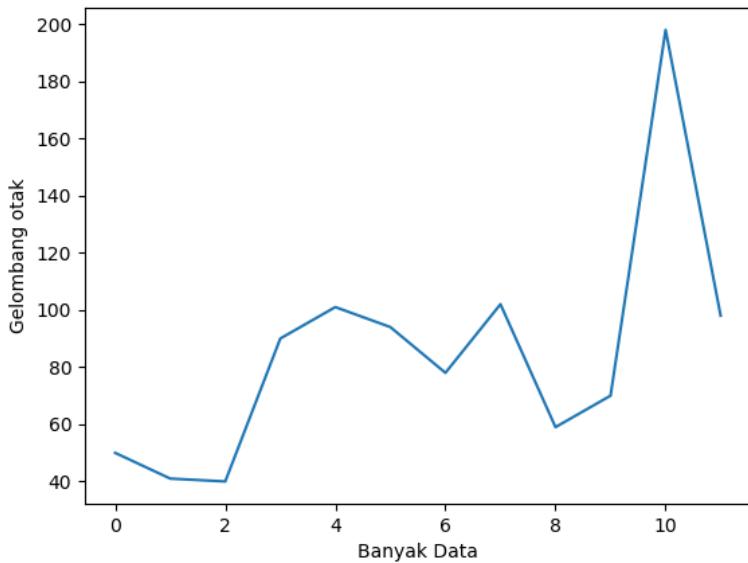
Pada source code diatas belum ada kustomisasi marker yang dibuat, lalu hasilnya akan seperti pada gambar 4.13:

Sekarang coba kita bandingkan dengan source code berikut yang ditambahkan marker untuk kustomisasi penanda label x Seperti pada listing 4.10 :

```

1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12
13 plt.plot(x, marker='o')
14
15 plt.xlabel('Banyak Data')

```



Gambar 4.13 Tampilan setting tanpa marker

```
16 plt.ylabel('Gelombang otak')  
17  
18 plt.show()
```

Listing 4.10 Skrip Setting Dengan Marker

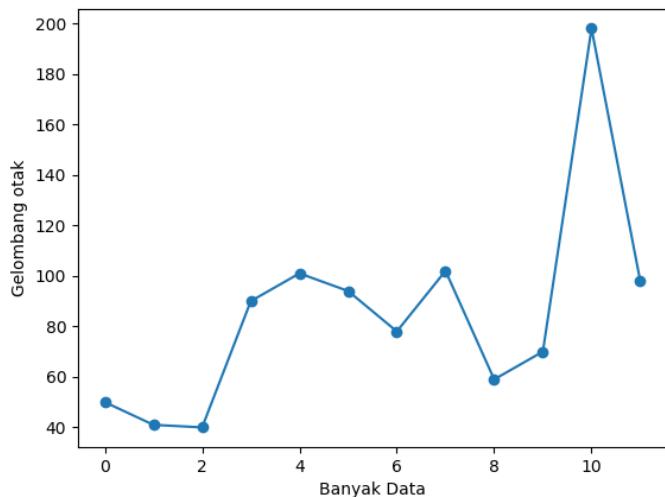
dan hasilnya akan seperti pada gambar 4.14:

Berbeda bukan sama yang gambar sebelumnya ? nah itu bisa kita kustomisasi dengan menambahkan marker pada plt.plot(x, marker='o') bisa juga o diganti dengan x dan sebagainya.

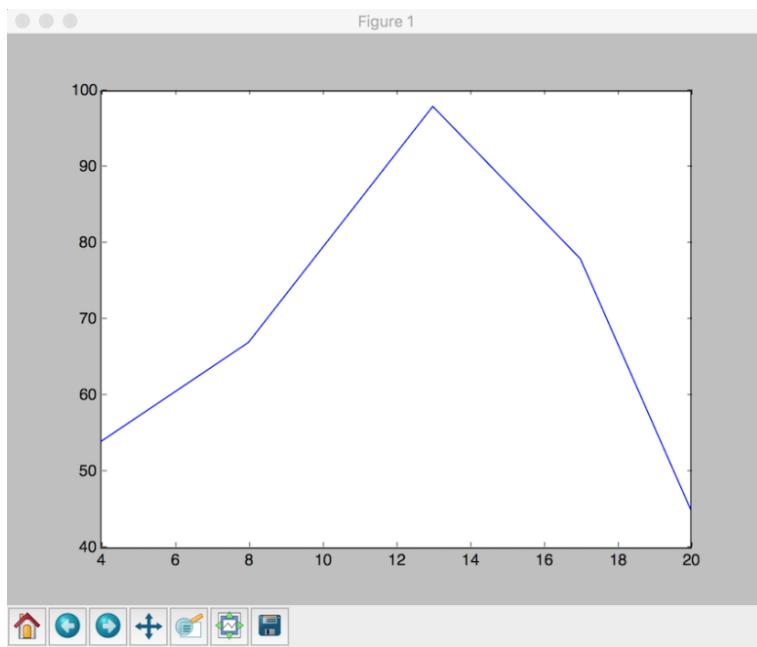
4.5.4 Jenis Plot

Pada pembahasan diawal sudah kita bahas tentang dasar-dasar plot, disini akan kita berikan hasil tampilkan pada jenis-jenis plot yang ada [4].

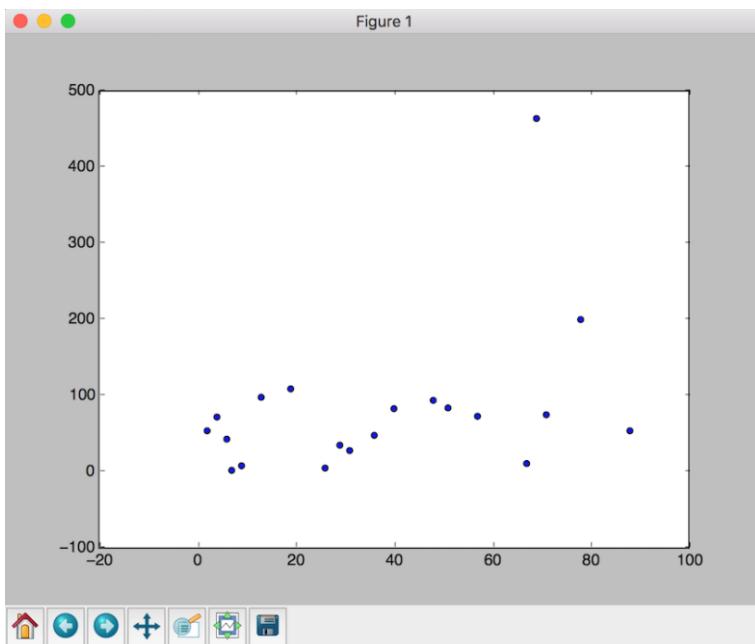
1. Plot Garis, seperti pada gambar 4.15:
2. Plot Sebaran, seperti pada gambar 4.16:
3. Histogram, seperti pada gambar 4.17:
4. Bar Charts, seperti pada gambar 4.18:
5. Pie, seperti pada gambar 4.19:
6. Hist, seperti pada gambar 4.20:



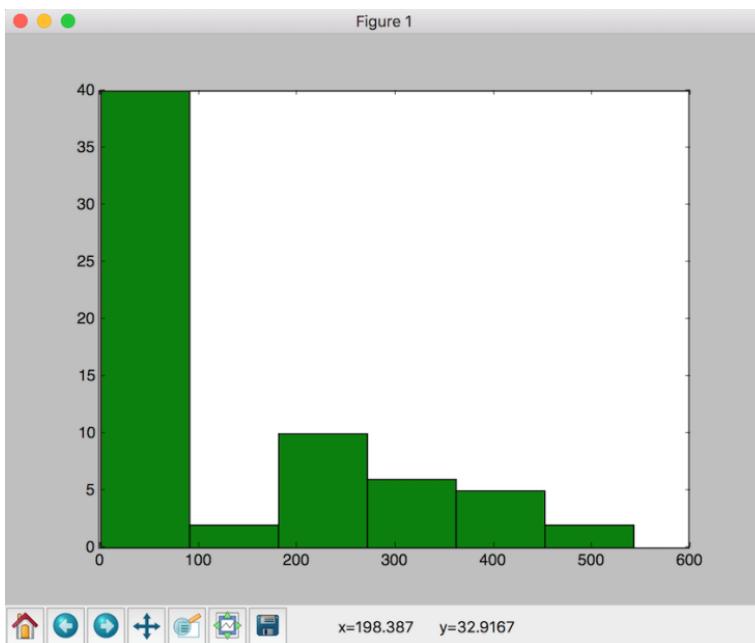
Gambar 4.14 Tampilan setting dengan marker



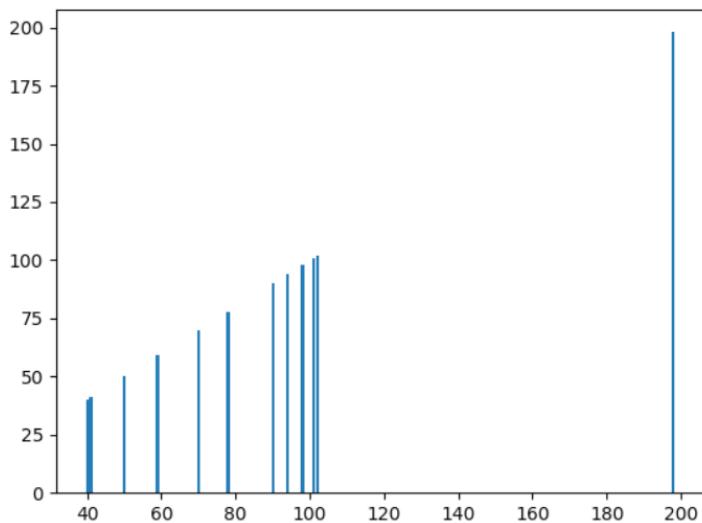
Gambar 4.15 Tampilan jenis plot garis



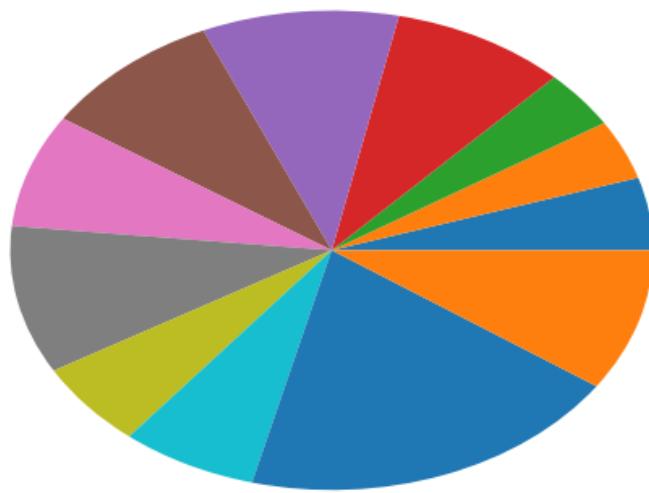
Gambar 4.16 Tampilan jenis plot sebaran



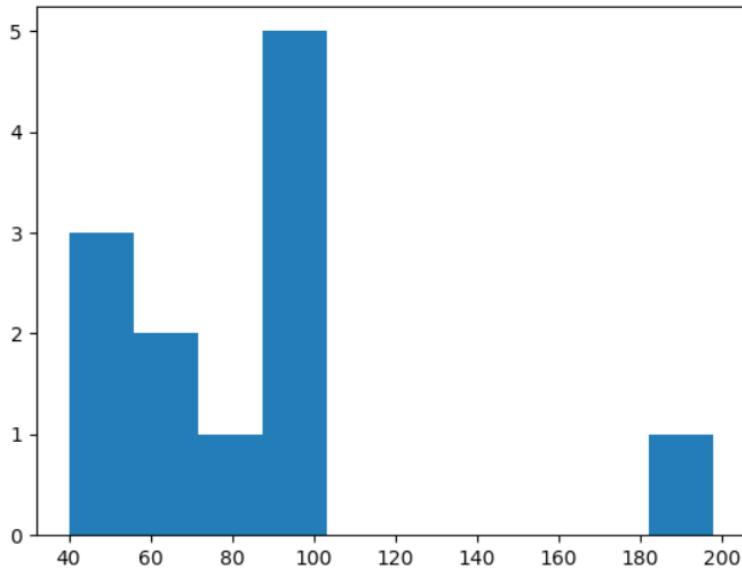
Gambar 4.17 Tampilan jenis plot histogram



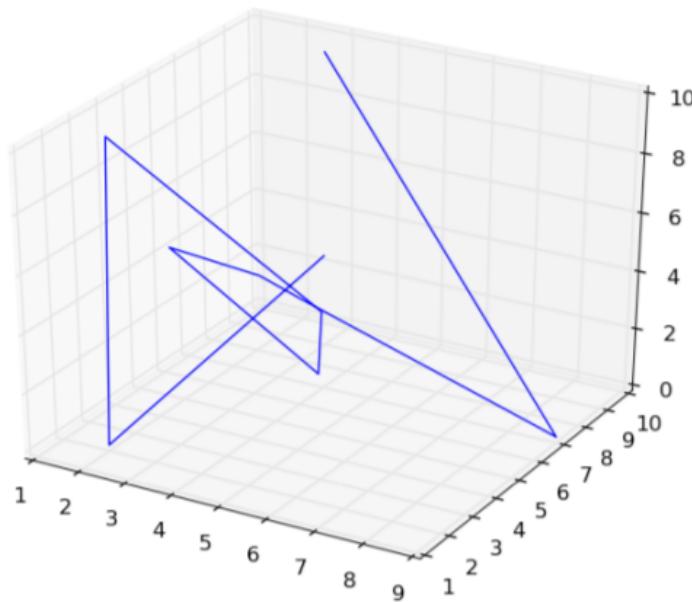
Gambar 4.18 Tampilan jenis plot bar charts



Gambar 4.19 Tampilan jenis plot pie

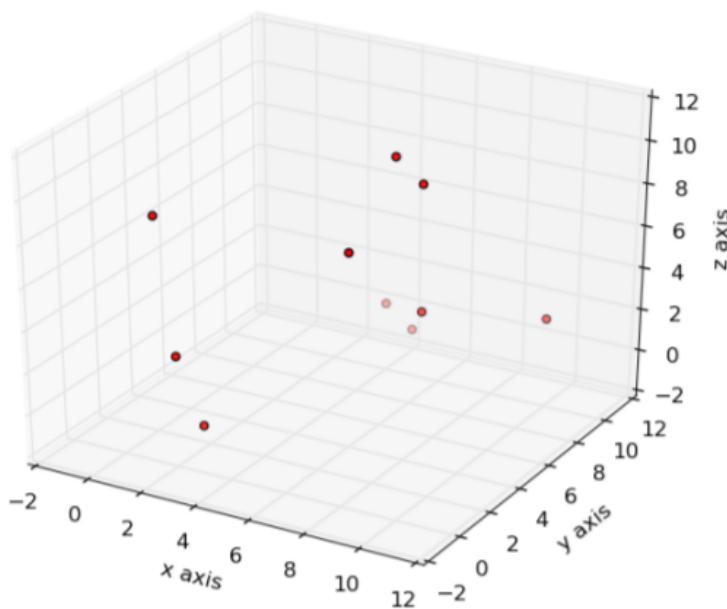


Gambar 4.20 Tampilan jenis plot hist



Gambar 4.21 Tampilan jenis plot 3D Line

7. 3D Line, seperti pada gambar 4.21:
8. 3D Scatter Plot, seperti pada gambar 4.22:



Gambar 4.22 Tampilan jenis plot 3D Scatter

9. 3D Bar Charts, seperti pada gambar 4.23:

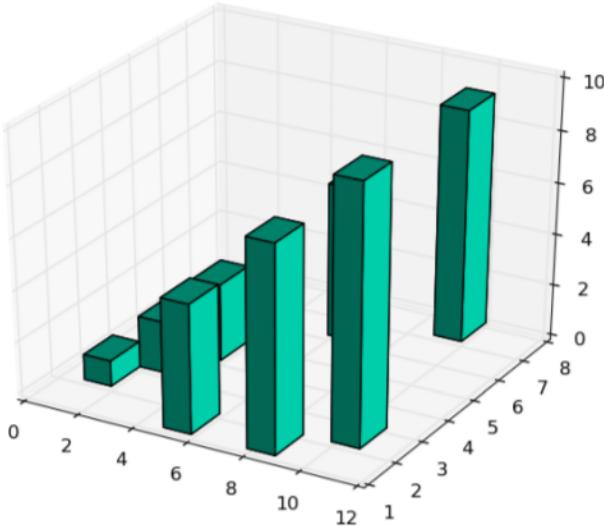
4.6 Plot Ditampilkan Dan Disimpan

Sekarang kita akan membahas bagaimana cara untuk membuat plot itu bisa disimpan sebagai gambar tanpa kita panggil melalui python.

4.6.1 Plot Ditampilkan

Seperti tutorial yang sebelumnya bagaimana kita sudah mempelajari bagaimana menampilkan data menjadi sebuah plot, kita ulas kembali bagaimana caranya, dengan memanfaatkan script diawal tadi Seperti pada listing 4.11 :

```
1 import matplotlib.pyplot as plt
2 import csv
3
4 x = []
5 y = []
```



Gambar 4.23 Tampilan jenis plot 3D Charts

```

7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12
13 plt.plot(x, marker='o')
14
15 plt.title('Data Gelombang Otak')
16
17 plt.xlabel('Banyak')
18 plt.ylabel('Panjang Gelombang')
19
20 plt.show()

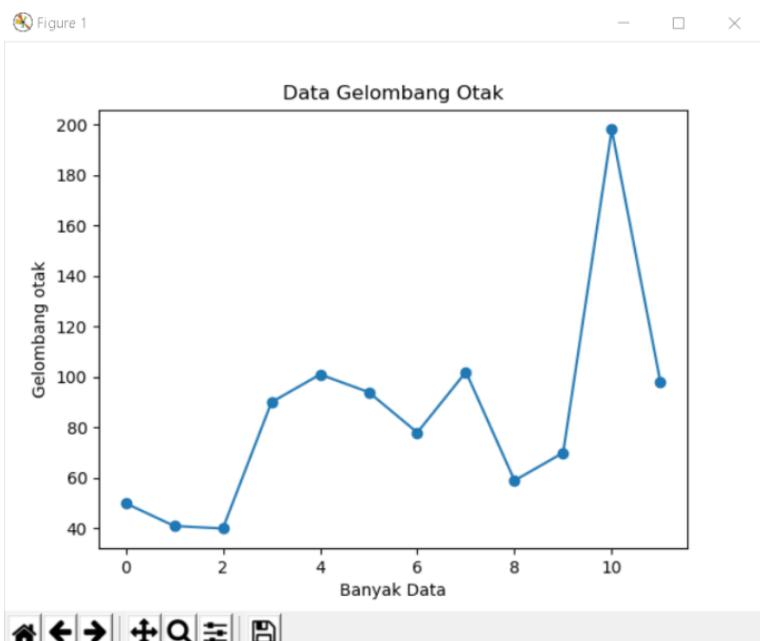
```

Listing 4.11 Skrip Plot ditampilkan

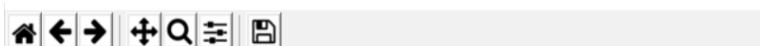
Hasilnya juga akan sama seperti tutorial diawal tadi seperti pada gambar 4.24:

4.6.2 Plot Disimpan

Sekarang kita akan membahas bagaimana suatu data yang sudah kita transformasi menjadi sebuah plot bisa tersimpan sebagai gambar. Script sama seperti Tabel 6.10 tadi, disini kita fokus ke hasil pada Gambar 6.21, ada sebuah menu dibawah hasil yang ditampilkan. Perhatikan gambar 4.25:

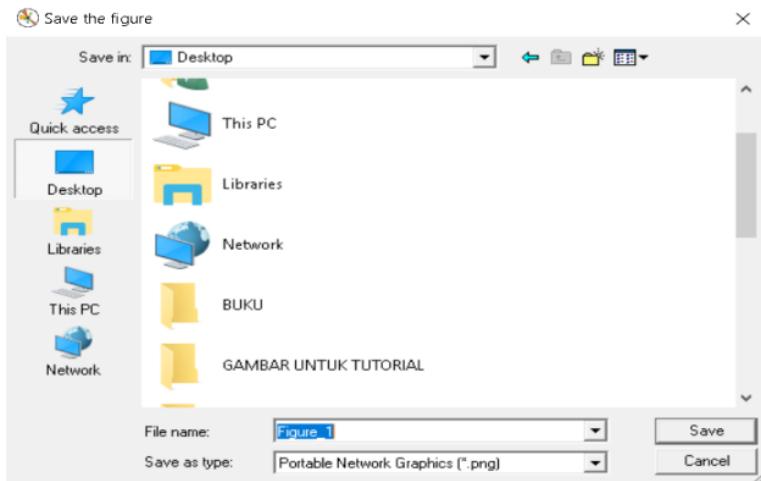


Gambar 4.24 Hasil plot ditampilkan



Gambar 4.25 Menu plot disimpan

Dengan memanfaatkan menu save yang ada pada tampilan diatas maka secara otomatis kita bisa menyimpan hasil plot menjadi gambar ke lokasi file yang kita mau seperti pada gambar 4.26:



Gambar 4.26 Proses plot disimpan

4.7 Penanganan Error Pada Pemasangan Matplotlib

Disini saya menggunakan sistem operasi windows, sebelum anda menginstall matplotlib kita harus memasang python terlebih dahulu. Disini saya menggunakan python 2,7.

Langsung saja kita pergi ke CMD lalu ketikan pip install matplotlib seperti pada gambar 4.27:

```
C:\Users\user>pip instal matplotlib
ERROR: unknown command "instal" - maybe you meant "install"
```

Gambar 4.27 Error Pemasangan Matplotlib

Gambar 4.27 adalah tampilan error pada saat kita install matplotlib, coba anda teliti dimanakah letak error tersebut ? Error terletak pada sintaks ‘pip instal matplotlib’ yang seharunya ‘pip install matplotlib’ tampaknya hanya hal kecil tapi dengan kita salah membuat sintaks maka hasil juga akan error sehingga perintah yang benar seperti pada gambar 4.28:

Jika tampilan sudah seperti gambar 4.28, maka matplotlib telah terpasang dan siap digunakan [2].

```
C:\Users\user>pip install matplotlib
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.7.
Requirement already satisfied: matplotlib in c:\python27\lib\site-packages (2.2.3)
Requirement already satisfied: cycler>=0.10 in c:\python27\lib\site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>2.1 in c:\python27\lib\site-packages (from matplotlib) (2.7.5)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\python27\lib\site-packages (from matplotlib) (1.0.1)
Requirement already satisfied: numpy>=1.7.1 in c:\python27\lib\site-packages (from matplotlib) (1.15.4)
Requirement already satisfied: pytz!=2.0.4,>=2.1.2,<=2.1.6,>=2.0.1 in c:\python27\lib\site-packages (from matplotlib) (2.3.0)
Requirement already satisfied: pytz in c:\python27\lib\site-packages (from matplotlib) (2018.7)
Requirement already satisfied: backports.functools-lru-cache in c:\python27\lib\site-packages (from matplotlib) (1.5)
Requirement already satisfied: six>=1.10 in c:\python27\lib\site-packages (from matplotlib) (1.12.0)
Requirement already satisfied: setuptools in c:\python27\lib\site-packages (from kiwisolver>1.0.1->matplotlib) (39.0.1)

You are using pip version 19.0, however version 19.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\user>
```

Gambar 4.28 Pemasangan Matplotlib

4.8 Error Pada Saat Menggambar Plot Dasar

Sekarang kita akan membahas tentang membuat sebuah plot dasar, atau membaca sebuah data yang berupa angka menjadi sebuah plot.

4.8.1 Error Pada Plot Garis

Kita akan membahas sebuah contoh sederhana dalam menggambar sebuah plot garis menggunakan matplotlib. Dalam contoh ini, kita akan membuat sebuah garis dari data angka yang memiliki sumbu x dan y. Untuk data yang akan kita buat sebagai plot adalah sebagai berikut :

```
x = (4, 8, 13, 17, 20)
y = (54, 67, 98, 78, 45)
```

Ini dapat dilakukan dengan menggunakan script Seperti pada listing 4.12 :

```
1 import matplotlib.pyplot as plt
2 plt.plot([4,8,13,17,20],[54, 67, 98, 78, 45])
3 plt.show()
```

Listing 4.12 Skrip Plot Garis

seperti pada gambar 4.29:

```
1 import matplotlib.pyplot as plt
2 plt.plot([4,8,13,17,20],[54, 67, 98, 78, 45])
3 plt.show()
4 |
```

Gambar 4.29 Skrip Error Plot Garis

seperti pada gambar 4.30:

Pada contoh di atas saya membuat file dengan nama test.py, dan saya panggil melalui python. Sintaks diatas adalah error karena plt.show() tidak sejajar dengan

```
C:\Users\user\Desktop>python test.py
  File "test.py", line 3
    plt.show()
    ^
IndentationError: unexpected indent
```

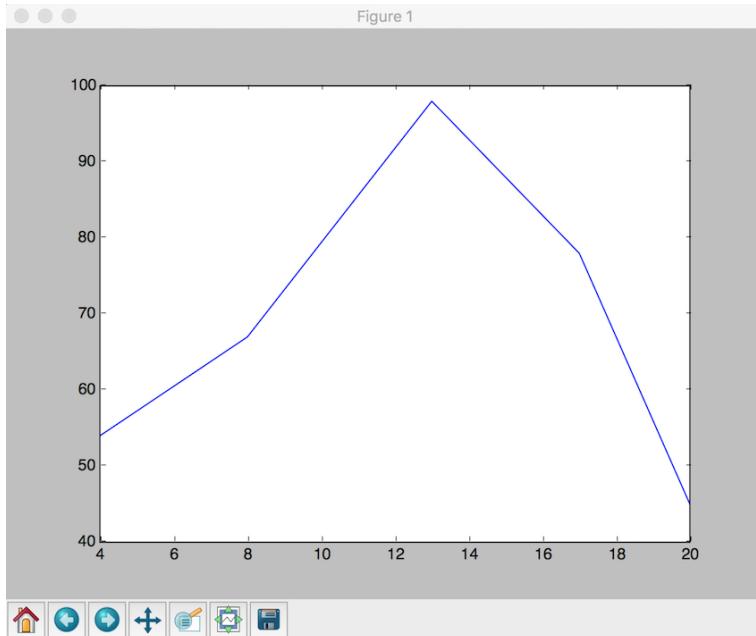
Gambar 4.30 Error Plot Garis

baris kedua sehingga di baca oleh program plt.show() itu adalah turunan dari baris kedua, sehingga solusinya seperti pada gambar 4.31:

```
1 import matplotlib.pyplot as plt
2 plt.plot([4,8,13,17,20],[54, 67, 98, 78, 45])
3 plt.show()
```

Gambar 4.31 Solusi Error Plot Garis

Dan jika dipanggil melalui python hasilnya akan seperti pada gambar 4.32:



Gambar 4.32 Error Plot Garis

Gambar 4.32 adalah error kedua pada program plot garis, disini errornya adalah kita tidak meletakkan koma di antara sumbu x dan y, bisa kita temui di plt.plot([4,8,13,17,20] [54, 67, 98, 78, 45]) yang seharusnya seperti pada gambar 4.33:

```

1 import matplotlib.pyplot as plt
2 plt.plot([4,8,13,17,20], [54, 67, 98, 78, 45])
3 plt.show()
4

```

Gambar 4.33 Solusi Error Plot Garis

4.8.2 Plot Sebaran

Sebuah plot sebaran adalah sebuah grafik yang menunjukkan hubungan antara dua set data, di plot sebaran ini digambarkan melalui titik yang saling terhubung ke angka-angka yang ada di sumbu x dan y. Di dalam contoh ini, kita akan membahas menggambar sebuah plot sebaran menggunakan matplotlib.

Untuk data yang akan kita buat sebagai plot adalah sebagai berikut :

```
x = [2,4,6,7,9,13,19,26,29,31,36,40,48,51,57,67,69,71,78,88]
y = [54,72,43,2,8,98,109,5,35,28,48,83,94,84,73,11,464,75,200,54]
```

Plot sebaran dapat digambarkan menggunakan script Seperti pada listing 4.13 :

```

1 import matplotlib.pyplot as plt
2 x =[2,4,6,7,9,13,19,26,29,31,36,40,48,51,57,67,69,71,78,88]
3 y =[54,72,43,2,8,98,109,5,35,28,48,83,94,84,73,11,464,75,200,54]
4 plt.scatter(x,y)
5 plt.show()
```

Listing 4.13 Skrip Plot Sebaran

Masalah atau error yang biasa timbul adalah seperti pada gambar 4.34:

```
C:\Users\user\Desktop>python test.py
  File "test.py", line 4
    plt.scatter(x y)
               ^
SyntaxError: invalid syntax
```

Gambar 4.34 Error Plot Sebaran

Solusinya adalah dengan menambahkan koma di antara x dan y (x,y) Atau seperti pada gambar 4.35:

Dalam contoh lain error bisa menjadi seperti pada gambar 4.36:

Solusinya adalah terdapat pada plt.scatter(y) yang seharunya menjadi plt.scatter(x,y) karena tidak mungkin kita mau menampilkan gambar yang sudah kita definisikan x

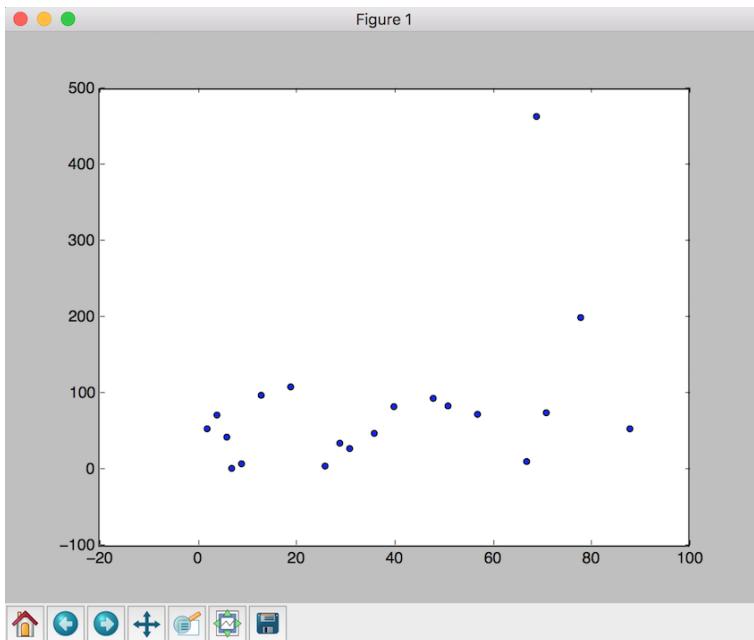
```
1 import matplotlib.pyplot as plt
2 x = [2, 4, 6, 7, 9, 13, 19, 26, 29, 31, 36, 40, 48, 51, 57, 67, 69, 71, 78, 88]
3 y = [54, 72, 43, 2, 8, 98, 109, 5, 35, 28, 48, 83, 94, 84, 73, 11, 464, 75, 200, 54]
4 plt.scatter(x,y)
5 plt.show()
```

Gambar 4.35 Solusi Error Plot Sebaran

```
C:\Users\user\Desktop>python test.py
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    plt.scatter(y)
TypeError: scatter() takes at least 2 arguments (1 given)
```

Gambar 4.36 Error Plot Sebaran

dan y tetapi kita hanya menampilkan 1 sumbu saja, Dalam contoh ini, hasilnya akan menjadi seperti pada gambar 4.37:



Gambar 4.37 Solusi Error Plot Sebaran

4.8.3 Error Pada Histogram

Plot histogram adalah sebuah grafik yang menampilkan frekuensi data menggunakan sebuah batang, dimana angka dikelompokkan dalam rentang yang tertentu. Dengan kata lain, frekuensi setiap elemen data di dalam daftar ditunjukkan menggunakan histogram. Angka yang dikelompokkan dalam bentuk rentang tertentu. Mari lihat contoh untuk lebih mengerti ini. Contoh data yang ingin kita tentukan histogramnya adalah sebagai berikut: $x = [2, 4, 6, 5, 42, 543, 5, 3, 73, 64, 43, 97, 63, 76, 63, 8, 73, 97, 23, 45, 56, \dots, 309]$. Script Python yang dapat kita gunakan untuk menampilkan histogram pada data di atas adalah Seperti pada listing 4.14 :

```

1 import matplotlib.pyplot as plt
2 x =
3     [2,4,6,5,42,543,5,3,73,64,42,97,63,76,63,8,73,97,23,45,56,...,309]
4 num_bins = 6
5 n, bins, patches = plt.hist(x, num_bins, facecolor = 'green')
6 plt.show()

```

Listing 4.14 Skrip Histogram

Error yang biasa ditemukan adalah seperti pada gambar 4.38:

```
C:\Users\user\Desktop>python test.py
  File "test.py", line 2
    x = [2,4,6,5,42,543,5,3,73,64,42,97 63,76,63,8,73,97,23,45,56,89,45,3,23,2,5,78,23,56,
          ^
SyntaxError: invalid syntax
```

Gambar 4.38 Error Histogram

Solusi untuk error diatas adalah pada saat mengetik script ada yang terlewat tanpa memberikan koma setelah angka, disini kita lupa memberi koma pada 97 dan 63. Seharusnya seperti pada gambar 4.39:

```
2 x = [2,4,6,5,42,543,5,3,73,64,42,97,63,76,63,8,73,97,23,45,56,89,45,3,23,2,5,78,23,56,
```

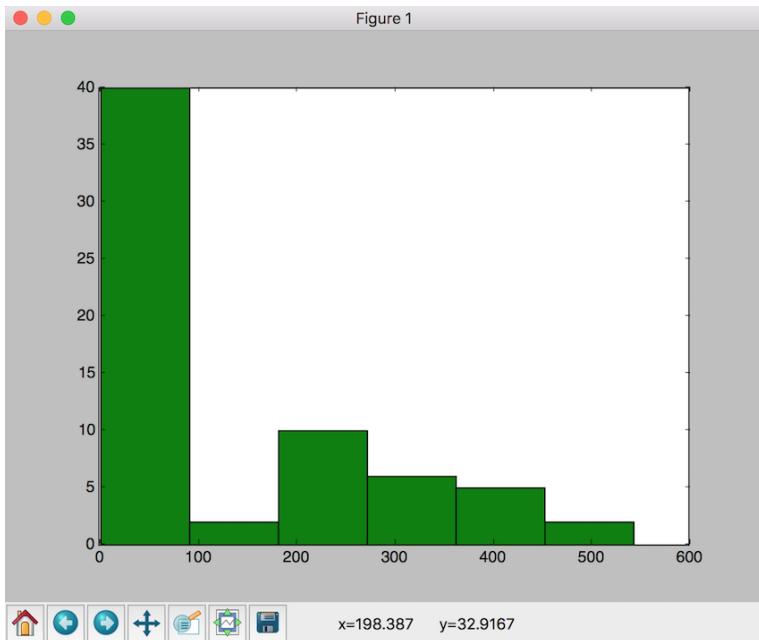
Gambar 4.39 Solusi Error Histogram

Kalau dari error yang lain adalah kita sebagai orang indonesia yang masih awam dengan sebutan warna atau color untuk program, tanpa sengaja kita mendefenisikan color=hijau. Itu jelas salah karena komputer atau program tidak akan mengerti bahasa itu, disini dapat kita ubah dengan facecolor = 'green' atau seperti pada gambar 4.40:

```
n, bins, patches = plt.hist(x, num_bins, facecolor = 'green')
```

Gambar 4.40 Solusi Error Histogram

Hasil tampilannya seperti pada gambar 4.41:



Gambar 4.41 Tampilan Histogram

4.8.4 Error Pada Saat Kustomisasi Plot

Sekarang kita dapat menambahkan judul, label untuk garis horisontal dan vertikal dengan memanggil fungsi `xtitle`, `xlabel`, dan `ylabel` Seperti pada listing 4.15 :

```

1 import matplotlib.pyplot as plt
2 plt.plot([1, 2, 3, 4], [13, 17, 19, 33]) []
3 plt.title("contoh judul pada matplotlib")
4 plt.xlabel("label x matplotlib")
5 plt.ylabel("label y matplotlib")
6 plt.show()

```

Listing 4.15 Kustomisasi Plot

Pada script diatas sudah ada fungsi `title`, `xlabel` dan `ylabel`.Tetapi ada beberapa error yang mungkin didapat jika kita salah pada pengetikan, seperti pada gambar 4.42:

Solusi dari error diatas adalah dengan menambahkan tanda kutip pada setiap kita mendefenisikan `title`, sehingga seperti pada gambar 4.43:

seperti pada gambar 4.44: Error 4.44 adalah kesalahan penulisan pada fungsi `plt.title` yang seharusnya digunakan `plt.title` (tanpa double T) atau seperti pada gambar 4.45:

Hasil tampilannya seperti pada gambar 4.46:

```
C:\Users\user\Desktop>python test.py
  File "test.py", line 4
    plt.title(Data Gelombang Otak)
                           ^
SyntaxError: invalid syntax
```

Gambar 4.42 Error Kustomisasi Plot

```
plt.title('Data Gelombang Otak')

plt.xlabel('Banyak')
plt.ylabel('Panjang Gelombang')
```

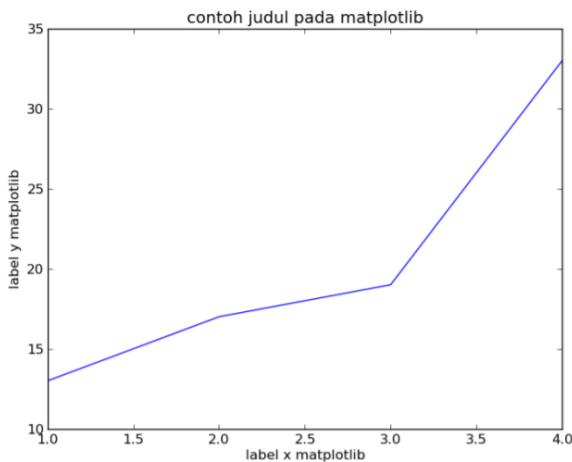
Gambar 4.43 Solusi Error Kustomisasi Plot

```
C:\Users\user\Desktop>python test.py
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    plt.tittle('Data Gelombang Otak')
AttributeError: 'module' object has no attribute 'tittle'
```

Gambar 4.44 Error Kustomisasi Plot

```
plt.title('Data Gelombang Otak')
```

Gambar 4.45 Solusi Error Kustomisasi Plot



Tabel ringkasan fungsi:

Fungsi	Keterangan
plot	melakukan plot
title	memberi judul pada gambar plot
xlabel	memberi nama label untuk garis x
ylabel	memberi nama label untuk garis y
show	menampilkan gambar plot

Gambar 4.46 Tampilan Kustomisasi Plot

4.9 Error Pada Pembacaan data dari sebuah file.txt

4.9.1 Membuat/memasukkan data melalui notepad

Cara yang pertama ialah dengan memasukkan seluruh data berupa angka ke sebuah halaman notepad dan kemudian disimpan sebagai file.txt (disini saya membuat nama file hasil.txt) seperti pada gambar 4.47:



The screenshot shows a Windows Notepad window titled "hasil - Notepad". The menu bar includes File, Edit, Format, View, and Help. The main content area contains the following numerical data:

```

50
41
40
90
101
94
78
102
59
70
198
98

```

Gambar 4.47 Data Hasil.txt

4.9.2 Error Pada Pembacaan data

Setelah kita berhasil membuat file dengan nama hasil.txt, kemudian kita akan memanggil file tersebut kesebuah kode program, sehingga source code akan Seperti pada listing 4.17 :

```

1 f = open("hasil.txt", "r")
2 print(f.read())

```

Listing 4.16 Skrip Pembacaan data

Source code diatas bernama ‘ bacadata.py ’ lalu dengan menggunakan python kita akan memanggil bacadata.py tersebut. Cara pemanggilan sebuah file python adalah sebagai berikut :

1. Buka CMD
2. Arahkan ke direktori tempat penyimpanan file python yang kamu buat (disini saya meletakkan file python bacadata.py tersebut di desktop)
3. Lalu panggil dengan perintah berikut seperti pada gambar 4.48: Adapun error-error pada script diatas ialah seperti pada gambar 4.49:

```
C:\Users\user\Desktop>python bacadata.py
```

Gambar 4.48 Panggil File bacadata.py

```
C:\Users\user\Desktop>python bacadata.py
  File "bacadata.py", line 3

          ^
SyntaxError: invalid syntax
```

Gambar 4.49 Error pada pembacaan data

```
f = open("hasil.txt", "r")
print(f.read())
```

Gambar 4.50 Script Error pada pembacaan data

Error diatas didapatkan dari script dibawah ini seperti pada gambar 4.50:

Yang membedakan script diatas dengan script diawal iyalah kita secara tidak sengaja lupa memberi kurung penutup pada script diatas, sehingga solusinya ialah seperti pada gambar 4.51:

```
f = open("hasil.txt", "r")
print(f.read())
```

Gambar 4.51 Solusi Error pada pembacaan data

Adapun hasil dari script pada pembacaan data diatas ialah seperti pada gambar 4.52:

4.9.3 Error Pemanggilan data ke sebuah plot

Seperti pada listing ?? :

```
1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
```

```
C:\Users\user\Desktop>python bacadata.py
50
41
40
90
101
94
78
102
59
70
198
98
```

Gambar 4.52 Tampilan pada pembacaan data

```
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12 plt.plot(x, marker='o')
13 plt.title('Data Gelombang Otak')
14 plt.xlabel('Banyak Data')
15 plt.ylabel('Gelombang otak')
16 plt.show()
```

Listing 4.17 Skrip pemanggilan data ke sebuah plot

Pada script diatas kita akan memanggil atau membaca file yang telah kita buat tadi yang bernama hasil.txt, tetapi biasa terdapat error-error pada script tersebut yaitu seperti pada gambar 4.53:

```
C:\Users\user\Desktop>python panggildata.py
Traceback (most recent call last):
  File "panggildata.py", line 10, in <module>
    x.append(int(row[1]))
IndexError: list index out of range
```

Gambar 4.53 Error pemanggilan data

Error pada gambar diatas terdapat pada `x.append(int(row[1]))` disini artinya kita membaca data pada baris kedua, jika [0] maka baris pertama dan [2] maka baris ke-dua dan seterusnya. Itulah yang menyebabkan script bisa jadi error, sehingga solusinya adalah mengganti angka 1 dengan 0 seperti pada gambar 4.54:

```
x.append(int(row[0]))
```

Gambar 4.54 Solusi Error pemanggilan data

Error lainnya seperti pada gambar 4.55:

```
C:\Users\user\Desktop>python panggildata.py
  File "panggildata.py", line 9
    for row in plots
          ^
```

Gambar 4.55 Error pemanggilan data

Pada gambar diatas coba kita teliti kembali pada script yang diawal, `for row in plots`, seharusnya setelah itu kita tambahkan ‘:’ agar tidak error seperti pada gambar 4.56:

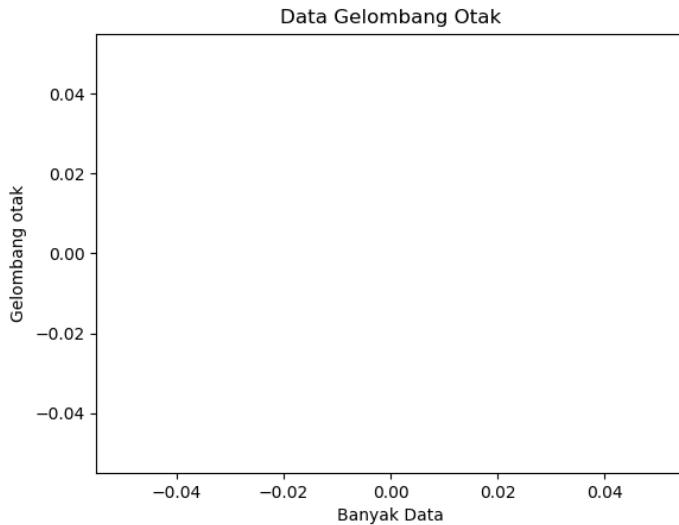
```
for row in plots:
```

Gambar 4.56 Solusi Error pemanggilan data

Error lainnya seperti pada gambar 4.57:

Pada gambar diatas didapatkan dari script berikut Seperti pada listing 4.18 :

```
1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12 plt.plot(marker='o')
13
14 plt.title('Data Gelombang Otak')
15
16 plt.xlabel('Banyak Data')
```



Gambar 4.57 Error tampilan pemanggilan data

```
18 plt.ylabel('Gelombang otak')
19
20 plt.show()
```

Listing 4.18 Contoh Skrip error pemanggilan data ke sebuah plot

Namun kenapa masih kosong tanpa ada gambaran sebuah plot ? Itu karena kita tidak meletakkan sumbu yang ingin dibaca oleh program, yaitu sumbu x, error terdapat pada baris ke 13, yang seharusnya seperti pada gambar 4.58:

```
...  
plt.plot(x, marker='^o^')
```

Gambar 4.58 Solusi Error pemanggilan data

Error lainnya seperti pada gambar 4.59:

Adapun error diatas karena kita mengganti marker menjadi A dan tidak bisa dipahami oleh program, adapun marker yang bisa kita buat pada matplotlib yaitu seperti pada gambar 4.60:

Sekarang contoh jika kita ingin mengubah marker menjadi square yaitu seperti pada gambar 4.61:

Adapun hasil dari script diatas jika mengubah marker menjadi s ialah seperti pada gambar 4.62:

```
C:\Users\user\Desktop>python panggildata.py
Traceback (most recent call last):
  File "panggildata.py", line 13, in <module>
    plt.plot(x, marker='A')
  File "C:\Python27\lib\site-packages\matplotlib\pyplot.py", line 3363, in plot
    ret = ax.plot(*args, **kwargs)
  File "C:\Python27\lib\site-packages\matplotlib\_init_.py", line 1867, in inner
    return func(ax, *args, **kwargs)
  File "C:\Python27\lib\site-packages\matplotlib\axes\_axes.py", line 1528, in plot
    for line in self._get_lines(*args, **kwargs):
  File "C:\Python27\lib\site-packages\matplotlib\axes\_base.py", line 406, in _grab_next_args
    for seg in self._plot_args(this, kwargs):
  File "C:\Python27\lib\site-packages\matplotlib\axes\_base.py", line 396, in _plot_args
    seg = func(x[:, j % nx], y[:, j % ny], kw, kwargs)
  File "C:\Python27\lib\site-packages\matplotlib\axes\_base.py", line 300, in _makeline
    seg = mlines.Line2D(x, y, **kw)
  File "C:\Python27\lib\site-packages\matplotlib\lines.py", line 397, in __init__
    self._marker = MarkerStyle(marker, fillstyle)
  File "C:\Python27\lib\site-packages\matplotlib\markers.py", line 189, in __init__
    self.set_marker(marker)
  File "C:\Python27\lib\site-packages\matplotlib\markers.py", line 272, in set_marker
    '{0}'.format(marker))
ValueError: Unrecognized marker style A
```

Gambar 4.59 Error pemanggilan data

4.10 Error Pada Modifikasi Hasil Plot

4.10.1 Error Setting Label

Kita dapat Kustomisasi Tampilan pada plot, pertama kita akan membahas tentang label. Anda dapat menambahkan judul, label untuk garis horisontal dan vertikal dengan menambahkan nama label untuk garis x dan y Disini ada contoh source code untuk menambahkan nama pada label, Seperti pada listing 4.19 :

```
1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12 plt.plot(x)
13 plt.xlabel('Banyak Data')
14 plt.ylabel('Gelombang otak')
15 plt.show()
```

Listing 4.19 Skrip Setting Label

Pada script diatas kita dapat mengubah kalimat yang terdapat pada xlabel dan ylabel. Tetapi setidaknya ada beberapa error yang biasa dialami pada script diatas yaitu seperti pada gambar 4.63:

Seperti yang error sebelumnya juga, kita secara tidak sengaja lupa memberi tanda kutip pada xlabel dan ylabel sehingga solusinya adalah seperti pada gambar 4.64:

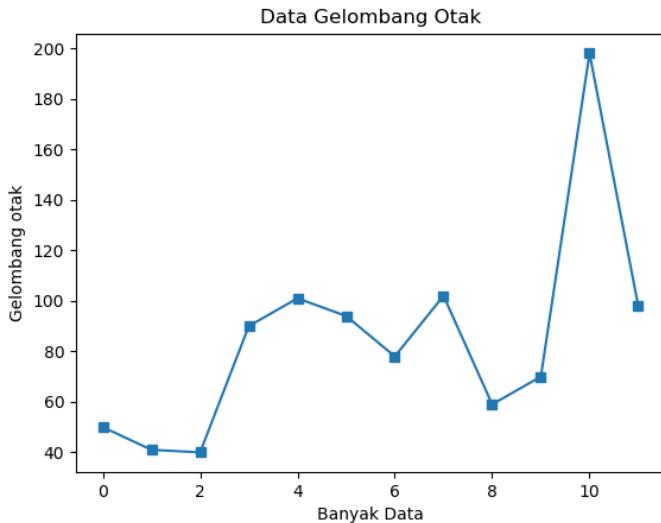
Adapun hasil dari setting label adalah seperti pada gambar 4.65: Yang bertanda

"_"	●	point
" , "	.	pixel
"o"	●	circle
"v"	▼	triangle_down
"^"	▲	triangle_up
"<"	◀	triangle_left
">"	▶	triangle_right
"1"	▼	tri_down
"2"	▲	tri_up
"3"	◀	tri_left
"4"	▶	tri_right
"8"	●	octagon
"s"	■	square
"p"	★	pentagon
"P"	✚	plus (filled)
"*"	★	star
"h"	●	hexagon1
"H"	●	hexagon2
"+"	+	plus
"x"	✗	x
"X"	✗	x (filled)
"D"	◆	diamond
"d"	◆	thin_diamond

Gambar 4.60 Marker Pada Matplotlib

```
plt.plot(x, marker='s')
```

Gambar 4.61 Skrip Marker Pada Matplotlib



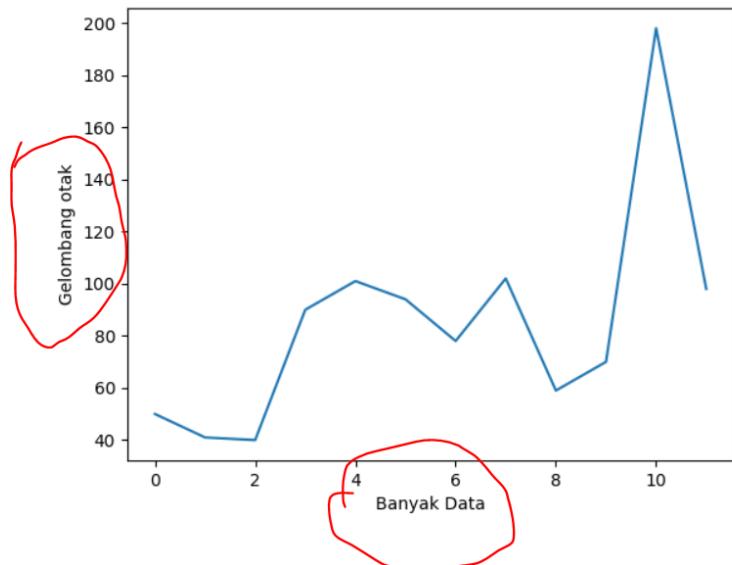
Gambar 4.62 Solusi Error Marker

```
C:\Users\user\Desktop>python panggildata.py
  File "panggildata.py", line 17
    plt.xlabel(Banyak Data)
                    ^
SyntaxError: invalid syntax
```

Gambar 4.63 Error Setting Label

```
plt.xlabel('Banyak Data')
plt.ylabel('Gelombang otak')
```

Gambar 4.64 Solusi Error Setting Label



Gambar 4.65 Tampilan Solusi Error Setting Label

merah ialah xlabel dan ylabel yang telah di setting

Adapun hasil yang ditampilkan pada source code diatas adalah seperti pada gambar 4.66:

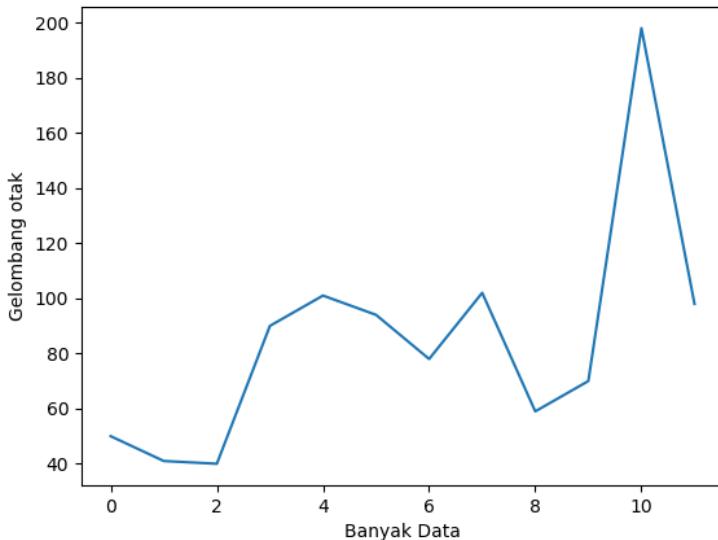
4.10.2 Error Setting Warna

Selanjutnya kita akan membahas bagaimana kustomisasi warna pada plot, dengan menggunakan facecolor = 'red' kita akan mengganti warna plot menjadi merah.

Disini ada contoh source code untuk mengganti warna pada plot, tetapi disini saya coba menggunakan plt.hist, Seperti pada listing 4.20 :

```

1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12
13 plt.hist(x, facecolor = 'red')
14 plt.xlabel('Banyak Data')
15 plt.ylabel('Gelombang otak')
```

**Gambar 4.66** Tampilan Setting Label

16 plt.show()

Listing 4.20 Skrip setting warna

Pada script diatas kita akan mengubah color atau warna pada plot hist yang akan kita baca dari data hasil.txt, tetapi ada beberapa error yang di dapat sebelum script diatas dibuat seperti pada gambar 4.67:

```
C:\Users\user\Desktop>python panggildata.py
Traceback (most recent call last):
  File "panggildata.py", line 12, in <module>
    plt.hist(x, facecolor = 'merah')
  File "C:/Python27/lib/site-packages/matplotlib/pyplot.py", line 3137, in hist
    stacked=stacked, normed=normed, data=data, **kwargs)
  File "C:/Python27/lib/site-packages/matplotlib/_init_.py", line 1867, in inner
    return func(ax, *args, **kwargs)
  File "C:/Python27/lib/site-packages/matplotlib/axes\_axes.py", line 6827, in hist
    p.update(kwargs)
  File "C:/Python27/lib/site-packages/matplotlib/artist.py", line 888, in update
    for k, v in props.items()]
  File "C:/Python27/lib/site-packages/matplotlib/artist.py", line 882, in _update_property
    return func(v)
  File "C:/Python27/lib/site-packages/matplotlib/patches.py", line 315, in set_facecolor
    self._set_facecolor(color)
  File "C:/Python27/lib/site-packages/matplotlib/patches.py", line 305, in _set_facecolor
    self._facecolor = colors.to_rgba(color, alpha)
  File "C:/Python27/lib/site-packages/matplotlib/colors.py", line 168, in to_rgba
    rgba = _to_rgba_no_colorcycle(c, alpha)
  File "C:/Python27/lib/site-packages/matplotlib/colors.py", line 212, in _to_rgba_no_colorcycle
    raise ValueError("Invalid RGBA argument: {!r}".format(orig_c))
ValueError: Invalid RGBA argument: 'merah'
```

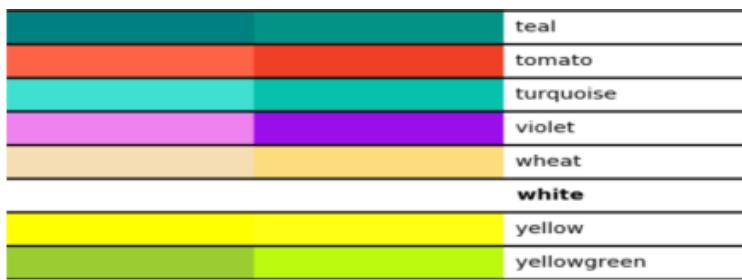
Gambar 4.67 Error Setting Warna

Pada gambar diatas kita salah memberikan warna pada facecolor yang scriptnya seperti pada gambar 4.68:

```
plt.hist(x, facecolor = 'merah')
```

Gambar 4.68 Solusi Error Setting Warna

Python tidak memahami pendefenisian merah untuk facecolor, adapun kurang lebih modul color pada matplotlib adalah seperti pada gambar 4.71:



Gambar 4.69 Color Pada Matplotlib 3

Adapun solusi dari error warna tersebut ialah dengan membuat nama warna sesuai modul yang ada pada matplotlib seperti pada gambar 4.72:

Adapun hasil dari script diatas ialah seperti pada gambar 4.73:

4.10.3 Error Pada Setting Marker

Selanjutnya kita akan kustomisasi marker pada plot, disini kita bisa mengganti penanda label pada x dan y Ikuti contoh source code berikut Seperti pada listing 4.21 :

```
1 import matplotlib.pyplot as plt
2 import csv
3
4 x=[]
5 y=[]
6
7 with open('hasil.txt', 'r') as csvfile:
8     plots= csv.reader(csvfile, delimiter=',')
9     for row in plots:
10         x.append(int(row[0]))
11
12
13 plt.plot(x, marker='o')
14 plt.xlabel('Banyak Data')
15 plt.ylabel('Gelombang otak')
16
```

	aqua
	aquamarine
	azure
	beige
	black
	blue
	brown
	chartreuse
	chocolate
	coral
	crimson
	cyan
	darkblue
	darkgreen
	fuchsia
	gold
	goldenrod
	green
	grey
	indigo
	ivory

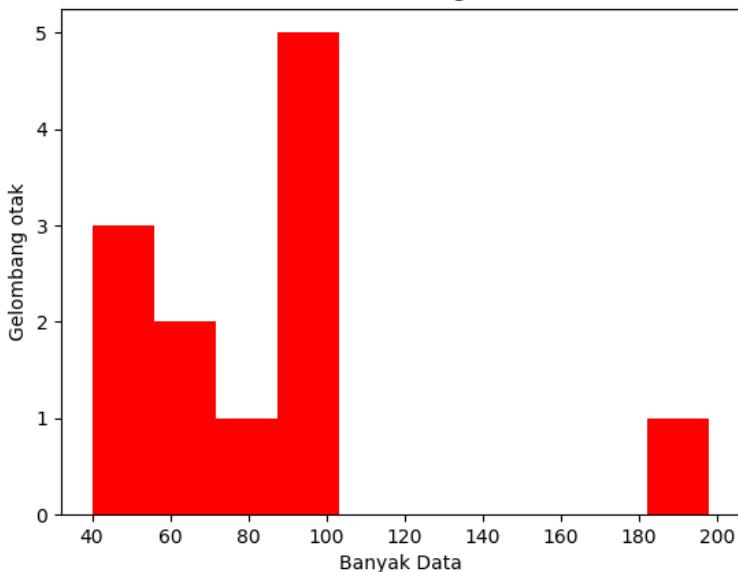
Gambar 4.70 Color Pada Matplotlib 2



Gambar 4.71 Color Pada Matplotlib 1

```
plt.hist(x, facecolor = 'red')
```

Gambar 4.72 Solusi Error Color Pada Matplotlib



Gambar 4.73 Tampilan Setting Warna

```
17 plt.show()
```

Listing 4.21 Skrip setting marker

Error yang paling sering dalam mengganti atau mengubah marker ialah kita tidak memperhatikan besar dan kecilnya huruf pada modul marker tersebut, sehingga jika kita mau membuat marker Square yaitu dengan huruf s (kecil) tetapi di script kita meletakkan S (besar) maka itu akan terjadi error seperti pada gambar 4.74:

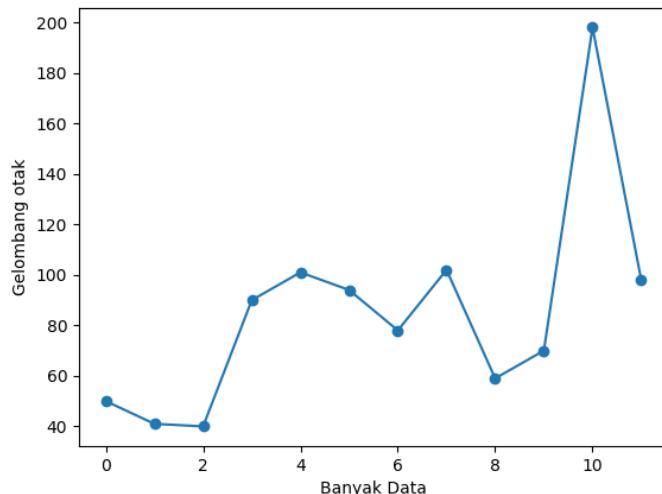
Solusinya ialah dengan mengikuti modul matplotlib yang ada dengan mengganti s seperti pada gambar 4.75:

Gambar 6.46 Solusi Error Setting Warna dan hasilnya akan seperti pada gambar 4.76:

```
C:\Users\user\Desktop>python panggildata.py
Traceback (most recent call last):
  File "panggildata.py", line 13, in <module>
    plt.plot(x, marker='S')
  File "C:/Python27/lib/site-packages/matplotlib/pyplot.py", line 3363, in plot
    ret = ax.plot(*args, **kwargs)
  File "C:/Python27/lib/site-packages/matplotlib/_init_.py", line 1867, in inner
    return func(ax, *args, **kwargs)
  File "C:/Python27/lib/site-packages/matplotlib/axes\_axes.py", line 1528, in plot
    for line in self._get_lines(*args, **kwargs):
  File "C:/Python27/lib/site-packages/matplotlib/axes\_base.py", line 406, in _grab_next_args
    for seg in self._plot_args(this, kwargs):
  File "C:/Python27/lib/site-packages/matplotlib/axes\_base.py", line 396, in _plot_args
    seg = func(x[:, j % ncy], y[:, j % ncy], kw, kwargs)
  File "C:/Python27/lib/site-packages/matplotlib/axes\_base.py", line 300, in _makeline
    seg = mlines.Line2D(x, y, **kw)
  File "C:/Python27/lib/site-packages/matplotlib/lines.py", line 397, in __init__
    self._marker = MarkerStyle(marker, fillstyle)
  File "C:/Python27/lib/site-packages/matplotlib/markers.py", line 189, in __init__
    self.set_marker(marker)
  File "C:/Python27/lib/site-packages/matplotlib/markers.py", line 272, in set_marker
    '%0}'.format(marker))
ValueError: Unrecognized marker style S
```

Gambar 4.74 Error Setting Marker

```
plt.plot(x, marker='S')
```

Gambar 4.75 Solusi Setting MArker**Gambar 4.76** Tampilan Setting Marker

BAB 5

PENGENALAN FLASK

5.1 Definisi *Micro Framework*

Microframework adalah istilah yang digunakan untuk merujuk pada kerangka kerja web minimalis. Kerangka ini sangat berbeda dengan kerangka kerja tumpukan penuh. Juga tidak memiliki sebagian besar fungsionalitas secara umum yang ada dalam kerangka kerja aplikasi web lengkap, seperti:

1. Akun, otentikasi, otorisasi, peran, dll.
2. Abstraksi basis data melalui pemetaan objek-relasional.
3. Validasi *input* dan sanitasi *input*.
4. Mesin *template web*.

Biasanya, sebuah *microframework* memfasilitasi untuk menerima permintaan HTTP, merutekan permintaan HTTP ke *controller* yang sesuai, mengirim *controller*, dan mengembalikan respons HTTP. *Microframeworks* seringkali dirancang khusus untuk membangun API untuk layanan atau aplikasi lain. Misalnya, Lumen *microframework* yang dirancang untuk pengembangan *Microservices* dan pengembangan API.

Microframework merupakan sebuah *tool* yang digunakan untuk *project* yang lebih kecil dan penggunaan untuk kasus yang spesifik. Ini sama saja dengan menyederhanakan *framework* agar lebih mudah dalam implementasi dan menyediakan *testing* dan *deployment* yang lebih cepat. *Microframework* mengeluarkan banyak sekali komponen yang ada pada pengaturan *full-stack*, termasuk diantaranya:

1. *Web template engine*;
2. *Input validation*;
3. *Database abstraction*;
4. *Roles, accounts, and authentication*.

Kerugian menggunakan *microframework* adalah saat *project* mulai tumbuh besar dengan cepat, dimana *microframework* tidak memiliki fitur yang dibutuhkan untuk mengakomodasi pertumbuhan website. Dengan kata lain kamu kehilangan fleksibilitas. *Micro-framework* lebih baik saat digunakan untuk *project* kecil yang membutuhkan kesederhanaan, *overhead* yang rendah dan *deployment* yang cepat. *Developer* yang sudah berpengalaman bisa saja menggunakan *microframework* pada awal *project* dan menambahkan tambahan *microframework* jika diperlukan. Hal ini merupakan pilihan yang menarik, tetapi untuk pemula dan *developer* menengah harus menghindari ini [2].

5.2 Jenis-Jenis *Framework* Python serta Kelebihan dan Kekurangan

5.2.1 Django

Django adalah kerangka kerja web Python yang memungkinkan individu dalam pengembangan yang bersih dan cepat. Kerangka kerja web secara umum dikatakan sebagai campuran komponen yang membantu pengembang mengembangkan situs web lebih cepat dan mudah. Karena itu, ini adalah kerangka kerja sumber bebas dan terbuka. Ini dapat disebut sebagai kerangka kerja yang memungkinkan pengembang untuk mengambil konsep penyelesaian secepat mungkin. Django sebagai kerangka kerja membantu mengurangi beberapa kesalahan keamanan umum yang dapat diawasi dengan mudah saat mengembangkan aplikasi. Skalabilitas adalah fitur lain yang disediakan oleh kerangka ini.

Django memiliki *tagline* “*The web framework for perfectionists with deadlines*”, bagaimana tidak, karena secara *default* Django sudah memiliki berbagai modul umum yang biasa digunakan ketika mengembangkan aplikasi web.

Kelebihan:

1. Cepat.

Ini telah dirancang sedemikian rupa untuk membantu pengembang membuat aplikasi secepat mungkin. Dari ide, produksi hingga rilis, Django membantu menjadikannya efektif dan efisien. Dengan demikian itu menjadi solusi ideal bagi pengembang yang memiliki fokus utama pada tenggat waktu.

2. Penuh dimuat.

Ini bekerja dengan cara yang mencakup puluhan tambahan untuk membantu dengan otentifikasi pengguna, peta situs, administrasi konten, umpan RSS, dan banyak lagi hal-hal seperti itu. Aspek-aspek ini membantu dalam melaksanakan proses pengembangan web sepenuhnya.

3. Aman.

Ketika melakukannya di Django, dipastikan bahwa pengembang tidak melakukan kesalahan yang terkait dengan keamanan. Beberapa kesalahan umum termasuk injeksi SQL, pemalsuan permintaan lintas situs, clickjacking dan skrip lintas situs. Untuk mengelola nama pengguna dan kata sandi secara efektif, sistem otentifikasi pengguna adalah kuncinya.

4. Dapat diukur.

Untuk memenuhi permintaan lalu lintas terberat, manfaat kerangka Django dapat dilihat. Oleh karena itu, situs tersibuk menggunakan media ini untuk dengan cepat memenuhi permintaan lalu lintas.

5. Serbaguna.

Manajemen konten, *platform* komputasi ilmiah, dan bahkan organisasi besar, semua aspek ini dikelola secara efisien dengan penggunaan Django.

6. Dokumentasi yang sangat lengkap dan kamu tidak perlu banyak - banyak *googling* karena sudah disediakan contoh.
7. Modul administrasi yang *auto generate* sesuai dengan model yang didefinisikan di dalam aplikasi. Lebih dari sekedar *CRUD generator*.
8. Sistem migrasi *database* otomatis yang tidak perlu kamu tulis *script*-nya. Cukup mengubah class dan struktur *database* pun berubah sesuai perubahan terakhir.
9. Memiliki sistem form yang kokoh.
10. Sudah *built-in* untuk sistem autentikasi dan roles bila Anda menggunakan *relational database* yang didukung Django seperti MySQL dan PostgreSQL.
11. Memiliki ekstensi - ekstensi yang bisa membuat kamu lebih produktif seperti *Django Rest Framework*, *Django Rest Auth*, *Django Celery*, *Django Mongo-engine*, *GeoDjango*, dan lainnya.
12. Memiliki *template engine* sendiri yang lebih *powerful*.
13. Kompatibilitas dengan berbagai modul dan *library* lain.

Kekurangan:

1. Menggunakan pola perutean, tentukan URL-nya
2. Django terlalu monolitik

3. Semuanya didasarkan pada Django ORM
4. Komponen dikerahkan bersama
5. Pengetahuan tentang sistem lengkap diperlukan untuk bekerja.

5.2.2 Flask

Python adalah Flask, yang merupakan kerangka kerja mikro untuk Python berdasarkan teknologi seperti Werkzeug, Jinja 2. Flask pada dasarnya adalah kerangka kerja web Python yang dibangun dengan inti kecil dan selanjutnya mudah untuk memperpanjang ekstensi Flask lebih berorientasi Python daripada Django karena beberapa alasan yang jelas. Karena ada sedikit kode *boilerplate* yang harus ditangani oleh pengembang, Flask adalah kerangka kerja web yang mungkin tidak perlu dikerjakan pengembang lebih lama untuk memahami mereka. Banyak aplikasi terkenal di luar sana ditulis dalam kerangka kerja Flask seperti Pinterest, LinkedIn dan halaman web komunitas untuk Flask itu sendiri [3].

Flask sendiri dapat dikatakan sebagai *web framework* yang fleksibel terhadap library apapun untuk Python. Selain itu dokumentasinya yang jelas membuat Flask sangat diminati oleh kawula muda.

Kelebihan:

1. *Framework* yang mudah untuk digunakan dan dipahami.
2. Berisi pengembangan server dan *debugger*.
3. *RESTfull request dispatching*.
4. Menggunakan Jinja2 *template engine*.
5. Dukungan untuk *secure cookies* pada sisi *client*.
6. 100% *Web Server Gateway Interface* (WSGI).
7. Berbasis *Unicode* yaitu suatu standar yang dirancang untuk mengizinkan *text* dan *symbol* dari semua tulisan untuk menampilkan dan dimanipulasi secara konsisten oleh komputer.
8. Dokumentasi yang ekstensif.
9. Kompatibilitas *Google App Engine*.

Kekurangan:

1. Fungsionalitas

Beberapa fitur Flask yang perlu kamu ketahui antara lain:

1. *Built-in development server* dan *debugger*.
2. Terintegrasi dengan unit *testing*.

3. RESTful.
4. Menggunakan *template engine* Jinja2.
5. Mendukung *secure cookie*.
6. 100% mendukung WSGI 1.0.
7. *Unicode based*.
8. Dokumentasi yang baik.
9. Komunitas yang kuat.

5.2.3 Tornado

Tornado adalah salah satu kerangka kerja web terbaik dari bahasa pemrograman Python. Kerangka kerja ini memungkinkan pendekatan yang lebih bersih untuk pemrograman server Web dan memiliki fokus yang tajam pada operasi non-pemblokiran, dapat meningkatkan skala ke sejumlah besar koneksi terbuka [4].

Kelebihan:

1. Dukungan bawaan

Tornado hadir dengan dukungan bawaan dan menemukan solusi untuk sebagian besar aspek pengembangan Web yang membosankan seperti template, pelokalan, cookie yang ditandatangani, dll. Tornado juga memungkinkan pengguna untuk mencampurnya dengan kerangka kerja lain, dengan cuplikan yang sesuai, sesuai untuk kebutuhan mereka.

2. Koneksi serentak

Tornado menawarkan layanan waktu nyata dan mendukung sejumlah besar koneksi konkuren, streaming HTTP (protokol komunikasi yang diterapkan oleh Apple Inc) dan polling panjang (ini adalah teknologi sembur, yang memungkinkan mekanisme pendorong emulatif dalam keadaan di yang dorongan nyata tidak mungkin). Dengan Tornado, sangat mudah untuk menulis layanan waktu nyata. FriendFeed memelihara koneksi terbuka, terutama untuk penggunanya yang sering terlibat.

3. Kinerja tinggi.

Ini adalah fitur paling menarik dari Tornado. Ini sangat cepat dibandingkan dengan semua kerangka kerja Python Web lainnya. Mempertimbangkan keluaran dasarnya, kecepatannya sekitar empat kali lebih tinggi dan juga cukup efisien.

Tornado memiliki beberapa modul utama seperti:

1. *Web framework*.
2. *HTTP Server and Client*.
3. *Asynchronous Networking, Coroutines and Concurrency*.
4. *Utilities*.

5.2.4 Falcon

Falcon merupakan pustaka WSGI yang membantu membangun API web dengan kecepatan lebih cepat. Saat Anda membuat kerangka HTTP API selain Falcon dapat memuat banyak dependensi dan abstraksi yang tidak diperlukan. Falcon, di sisi lain, mengurangi semua ketergantungan dan menyediakan pengembang untuk mengembangkan desain yang lebih bersih yang memungkinkan gaya arsitektur HTTP dan REST[5].

Falcon mengklaim bahwa ia dapat menangani lebih banyak permintaan dengan perangkat keras yang sama jika sedang ditangani oleh kerangka kerja lain. Kerangka kerja ini bertujuan untuk memiliki cakupan kode 100%, sehingga membuatnya lebih andal. Sebagian besar fitur di atas dimungkinkan karena Falcon hanya mempertahankan 2 dependensi pihak ketiga seperti enam, mimeparse. Sesuai dengan halaman Falith Github perusahaan seperti RackSpace, OpenStack dan LinkedIn menggunakan Falcon.

Dengan tagline *The Minimalist Python WSGI Framework*, Falcon siap menyuguhkan berbagai fitur yang dapat mempermudah kamu membangun sebuah RESTful API. Falcon merupakan *high performance web framework* yang dapat digunakan untuk membangun HTTP API dan backend apps.

Kelebihan:

1. Cepat

Salah satu persyaratan paling penting dari cloud API adalah mereka harus menanggapi permintaan secepat mungkin. Ini menjadi fitur penting dalam skenario real-time ketika jumlah permintaan bersamaan tinggi. Falcon adalah salah satu kerangka kerja tercepat yang tersedia.

2. Cahaya

Kerangka kerja dengan jejak ketergantungan yang lama menjadi sangat sulit untuk disatukan dalam berbagai lingkungan karena pembatasan yang diberlakukan oleh ketergantungan tersebut. Falcon hanya memiliki dua dependensi: enam (pustaka kompatibilitas Python 2 dan 3, yang memfasilitasi basis kode untuk bekerja pada Python 2 dan 3 tanpa memerlukan perubahan apa pun) dan mimeparse (yang menyediakan fungsi seperti parsing nama tipe-mime). Ini membuat Falcon lebih mudah untuk diuji dan digunakan.

3. Fleksibel

Falcon tidak membatasi pengembang ketika memilih perpustakaan sehubungan dengan database, otorisasi, dll. Pengembang dapat memilih perpustakaan yang mereka sukai, yang cocok dengan persyaratan skenario proyek saat ini.

Kekurangan:

1. Tidak cocok untuk melayani halaman HTML.
2. Belum tentu lebih cepat daripada Flask.

5.2.5 Hug

Hug adalah kerangka kerja berbasis web Python yang lain memberi para pengembang fleksibilitas mengembangkan API dan memungkinkan dapat mengkonsumsinya sesuka mereka. Pengembangan API telah disederhanakan secara drastis melalui beberapa antarmuka. Biarlah itu pengembangan lokal atau melalui HTTP atau bahkan melalui antarmuka baris perintah (CLI), sejauh ini merupakan cara modern tercepat untuk mengembangkan API. Kerangka kerja Hug telah dibangun dengan fokus tunggal pada kinerja dalam pikiran. Dikatakan mengkonsumsi sumber daya hanya bila diperlukan dan selanjutnya dikompilasi menggunakan Cython untuk mencapai angka-angka luar biasa pada kinerja. Dengan semua alasan yang jelas ini, Hug mencuri mahkota sebagai kerangka kerja web tercepat untuk Python 3.

Kelebihan:

1. Jadikan mengembangkan API yang digerakkan oleh Python sesingkat definisi tertuli.
2. Kerangka kerja harus mendorong kode yang mendokumentasikan sendiri.
3. Itu harus cepat. Pengembang seharusnya tidak pernah merasa perlu mencari di tempat lain karena alasan kinerja.
4. Tes penulisan untuk API yang ditulis di atas pelukan harus mudah dan intuitif.
5. Sihir yang dilakukan sekali, dalam kerangka kerja API, lebih baik daripada mendorong masalah yang disetel ke pengguna kerangka API.
6. Menjadi dasar untuk API Python generasi berikutnya, merangkul teknologi terbaru.

Kekurangan:

1. Hanya dapat memuat kode sedikit.

5.2.6 Sanic

Sanic adalah kerangka kerja web Python *cocokuntuk Python 3.5* yang dibangun di atas *uvloop* dan dirancang untuk respons HTTP yang lebih cepat melalui penanganan permintaan yang tidak sinkron. Karena struktur internal dan ketergantungannya yang kuat pada uvloop, itu tidak dapat dikembangkan atau digunakan pada lingkungan Windows. Sampai saat ini, Sanic masih dalam tahap pengembangan dan dianggap sebagai bayi di antara kerangka web lain yang tersedia untuk Python. Dengan ini, ada sejumlah kode yang telah ditulis di sekitar Sanic agar Anda dapat menggunakan-nya untuk keperluan bisnis yang kompleks. Mengingat masih dalam pengembangan, tidak ada banyak aplikasi atau ekstensi untuk Sanic dibandingkan dengan Flask atau Django. Mengingat semua itu, kerangka kerja ini memungkinkan Anda untuk mengambil keuntungan dari sintaks async atau menunggu untuk mendefinisikan

fungsi asinkron Anda sendiri. Ini memberikan kekuatan menulis aplikasi asinkron seperti apa yang dapat dicapai dengan menggunakan Node.js.

Kelebihan:

1. Server yang dikonfigurasikan untuk harus dilampirkan ke aplikasi yang ada.
2. Aplikasi Sanic dapat menentukan rute reguler yang akan hidup berdampingan dengan server Engine.IO. Pola khas adalah menambahkan rute yang melayani aplikasi klien dan file statis terkait dengan aplikasi ini.

Kekurangan:

- Tidak memiliki banyak aplikasi atau ekstensi.

5.2.7 Aiohttp

Kerangka kerja asinkron yang sangat bergantung pada dan menggunakan fitur Python 3.5+ seperti async dan menunggu. Kerangka kerja ini tidak hanya kerangka kerja server web tetapi juga bertindak sebagai kerangka kerja klien juga, karena mendukung baik WebSocket Server dan Klien. Ini adalah kerangka kerja terkenal yang memanfaatkan perpustakaan asinkron populer - asyncio yang ada di sana sejak awal perpustakaan. aiohttp seperti Flask menyediakan objek permintaan dan router untuk memungkinkan pengalihan permintaan ke fungsi yang dikembangkan untuk managannya. Sebagai pengembang layanan-mikro, Anda bisa fokus membangun pandangan seperti yang akan Anda lakukan dengan Flask.

Kelebihan:

1. Efisiensi.

Menangani jumlah permintaan yang setara dengan server yang lebih sedikit atau lebih kecil dibandingkan dengan sinkronisasi. Skalabilitas dibatasi oleh jumlah koneksi soket terbuka dalam proses tunggal vs. jumlah utasproses bersamaan untuk sinkronisasi kerangka kerja web (ribuan hingga puluhan ribu untuk async vs. puluhan hingga ratusan untuk sinkronisasi). Server kecil (dalam hal CPU dan memori) menjalankan web async layanan dalam satu proses akan cocok dan seringkali mengungguli yang lebih besar server menjalankan layanan web sinkronisasi menggunakan puluhan hingga ratusan utas/proses.

2. Mampu menangani sejumlah besar permintaan bersamaan.

Mengizinkan fungsionalitas *push* yang efisien melalui soket web, EventSource, atau koneksi berumur panjang lainnya

Kekurangan:

- Tunggal.

Memiliki model yang lebih kompleks untuk dipikirkan status bersama dan bagaimana hal itu dapat berubah dari satu proses sinkronisasi, harus diingat bahwa status bersama dapat berubah di antara saat-saat menghasilkan kontrol ke loop acara dan mengembalikan kontrol ke kode Anda.

5.2.8 Piramid

Kerangka kerja yang telah dikembangkan atau dibangun untuk aplikasi yang lebih besar. Piramida, nama itu sendiri menunjukkan bahwa itu fleksibel, tidak seperti Django yang menawarkan pendekatan "semuanya di dalam kotak". Aplikasi web dibangun menggunakan Pyramid, mulai dari modul file tunggal dan kemudian proyek-proyek ini berkembang menjadi proyek yang lebih besar dan ambisius dalam waktu singkat. Kerugian dari kerangka kerja web ini adalah dokumentasi mereka sendiri, yang tidak terlalu jelas dan terkadang membingungkan. Chameleon Pyramid dipasang untuk menggunakan templat Chameleon alih-alih templat Jinja. Dibutuhkan beberapa waktu dalam mengembangkan aplikasi file tunggal dengan Pyramid, tetapi kemudian, ini dapat ditingkatkan lebih cepat karena pengaturan awal lebih keras dan rawan kesalahan.

Kelebihan:

1. Fleksibilitas.

Sistem rendering template, cara menyambungkan ke database, cara memetakan url ke tampilan , semacam sistem otentifikasi, dan banyak hal lainnya. Piramida sangat bagus karena semua komponen ini dapat ditukar. Anda dapat memilih mesin rendering template, memiliki dua cara berbeda untuk memetakan url ke tampilan dan dapat menggunakan keduanya di aplikasi yang sama, dapat menggunakan metode apa pun yang ingin disambungkan ke database (meskipun SQLAlchemy umumnya digunakan), dan bahkan dapat terhubung ke beberapa database dari tipe yang sangat berbeda.

2. Kemudahan AJAX.

Penggunaan dekorator dan tampilan XHR membuatnya sangat mudah untuk mendapatkan permintaan AJAX untuk pergi ke tempat yang Anda inginkan. Menjelaskan XHR, dekorator, atau AJAX jauh di luar cakupan tutorial ini, tetapi saya jamin penjelasannya ada di luar sana.

3. Dukungan SQLAlchemy

SQLAlchemy adalah hal yang sangat kuat, banyak orang yang sangat pintar berpikir itu adalah ORM terbaik di sekitar. Jika Anda memilih Django Anda memilih untuk tidak menggunakan SQLAlchemy. Itu hanya masalah besar jika aplikasi Anda sangat (SQL) database intensif - jika Anda ingin melakukan pertanyaan rumit dengan cara yang waras. Di sisi lain, jika aplikasi Anda sederhana maka SQLAlchemy tidak akan banyak membantu, tetapi tidak ada salahnya untuk memilikiinya.

Kekurangan:

1. Fungsionalitas

Beberapa fitur Pyramid:

1. Kompatibel dengan berbagai template engine seperti Jinja2, Chameleon, dan Mako.

2. Mempunyai sistem form yang handal.
3. Menggunakan SQL Alchemy untuk teknologi database.
4. Memiliki bootstraper.
5. Function decorator.
6. Asset management.
7. Event dan subscriber

5.2.9 Web2py

Merupakan salah satu *full-stack enterprise framework* yang *free* dan *open source* untuk membangun aplikasi web berbasis *database* yang aman. Web2Py merupakan salah satu *web framework* yang masih ada hingga hari ini. Tidak hanya soal *url routing*, Web2Py pun memiliki *template engine* yang cukup *powerful* untuk membuat halaman web.

Beberapa fitur yang dimiliki oleh Web2py antara lain:

1. Dibuat oleh komunitas terpercaya.
2. Selalu backward compatible.
3. Mudah digunakan.
4. Dapat berjalan di banyak sistem operasi.
5. Dapat berjalan di banyak web server.
6. Dapat "berbicara" ke SQLite3, PostgreSQL, MySQL, MSSQL, FireBird, Oracle, IBM DB2, Informix, Ingres, dan Google App Engine.
7. Aman dari cross site scripting, injection flaws, dan eksekusi file berbahaya.
8. Mengajarkan penggunanya apa arti MVC yang sesungguhnya.
9. Kompatibel dengan berbagai protokol seperti REST, RSS, HTML, REST, XML-RPC, dan lainnya.
10. Dukungan berbagai modul dan library yang sudah disediakan oleh Web2py

5.2.10 TurboGears2

TurboGears2 adalah kerangka web berbasis Python yang didasarkan pada paradigma ObjectDispatch. Ini secara khusus dimaksudkan untuk memungkinkan untuk menulis aplikasi kecil dan ringkas dalam mode Minimal dan aplikasi yang jauh lebih kompleks dalam Mode Stack Penuh. Fitur TurboGears2 adalah ORM dengan dukungan

multi-database nyata, dan juga mendukung partisi data horizontal, sistem widget untuk menyederhanakan pengembangan aplikasi AJAX.

Kelebihan:

1. Dukungan Web Server Gateway Interface (WSGI)
2. Sistem widget yang mempermudah pembuatan aplikasi AJAX
3. Mendukung multi data-exchange format
4. Dapat membuat pluggable application
5. Template engine yang sangat designer friendly

Kekurangan:

1. Fungsionalitas

5.2.11 Cherrypy

CherryPy yang merupakan kerangka kerja berikutnya dalam daftar, yang digunakan untuk menjadi jalan antara masalah dan programmer. Aplikasi web yang dibangun menggunakan kerangka CherryPy terlihat seperti aplikasi Python lainnya dan berjalan tanpa memberikan pengaturan rumit dan penyesuaian terbaik. Bersamaan dengan itu, ia juga memperluas dukungannya ke berbagai jenis server web seperti Apache, IIS dan banyak lagi. Karena itu, CherryPy mengemasnya bersama server web, sehingga aplikasi dapat digunakan di mana pun Python diinstal. Ini juga memungkinkan Anda untuk memulai beberapa server HTTP sekaligus. Tidak ada paksaan yang diberlakukan oleh CherryPy untuk menggunakan mesin templat tertentu, ORM atau pustaka JavaScript dan karenanya kami sebagai pengembang memiliki pilihan untuk memilih mana yang sesuai dengan kebutuhan kami dengan lebih baik.

Kelebihan:

1. CherryPy berjalan dengan mudah di lingkungan yang kompatibel dengan WSGI seperti server web Apache dan bahkan dapat dijalankan di server mandiri tanpa gangguan langkah otorisasi dan akses backend.
2. Pengembangan aplikasi web adalah tersedianya pilihan penyesuaian, dengan demikian menarik lebih banyak peminat untuk itu. Dan bit penyesuaian hanya dimungkinkan karena CherryPy menawarkan berbagai macam fitur dan alat untuk memungkinkannya bagi pengembang.

Kekurangan:

1. Tidak memiliki sistem templating sendiri sehingga harus memilih template yang cocok.

CherryPy memiliki dukungan seperti berikut:

1. URL routing.

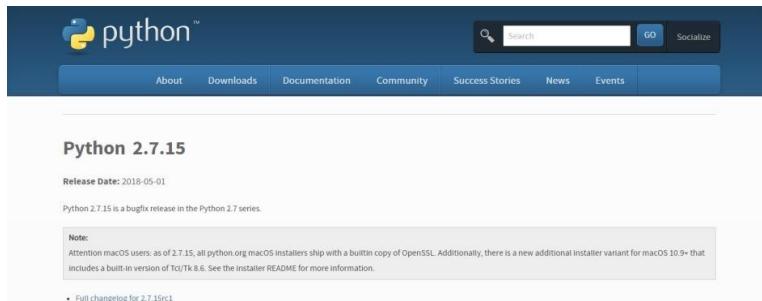
2. Static file management.
3. Managable configuration.
4. Kompatibel dengan WSGI dan HTTP/1.1.
5. Built-in profiling, coverage, dan dukungan testing.
6. Dukungan terhadap session, authentication, static content, dan banyak lagi.
7. Dukungan bawaan untuk caching dan encoding.
8. Sistem konfigurasi yang enak.

5.3 Instalasi dan Hello World di Flask

5.3.1 Instalasi Python 2.7.

Mulai dengan tutorial dalam menginstall Python 2.7. Python ini digunakan untuk code pembaca data dari sinyal gelombang otak yang telah dihasilkan oleh alat EEG yaitu NeuroSky Mindwave. Baiklah langsung kita mulai saja:

1. Pertama-tama silahkan download software dari python versi 2 di laptop anda. Download python versi 2.7.15 dari situs web resminya di www.python.org. Silahkan sesuaikan dengan kapasitas laptop anda, bisa yang win 32 atau yang win 64. Contoh downloadnya seperti pada gambar 5.1.



Gambar 5.1 Download Softfile Python 2.7.

2. Setelah berhasil mendownload pythonnya silahkan lakukan instalasi seperti biasa anda lakukan. Setelah selesai instalasi pythonnya silahkan cek di *Command Prompt*, apakah python telah terbaca/running disesuaikan dengan laptop anda atau belum. Contoh pengecekan di Command Prompt seperti pada gambar 5.2
3. Apabila tampilannya telah sesuai dengan gambar 5.2, maka python anda siap digunakan.

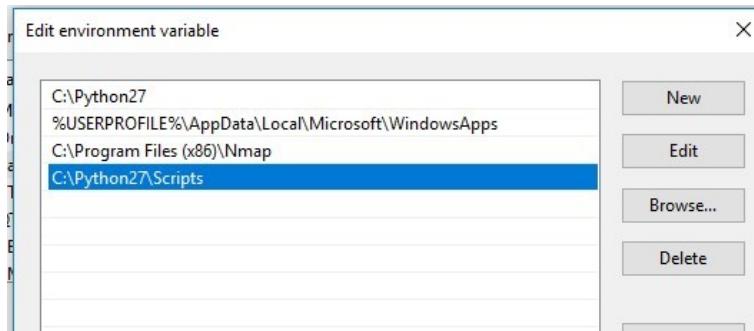
```
Command Prompt - python
Microsoft Windows [Version 10.0.17134.407]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ASUS>python
Python 2.7.15 (v2.7.15:c0089a3ea3, Apr 30 2018, 16:22:17) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Gambar 5.2 Pengecekan Python 2.7.

- Pastikan python yang terbaca versi 2.7.15 atau bahkan belum ada sama sekali silahkan lakukan konfigurasi ini:

- Silahkan buka *Control Panel* anda.
- Pilih System and Security.
- Kemudian pilih lagi system.
- Lalu di bagian kiri tampilan ada sub menu Advanced system setting.
- Pada sub menu tersebut silahkan pilih button Environment Variabel.
- Silahkan ganti path dengan C:\Python27 dan C:\Python27\Scripts (Lokasi anda menyimpan mentahan python yang telah anda install tadi).
- Maka tampilannya akan seperti pada gambar 5.3.



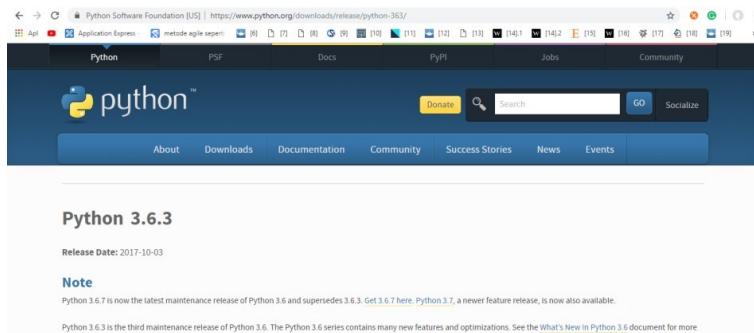
Gambar 5.3 Pengubahan Environment Python 2.7.

- Jangan lupa untuk memasukkan script dari pythonnya sehingga benar-benar bisa terbaca untuk pipnya. Silahkan klik button ok sampai selesai.
- Setelah itu, lakukan pengecekan ulang di Command Prompt maka hasilnya akan berubah menjadi versi 2.7.15.

5.3.2 Instalasi Python 3.6

Selanjutnya kita mulai dengan tutorial dalam menginstall Python 3.6. Python ini digunakan untuk code pengolahan data csv ke dalam flask python.

- Pertama-tama silahkan download software dari python versi 3 di laptop anda. Download python versi 3.6.3 dari situs web resminya yaitu <https://www.python.org/>. Silahkan sesuaikan dengan kapasitas laptop anda, bisa yang win 32 atau yang win 64 (32 bit / 64 bit). Contoh downloadnya seperti seperti pada gambar 5.4.



Gambar 5.4 Download Softfile Python 3.6.

- Setelah berhasil mendownload mentahan pythonnya silahkan lakukan instalasi seperti biasa anda lakukan. Setelah selesai instalasi pythonnya silahkan check di Command Prompt, apakah Pythonnya telah terbaca / running disesuaikan dengan laptop anda atau belum. Contoh pengecekan di Command Prompt seperti pada gambar 5.5

```
Command Prompt - python3
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

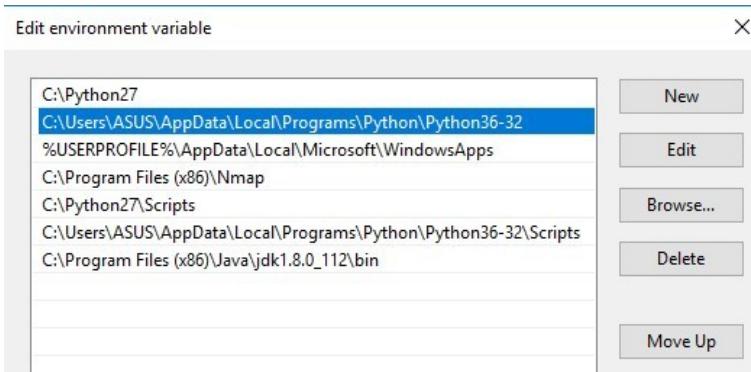
C:\Users\ASUS>python
ImportError: No module named site

C:\Users\ASUS>python3
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Gambar 5.5 Pengecekan Python 3.6.

- Apabila tampilannya telah sesuai dengan contoh diatas, maka python anda siap digunakan.
- Pastikan python yang terbaca versi 3.6.3 atau bahkan belum ada sama sekali silahkan lakukan konfigurasi ini:
 - Silahkan buka Control Panel anda.
 - Pilih System and Security.
 - Kemudian pilih lagi system.

- Lalu di bagian kiri tampilan ada sub menu Advanced system setting.
- Pada sub menu tersebut silahkan pilih button Environment Variabel.
- Silahkan ganti path dengan C :\Python36 dan C :\Python36\Scripts (Lokasi anda menyimpan mentahan python yang telah anda install tadi).
- Maka tampilannya akan seperti pada gambar 5.6.



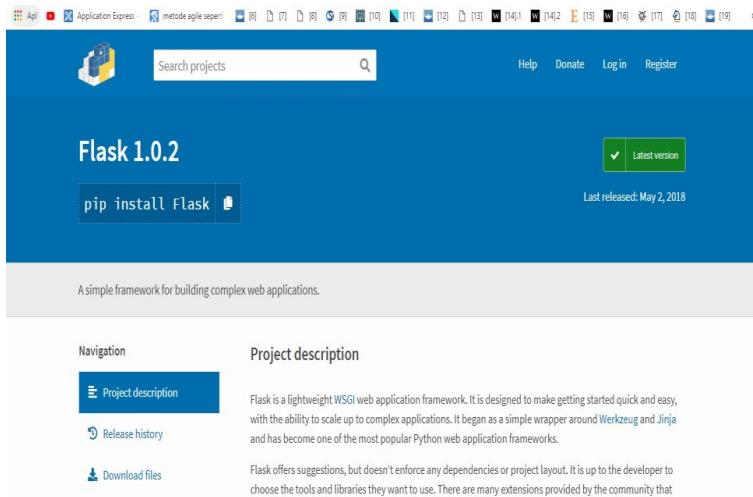
Gambar 5.6 Pengubahan Environment Python 3.6.

- Jangan lupa untuk memasukkan script dari pythonnya sehingga benar-benar bisa terbaca untuk pipnya. Silahkan klik button ok sampai selesai.
- Setelah itu, lakukan pengecekan ulang di Command Prompt maka hasilnya akan berubah menjadi versi 3.6.3
- Setelah semua tahap di atas selesai, maka silahkan lanjutkan ke tahap berikutnya.

5.3.3 Instalasi Framework Flask

Tutorial selanjutnya ialah kita akan menginstall Framework Flask di komputer/laptop kita sehingga bisa digunakan untuk tutorial selanjutnya. Perlu kita ketahui bahwa flask merupakan *microframework* dari python, dengan penggunaannya, aktivitas apapun yang kita lakukan baik pengolahan data dan lain sebagainya akan terasa lebih mudah dan rapih. Ayo kita mulai instalasinya!

1. Pertama, silahkan nyalakan PC/laptop Anda.
2. Kemudian apabila PC/laptop Anda sudah siap digunakan, silahkan buka web browser seperti Chrome, Mozilla Firefox, dsb.
3. Selanjutnya ketikkan pada web browser Anda alamat laman resmi <https://pypi.org/project/> untuk mengunduh Flask. Contohnya akan nampak seperti pada gambar 5.7.
4. Setelah proses mengunduh selesai, silahkan buka Command Prompt di PC/laptop Anda.

**Gambar 5.7** Download Flask

5. Kemudian silahkan ketikkan perintah `pip install flask`.
6. Setelah mengetikkan perintah tersebut, silahkan tekan enter maka prosesnya akan berjalan. Hasilnya akan nampak seperti pada gambar 5.8.

```
C:\Users\admin\AppData\Local\Programs\Python\Python36\Scripts>pip install flask
Collecting flask
  Using cached Flask-0.12.2-py2.py3-none-any.whl
Collecting itsdangerous>=0.21 (from flask)
  Using cached itsdangerous-0.24.tar.gz
Collecting Werkzeug>=0.7 (from flask)
  Using cached Werkzeug-0.12.2-py2.py3-none-any.whl
Collecting Jinja2>=2.4 (from flask)
  Using cached Jinja2-2.9.6-py2.py3-none-any.whl
Collecting click>=2.0 (from flask)
  Using cached click-6.7-py2.py3-none-any.whl
Collecting MarkupSafe>=0.23 (from Jinja2>=2.4->flask)
  Using cached MarkupSafe-1.0.tar.gz
Installing collected packages: itsdangerous, Werkzeug, MarkupSafe, Jinja2, click, flask
  Running setup.py install for itsdangerous ... done
  Running setup.py install for MarkupSafe ... done
Successfully installed Jinja2-2.9.6 MarkupSafe-1.0 Werkzeug-0.12.2 click-6.7 flask-0.12.2 itsdangerous-0.24
```

Gambar 5.8 Proses Instalasi Flask

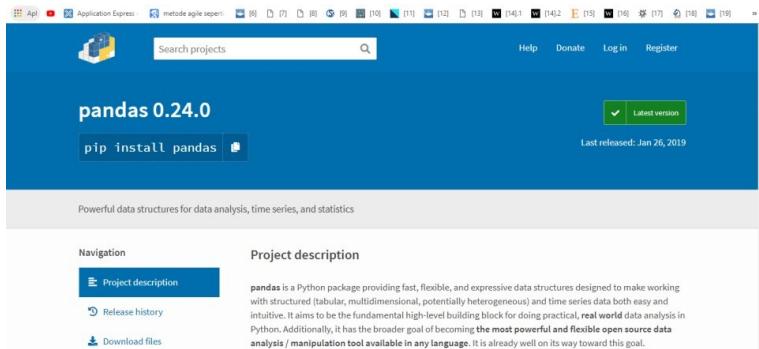
7. Proses instalasi sudah berhasil apabila tampilan pada PC/laptop Anda sudah sama seperti pada gambar 5.8. Namun jika masih terjadi *error*, coba lakukan kembali langkah-langkah instalasi flask, dan pastikan Anda telah melakukan semua proses sesuai dengan tutorial di buku ini. Setelah flask sudah terpasang pada PC/laptop Anda, silahkan untuk melanjutkan ke tutorial selanjutnya.

5.3.4 Instalasi Library Pandas

Perlu kita ketahui bahwa pandas merupakan *library* dari bahasa pemrograman python. Library ini digunakan untuk pemrosesan data analitik. Mengapa kita gunakan? Karena

memang pandas ini akan mengolah data analitik dari CSV file yang berisikan data sinyal gelombang otak yang bisa kita baca dan tangkap dari aktifitas tertentu (lampa sein saat bermotor). Ayo kita mulai instalasinya!

1. Pertama silahkan nyalakan PC/laptop Anda.
2. Kemudian apabila PC/laptop Anda sudah siap digunakan, silahkan buka web browser seperti Chrome, Mozilla Firefox, dsb.
3. Selanjutnya ketikkan pada web browser Anda alamat laman resmi <https://pypi.org/project/pandas/> mengunduh pandas. Contohnya nampak seperti pada gambar 5.9.



Gambar 5.9 Download Pandas

4. Setelah proses mengunduh selesai, silahkan buka Command Prompt di PC/laptop Anda.
5. Kemudian silahkan ketikkan perintah `pip install pandas`.
6. Setelah mengetikkan perintah tersebut, silahkan tekan enter maka prosesnya akan berjalan. Silahkan tunggu untuk melihat hasilnya yang akan nampak seperti pada gambar 5.10.
7. Proses instalasi sudah berhasil apabila tampilan pada PC/laptop Anda sudah sama seperti pada gambar ???. Namun jika masih terjadi *error*, coba lakukan kembali langkah-langkah instalasi pandas, dan pastikan Anda telah melakukan semua proses sesuai dengan tutorial di buku ini.
8. Setelah pandasnya terpasang di komputer/laptop anda maka silahkan lanjutkan ke tutorial selanjutnya.

5.3.5 Contoh Penerapan Fungsi Pada Flask Python

Flask adalah kerangka kerja aplikasi web mikro yang ditulis dalam bahasa pemrograman Python dan berdasarkan Werkzeug toolkit dan template engine Jinja2. Berlisensi BSD.

```
C:\Windows\system32>pip install pandas
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/7b/5c/51ad4b4f431b5735b239
  6da44d44bd0f62bad209e0be46311?10162b/pandas-0.24.0-cp36-cp36m-win_amd64.whl
    100% [██████████] 8.8MB 867kB/s
Collecting numpy>=1.12.0 <from pandas>
  Downloading https://files.pythonhosted.org/packages/31/7e/8905636f7e4f9b9d7078
  aa0e701500634f832f145855a11beb098d3b0fb1/numpy-1.16.0-cp36-cp36m-win_and64.whl
    100% [██████████] 11.9MB 506kB/s
Collecting pytz>=2011k <from pandas>
  Using cached https://files.pythonhosted.org/packages/61/28/1d3920e4d1d50b19bc5
  d24398a7cd85cc7b9a75a490579d530c57622d34/pytz-2018.9-py2.py3-none-any.whl
Collecting python-dateutil>=2.5.0 <from pandas>
  Using cached https://files.pythonhosted.org/packages/74/68/d87d9b36af36f44254a
  8d512chfc48369103a3b9e474be9hdfe536abfc45/python_dateutil-2.7.5-py2.py3-none-any
  .whl
Collecting six>=1.5 <from python-dateutil>=2.5.0->pandas>
  Using cached https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecfe
  898238ce23f502a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.whl
Installing collected packages: numpy, pytz, six, python-dateutil, pandas
Successfully installed numpy-1.16.0 pandas-0.24.0 python-dateutil-2.7.5 pytz-201
8.9 six-1.12.0
```

Gambar 5.10 Proses Instalasi Pandas

Langkah-langkah membangun Flask:

1. Pertama, pastikan bahwa python telah terinstal pada komputer.
2. Kemudian install framework python yaitu Flask pada CMD.
3. Buka CMD.
4. Kemudian ketikkan `pip install flask`. Contohnya seperti pada gambar 5.11.

```
D:\>pip install flask
Requirement already satisfied: flask in c:\python27\lib\site-packages (1.0.2)
Requirement already satisfied: Jinja2>=2.10 in c:\python27\lib\site-packages (from flask) (2.10)
Requirement already satisfied: itsdangerous>=0.24 in c:\python27\lib\site-packages (from flask) (1.1.0)
Requirement already satisfied: click>=3.0 in c:\python27\lib\site-packages (from flask) (7.0.0)
Requirement already satisfied: Werkzeug>=0.14 in c:\python27\lib\site-packages (from flask) (0.14.1)
Requirement already satisfied: MarkupSafe>=0.23 in c:\python27\lib\site-packages (from Jinja2>=2.10->flask) (1.1.0)
Requirement already satisfied: six>=1.5 in c:\python27\lib\site-packages (from Jinja2>=2.10->flask) (1.1.0)
```

Gambar 5.11 Instalasi Pip Flask

5. Gambar 5.11 menunjukkan bahwa laptop anda telah memiliki flask, jadi sudah bisa digunakan.
6. Kemudian masuklah pada *text editor* seperti sublime, *notepad++*, dan lain sebagainya.
7. Kemudian silahkan untuk mulai memasukkan perintah-perintah di Flask.
8. Sebelum mencobanya, alangkah baiknya jika kita mempelajari flask terlebih dahulu.
9. Selanjutnya kita masuk ke bagian perintah python flask. Pada bagian ini kita akan mencoba membuat *hello world*. Berikut ini *source code* untuk membuat hello world seperti pada listing 5.1.

```

1 def hello():
2     return "Hello World!"
```

Listing 5.1 Contoh kode program hello.py

Contoh *source code* main.py bisa dilihat seperti pada listing 5.2.

```

1 from flask import Flask
2 from hello import hello
3
4 app = Flask(__name__)
5
6 @app.route('/oke', methods=['GET'])
7 def annisa():
8     args = request.args
9     if request.method == 'GET':
10         if not len(args) is 0:
11             if 'word' in args:
12                 response = hello(response, args['word'])
13             return response
14
15 if __name__ == "__main__":
16     app.run(debug=True)
```

Listing 5.2 Contoh kode program main.py

10. Masukkan perintah `import hello` untuk memanggil dan menghubungkan fungsi ke dalam file flask ini.
11. Pada file flask juga didefinisikan pemanggilan untuk fungsi hello world nya.
12. Pemanggilannya berupa, apabila parameter word difungsikan maka akan langsung terhubung dengan fungsi hello world dan tulisan hello worldnya akan tampil sesuai dengan request GetSilahkan jalankan file pada CMD, maka hasilnya akan nampak seperti gambar 5.12.

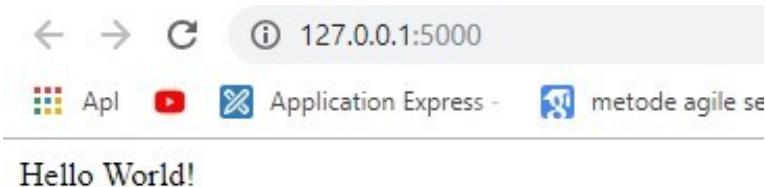
```
D:\PERKULIAHAN\TUTORIAL PROYEK 2>flask run
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 5.12 Pemanggilan Fungsi Flask Opsi 1

13. Atau bisa juga seperti pada gambar 5.13.
14. Copy URL yang ada pada CMD, lalu cek URL di Web Browser. Maka hasilnya akan nampak seperti pada gambar 5.14.

```
D:\PERKULIAHAN\TUTORIAL PROYEK 2>python check.py
 * Serving Flask app "check" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 179-326-677
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 5.13 Pemanggilan Fungsi Flask Opsi 2



Gambar 5.14 Output Hello World

5.4 Penanganan Error

5.4.1 Penanganan Error pada Python

5.4.1.1 Contoh kasus 1 Untuk tutorial ini, akan dicontohkan penanganan error untuk contoh Python. Ini contoh yang pertama. Sebelum menangani error. Akan dijelaskan terlebih dahulu tutorial untuk pembangunan Pythonnya kemudian akan masuk kepada penanganan error yang ada. Silahkan simak dan ikuti tutorial berikut:

1. Pertama, pastikan bahwa python telah terinstal di PC/laptop Anda. Anda dapat menggunakan python versi 2 maupun python versi 3. Untuk tutorial pertama ini, kita akan menggunakan Python 3.
2. Selanjutnya silahkan buka text editor. Untuk text editornya bisa apa saja. Namun, untuk tutorial ini dicontohkan dengan menggunakan Sublime 3.
3. Selanjutnya silahkan buat script Pythonnya, dengan nama file satu.py. Scriptnya dicontohkan seperti pada listing 5.3.

```
1 def hello(nama):
2     sambutan = "Assalamualaikum"
3     penggabungan = sambutan + " " + nama
```

Listing 5.3 File satu.py

4. Apabila telah membuat script seperti pada listing 5.3, maka silahkan buka Command Promp (CMD).

5. Kemudian pada CMD silahkan masuk ke tempat file anda tersimpan
6. Setelah masuk, silahkan check terlebih dahulu dengan perintah “Python2 1.py”
7. Apabila jalan maka berhasil. Silahkan keluar dari tempat file tersebut. Pada CMD silahkan ketikkan Python3
8. Kemudian silahkan eksekusi fungsi yang ada pada file tersebut untuk melihat hasilnya. Untuk pengeksekusianya seperti pada gambar 5.15.

```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import satu
>>> satu.hello("Anisa")
>>>
```

Gambar 5.15 Eksekusi Fungsi Percobaan Pertama

9. Ketika dieksekusi seperti ini. Tapi kenapa tidak ada hasilnya? Padahal untuk fungsi dan parameternya telah disetting. Coba perhatikan lagi pada script Pythonnya. Ternyata ada yang kurang, selain inputan seharusnya ada keluaran
10. Nah untuk perintah keluarannya belum ada. Untuk perintahnya bisa Print. Silahkan ganti script satu.py seperti listing 5.4.

```
1 def hello(nama):
2     sambutan = "Assalamualaikum"
3     penggabungan = sambutan + " " + nama
4     print penggabungan
```

Listing 5.4 File baru satu.py

11. Setelah mengganti script diatas, maka silahkan buka CMD. Masuk ke tempat penyimpanan file.
12. Kemudian ketikan perintah Python3 (untuk masuk ke Pythonnya)
13. Silahkan eksekusi ulang dengan cara yang sama
14. Hasilnya seperti pada gambar 5.16.

```
D:\Buat Praktek>python3
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import satu
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "D:\Buat Praktek\satu.py", line 5
      print penggabungan
          ^

```

Gambar 5.16 Eksekusi Fungsi Percobaan Kedua

15. Ternyata malah menghasilkan error yang berbeda. Nah untuk error berikut dinyatakan bahwa ada yang salah dengan perintah keluaran diatas.
16. Setelah ditelusuri ternyata salahnya pada penulisan perintah. Mengapa? Dikarenakan kita menggunakan script Python yang sangat sensitif jadi kita harus menyamaikan perintah dengan versi Python yang digunakan. Ternyata untuk pemanggilannya tadi kita menggunakan jenis tulisan perintah untuk Python 2. Seharusnya untuk Python3 kita harus menambahkan beberapa tanda yaitu () tanda kurung pada parameternya sehingga bisa dijalankan
17. Silahkan ganti code seperti pada listing 5.5 berikut :

```

1 def hello(nama):
2     sambutan = "Assalamualaikum"
3     peng gabungan = sambutan + " " + nama
4     print (peng gabungan)

```

Listing 5.5 File baru satu.py

18. Setelah mengganti code seperti contoh diatas maka silahkan buka kembali CMD. Command Prompt lalu masuk ke file tempat penyimpanan file satu.py
19. Silahkan lakukan kembali perintah pemanggilannya seperti pada gambar 5.17 berikut :

```

Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import satu
>>> satu.hello("Annisa")
Assalamualaikum Annisa

```

Gambar 5.17 Eksekusi Fungsi Percobaan Ketiga

20. Hasil yang benar akan nampak seperti gambar tersebut. Dimana ketika telah berhasil melakukan penyettingan pada parameter yang dieksekusi maka fungsi dari file tersebut akan mengeluarkan keluaran yang sesuai
21. Keluarannya tentunya akan berupa string. Semuanya sesuai dengan settingan yang telah dibuat pada file satu.py

5.4.1.2 Contoh kasus 2 Untuk tutorial ini, akan dicontohkan penanganan error untuk contoh Python. Ini contoh yang kedua. Sebelum menangani error. Akan dijelaskan terlebih dahulu tutorial untuk pembangunan Pythonnya kemudian akan masuk kepada penanganan error yang ada. Silahkan simak dan ikuti tutorial berikut: (Contoh kedua ini akan muncul error yang sama namun dengan penyelesaian yang berbeda dari contoh sebelumnya)

1. Pertama-tama silahkan pastikan bahwa di laptop anda telah terinstall Python. Python yang digunakan ada Python 2 dan Python 3. Untuk tutorial pertama ini, kita akan menggunakan Python 3.

2. Selanjutnya silahkan buka text editor. Untuk text editornya bisa apa saja. Namun, untuk tutorial ini dicontohkan dengan menggunakan Sublime 3.
3. Selanjutnya silahkan buat script Pythonnya, dengan nama file satu.py. Scriptnya dicontohkan seperti pada listing 5.6.

```
1 def hello(nama):  
2     sambutan = "Assalamualaikum"  
3     penggabungan = sambutan + " " + nama
```

Listing 5.6 File baru satu.py

4. Apabila telah membuat script seperti diatas maka silahkan buka CMD. CMD adalah Command Prompt, software bawaan laptop anda. Kemudian pada CMD silahkan masuk ke tempat file anda tersimpan. Setelah masuk, silahkan check terlebih dahulu dengan perintah “ Python2 1.py ”
5. Apabila jalan maka berhasil. Silahkan keluar dari tempat file tersebut. Pada CMD silahkan ketikkan Python3
6. Kemudian silahkan eksekusi fungsi yang ada pada file tersebut untuk melihat hasilnya. Untuk pengeksekusianya seperti pada gambar 5.18.

```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import satu  
>>> satu.hello("Anisa")  
>>>
```

Gambar 5.18 Eksekusi Fungsi Percobaan Keempat

7. Ketika dieksekusi seperti ini. Tapi kenapa tidak ada hasilnya? Padahal untuk fungsi dan parameternya telah disetting
8. Coba perhatikan lagi pada script Pythonnya. Ternyata ada yang kurang, selain inputan seharusnya ada keluaran. Nah untuk perintah keluarannya belum ada. Untuk perintahnya bisa Return. Contoh yang pertama menggunakan print dan ternyata seharusnya kita menggunakan perintah Return karena lebih sering dan memang seharusnya dipakai untuk pengembalian keluaran pada fungsi Python
9. Silahkan ganti script satu.py seperti pada listing 5.7.

```
1 def hello(nama):  
2     sambutan = "Assalamualaikum"  
3     penggabungan = sambutan + " " + nama  
4     return penggabungan
```

Listing 5.7 File baru satu.py

10. Setelah mengganti script diatas, maka silahkan buka CMD. Masuk ke tempat penyimpanan file. Kemudian ketikkan perintah Python3 (untuk masuk ke Pythonnya). Silahkan eksekusi ulang dengan cara yang sama

```
D:\Buat Praktek>python3
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import satu
>>> satu.hello("Annisa")
'Assalamualaikum Annisa'
```

Gambar 5.19 Eksekusi Fungsi Percobaan Kelima

11. Hasilnya seperti pada gambar 5.19.
12. Hasil yang benar akan nampak seperti gambar 5.19.
13. Dimana ketika telah berhasil melakukan penyettingan pada parameter yang dieksekusi maka fungsi dari file tersebut akan mengeluarkan keluaran yang sesuai. Keluarannya tentunya akan berupa string
14. Semuanya sesuai dengan settingan yang telah dibuat pada file satu.py

5.4.1.3 Contoh kasus 3 Untuk tutorial ini, akan dicontohkan penanganan error untuk contoh Python. Ini contoh yang ketiga. Sebelum menangani error. Akan dijelaskan terlebih dahulu tutorial untuk pembangunan Pythonnya kemudian akan masuk kepada penanganan error yang ada. Silahkan simak dan ikuti tutorial berikut:

1. Pertama-tama silahkan pastikan bahwa di laptop anda telah terinstall Python. Python yang digunakan ada Python 2 dan Python 3. Untuk tutorial pertama ini, kita akan menggunakan Python 3.
2. Selanjutnya silahkan buka text editor. Untuk text editornya bisa apa saja. Namun, untuk tutorial ini dicontohkan dengan menggunakan Sublime 3
3. Selanjutnya silahkan buat script Pythonnya, dengan nama file dua.py
4. Scriptnya dicontohkan seperti pada listing5.8

```
1 def hello(nama):
2     sambutan = "Assalamualaikum"
3     penggabungan = sambutan + " " + nama
4     return (penggabungan)
```

Listing 5.8 File baru dua.py

5. Apabila telah membuat script seperti diatas maka silahkan buka CMD. CMD adalah Command Prompt, software bawaan laptop anda. Kemudian pada CMD silahkan masuk ke tempat file anda tersimpan. Setelah masuk, silahkan check terlebih dahulu dengan perintah “ Python2 dua.py “
6. Apabila jalan maka berhasil, silahkan keluar dari tempat file tersebut. Pada CMD silahkan ketikkan Python3, kemudian silahkan eksekusi fungsi yang ada pada file tersebut untuk melihat hasilnya.
7. Untuk pengeksekusiannya seperti pada gambar 5.20

```
D:\Buat Praktek>python3
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import dua
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "D:\Buat Praktek\dua.py", line 2
    sambutan = "Assalamualaikum"
           ^
IndentationError: expected an indented block
>>>
```

Gambar 5.20 Eksekusi Fungsi Percobaan Keenam

8. Ketika dieksekusi, hasilnya nampak seperti gambar 5.20.
9. Tapi kenapa tidak ada hasilnya? Padahal untuk fungsi dan parameternya telah disetting. Coba perhatikan lagi pada script Pythonnya. Ternyata ada yang kurang, selain inputan seharusnya ada keluaran. Nah untuk errornya sendiri ternyata ada pada blokingannya. Dimana Python sangatlah sensitif maka scriptnya juga harus benar. Kita harus menyamakan perintah dengan benar. Penanganganannya yaitu memberikan SPASI ataupun TAB pada perintah untuk memasukkan inputan dengan benar. Harus sesuai dengan perintah yang lainnya agar dapat dieksekusi dengan benar
10. Silahkan ganti script dua.py seperti pada listing 5.9.

```
1 def hello(nama):
2     sambutan = "Assalamualaikum"
3     penggabungan = sambutan + " " + nama
4     return penggabungan
```

Listing 5.9 File baru dua.py

11. Setelah mengganti script diatas, maka silahkan buka CMD. Masuk ke tempat penyimpanan file. Kemudian ketikan perintah Python3 (untuk masuk ke Pythonnya). Silahkan eksekusi ulang dengan cara yang sama
12. Hasilnya seperti pada gambar 5.21.

```
D:\Buat Praktek>python3
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import dua
>>> dua.hello("Annisa")
'Assalamualaikum Annisa'
>>>
```

Gambar 5.21 Eksekusi Fungsi Percobaan Ketujuh

13. Hasil yang benar akan nampak seperti gambar 5.21.
14. Dimana ketika telah berhasil melakukan penyettingan pada parameter yang dieksekusi maka fungsi dari file tersebut akan mengeluarkan keluaran yang sesuai. Keluarannya tentunya akan berupa string. Semuanya sesuai dengan settingan yang telah dibuat pada file dua.py

5.4.1.4 Contoh kasus 4 Untuk tutorial ini, akan dicontohkan penanganan error untuk contoh Python. Ini contoh yang empat. Sebelum menangani error. Akan dijelaskan terlebih dahulu tutorial untuk pembangunan Pythonnya kemudian akan masuk kepada penanganan error yang ada. Silahkan simak dan ikuti tutorial berikut:

1. Pertama-tama silahkan pastikan bahwa di laptop anda telah terinstall Python. Python yang digunakan ada Python 2 dan Python 3. Untuk tutorial pertama ini, kita akan menggunakan Python 3.
2. Selanjutnya silahkan buka text editor. Untuk text editornya bisa apa saja. Namun, untuk tutorial ini dicontohkan dengan menggunakan Sublime 3. Selanjutnya silahkan buat script Pythonnya, dengan nama file dua.py
3. Scriptnya dicontohkan seperti pada listing 5.10

```

1 def hello(nama):
2     sambutan = "Assalamualaikum"
3     penggabungan = sambutan + " " + nama
4     return penggabungan

```

Listing 5.10 File baru dua.py

4. Setelah membuat script diatas, maka silahkan buka CMD. Masuk ke tempat penyimpanan file. Kemudian ketikan perintah Python3 (untuk masuk ke Pythonnya)
5. Silahkan eksekusi mengikuti perintah berikut seperti pada gambar 5.22

```

D:\Buat_Praktek>python3
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import dua
>>> satu.hello("Annisa")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'satu' is not defined
>>> satu.hello("Annisa")^Z
  File "<stdin>", line 1
    satu.hello("Annisa")^A

```

Gambar 5.22 Eksekusi Fungsi Percobaan Kedelapan

6. Hasil yang benar akan nampak seperti gambar tersebut.
7. Terjadi error pada eksekusi tersebut. Ternyata untuk error ini terjadi karena salah perintah dalam pengeksekusian. Pada perintah tersebut, kita memasukkan “ import dua “. Maka akan masuk ke file dua.py
8. Namun, setelah itu, kita malah memasukkan perintah “ satu.hello(“annisa”). Yang kalau diartikan, maka perintah tersebut ditujukan untuk file satu.py. Maka dari itu, kita harus mengganti perintah yang kita masukkan dalam pengeksekusian. Kalau mengeksekusi file dua.py maka import dan arahan perintahnya harus ke file dua.py

```
D:\Buat Praktek>python3
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import dua
>>> dua.hello("Annisa")
'Assalamualaikum Annisa'
>>>
```

Gambar 5.23 Eksekusi Fungsi Percobaan Kesembilan

9. Silahkan eksekusi lagi dengan perintah berikut seperti pada gambar 5.23.
10. Apabila perintah telah sesuai, maka akan menampilkan hasil yang seperti contoh diatas. Keluarannya tentunya akan berupa string. Semuanya sesuai dengan settingan yang telah dibuat pada file dua.py. Dengan permasalahan yang sama, apabila kita secara tidak sengaja menemukan error juga ketika mengeksekusi file satu.py maka penyelesaiannya mirip seperti contoh diatas
11. Dimisalkan untuk file satu.py errornya seperti pada gambar 5.24.

```
D:\Buat Praktek>python3
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import satu
>>> dua.hello("Annisa")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'dua' is not defined
>>>
```

Gambar 5.24 Eksekusi Fungsi Percobaan Kesepuluhur

12. Nah, tentunya kita sudah tau bahwa salahnya pada pemanggilan yang harusnya mengarah kepada file dua.py. Silahkan dibenarkan lagi.
13. Ikuti contoh eksekusi seperti pada gambar 5.25 untuk mendapatkan hasil yang benar.

```
D:\Buat Praktek>python3
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import satu
>>> satu.hello("Annisa")
'Assalamualaikum Annisa'
```

Gambar 5.25 Eksekusi Fungsi Percobaan Kesebelas

14. Nah, penyelesaiannya seperti diatas. Kita harus memperhatikan perintah yang kita gunakan untuk eksekusi.

5.4.1.5 Contoh kasus 5 Untuk tutorial ini, akan dicontohkan penanganan error untuk contoh Python. Ini contoh yang kelima. Sebelum menangani error. Akan dijelaskan terlebih dahulu tutorial untuk pembangunan Pythonnya kemudian akan masuk

kepada penanganan error yang ada. Silahkan simak dan ikuti tutorial berikut: (Contoh Python ini digunakan dalam pembuatan script python untuk pembacaan gelombang otak)

- Pertama-tama silahkan pastikan bahwa di laptop anda telah terinstall Python. Python yang digunakan ada Python 2 dan Python 3. Untuk tutorial pertama ini, kita akan menggunakan Python 2.
- Selanjutnya silahkan buka text editor. Untuk text editornya bisa apa saja. Namun, untuk tutorial ini dicontohkan dengan menggunakan Sublime 3.
- Selanjutnya silahkan buat script Pythonnya, dengan nama file coba.py
- Scriptnya dicontohkan seperti pada listing 5.13.

```

1 import mindwave, time
2 import keyboard
3 import csv
4
5 headset = mindwave.Headset('COM4', '1425')
6 time.sleep(2)
7 a=0
8 headset.connect()
9 print "Connecting..."
10
11 while headset.status != 'connected':
12     time.sleep(0.5)
13     if headset.status == 'standby':
14         headset.connect()
15         print "Retrying connect..."
16 print "Connected."
17
18 while True:
19     print "raw_value: %s : %s" % (headset.raw_value, a)
20     writer.writerow({'RawValue': headset.raw_value, 'sign': a})

```

Listing 5.11 File coba.py

- Pada code diatas mendefinisikan perintah dimana fungsinya akan dipanggil dan berhubungan dengan mindwave.py. Seperti contoh perintah headset.connect dimana fungsinya akan diproses di mindwave.py sehingga apabila hasilnya benar dan sesuai maka akan muncul kata “connecting” seperti perintah print diatas. Apabila alat dan code sesuai maka fungsinya akan jalan, baik itu connected, standby dll.
- Nah untuk pembacaan sinyalnya, kita menggunakan variabel dari mindwave.py. Untuk variabelnya bisa dicoba yang attention / blink / meditation dll secara bersamaan juga bisa asal perintahnya benar.
- Perhatikan code ini:

```

1 headset = mindwave.Headset('COM4', '1425')
2 time.sleep(2)

```

Listing 5.12 File coba.py

8. Untuk mengkoneksikan code dengan alat sehingga terhubung maka serial dari alat tersebut harus benar. Portnya yaitu COM4 dapat dilihat dengan cara :
 - Buka pengaturan laptop anda
 - Kemudian pilih device manager
 - Lalu pilih alat yang anda gunakan misalnya : Mindwave Neurosky
 - Apabila telah diklik maka akan muncul portnya yaitu COM4
 - Untuk serialnya dapat dilihat dari alatnya sendiri
 - Pengecekannya yaitu di dalam tempat baterai. Serialnya 1425
 - Port dan serial itu beda-beda tergantung dari alatnya masing-masing
9. Time.sleep itu gunanya untuk memberikan jeda pembacaan sinyal sehingga nilai gelombang yang dibaca tidak terlalu cepat dan dapat lebih terpantau. Lebih jelasnya lagi, mengapa time.sleepnya bernilai (0.001953125) dikarenakan :
10. Silahkan perhatikan ini:

```

1 print "raw_value: %s : %s" % (headset.raw_value , a)
2 writer.writerow({ 'RawValue': headset.raw_value , 'sign': a })

```

Listing 5.13 File coba.py

Measures

Outputs 12 bit Raw-Brainwaves (3 - 100Hz) with Sampling rate at 512Hz

Outputs EEG power spectrums (Alpha, Beta, etc.)

Outputs NeuroSky proprietary eSense meter such as Attention,
Meditation, and other future meters

EEG/ECG signal quality analysis (can be used to detect poor contact and
whether the device is off the head)

Gambar 5.26 Deskripsi Pengukuran

11. Sampling rate dari alat yang digunakan ialah 512Hz jadi tentunya kita harus mengikuti dan menyesuaikannya dengan baik, untuk code dan alatnya. Nah kemudian 512Hz tersebut kita bagi dengan 1 (detik) jadi hasilnya akan menjadi jeda antara sinyal yang telah ditangkap/ dibaca dari alat. $1/512 = 0.001953125$. nah nilai itu yang akan menjadi jeda pada code. Selain itu. Kita juga harus memastikan apakah benar bahwa dalam 1 detik itu ada 512 data yang dibaca dan ditangkap oleh alat Dan sesuai hasil dari pengujinya sudah dibuktikan dalam setiap detiknya ada sekitar 480-520an data perdetik.
12. Setelah semua langkah di atas telah dilakukan, kita akan mencoba running coba.py melalui cmd, dengan cara :

13. Buka Command Prompt di komputer/laptop anda masing-masing. Silahkan pada CMD arahkan ke folder / directory tempat file coba.py anda simpan
14. Namun sebelum itu pastikan alat sudah terpasang baterai dan lampu indikator berwarna merah. Masukkan USB ke port yang ada di laptop anda. USBnya tentu milik dari alat mindwavenya
15. Kemudian silahkan masukkan perintah python coba.py
16. Apabila pas anda jika terjadi ERROR seperti gambar 5.27.

```
C:\Users\dikasukmap\Desktop>python coba.py
  File "coba.py", line 17
    print "raw_value: %s" % (headset.raw_value)
                                         ^
IndentationError: expected an indented block
C:\Users\dikasukmap\Desktop>
```

Gambar 5.27 Error pada program coba.py

17. ERROR diatas sebenarnya menunjukan bahwa indentasi codingan anda tak sesuai maka yang perlu dilakukan ialah rapikan code anda sesuai dengan perintah-perintah terkait. Biasanya ERROR ini terjadi ketika code anda tak berada pada tempatnya maka silahkan sesuaikan dengan SPACI ataupun TAB.
18. Silahkan Run kembali
19. Ketika di Run, dan terjadi ERROR ditengah jalan seperti pada gambar 5.28

```
raw_value: 67
raw_value: 104
raw_value: 8
Exception in thread Thread-1:
Traceback (most recent call last):
  File "C:\Python27\lib\threading.py", line 801, in __bootstrap_inner
    self.run()
  File "C:\Users\dikasukmap\Desktop\mindwave.py", line 73, in run
    self.parse_payload(payload)
  File "C:\Users\dikasukmap\Desktop\mindwave.py", line 143, in parse_payload
    raw=ord(value[0])>256+ord(value[1])
IndexError: string index out of range
```

Gambar 5.28 Keterangan error pada program coba.py

20. Maka anda harus menerapkan Try and Except pada codingan mindwavenya.
21. Contohnya seperti pada listing 5.14

```
1 if code == RAW_VALUE:
2     try:
3         anu = value[0]
4         itu = value[1]
5     except IndexError:
6         anu = "0"
```

```
7     itu = "0"
```

Listing 5.14 File try_except.py

22. Penggunaan try dan except yaitu untuk menangani error pada code yang dihindari untuk terjadi ketika sedang membaca sinyal. Dimisalkan penanganan errornya untuk penanganan index. Selain index sebenarnya bisa juga untuk menangani error pada IO, database, dictionary dan lain-lain.
23. Setelah penerapan tersebut silahkan re-run file coba.py. Maka tidak akan terjadi error lagi.

5.4.1.6 Contoh kasus 6 Untuk tutorial ini, akan dicontohkan penanganan error untuk contoh Python. Ini contoh yang keenam. Sebelum menangani error. Akan dijelaskan terlebih dahulu tutorial untuk pembangunan Pythonnya kemudian akan masuk kepada penanganan error yang ada. Silahkan simak dan ikuti tutorial berikut: (Contoh Python ini digunakan dalam pembuatan script python untuk pembacaan gelombang otak)

1. Pertama-tama silahkan pastikan bahwa di laptop anda telah terinstall Python. Python yang digunakan ada Python 2 dan Python 3. Untuk tutorial pertama ini, kita akan menggunakan Python 2.
2. Selanjutnya silahkan buka text editor. Untuk text editornya bisa apa saja. Namun, untuk tutorial ini dicontohkan dengan menggunakan Sublime 3.
3. Selanjutnya silahkan buat script Pythonnya, dengan nama file coba.py
4. Scriptnya dicontohkan seperti pada listing 5.15

```
1 import mindwave, time
2 import keyboard
3 import csv
4
5 headset = mindwave.Headset('COM4', '1425')
6 time.sleep(2)
7 a=0
8
9 headset.connect()
10 print "Connecting..."
11
12 while headset.status != 'connected':
13     time.sleep(0.5)
14     if headset.status == 'standby':
15         headset.connect()
16         print "Retrying connect..."
17 print "Connected."
18
19 while True:
20     print "raw_value: %s : %s" % (headset.raw_value, a)
```

Listing 5.15 File coba_lagi.py

5. Pada code diatas mendefinisikan perintah dimana fungsinya akan dipanggil dan berhubungan dengan mindwave.py
6. Seperti contoh perintah headset.connect dimana fungsinya akan diproses di mindwave.py sehingga apabila hasilnya benar dan sesuai maka akan muncul kata “connecting” seperti perintah print diatas. Apabila alat dan code sesuai maka fungsinya akan jalan, baik itu connected, standby dll.
7. Nah untuk pembacaan sinyalnya, kita menggunakan variabel dari mindwave.py. Sebenarnya penjelasan lengkap untuk tutorial pembangunan script python ini sama dengan contoh kasus kelima
8. Namun, akan muncul error yang berbeda.
9. Setelah semua langkah di atas telah dilakukan, kita akan mencoba running coba.py melalui cmd, dengan cara :
10. Buka Command Prompt di komputer/laptop anda masing-masing. Silahkan pada CMD arahkan ke folder / directory tempat file coba.py anda simpan.
11. Namun sebelum itu pastikan alat sudah terpasang baterai dan lampu indikator berwarna merah. Masukkan USB ke port yang ada di laptop anda. USBnya tentu milik dari alat mindwavenya
12. Kemudian silahkan masukkan perintah python coba.py. Maka akan muncul error dimana hasilnya tidak akan muncul
13. Hal tersebut dikarenakan ada perintah yang kurang dalam script. Seharusnya kita juga memasukkan perintah seperti pada listing 5.16 berikut:

```
writer.writerow({ 'RawValue' : headset.raw_value , 'sign' : a })
```

Listing 5.16 File coba_lagi.py

14. Silahkan Run kembali, maka file akan mengeluarkan sinyal datanya.

5.4.2 Penanganan Error dalam Flask

5.4.2.1 Contoh kasus 1 Untuk tutorial ini, akan dicontohkan penanganan error untuk contoh Python Flask. Ini contoh yang pertama. Sebelum menangani error. Akan dijelaskan terlebih dahulu tutorial untuk pembangunan Pythonnya kemudian akan masuk kepada penanganan error yang ada. Silahkan simak dan ikuti tutorial berikut:

1. Pertama-tama silahkan pastikan bahwa di laptop anda telah terinstall Python. Python yang digunakan ada Python 2 dan Python 3. Untuk tutorial pertama ini, kita akan menggunakan Python 3.
2. Selanjutnya silahkan buka text editor.Untuk text editornya bisa apa saja. Namun, untuk tutorial ini dicontohkan dengan menggunakan Sublime 3.

3. Selanjutnya silahkan buat script Python Flasknnya. Script lengkap Python Flasknnya dapat ditemukan di Tutorial Proyek yang telah dibuat sebelumnya
4. Contoh Potongan scriptnya seperti pada listing 5.17.

```
1 from flask import Flask, request, jsonify
2
3 # import fungsi
4 from load_data import load_data
5 from get_all_data import get_all_data
6 from get_column import get_column
7 from get_top_five import get_top_five
8 from sorting import sorting
9 from convert_json import convert_json
10 from delete_column import delete_column
11 from delete_all_data import delete_all_data
12 from insert_data import insert_data
13 from get_info import get_info
14 from get_row import get_row
15 from get_row_field import get_row_field
16 from update_data import update_data
17 from delete_data import delete_data
18 from get_row_json import get_row_json
19 from get_all_data_json import
```

Listing 5.17 File import.py

5. Apabila telah membuat script seperti diatas maka silahkan buka CMD. CMD adalah Command Prompt, software bawaan laptop anda. Kemudian pada CMD silahkan masuk ke folder tempat penyimpanan file. Kemudian aktifkan dulu environtment dari file tersebut sehingga Flasknnya dapat berjalan.
6. Untuk pengeksekusiannya seharusnya nampak seperti pada gambar 5.29.

```
(env) D:\Buat Praktek\done>python3 main.py
* Serving Flask app "main" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 201-749-500
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 5.29 Eksekusi Program import.py

7. Namun, ketika dieksekusi malah hasilnya nampak seperti gambar berikut:
8. Error tersebut menyatakan bahwa kita tidak mendefinisikan module pandas pada file codingan yang telah dibuat
9. Maka kita harus memasukkan perintah seperti pada listing 5.18.

```
| import pandas as pd
```

Listing 5.18 File import.py

10. Perintah tersebut ditaruh setelah perintah pengimportan module yang lain seperti jsonify, request maupun flask. Setelah itu silahkan buka CMD kembali. Pada CMD silahkan lakukan perintah yang sama yaitu menjalankan file Main.py dengan mengaktifkan environmentnya terlebih dahulu.
11. Contohnya nampak seperti pada gambar 5.30.

```
Traceback (most recent call last):
  File "main.py", line 6, in <module>
    from load_data import load_data
  File "D:\Buat Praktek\done\load_data.py", line 1, in <module>
    import pandas
ModuleNotFoundError: No module named 'pandas'
```

Gambar 5.30 Eksekusi Program import.py

12. Apabila perintah yang anda lakukan nampak seperti pada contoh diatas, berarti flasknya telah berhasil berjalan. Silahkan untuk pengujinya bisa dilihat dengan masuk ke dalam URL yang diperlihatkan pada CMD. Pengujian bisa dilakukan dengan memasukkan URL ke dalam web browser biasa maupun Postman
13. Contoh pengujian ini akan menggunakan Postman biar lebih mudah dan rapih.
14. Hasilnya akan nampak seperti gambar 5.31, dimana perintahnya telah disesuaikan dengan request yang ada dalam file Main.py

```
D:\Buat Praktek\done>env\scripts\activate
(env) D:\Buat Praktek\done>python3 main.py
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 201-749-500
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 5.31 Eksekusi Program import.py

15. Apabila mendapatkan hasil seperti gambar 5.32, maka tutorial ini berhasil.

5.4.2.2 Contoh kasus 2 Untuk tutorial ini, akan dicontohkan penanganan error untuk contoh Python Flask. Ini contoh yang kedua. Sebelum menangani error. Akan dijelaskan terlebih dahulu tutorial untuk pembangunan Pythonnya kemudian akan masuk kepada penanganan error yang ada. Silahkan simak dan ikuti tutorial berikut:

The screenshot shows a Postman interface with a GET request to `http://127.0.0.1:5000/dataset?field=raw_value`. The response status is 200 OK. The response body is a JSON array containing 16 elements, each with the value "raw_value".

```

[{"id": 1, "value": "raw_value"}, {"id": 2, "value": "raw_value"}, {"id": 3, "value": "raw_value"}, {"id": 4, "value": "Fedia"}, {"id": 5, "value": "raw_value"}, {"id": 6, "value": "raw_value"}, {"id": 7, "value": "raw_value"}, {"id": 8, "value": "raw_value"}, {"id": 9, "value": "raw_value"}, {"id": 10, "value": "raw_value"}, {"id": 11, "value": "raw_value"}, {"id": 12, "value": "raw_value"}, {"id": 13, "value": "raw_value"}, {"id": 14, "value": "raw_value"}, {"id": 15, "value": "raw_value"}, {"id": 16, "value": "raw_value"}]
  
```

Gambar 5.32 Hasil Eksekusi Program import.py

1. Pertama-tama silahkan pastikan bahwa di laptop anda telah terinstall Python. Python yang digunakan ada Python 2 dan Python 3. Untuk tutorial pertama ini, kita akan menggunakan Python 3.
2. Kita akan mengeksekusi main.py, sehingga kita bisa menemukan beberapa error. Karena kita akan mengeksekusi main.py, kita juga akan membangun terlebih dahulu fungsi yang akan dimasukkan dalam main.py (flask).
3. Silahkan buka text editor. Gunakan sublime 3.
4. Kemudian silahkan buat script seperti pada listing 5.21.

```

1 def get_top_five(dataframe, order_by):
2     return dataframe.head()
  
```

Listing 5.19 File get_top.five.py

5. Fungsi tersebut disimpan dengan nama file get_top_five.py. Fungsi tersebut telah dimasukkan kedalam Flask yaitu main.py sehingga ketika dijalankan fungsinya akan dapat dieksekusi di file flask yang sama dengan fungsi yang lain. Fungsinya yaitu untuk menampilkan 5 data teratas ataupun terbawah pada file CSV.
6. Adapun code flask yang mengeksekusi fungsi ini ialah nampak seperti pada listing 5.20.

```

1
2 @app.route('/dataset', methods=['GET', 'POST', 'DELETE'])
3 def dataset():
4     args = request.args
5     response = datafram
6     if request.method == 'GET':
7         if 'topfive' in args:
  
```

Listing 5.20 File main.py

7. Silahkan masuk ke CMD. Jalankan file main.py dengan masuk ke folder penyimpanannya. Kemudian aktifkan dulu environment dari file tersebut sehingga Flasknya dapat berjalan.
8. Untuk pengeksekusiannya seharusnya nampak seperti pada gambar 5.33.

```
(env) D:\Buat Praktek\done>python3 main.py
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 201-749-500
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 5.33 Eksekusi Program main.py

9. Silahkan untuk pengujinya bisa dilihat dengan masuk ke dalam URL yang diperlihatkan pada CMD. Pengujian bisa dilakukan dengan memasukkan URL ke dalam web browser biasa maupun Postman. Contoh pengujian ini akan menggunakan Postman biar lebih mudah dan rapih.
10. Hasilnya akan nampak seperti pada gambar 5.34, dimana perintahnya telah disesuaikan dengan request yang ada dalam file main.py.
11. Pengujian diatas menunjukkan hasil untuk 5 angka teratas, dengan perintah /dataset?topfive=first
12. Kemudian untuk pengujian hasil 5 angka terbawah, dengan perintah “/dataset?topfi... seperti pada gambar 5.35.
13. Hasil pengujinya tidak berbeda dengan perintah ?topfive=first. Mengapa? Terjadi error atau kesalahan disini. Ternyata setelah dicheck, pada fungsi get_top_five.py hanya mendefinisikan fungsi untuk keluaran 5 data teratas (headframe). Apabila kita ingin tetap menampilkan 5 data terbawah maka kita harus menambahkan perintah berikut pada file get_top_five.py seperti pada listing 5.21.

```
1 if order_by == 'first' else dataframe.tail()
```

Listing 5.21 File get_top_five.py

14. Setelah menambahkan perintah tersebut silahkan disimpan. Kemudian buka kembali CMD. Kemudian aktifkan ulang environment dari main.py
15. Lalu silahkan masuk ke Postman untuk pengujinya
16. Dengan perintah /dataset?topfive=last, maka hasilnya nampak seperti pada gambar 5.36.

The screenshot shows a POSTMAN interface with the following details:

- Method: GET
- URL: `http://127.0.0.1:5000/dataset?topfive=first`
- Authorization tab is selected.
- Headers tab shows (1) header.
- Body tab is selected.
- Pre-request Script tab is present.
- Type dropdown is set to No Auth.
- Body tab is selected.
- Cookies tab is present.
- Headers tab shows (4) headers.
- Test Results tab is present.
- Pretty, Raw, Preview, and HTML dropdown (set to HTML) are present.
- Copy icon is present.
- Data table:

i	raw_value	raw_value		
1	-129	0		
2	yeay	90.0	R	
3	1	-93.0	0	
4	Fadila	100.0	Sukses	
5	Fadila	100.0	A	
6	annisa_fathoroni	100.0	A	

Gambar 5.34 Hasil Eksekusi Program main.py

i	1	raw_value	-129	0
2	0	yeay	90.0	R
3	1	raw_value	-93.0	0
4	2	Fadila	100.0	Sukses
5	3	Fadila	100.0	A
6	4	annisa_fathoroni	100.0	A

Gambar 5.35 Hasil Eksekusi Program main.py

```

1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
2  "<http://www.w3.org/TR/html4/loose.dtd">
3  <html>
4  <head>
5      <title>KeyError: &quot;['sign'] not found in axis&quot; // Werkzeug Debugger</title>
6      <link rel="stylesheet" href=?__debugger__yes&cmd=resource&f=style.css"
7      type="text/css">
8      <!-- We need to make sure this has a favicon so that the debugger does
9      not by accident trigger a request to /favicon.ico which might
10     change the application state. -->
11     <link rel="shortcut icon"
12     href=?__debugger__yes&cmd=resource&f=console.png"
13     <script src=?__debugger__yes&cmd=resource&f=jquery.js"></script>
14     <script src=?__debugger__yes&cmd=resource&f=debugger.js"></script>
15     <script type="text/javascript">
16         TRACEBACK = 160008912,
17         CONSOLE_MODE = false,
18         FVAI_FX = true.

```

Gambar 5.36 Hasil Eksekusi Program main.py

5.4.2.3 Contoh kasus 3 Untuk tutorial ini, akan dicontohkan penanganan error untuk contoh Python Flask. Ini contoh yang ketiga. Sebelum menangani error. Akan dijelaskan terlebih dahulu tutorial untuk pembangunan Pythonnya kemudian akan masuk kepada penanganan error yang ada. Silahkan simak dan ikuti tutorial berikut:

1. Pertama-tama silahkan pastikan bahwa di laptop anda telah terinstall Python. Python yang digunakan ada Python 2 dan Python 3. Untuk tutorial pertama ini, kita akan menggunakan Python 3.
2. Kita akan mengeksekusi main.py, sehingga kita bisa menemukan beberapa error. Karena kita akan mengeksekusi main.py, kita juga akan membangun terlebih dahulu fungsi yang akan dimasukkan dalam main.py (flask)
3. Silahkan buka text editor. Gunakan sublime 3
4. Kemudian silahkan buat script seperti pada listing 5.22.

```
1 def delete_all_data(dataframe):  
2     dataframe.drop(['raw_value'], axis=1)
```

Listing 5.22 File delete_column.py

5. Fungsi tersebut disimpan dengan nama file delete_column.py
6. Fungsi tersebut telah dimasukkan kedalam Flask yaitu main.py sehingga ketika dijalankan fungsinya akan dapat dieksekusi di file flask yang sama dengan fungsi yang lain. Fungsinya yaitu untuk menghapus data sesuai dengan kolom yang telah dijadikan parameter/argumen pada file fungsi tersebut
7. Adapun code flask yang mengeksekusi fungsi ini ialah nampak seperti pada listing 5.23.

```
1 elif request.method == 'DELETE':  
2     if not len(args) is 0:  
3         if 'field' in args:  
4             response = delete_column(response, args['field'])
```

Listing 5.23 File main.py

8. Silahkan masuk ke CMD. Jalankan file main.p dengan masuk ke folder penyimpanannya. Kemudian aktifkan dulu environtment dari file tersebut sehingga Flasnnya dapat berjalan.
9. Untuk pengeksekusiannya seharusnya nampak seperti pada gambar dibawah:
10. Silahkan untuk pengujianya bisa dilihat dengan masuk ke dalam URL yang diperlihatkan pada CMD. Pengujian bisa dilakukan dengan memasukkan URL ke dalam web browser biasa maupun Postman. Contoh pengujian ini akan menggunakan Postman biar lebih mudah dan rapih.
11. Hasilnya akan nampak seperti gambar 5.37, dimana perintahnya telah disesuaikan dengan request yang ada dalam file main.py.

```
(env) D:\Buat Praktek\done>python3 main.py
* Serving Flask app "main" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 201-749-500
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

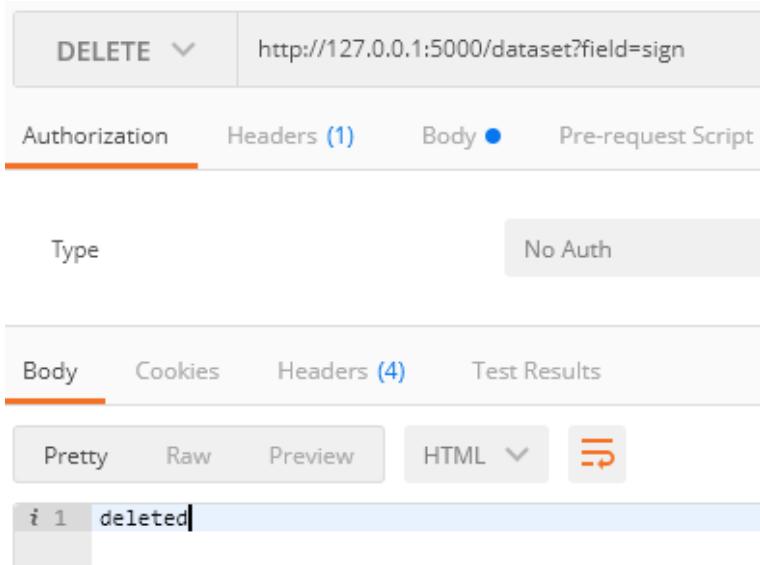
Gambar 5.37 Eksekusi Program main.py

12. Terjadi error.
13. Error terjadi karena pada file delete_all_data.py yang terhubung dengan fungsi delete_column.py tidak sesuai. Perintah yang dimasukkan tidak di definisikan pada file delete_all_data.py yaitu untuk menghapus field “ sign “. Maka harus ditambahkan parameter/argument tersebut
14. Dan tampilan delete_all_data.py nampak seperti pada listing 5.24.

```
1 # menghapus semua data dari csv
2 def delete_all_data(dataframe):
3     dataframe.drop(['raw_value', 'data', 'sign'], axis=1)
```

Listing 5.24 File delete_all_data.py

15. Nah bisa diliat bahwa pada fungsi didefinisikan field apa saja yang bisa dihapus. Tentunya field ini akan dapat dieksekusi melalui file delete_column.py.
16. Ada 3 kolom yang menjadi field yang dapat dieksekusi yaitu Raw_value, data dan sign. Nah parameter signnya sudah ada, maka apabila dijalankan kembali tidak akan terjadi error.
17. Silahkan kita check ulang fungsinya. Masuk lagi ke CMD untuk merefresh atau membuka kembali file Flasknya yaitu Main.py. Jangan lupa untuk mengaktifkan environtmentnya seperti pada contoh-contoh yang telah dijelaskan sebelumnya
18. Hasil pengujianannya nampak seperti pada gambar 5.38.
19. Apabila hasilnya nampak seperti pada gambar 5.38, maka fungsinya telah berhasil dieksekusi sesuai request yang ada di Python Flask
20. Maka telah selesai pula tutorial penanganan error ini.



Gambar 5.38 Hasil Eksekusi Program main.py

BAB 6

ROUTE DAN URL DI FLASK

6.1 Aturan Route di Flask

Kerangka kerja web modern menggunakan teknik route untuk membantu pengguna mengingat URL aplikasi. Berguna untuk mengakses halaman yang diinginkan secara langsung tanpa harus bermigrasi dari halaman beranda. Penggunaan route () dalam Flask digunakan untuk mengikat URL ke suatu fungsi. Routing merupakan suatu modul dalam sebuah aplikasi yang berfungsi untuk mengatur jalannya aplikasi yang berbasis web.

Dekorator python adalah fungsi yang digunakan untuk mengubah fungsi lainnya. Ketika fungsi yang didekorasi pada app.route dipanggil, dekorator dipanggil sebagai gantinya. Dekorator kemudian dapat mengambil tindakan, memodifikasi argumen, menghentikan eksekusi atau memanggil fungsi asli. Untuk membuat routing pada python flask langkah pertama pastikan teman-teman telah menginstall python dan telah menginstall package flask yang akan anda gunakan. Untuk struktur penulisan routing pada flask yaitu seperti pada listing 10.1.

```
1 from flask import Flask  
2  
3 app = Flask(__name__)
```

```

4
5 @app.route("/")
6 def hello():
7     return "Hello World!"
8
9 if __name__ == "__main__":
10    app.run(debug=True)

```

Listing 6.1 File app.py

Pada baris pertama flask adalah framework di sini, sedangkan Flask adalah tipe data kelas Python. Dengan kata lain, Flask adalah prototipe yang digunakan untuk membuat aplikasi web. setelah mengimpor Flask, maka perlu membuat turunan dari kelas Flask untuk aplikasi web yang akan dibuat. Itulah yang dilakukan app = Flask(__name__) adalah variabel khusus yang mendapatkan nilai string "__main__" ketika Anda menjalankan skrip kode diatas.

@app.route adalah dekorator pada flask yang digunakan untuk mencocokkan URL dan memberi tahu Flask URL apa yang harus digunakan untuk melihat fungsi di aplikasi Flask. Kemudian mendefinisikan fungsi yang mengembalikan string "Hello World!". Fungsi itu dipetakan ke URL beranda '/'. Fungsi hello() digunakan untuk menghasilkan URL dan kemudian Return "Hello world" mengembalikan pesan yang ingin ditampilkan di browser pesan yang akan ditampilkan pada browser adalah 'Hello World'.

Karena __name__ akan sama dengan "__main__" dan metode app.run () akan dieksekusi. Teknik ini memungkinkan programmer untuk mengontrol perilaku skrip. Perhatikan juga bahwa ada parameter debug ke true. Itu akan mencetak kemungkinan kesalahan Python di halaman web membantu untuk melacak kesalahan. Namun, dalam lingkungan produksi, jika ingin mengurnya ke False untuk menghindari masalah keamanan.

1. Simpan script diatas dengan nama app.py kemudian jalankan menggunakan command prompt seperti pada gambar 6.1.

```

Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

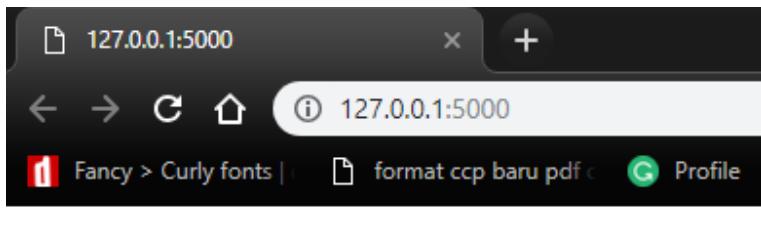
C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 251-765-568
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

Gambar 6.1 Pengujian helloworld

2. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.

3. Jalankan program di `http://127.0.0.1:5000/` maka akan muncul tampilan seperti pada gambar 6.2.



Gambar 6.2 Hasil Pengujian helloworld

4. @app.route harus selalu menjadi dekorator tampilan terluar. Didalam dekorator @app.route("/"), slash ('/') artinya di mainroot anda bisa menambahkan filled apa saja yang ingin anda tambahkan misalnya @app.route("/contoh").

6.2 Aturan URL di Flask

Selain itu anda juga bisa memodifikasinya sesuai keinginan anda agar lebih mudah untuk mengingat suatu URL aplikasi seperti pada listing 6.3.

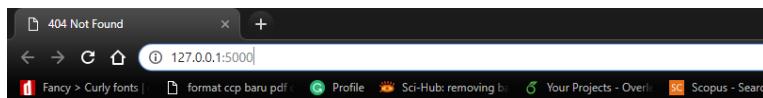
```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route("/contoh")
6 def hello():
7     return "Hello World!"
8
9 if __name__ == "__main__":
10    app.run(debug=True)
```

Listing 6.2 Contoh File app.py

1. @app.route("/contoh") digunakan sebagai Url yang nanti akan anda panggil di browser.
2. Simpan script code diatas kemudian running kembali kode tersebut seperti pada gambar 6.3.
3. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
4. Jalankan program di `http://127.0.0.1:5000/` maka akan muncul tampilan seperti pada gambar 6.4.
5. Error terjadi dikarnakan URL pada direktori @app.route ditambahkan dengan ("contoh") sehingga URL yang diminta tidak ditemukan di server.

```
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

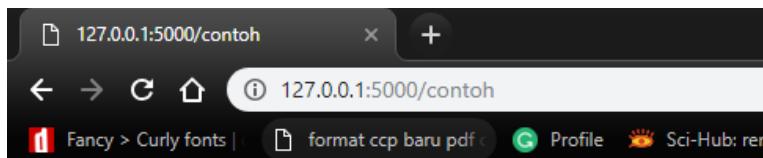
C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 251-765-568
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 6.3 Pengujian url contoh**Not Found**

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

Gambar 6.4 Hasil Pengujian url contoh

6. Maka dari itu tambahkan /contoh pada Url flask, seperti ini <http://127.0.0.1:5000/contoh> kemudian refresh web browser, maka akan muncul tampilan web seperti pada gambar 6.5.



Hello World!

Gambar 6.5 Hasil Pengujian url contoh**6.3 Berbagai Macam Jenis Kreasi URL di Flask**

Anda bisa kreasikan URL pada flask untuk membantu pengguna mengingat URL aplikasi. Selain URL /contoh seperti gambar diatas anda bisa memanage Url dalam flask, misalnya anda akan mencoba parameter get pada URL flask seperti pada listing ??.

```
1 from flask import Flask
2 app = Flask(__name__)
```

```
3 @app.route ('/<contoh2>')
4 def index(contoh2):
5     return contoh2
6
7
8 if __name__ == "__main__":
9     app.run(debug=True)
```

Listing 6.3 Contoh File app.py variabel

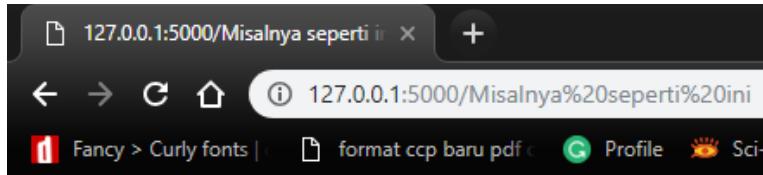
1. URL yang ada pada direktori @app.route di ubah menjadi sebuah variable ('/contoh2'), artinya apapun yang ditulis setalah slash (/) pada browser maka akan di keluarkan di browser.
2. Return contoh2 akan mengembalikan fungsi yang sudah dibuat dna akan menampilkan-nya ke browser.
3. Simpan script code diatas kemudian running kembali kode tersebut seperti pada gambar 6.6.

```
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 251-765-568
- Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

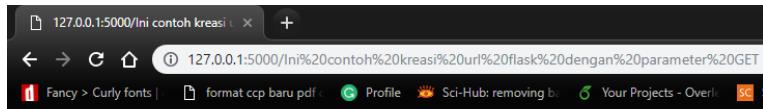
Gambar 6.6 Pengujian variabel

4. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
5. Jalankan program di <http://127.0.0.1:5000/> kemudian tuliskan kata atau kalimat yang di inginkan setealah slash (/) seperti pada gambar 6.7 dan 6.8.
6. Apa yang terjadi jikalau anda menuliskan yang aneh pada url di web browser misalnya seperti pada gambar 6.9, 6.10, 6.11, 6.12, dan 6.13.
7. Itulah tadi contoh sederhana pada URL flask dengan menggunakan Parameter Get.
8. Selain contoh diatas anda akan coba menggunakan simbol-simbol untuk meng-ganti alamat url yang akan anda panggil di browser
9. Misalnya seperti pada listing 6.4.



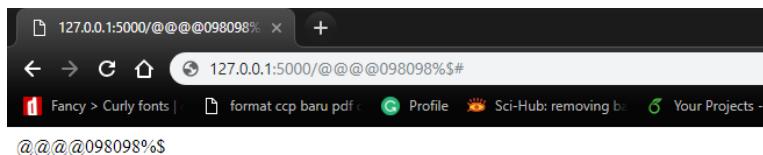
Misalnya seperti ini

Gambar 6.7 Hasil Pengujian Variabel 1



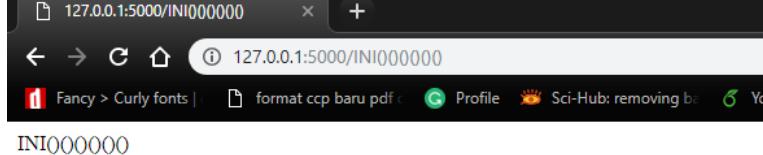
Ini contoh kreasi url flask dengan parameter GET

Gambar 6.8 Hasil Pengujian Variabel 2



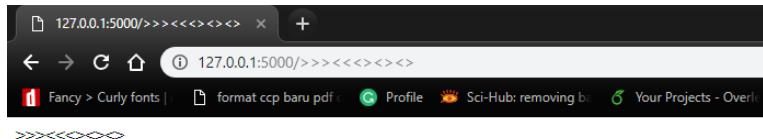
@@@@098098%\$

Gambar 6.9 Hasil Pengujian Variabel 3



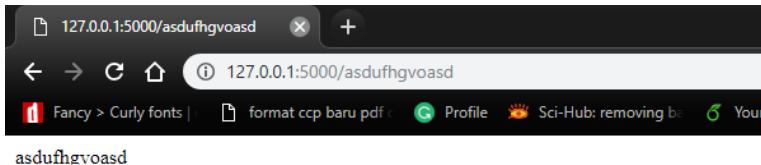
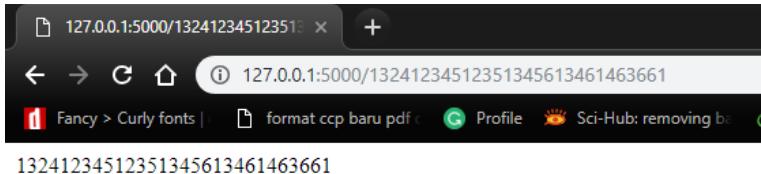
INI000000

Gambar 6.10 Hasil Pengujian Variabel 4



>><<><>

Gambar 6.11 Hasil Pengujian Variabel 5

**Gambar 6.12** Hasil Pengujian Variabel 6**Gambar 6.13** Hasil Pengujian Variabel 7

```

1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/coba%symbol')
6 def hello():
7     return "Hello World!"
8
9 if __name__ == '__main__':
10    app.run(debug = True)

```

Listing 6.4 Contoh File app.py simbol

10. Simpan script code diatas kemudian running kembali kode tersebut seperti pada gambar 6.14.

```

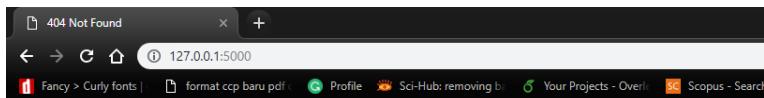
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 251-765-568
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
■

```

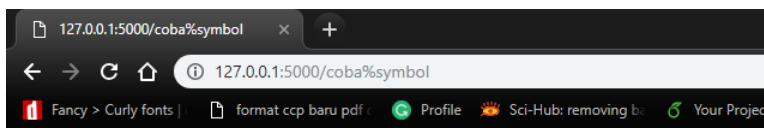
Gambar 6.14 Pengujian simbol

11. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
12. Jalankan program di <http://127.0.0.1:5000/> maka akan muncul halaman web seperti pada gambar 6.15.



Gambar 6.15 Hasil Pengujian simbol 1

13. Maka akan terjadi eror seperti ini, ini terjadi dikarnakan URI yang di panggil tidak di temukan oleh server.
14. Dalam scrip kode yang sudah anda buat anda membuat url baru dengan kata (coba%symbol), sekarang anda coba menambahkan kata tersebut setelah slash di alamat url web <http://127.0.0.1:5000/coba%symbol> seperti pada gambar 6.16.



Gambar 6.16 Hasil Pengujian simbol 2

15. Anda bisa mencoba dengan menambahkan beberapa code html kedalam scrip code yang sudah anda buat, misalnya tulisan yang akan di tampilkan tampak lebih besar, contoh seperti pada listing 6.5.

```

1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def hello():
7     return "<h1>Hello World!</h1>"
8
9
10 if __name__ == '__main__':
11     app.run(debug = True)

```

Listing 6.5 Contoh File app.py heading text

16. return ”

Hello World!

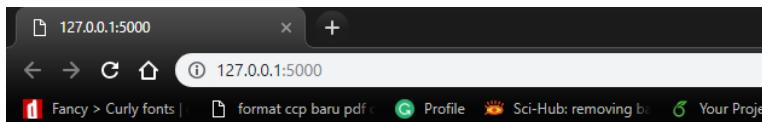
” ini akan menampilkan tulisan Hello World dengan ukuran yang lebih besar, simpan code diatas kemudian jalankan.
17. Simpan skrip kode diatas kemudian jalankan kembali program seperti pada gambar 6.17.

```
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 251-765-568
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 6.17 Pengujian heading text

18. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
19. Jalankan program di <http://127.0.0.1:5000/> maka akan muncul halaman web seperti pada gambar 6.18.



Hello World!

Gambar 6.18 Hasil Pengujian heading text

20. Anda juga bisa kreasiikan @app.route dan url yang berbeda beda dalam 1 file.py misalnya seperti pada listing 6.6.

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def hello():
6     return "Hello World!"
7
8 @app.route('/anu')
9 def nya():
10    return "<h1>GEDE!<h1>"
```

```

12 @app.route('/anu%nya')
13 def itu():
14     return "<h1>panjang!<h1>"
15
16 if __name__ == '__main__':
17     app.run(debug = True)

```

Listing 6.6 Contoh File app.py contoh lain

21. Simpan skrip kode diatas kemudian jalankan kembali program seperti pada gambar 6.19.

```

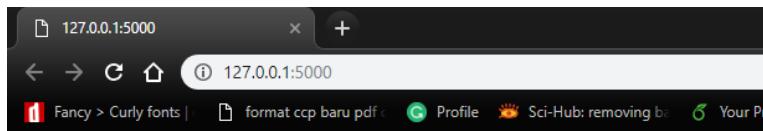
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 251-765-568
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

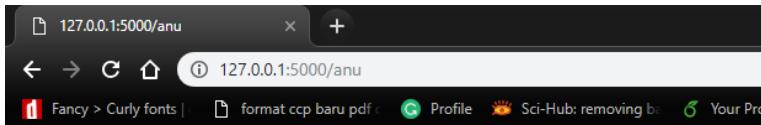
```

Gambar 6.19 Pengujian contoh lain

22. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
 23. Jalankan program di <http://127.0.0.1:5000/> maka akan muncul halaman web seperti pada gambar 6.20.

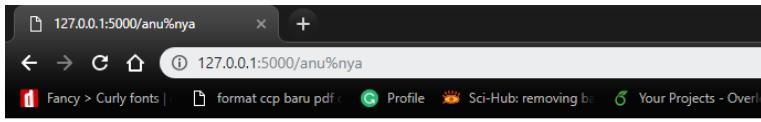
**Gambar 6.20** Hasil Pengujian contoh lain 1

24. Jika anda tambahkan /anu seperti didalam scrip kode pada alamat url <http://127.0.0.1:5000/anu> maka tampilannya akan seperti pada gambar 6.21.
 25. Dan yang yetakhir menggunakan symbol % pada URL yang sudah di buat pada skrip kode maka tampilan pada web browser akan seperti pada gambar 6.22.
 26. Di flask anda bisa sesuka hati mengatur URL apa yang akan di gunakan untuk memanggil sebuah progam yang akan dijalankan, dengan tujuan untuk lebih mempermudah si user dalam mengingat url aplikasi seperti pada listing 6.7.



GEDE!

Gambar 6.21 Hasil Pengujian contoh lain 2



panjang!

Gambar 6.22 Hasil Pengujian contoh lain 3

```

1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route("/cobacoba")
6 def hello():
7     return "Hello World!"
8
9 if __name__ == "__main__":
10     app.run(debug=True)

```

Listing 6.7 Contoh File app.py cobacoba 1

27. Simpan skrip kode diatas kemudian jalankan kembali program seperti pada gambar 6.23.

```

Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

```

```

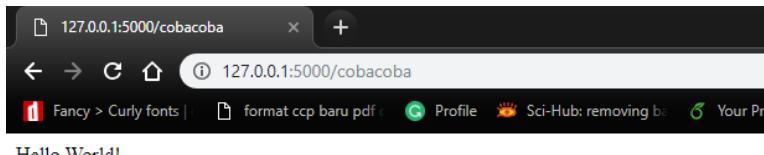
C:\Users\PC_10\royek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 251-765-568
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

Gambar 6.23 Pengujian cobacoba 1

28. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.

29. Jalankan program di <http://127.0.0.1:5000/cobacoba> maka akan muncul halaman web seperti pada gambar 6.24.



Gambar 6.24 Hasil Pengujian cobacoba 1

30. Pada listing 6.8.

```

1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route("/cobacoba")
6 def hello():
7     return "Hello World!"
8
9 if __name__ == "__main__":
10    app.run(debug=True)

```

Listing 6.8 Contoh File app.py cobacoba 2

31. Simpan skrip kode diatas kemudian jalankan kembali program seperti pada gambar 6.25.

```

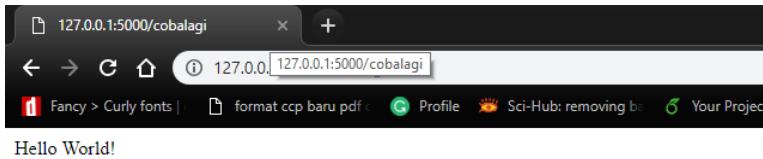
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 251-765-568
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

Gambar 6.25 Pengujian cobacoba 2

32. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
33. Jalankan program di <http://127.0.0.1:5000/cobacoba> maka akan muncul halaman web seperti pada gambar 6.26.
34. Contoh yang selanjutnya adalah seperti pada listing 6.9.



Gambar 6.26 Hasil Pengujian cobacoba 2

```

1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route("/sekalilagi")
6 def hello():
7     return "INI ADALAH HASILNYA!"
8
9 if __name__ == "__main__":
10     app.run(debug=True)

```

Listing 6.9 Contoh File app.py sekali lagi

35. Simpan skrip kode diatas kemudian jalankan kembali program seperti pada gambar 6.27.

```

Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

```

```

C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 251-765-568
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

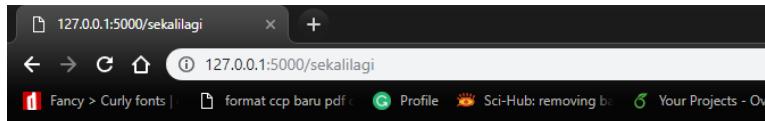
Gambar 6.27 Pengujian sekali lagi

36. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
 37. Jalankan program di <http://127.0.0.1:5000/cobalagi> maka akan muncul halaman web seperti pada gambar 6.28.
 38. Pada listing 6.10.

```

1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route("/cobacoba")

```



INI ADALAH HASILNYA!

Gambar 6.28 Hasil Pengujian sekali lagi

```

6 def hello():
7     return "Hello World!"
8
9 if __name__ == "__main__":
10    app.run(debug=True)

```

Listing 6.10 Contoh File app.py cobacoba 3

39. Selama URL ada di dalam decorator flask (@app.route) apapun URL yang dibuat dan kemudian di panggil ke browser maka akan menampilkan output sesuai dengan apa yang anda ingin tampilkan dengan mngembalikan fungsi yang sudah dibuat. Jika URL yang dimasukan kedalam alamat web tidak sesuai maka akan terjadi eror yang mana URL yang di minta tidak ditemukan oleh server flask. Kecuali URL yang di buat menggunakan variable seperti contoh sebelumnya, flask akan menampilkan string yang ditulis pada alamat url di browser.
40. Pada contoh sebelumnya parameter variable yang sudah dibuat menjadi parameter get pada flask python. flask memberikan kemudahan bagi para user untuk mengakses alamat web dengan menggunakan URL yang bisa di ubah ubah sesuka hati menggunakan karakter-karakter aneh didalam url itu salah satu kelebihan flask yang merupakan framework pada python yang sangat customizable bagi si pengguna flask python. Itulah tadi contoh sederhana dan penjelasan Route dan URL di Flask.

6.4 Penanganan Error

6.4.1 Error 1

Pada listing 9.3.

```

1 from flask import Flask , render_template , make_response , send_file ,
2     url_for
3 import json , time
4 import Tkinter
5 import tkMessageBox
6 import atexit
7 from time import clock
8 import request
9 import os

```

```
9
10 app = Flask(__name__)
11 #
12 #
13 #
14 #
15 #
16 def img():
17     PEOPLE_FOLDER = os.path.join('static', 'img')
18     app.config['UPLOAD_FOLDER'] = PEOPLE_FOLDER
19
20 @app.route('/')
21 def show_index():
22     img()
23     full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'EEG.jpg')
24     return render_template("index.html", user_image = full_filename)
25
26 def popup():
27     tkMessageBox.showinfo("Information","Created in Python.")
28
29 @app.route('/live')
30 def live():
31     popup()
32     img()
33     full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'pict.jpg')
34     return render_template('live.html', user = full_filename)
35
36 dataset = []
37 def myline():
38     f = open('tmp', 'r+')
39     line = f.readlines()
40
41
42     for x in line:
43         dataset.insert(0,x)
44     return
45
46 def fromDataset():
47     for x in dataset:
48         return int(x)
49
50 @app.route('/live-data')
51 def live_data():
52     myline()
53     data = [time.time() * 500, fromDataset()]
54     print time.ctime(time.time())
55     response = make_response(json.dumps(data))
56     response.content_type = 'application/json'
57     return response
58
59 @app.route('/live-tmp', methods=['GET'])
60 def livetmp():
61     f = open('tmp', 'r')
```

```

62     line = f.readlines()
63
64     for x in line:
65         response = make_response(json.dumps(x))
66         response.content_type = 'application/json'
67         return response
68
69 @app.route('/data', defaults={'req_path':''}, methods=['GET'])
70 @app.route('/<path:req_path>')
71 def simpanjson(req_path):
72     BASEDIR = '/Users/PC_10/Zroyek/example2/LIVE/data'
73     data_path = os.path.join(BASEDIR, req_path)
74     if os.path.isfile(data_path):
75         return send_file(data_path)
76     files = os.listdir(data_path)
77     return render_template('files.html', files=files)
78 #
79 #
80 #
81 #
82 #
83 #
84
85 def secondsToStr(t):#fungsi waktu
86     return "%d:%02d:%02d.%03d" % \
87             reduce(lambda ll ,b : divmod(ll[0] ,b) + ll[1:], [(t*1000,),1000,60,60])
88
89 def sapa(nama):
90     print("Hi, " + nama + ". Apa kabar?")
91 sapa('teman')
92
93 line = "="*40
94 def log(s, elapsed=None):
95     print line
96     print secondsToStr(clock()), '_', s
97     if elapsed:
98         print "Elapsed time:", elapsed
99     print line
100    print
101
102 def date():
103     date = time.time
104     print time.ctime(time.time())
105
106 def endlog():
107     end = clock()
108     elapsed = end-start #ngitung waktu awal sampai akhir
109     log("End Program", secondsToStr(elapsed))
110     date()
111     print ('Proyek dua')
112
113 def now():
114     return secondsToStr(clock())
115
116

```

```

117 start = clock()
118 atexit.register(endlog)
119 log("Start Program")
120
121 def print_info( nama , Npm ):
122     print ("Nama: " , nama)
123     print ("Npm: " , Npm)
124 print_info( Npm=1164035 , nama = "Eko Cahyono Putro" )
125 print_info( Npm=1164049 , nama = "Nur Arkhamia Batubara" )
126 #
127 #
128 #
129 #
130 #
131 if __name__ == '__main__':
132     app.run(debug=True , host='127.0.0.1' , port=8080)

```

Listing 6.11 File flasklivechart.py

1. Simpan skrip kode diatas kemudian namakan file tersebut menjadi flasklivechart.py, jalankan file tersebut menggunakan cmd dengan menuliskan perintah python flasklivechart.py seperti pada gambar 6.29.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: python + -x

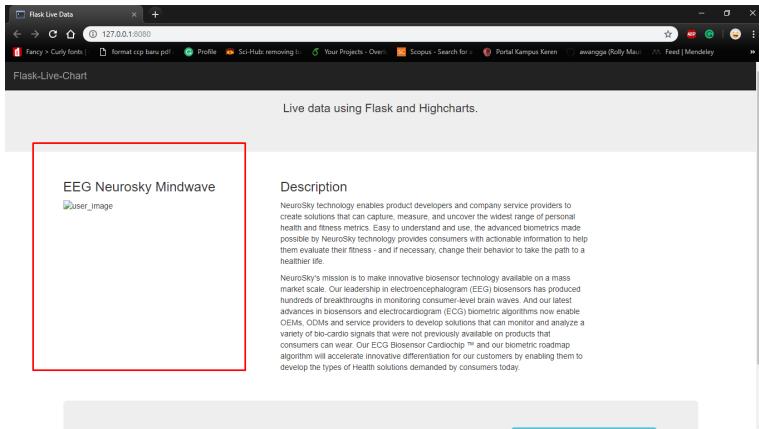
('Nama: ', 'Eko Cahyono Putro')
('Npm: ', 1164035)
('Nama: ', 'Nur Arkhamia Batubara')
('Npm: ', 1164049)
* Serving Flask app "flasklivechart" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
Hi, teman. Apa kabar?
=====
0:00:00.000 - Start Program
=====

('Nama: ', 'Eko Cahyono Putro')
('Npm: ', 1164035)
('Nama: ', 'Nur Arkhamia Batubara')
('Npm: ', 1164049)
* Debugger is active!
* Debugger PIN: 988-858-628
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)

```

Gambar 6.29 Pengujian file flasklivechart.py

2. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
3. Jalankan program di <http://127.0.0.1:8080> kemudian akan muncul tampilan seperti berikut:
4. Akan tetapi terdapat suatu masalah pada halaman home seperti pada gambar 6.30.
5. Muncul eror seperti ini pada Command Prompt seperti pada gambar 6.31.



Gambar 6.30 Hasil Pengujian file flasklivechart.py

```
WindowsError: [Error 3] The system cannot find the path specified: u'/Users/PC 10/Zroyek/example2/LIVE/data\\data\\img\\EEG.jpg\\*.*'
127.0.0.1 - - [07/Febr/2019 02:30:28] "GET /favicon.ico HTTP/1.1" 500 -
Traceback (most recent call last):
```

Gambar 6.31 CMD file flasklivechart.py

6. Dan pada command Prompt ditemukan eror seperti berikut:
7. Windows Eror: [Error 3] The system cannot find the path specified: u'/Users/PC 10/Zroyek/example2/LIVE/data
data/img/EEG.jpg
*,'

6.4.2 Error 2

1. Melanjutkan pembahasan Error 1 yang saling berkaitan dengan Error 2 ini
2. Masalah yang terjadi pada aplikasi yang sudah dibuat adalah gambar di program flask tidak muncul, function yang ada dalam kode program adalah seperti pada listing 6.12.

```
1 def img():
2     PEOPLE_FOLDER = os.path.join('data', 'img')
3     app.config['UPLOAD_FOLDER'] = PEOPLE_FOLDER
4
5 @app.route('/')
6 def show_index():
7     img()
8     full_filename = os.path.join(app.config['UPLOAD_FOLDER'],
9         'EEG.jpg')
10    return render_template("index.html", user_image =
full_filename )
```

Listing 6.12 File flasklivechart.py edit

3. fungsi def img() dan def show_index() saling berhubungan karena fungsi show_index() akan menampilkan gambar dengan merender file index.html yang ada pada folder templates dan fungsi img() akan menggabungkan folder dan menentukan path yang digunakan serta akan menampilkan gambar di web browser.
4. Disini kita akan mencoba memodifikasi kode yang digunakan untuk menampilkan gambar pada halaman HOME agar gambar dapat ditampilkan di browser.
5. Dan untuk struktur projek yang di kerjakan adalah seperti pada gambar 6.32.



Gambar 6.32 Struktur Projek

6. Dalam hasil yang ditampilkan windows error mengatakan bahwa sistem tidak mampu menemukan jalur yang ditentukan.
7. Maksut jalur disini adalah path yang digunakan untuk mendeteksi dimana file gambar yang akan ditampilkan berada.
8. Dalam projek flask yang sudah dibuat file img terdapat pada folder img yang disimpan didalam folder static dan gambar yang akan ditampilkan di web browser adalah gambar yang bernama EEG.jpg seperti pada gambar 6.33.
9. Function yang menentukan path gambar adalah seperti pada listing 6.13.

```

1 def img():
2     PEOPLE_FOLDER = os.path.join('data', 'img')
3     app.config['UPLOAD_FOLDER'] = PEOPLE_FOLDER

```

Listing 6.13 File flasklivechart.py edit



Gambar 6.33 Alat yang akan ditampilkan

10. Setelah diteliti ada yang salah pada os.path.join, bukannya menggabungkan folder yang menyimpan gambar malah menggabungkan folder yang lain.
11. Didalam folder ‘data’ tidak ada folder img yang menyimpan gambar EEG.jpg melainkan folder static yang didalamnya terdapat folder img yang menyimpan gambar EEG.jpg
12. Jadi yang harus kita lakukan adalah mengganti ‘data’ dengan ‘static’ karena folder img tada didalam folder static yang menyimpan gambar EEG.jpg
13. PEOPLE_FOLDER = os.path.join('static', 'img')

BAB 7

HTTP GET METHOD REQUEST

7.1 HTTP Method Request

7.1.1 Pengertian HTTP Method Request

Protokol HTTP adalah protokol permintaan atau respon. Klien mengirimkan permintaan keserver dalam bentuk metode permintaan, URL, dan versi protokol, diikuti oleh pesan seperti MIME yang berisi perubahan permintaan, informasi klien, dan kemungkinan ontent tubuh melalui koneksi dengan server[6]. Protokol ini sangat ringan serta generik dan tidak berstatus sehingga dapat dipergunakan oleh tipe dokument apa saja. Method adalah sekumpulan kode yang diberi nama, untuk merujuk kesekumpulan kode yang ada kemuadian digunakan sebuah nama yang disebut dengan nama method. Method sendiri mempunyai parameter sebagai input (masukan) dan nilai kembalian sebagai output (keluaran). Request adalah permintaan dimana fungsi ini digunakan sebagai istilah ataupun kinerja dalam pengembalian nilai dari masukan yang dieksekusi.

Berdasarkan beberapa penjelasan diatas, maka untuk pengertian dari HTTP Method Request sendiri merupakan seperangkat metode permintaan untuk menunjukkan tindakan yang diinginkan yang akan dilakukan untuk sumber daya tertentu. Meskipun

mereka juga bisa menjadi kata benda, metode permintaan ini kadang-kadang disebut sebagai verba HTTP. Masing-masing menerapkan semantik yang berbeda, namun beberapa fitur umum digunakan bersama oleh mereka adalah misalnya Metode pertintaan dapat berupa safe, idempotent, atau cacheable.

7.1.2 Jenis-jenis HTTP Method Request

1. GET : akan dijelaskan pada point berikutnya.
2. HEAD : Metode HEAD meminta tanggapan yang identik dengan permintaan GET, namun tanpa respon body.
3. POST : Metode POST digunakan untuk mengirimkan entitas ke sumber daya yang ditentukan, sering menyebabkan perubahan pada keadaan atau efek samping pada server.
4. PUT : Metode PUT menggantikan semua representasi terkini dari sumber target dengan muatan permintaan.
5. DELETE : Metode DELETE akan menghapus sumber daya yang ditentukan
6. CONNECT : Metode CONNECT menetapkan terowongan keserver yang diidentifikasi oleh sumber target.
7. OPTIONS : Metode OPTIONS digunakan untuk menggambarkan opsi komunikasi untuk sumber target.
8. TRACE : metode TRACE ini yaitu untuk melakukan tes pesan loop-back disepanjang jalan menuju sumber daya target.
9. PATCH : Metode PATCH digunakan untuk menerapkan modifikasi sebagian pada sumber daya.

7.1.3 Penjelasan Lengkap HTTP Get Method

Metode GET digunakan untuk meminta representasi sumber daya yang ditentukan. permintaan menggunakan Get seharusnya hanya mengambil data. GET adalah salah satu metode HTTP yang paling umum digunakan baik dalam pengimplementasian biasa ataupun sudah dalam bentuk pengujian. Hal yang harus diperhatikan dalam Method Get yaitu :

- Permintaan GET dapat di-cache.
- Permintaan GET tetap ada dalam riwayat browser.
- Permintaan GET dapat ditandai.
- Permintaan GET tidak boleh digunakan saat berurusan dengan data sensitif.

- Permintaan GET memiliki batasan panjang.
- Permintaan GET hanya digunakan untuk meminta data (tidak dimodifikasi).
- Permintaan GET dibatasi oleh panjang string sebanyak 2047 karakter.
- Permintaan GET memungkinkan pengunjung langsung memasukkan nilai variable pada form proses.
- Variabel diambil dengan \$_REQUEST["nama"] atau \$_GET["nama"]

7.1.4 Pembacaan HTTP Get Method

Data dikirimkan dalam HTTP Request dalam dua cara, tergantung dari method yang dikirimkan, yaitu :

1. Melalui URL, dengan parameter yang diberikan. Digunakan oleh GET.
2. Melalui entity body dalam HTTP Request. Digunakan untuk POST dan PUT.

Pada prakteknya terdapat satu cara lagi untuk mengirimkan data, yaitu melalui cookie, tetapi penggunaan cookie tidak akan terlalu efektif karena cookie dirancang untuk menyimpan data status pengguna.

7.1.5 Pembacaan Data pada URL

Pembacaan data yang dikirimkan melalui URL biasanya dilakukan untuk request dengan method GET. Untuk melihat bagaimana GET mengirimkan data, kita terlebih dahulu harus mengerti tentang sintaks penulisan URL. Secara umum, sebuah URL memiliki sintaks seperti berikut :

```
<schema>://<user>:<password>@<host>:<port>/<path>?<query>#<fragment>
```

Listing 7.1 Contoh kode untuk schema

Apa makna dari setiap bagian dari URL yang dijelaskan pada 7.1? Pada tabel 7.1, anda dapat melihat makna dan maksud dari contoh URL yang telah diberikan.

7.2 Mekanisme HTTP Method Request

7.2.1 Mekanisme / Alur kerja HTTP Get Method

Mekanisme adalah suatu rangkaian kerja sebuah alat yang digunakan dalam menyelesaikan sebuah masalah yang berkaitan dengan proses kerja, tujuannya untuk menghasilkan hasil yang mekasimal serta mengurangi datangnya atau munculnya kegagalan. untuk mekanisme HTTP Get Method sendiri dapat diperhatikan sebagai berikut :

1. Silahkan membuat dan membangun sebuah URL API.

Tabel 7.1 Penjelasan Schema

Nama	Deskripsi	Harus ada ?
schema	Protokol yang digunakan	Ya
user	Nama pengguna	Tidak
password	Password untuk nama pengguna	Tidak
hots	Hostname atau IP	Ya
port	Port yang akan diakses. Beberapa atau sebagian protokol memiliki port standar yaitu seperti HTTP = 80	Tergantung Protokol
paht	Lokasi data pada server	Tergantung Protokol
query	Digunakan untuk mengirimkan parameter kepada aplikasi.	Tidak
fragment	Nama dari bagian tertentu pada data (misalnya : judul pada buku)	Tidak

2. Didalam URL API (endpoint) tersebut kita akan menggunakan fungsi Method Get.
3. Kemudian didalam endpoint tersebut akan difungsikan inputan.
4. Inputan tersebut kemudian akan meghasilkan output (keluaran) dari Metgod GET tersebut.
5. Secara sederhana, garis besar mekanisme atau alur kerja Method Get nampak seperti penjelasan diatas.
6. Untuk tutorial pembangunan endpoint seperti pada point kedua akan dijelaskan pada point selanjutnya.

7.3 Contoh URL HTTP Get Method

Untuk pemberian contoh ini akan dibarengin dengan tutorial pembangunannya, jadi diharapkan teman-teman dapat dengan mudah memahami dan mudah dalam mengikuti contoh yang diberikan. Berikut tutorialnya :

1. Endpoint adalah perangkat komputasi jarak jauh yang berkomunikasi bolak-balik dengan jaringan yang terhubung dengannya. Fungsi-fungsi yang dipergunakan yaitu sebagai contoh berikut :

- Fungsi 1 : LoadData.py yaitu membaca file CSV.

```

1 import pandas
2
3 # membaca file csv
4 def load_data(filename):
```

```
5     return pandas.read_csv(filename, delimiter=';')
```

Listing 7.2 Contoh kode untuk membaca file CSV

- Fungsi 2 : Getalldata.py yaitu untuk menampilkan semua data dari file CSV.

```
1 def get_all_data(dataframe):
2     return dataframe
```

Listing 7.3 Contoh kode untuk menampilkan semua data dari file CSV

- Fungsi 3 : Getcolumn.py yaitu untuk menampilkan data dari kolom tertentu.

```
1 def get_column(dataframe, fieldname):
2     return dataframe.loc[:, [fieldname]]
```

Listing 7.4 Contoh kode untuk menampilkan data dari kolom

- Fungsi 4 : Gettopfive.py yaitu untuk menampilkan 5 data teratas dan terbawah.

```
1 def get_top_five(dataframe, order_by):
2     return dataframe.head() if order_by == 'first'
3 else dataframe.tail()
```

Listing 7.5 Contoh kode untuk menampilkan 5 data teratas dan terbawah

- Fungsi 5 : Sorting.py yaitu untuk mengurutkan data dari yang terbesar ke yang terkecil dan sebaliknya.

```
1 def sorting(dataframe, order_by):
2     asc = True if order_by == 'ascending' else
3 False
4     return dataframe.sort_index(ascending=asc)
```

Listing 7.6 Contoh kode untuk mengurutkan data

- Fungsi 6 : Convertjson.py yaitu untuk mengganti data kedalam format Json.

```
1 import json
2 def convert_json(dataframe):
3
4     return
5 json.loads(dataframe.to_json(orient='index'))
```

Listing 7.7 Contoh kode untuk mengganti data

- Fungsi 7 : Deletecolumn.py yaitu untuk menghapus kolom tertentu.

```
1 def delete_column(dataframe, column):
2     return dataframe.drop(column, axis=1)
```

Listing 7.8 Contoh kode untuk menghapus kolom tertentu

- Fungsi 8 : Deletealldata.py yaitu untuk Menghapus semua data pada file CSV.

```

1 def delete_all_data(dataframe):
2     dataframe.drop(['raw_value', 'data', 'sign'],
3     axis=1)

```

Listing 7.9 Contoh kode untuk menghapus semua data

- Fungsi 9 : Insertdata.py yaitu untuk menambahkan data.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:02:27 2019
4
5 @author: Rahmatul Ridha
6 """
7
8 from get_row_count import get_row_count
9
10 def insert_data(dataframe, newdata):
11     index = get_row_count(dataframe) + 1
12     dataframe.at[index] = newdata

```

Listing 7.10 Contoh kode untuk menambah data

- Fungsi 10 : Getinfo.py yaitu untuk Menampilkan data csv namun dengan format json bawaan dari library pandas yang akan berupa deskripsi.

```

1 def get_info(dataframe):
2     return dataframe.describe()

```

Listing 7.11 Contoh kode untuk Menampilkan data csv

- Fungsi 11 : Getrow.py yaitu untuk Menampilkan seluruh jumlah baris.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:20:06 2019
4
5 @author: Rahmatul Ridha
6 """
7
8 def get_row(dataframe, id):
9     return dataframe.loc[id, ['raw_value', 'data',
10 'sign']]

```

Listing 7.12 Contoh kode untuk Menampilkan seluruh jumlah baris

- Fungsi 12 : Getrowfield.py yaitu untuk Menampilkan data perbaris sesuai dengan kolom yang digunakan.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:22:00 2019
4
5 @author: Rahmatul Ridha
6 """
7
8 def get_row_field(dataframe, field):

```

```
9     return " {}:{}".format(field , dataframe[ field ])
```

Listing 7.13 Contoh kode untuk Menampilkan data perbaris

- Fungsi 13 : Updatedata.py yaitu untuk Mengubah data berdasarkan index id.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:20:57 2019
4
5 @author: Rahmatul Ridha
6 """
7
8 def update_data(dataframe , id , newdata):
9     dataframe.loc[id , :] = newdata
10    return dataframe
```

Listing 7.14 Contoh kode untuk Mengubah data berdasarkan index id

- Fungsi 14 : Deletedata.py yaitu untuk Menghapus data berdasarkan index id.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:20:45 2019
4
5 @author: Rahmatul Ridha
6 """
7
8 def delete_data(dataframe , id):
9
10    id = int(id) if type(id) != int else id
11    dataframe.drop([ id ], axis=0, inplace=True)
12    return dataframe
```

Listing 7.15 Contoh kode untuk Menghapus data berdasarkan index id

- Fungsi 15 : Getrowjson.py yaitu untuk Mengubah data menjadi Json.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:25:18 2019
4
5 @author: Rahmatul Ridha
6 """
7
8 from convert_json import convert_json
9 def get_row_json(dataframe):
10    return convert_json(dataframe)
```

Listing 7.16 Contoh kode untuk Mengubah data menjadi Json

- Fungsi 16 : Getalldatajson.py yaitu untuk Menampilkan seluruh data dalam bentuk format Json.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:25:20 2019
4
5 @author: Rahmatul Ridha
6 """
7
8 from convert_json import convert_json
9 def get_all_data_json(dataframe):
10    return convert_json(dataframe)

```

Listing 7.17 Contoh kode untuk Menampilkan seluruh data

- Fungsi 17 : Amountdata.py yaitu untuk Menghitung jumlah data secara keseluruhan dari kolom dan baris.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:25:21 2019
4
5 @author: Rahmatul Ridha
6 """
7
8 def jumlah_data(dataframe, field):
9    count = dict(row=dataframe.shape[0],
10 column=dataframe.shape[1])
11    if not field == 'all':
12        return count['row'] if field == 'row' else
13 count['column']
14    else:
15        return count

```

Listing 7.18 Contoh kode untuk Menghitung jumlah data

- Fungsi 18 : Getcolumncount.py yaitu untuk Menghitung jumlah kolom pada file.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:25:21 2019
4
5 @author: Rahmatul Ridha
6 """
7
8 def get_column_count(dataframe):
9    return dataframe.shape[1]

```

Listing 7.19 Contoh kode untuk Menghitung jumlah kolom

- Fungsi 19 : Getrowcount.py yaitu untuk Menghitung jumlah baris pada file.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:26:51 2019
4
5 @author: Rahmatul Ridha
6 """

```

```

7
8 def get_row_count(dataframe):
9     return dataframe.shape[0]

```

Listing 7.20 Contoh kode untuk Menghitung jumlah baris

- Fungsi 20 : Reloaddata.py yaitu untuk Mengembalikan nilai data.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:26:49 2019
4
5 @author: Rahmatul Ridha
6 """
7
8 def reload_data(filename, dataframe):
9     dataframe.to_csv(filename, index=False, sep=';')

```

Listing 7.21 Contoh kode untuk Mengembalikan nilai data

7.3.1 Endpoint 1 (URL API / Dataset)

Pembatasan pertama ialah dari Endpoint 1 dimana URL API nya yaitu dataset. Silahkan tuliskan code dibawah sebagai contoh pertama :

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 12 14:27:49 2019
4
5 @author: Rahmatul Ridha
6 """
7
8 @app.route('/dataset', methods=['GET', 'POST',
9 'DELETE'])
10 def dataset():
11     args = request.args
12     response = dataframe
13     if request.method == 'GET':
14         if not len(args) is 0:
15             if 'field' in args:
16                 response = get_column(response,
17 args['field']))
18             if 'topfive' in args:
19                 response = get_top_five(response,
20 args['topfive']))
21             if 'sort' in args:
22                 response = sorting(response,
23 args['sort']))
24             if 'format' in args:
25                 if args['format'] == 'json':
26                     response = jsonify(get_all_data_json(response))
27                 else:
28                     response = str(response)
29             else:
30                 response = str(response)

```

31 return response

Listing 7.22 Contoh kode untuk URL API

Pada tabel code diatas dapat dilihat bahwa fungsi yang digunakan ialah dataset dimana data set dilakukan pemanggilan fungsi yang ada di file-file yang telah di import. Seperti yang bisa dilihat bahwa pada fungsi dataset direalisasikan dengan metode GET, POST dan DELETE jadi disesuaikan dengan fungsi pada file-file yang dihubungkan atau dipanggil kedalam fungsi dataset. Yang akan dibahas disini adalah MEthod GETnya saja. Tutorialnya sebagai berikut :

- Setelah melakukan perintah 7.22
- Kemudian masukkan perintah yang sesuai
- Pendefinisian endpoint dataset
- Definisikan request atau permintaan metode yang digunakan. Ada Get
- Pada Method GET : Silahkan masukkan beberapa argument yang akan diolah.
- Argumentnya ialah parameter field, topfive, sort dan format
- Penjelasan Parameter :

All : Mengambil semua data dari file csv

Field : Mengambil data berdasarkan parameter field yaitu ada Raw-Value, sign dan daya

Topfive : Mengambil 5 data berdasarkan data paling atas dan 5 data paling bawah

Sort : Mengambil data dan diurutkan sesuai dengan urutan ascending (kecil ke besar) dan descending (besar ke kecil)

Format : Menampilkan data dengan format tulisan JSON dan Raw

- Selanjutnya pastikan setiap argument memberikan respon sesuai dengan fungsi masing-masing
- Kemudian reloaddata akan menampilkan data terbaru setelah dilakukan perintah delete
- Silahkan uji endpointnya sesuai dengan parameter tersebut menggunakan POSTMAN maka hasilnya akan seperti berikut :

- HASIL PENGUJIAN (Fungsi Dataset, method: GET):

1. Menampilkan data secara keseluruhan (/dataset) seperti pada gambar 7.1.
2. Menampilkan data sesuai dengan field yang dimasukkan (/dataset?field=data) seperti pada gambar 7.2.
3. Menampilkan data sesuai dengan field yang dimasukkan (/dataset?field=raw_value) seperti pada gambar 7.3.

i	1	raw_value	data	sign
2	1	bbbb	54654	R
3	2	raw_value	-93	0
4	3	raw_value	-120	0
5	4	jhgjhg	54654	R
6	5	raw_value	-120	0
7	6	raw_value	-120	0
8	7	raw_value	-120	0
9	8	raw_value	-120	0
10	9	raw_value	-120	0
11	10	raw_value	-120	0
12	11	raw_value	-120	0
13	12	raw_value	-120	0
14	13	raw_value	-267	0
15	14	raw_value	-250	0
16	15	raw_value	-250	0
17	16	raw_value	-250	0
18	17	raw_value	-250	0
19	18	raw_value	-250	0

Gambar 7.1 Hasil Pengujian 1 fungsi dataset (GET)

4. Menampilkan data dalam bentuk Json (/dataset?format=json) seperti pada gambar 7.4.
5. Menampilkan 5 data teratas (/dataset?topfive=first) seperti pada gambar 7.5.
6. Menampilkan data dengan urutan besar ke kecil (/dataset?field=data&sort=descending) seperti pada gambar 7.6.
7. Menampilkan 5 data teratas (/dataset?topfive=first) seperti pada gambar 7.7.
8. Menampilkan data dengan urutan besar ke kecil (/dataset?field=data&sort=descending) seperti pada gambar 7.8.

7.3.2 Endpoint 2 (URL API : /dataset/jumlah)

Pembahasan selanjutnya yaitu tutorial untuk pembuatan endpoint kedua (/dataset/jumlah). Silahkan tuliskan code seperti pada listing 7.23:

```

1 @app.route('/dataset/<int:id>', methods=['GET', 'POST', 'DELETE'])
2 def detail(id):
3     args = request.args
4     detailframe = get_all_data(dataframe), id)
5     if request.method == 'GET':
6         if not len(args) is 0:
7             if 'field' in args:

```

i	1	data
2	1	54654
3	2	-93
4	3	-120
5	4	54654
6	5	-120
7	6	-120
8	7	-120
9	8	-120
10	9	-120
11	10	-120
12	11	-120
13	12	-120
14	13	-267
15	14	-250
16	15	-250
17	16	-250
18	17	-250
19	18	-250

Gambar 7.2 Hasil Pengujian 2 fungsi dataset (GET)

i	1	raw_value
2	1	bbbb
3	2	raw_value
4	3	raw_value
5	4	jhgjhg
6	5	raw_value
7	6	raw_value
8	7	raw_value
9	8	raw_value
10	9	raw_value
11	10	raw_value
12	11	raw_value
13	12	raw_value
14	13	raw_value
15	14	raw_value
16	15	raw_value
17	16	raw_value
18	17	raw_value
...	...	-

Gambar 7.3 Hasil Pengujian 3 fungsi dataset (GET)

```

1  {
2      "1": {
3          "data": 54654,
4          "raw_value": "bbbb",
5          "sign": "R"
6      },
7      "2": {
8          "data": -93,
9          "raw_value": "raw_value",
10         "sign": "0"
11     },
12     "3": {
13         "data": -120,
14         "raw_value": "raw_value",
15         "sign": "0"
16     }

```

Gambar 7.4 Hasil Pengujian 4 fungsi dataset (GET)

i	1	raw_value	data	sign
2	1	bbbb	54654	R
3	2	raw_value	-93	0
4	3	raw_value	-120	0
5	4	jhgjhg	54654	R
6	5	raw_value	-120	0

Gambar 7.5 Hasil Pengujian 5 fungsi dataset (GET)

i	1			data
2	3866			201
3	3865			201
4	3864			201
5	3863			201
6	3862			201
7	3861			201
8	3860			201

Gambar 7.6 Hasil Pengujian 6 fungsi dataset (GET)

i	1	raw_value	data	sign
2	1	bbbb	54654	R
3	2	raw_value	-93	0
4	3	raw_value	-120	0
5	4	jhgjhg	54654	R
6	5	raw_value	-120	0

Gambar 7.7 Hasil Pengujian 7 fungsi dataset (GET)

i	1		data
2	3866	201	
3	3865	201	
4	3864	201	
5	3863	201	
6	3862	201	
7	3861	201	
8	3860	201	

Gambar 7.8 Hasil Pengujian 8 fungsi dataset (GET)

```

8         detailframe = get_row_field(detailframe , args['field']
9     })
10    if 'format' in args:
11        if args['format'] == 'json' and 'field' not in args:
12            detailframe = jsonify(get_row_json(detailframe))
13    else:
14        return jsonify(detailframe)
15    else:
16        detailframe = str(detailframe)
17    else:
18        detailframe = str(detailframe)
        return detailframe

```

Listing 7.23 Method Get dari Endpoint : URL API /dataset/jumlah

Pada listing 7.23 ada fungsi jumlah, pada fungsi ini di panggil lah fungsi jumlah_data yang bisa dieksekusi dengan beberapa nilai seperti misalnya kolom, row dan lain-lain. Bisa dilihat di bagian alur manipulasi data. Tutorial :

1. Silahkan melanjutkan contoh file diatas
2. Masukkan endpoint dataset/jumlah
3. Terapkan method Get : dengan fungsi jumlah dimana di dalamnya terdapat argument untuk parameter from (row dan column)
4. Penjelasan Parameter From :

- Column :Akan mengambil dan menampilkan jumlah kolom sesuai dengan kolom yang ada pada file CSV yang dieksekusi.
 - Row :Akan mengambil dan menampilkan jumlah baris sesuai dengan baris yang ada pada file CSV yang dieksekusi.
5. Kemudian pada argument/parameter from tersebut, fungsi jumlah_data dijadikan sebagai respon
 6. Fungsi jumlah_data akan menghitung berapa banyak jumlah baris dan kolom pada data (tertentu, sesuai dengan parameternya yang di get)
 7. Kemudian apabila pengeksekusian tanpa dibarengi dengan argument/parameter maka akan menampilkan jumlah keseluruhan.
- HASIL PENGUJIAN (Fungsi Dataset/jumlah , method: GET):
1. Menampilkan jumlah data secara keseluruhan (/dataset/jumlah) seperti pada gambar 7.9.

```
i 1 {"row": 3866, "column": 1}
```

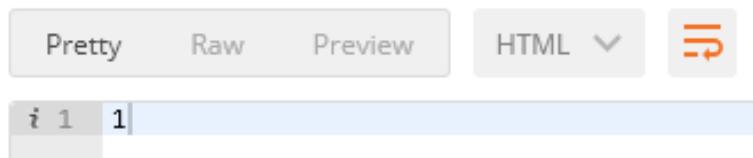
Gambar 7.9 Hasil Pengujian 1 fungsi dataset/jumlah (GET)

2. Menampilkan jumlah data row (/dataset/jumlah?from=row) seperti pada gambar 7.10.

```
i 1 3866
```

Gambar 7.10 Hasil Pengujian 2 fungsi dataset/jumlah (GET)

3. Menampilkan jumlah data row (/dataset/jumlah?from=column) seperti pada gambar 7.11.



Gambar 7.11 Hasil Pengujian 3 fungsi dataset/jumlah (GET)

7.3.3 Endpoint 3 (URL API : /dataset/<id>)

Pembahasan selanjutnya yaitu tutorial untuk pembuatan endpoint ketiga (/dataset/{id}). Silahkan perhatikan dan ikuti file seperti pada listing 7.24:

```
1 @app.route('/info', methods=['GET'])
2 def info():
3     return str(get_info(get_all_data(dataframe)))
```

Listing 7.24 Method Get dari Endpoint : URL API /dataset/<id>

Pada listing 7.24 terdapat fungsi detail. Di dalam fungsi tersebut dipergunakan untuk pengeksekusian data memakai fungsi yang diperlukan penentuan index id. Jadi kebanyakan dari hasil fungsi yang dipanggil dan dihubungkan kedalam fungsi detail. Tutorial :

1. Silahkan melanjutkan contoh file diatas
2. Masukkan endpoint dataset/id
3. Terapkan method Get : dengan fungsi jumlah dimana di dalamnya terdapat argument untuk parameter field (raw_value, data, sign) dan format (json)
4. Penjelasan Parameter :
 - All :Menampilkan data sesuai index id yang dipanggil tanpa parameter tertentu.
 - Field :Menampilkan dan mengambil data dari kolom tertentu dan disesuaikan dengan index id yang telah dipanggil
 - Format :Menampilkan dan mengambil data dengan format tulisan JSON dan RAW sesuai dengan index id yang dipanggil.
5. Kemudian pada argument/parameter field tersebut, fungsi get_row_field dijadikan sebagai respon
6. Fungsi get_row_field akan mengambil data dari row tersebut sesuai dengan field yang dimasukkan
7. Selanjutnya pada argument/parameter format, fungsi jsonify(get_all_data.json) dijadikan sebagai respon

8. Fungsi get_all_data.json akan mengambil data dari id tersebut dan ditampilkan dengan format json string.
 9. Kemudian reload_data akan menampilkan data terbaru setelah dilakukan perintah delete.
 10. Setelah semuanya selesai, maka tutorial untuk tahapan ini juga berhasil.
- HASIL PENGUJIAN (Fungsi Dataset/*id*, method: GET):
1. Menampilkan data berdasarkan id (/dataset/*id*) seperti pada gambar 7.12.

i	1	raw_value	Nan
	2	data	-99.0
	3	sign	Nan
	4	Name:	109, dtype: float64

Gambar 7.12 Hasil Pengujian 1 fungsi dataset/<id> (GET)

2. Menampilkan data sesuai id dengan format json (/dataset/1?format=json) seperti pada gambar 7.13.

```

1 ▾ { |
2   "data": -93,
3   "raw_value": null,
4   "sign": null
5 }

```

Gambar 7.13 Hasil Pengujian 2 fungsi dataset/<id> (GET)

3. Menampilkan data sesuai id dengan format raw (/dataset?format=raw) seperti pada gambar 7.14.

Gambar 9.14. Hasil pengujian 3 fungsi dataset/<id> (GET)

```
{  
    "data": -93.0,  
    "raw_value": null,  
    "sign": null  
}
```

Gambar 7.14 Hasil Pengujian 3 fungsi dataset/<id> (GET)

7.3.4 Endpoint 4 (URL API : /info)

Pembahasan selanjutnya ialah untuk endpoint terakhir yaitu endpoint info. Silahkan perhatikan file dibawah kemudian silahkan anda ikuti seperti pada listing 7.25:

```
1 @app.route('/info', methods=['GET'])  
2 def info():  
3     return str(get_info(get_all_data(dataframe)))
```

Listing 7.25 Method Get dari Endpoint : URL API /dataset/info

Pada file ini cukup sederhana fungsinya yaitu hanya untuk mendapatkan data berupa deskripsi bawaan dari library (bentuknya akan berbeda dari GET data sebelumnya)

Tutorial :

1. Silahkan melanjutkan contoh file diatas
2. Masukkan endpoint info
3. Terapkan method Get : dengan fungsi info diatas
4. Setelah itu silahkan eksekusi menggunakan CMD maka hasilnya akan sebagai berikut :
 - HASIL PENGUJIAN (Fungsi info, method: GET):

Menampilkan data berdasarkan info yang disimpan library pandas (/info) seperti pada gambar 7.15.

The screenshot displays a JSON response with the following structure:

i	1			-129
2	count	3864	.000000	
3	mean	55	.260093	
4	std	824	.002806	
5	min	-558	.000000	
6	25%	-104	.000000	
7	50%	20	.000000	
8	75%	90	.000000	
9	max	12336	.000000	

Gambar 7.15 Hasil Pengujian 1 fungsi dataset/info (GET)

7.4 Mendapatkan Parameter GET Python Flask

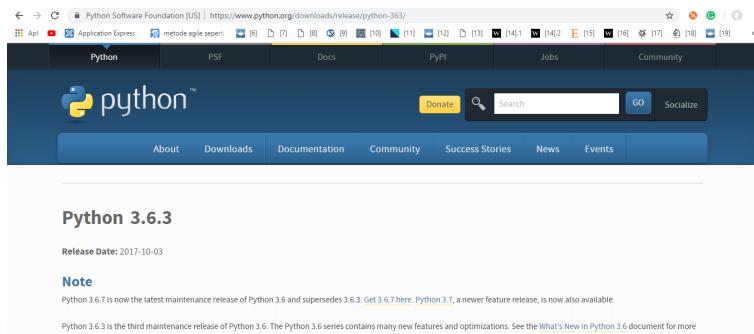
7.4.1 Penjelasan Parameter GET Python Flask

Parameter GET pada Python Flask ini saya lampirkan dan ujikan dalam bentuk file penuh dengan beberapa fungsi. File tersebut bernama Main.py. Untuk penerapan lebih dan contoh GETnya sudah saya tampilkan dan jelaskan sebelumnya pada point contoh URL GET. Namun, penggabungannya bersama Flask Python ada pada file ini. Silahkan diperhatikan penjelasan dan tutorialnya dan semoga dapat dimengerti.

Namun sebelum melanjutkan tutorial silahkan pertama-tama kita harus memastikan beberapa hal yaitu :

1. Persiapan Python : Instalasi Python 3.6 Instalasi Python dapat ditemukan di penjelasan sebelumnya (Sub Menu 1). Namun apabila masih belum paham dapat mengikuti langkah-langkah berikut:

- Pertama-tama silahkan download software dari python versi 3 di laptop anda.
- Download python versi 3.6.3 dari situs web resminya yaitu <https://www.python.org/>
- Silahkan sesuaikan dengan kapasitas laptop anda, bisa yang win 32 atau yang win 64 (32 bit / 64 bit)
- Contoh downloadnya seperti pada gambar 7.16.



Gambar 7.16 Instalasi Python

- Setelah berhasil melakukan pengunduhan/download aplikasi python tersebut, maka silahkan lakukan instalasi
- Instalasi dapat dilakukan seperti instalasi biasa pada umumnya
- Setelah selesai instalasi python, silahkan check di Command Prompt, apakah Pythonnya telah terbaca / running disesuaikan dengan laptop anda atau belum.
- Contoh pengecekan di Command Prompt seperti pada gambar 7.17.

```
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

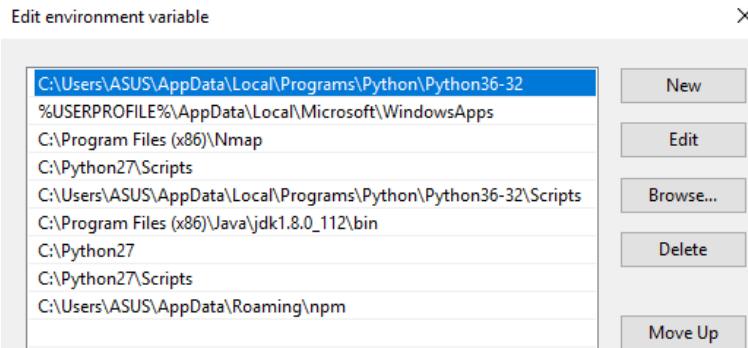
C:\Users\ASUS>python
ImportError: No module named site

C:\Users\ASUS>python3
Python 3.6.3 (v3.6.3:2f5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.

>>>
```

Gambar 7.17 Python CMD

- Apabila tampilannya telah sesuai dengan contoh diatas, maka python anda siap digunakan.
- Pastikan python yang terbaca versi 3.6.3 atau bahkan belum ada sama sekali silahkan lakukan konfigurasi ini :
 - Silahkan buka Control Panel anda
 - Pilih System and Security
 - Kemudian pilih lagi system
 - Lalu di bagian kiri tampilan ada sub menu Advanced system setting
 - Pada sub menu tersebut silahkan pilih button Environment Variabel
 - Silahkan ganti path dengan C:/Python36 dan C:/Python36/Scripts (Lokasi anda menyimpan mentahan python yang telah anda install tadi)
 - Maka tampilannya akan seperti pada gambar 7.18.

**Gambar 7.18** Device Manager (Env)

- Jangan lupa untuk memasukkan script dari python sehingga benar-benar bisa terbaca untuk pipnya.
- Silahkan klik button ok sampai selesai

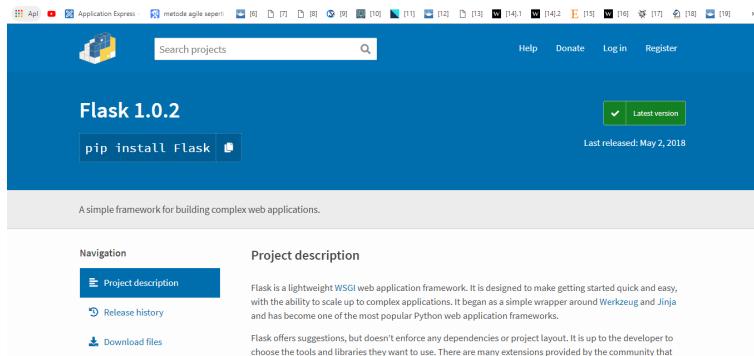
- Setelah itu, lakukan pengecekan ulang di Command Prompt maka hasilnya akan berubah menjadi versi 3.6.3
- Setelah semua tahap di atas selesai, maka silahkan lanjutkan ke tahap berikutnya

2. Persiapan Flask

Tutorial selanjutnya ialah kita akan menginstall Framework Flask di komputer/laptop kita sehingga bisa digunakan untuk tutorial selanjutnya. Untuk teman-teman ketahui flask sendiri merupakan microframework dari python, dengan penggunaannya, aktivitas apapun yang kita lakukan baik pengolahan data dll akan terasa lebih mudah dan rapih, kurang lebih seperti itu.

Instalasi Flask dapat ditemukan di penjelasan sebelumnya (Sub Menu 7). Namun apabila masih belum paham dapat mengikuti langkah-langkah berikut :

- Pertama-tama silahkan nyalakan laptop / komputer anda
- Kemudian apabila komputer / laptop anda telah bisa digunakan silahkan buka web browser
- Web browser yang digunakan bisa Chrome. Mozilla Firefox dll sesuai dengan keinginan anda.
- Selanjutnya pada web browser silahkan kunjungi website resmi berikut : <https://pypi.org/project/Flask/>
- Pada website tersebut silahkan download Flask
- Contohnya nampak seperti pada gambar 7.19.



Gambar 7.19 Instalasi Flask

- Setelah di download silahkan buka Command Prompt di komputer/laptop anda
- Kemudian silahkan ketikkan perintah (pip install flask)
- Setelah mengetikkan perintah tersebut, silahkan tekan enter maka prosesnya akan belajar

- Silahkan tunggu hasilnya
- Hasilnya akan nampak seperti pada gambar 7.20.

```
C:\Users\admin\AppData\Local\Programs\Python\Python36\Scripts>pip install flask
Collecting flask
  Using cached Flask-0.12.2-py2.py3-none-any.whl
Collecting itsdangerous>=0.21 (from flask)
  Using cached itsdangerous-0.24.tar.gz
Collecting Werkzeug>0.7 (from flask)
  Using cached Werkzeug-0.12.2-py2.py3-none-any.whl
Collecting Jinja2>=2.4 (from flask)
  Using cached Jinja2-2.9.6-py2.py3-none-any.whl
Collecting Click>=2.0 (from flask)
  Using cached Click-6.7-py2.py3-none-any.whl
Collecting MarkupSafe>=0.23 (from Jinja2>=2.4->flask)
  Using cached MarkupSafe-1.0.tar.gz
Installing collected packages: itsdangerous, Werkzeug, MarkupSafe, Jinja2, click, flask
  Running setup.py install for itsdangerous ... done
  Running setup.py install for MarkupSafe ... done
Successfully installed Jinja2-2.9.6 MarkupSafe-1.0 Werkzeug-0.12.2 click-6.7 flask-0.12.2 itsdangerous-0.24
```

Gambar 7.20 Install Flask di CMD

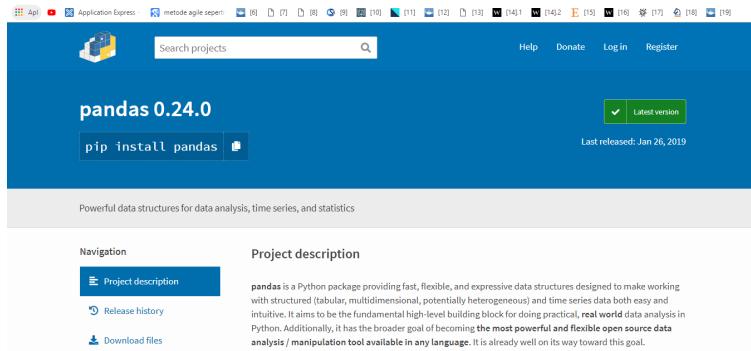
- Apabila tampilan pada proses di komputer/laptop anda telah nampak seperti gambar yang dicontohkan maka prosesnya berhasil
- Dan apabila masih terjadi error maka silahkan lakukan kembali langkah-langkahnya, bisa saja anda melewatkannya beberapa point pada tutorial ini
- Setelah flasknya terpasang di komputer/laptop anda maka silahkan lanjutkan ke tutorial selanjutnya.

3. Persiapan Library Pandas (untuk menyesuaikan dengan file CSV yang dieksplorasi)

Tutorial selanjutnya ialah kita akan menginstall Library Pandas di komputer/laptop kita sehingga bisa digunakan untuk tutorial selanjutnya. Untuk teman-teman ketahui pandas sendiri merupakan library dari bahasa pemrograman Python. Library ini digunakan untuk pemrosesan data analitik. Mengapa kita gunakan? Karena memang pandas ini akan mengolah data analitik dari CSV file yang berisikan data sinyal gelombang otak yang kita baca dan tangkap dari aktifitas tertentu (lampu sein saat bermotor). Nah mari kita mulai:

Instalasi Pandas dapat ditemukan di penjelasan sebelumnya (Sub Menu 7). Namun apabila masih belum paham dapat mengikuti langkah-langkah berikut :

- Pertama-tama silahkan nyalakan laptop / komputer anda
- Kemudian apabila komputer / laptop anda telah bisa digunakan silahkan buka web browser
- Web browser yang digunakan bisa Chrome. Mozilla Firefox dll sesuai dengan keinginan anda.
- Selanjutnya pada web browser silahkan kunjungi website resmi berikut : <https://pypi.org/project/pandas/>
- Pada website tersebut silahkan download Pandas
- Contohnya nampak seperti pada gambar 7.21.



Gambar 7.21 Instalasi Pandas

- Setelah di download silahkan buka Command Prompt di komputer/laptop anda
- Kemudian silahkan ketikkan perintah (pip install pandas)
- Setelah mengetikkan perintah tersebut, silahkan tekan enter maka prosesnya akan belajar
- Silahkan tunggu hasilnya
- Hasilnya akan nampak seperti pada gambar 8.15.

```
C:\Windows\system32>pip install pandas
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/7b/5c/51ad4b4f431b5735b239
6da44d44bd6f624bd209e0be46311710162b/pandas-0.24.0-cp36-cp36m-win_amd64.whl
(8.7MB)
    100% : [██████████] 8.8MB 867kB/s
Collecting numpy<1.12.0,>=1.11.0 (from pandas)
  Downloading https://files.pythonhosted.org/packages/31/7e/8905636f7e4f9b9d7078
aa0e79150634f832f145855a11beb099d3b0fb1/numpy-1.16.0-cp36-cp36m-win_amd64.whl
(11.9MB)
    100% : [██████████] 11.9MB 506kB/s
Collecting pytz>=2011k (from pandas)
  Using cached https://files.pythonhosted.org/packages/61/28/1d3920e4d1d50b19bc5
d24398a7cd85cc7b9a75a499579d53057522d34/pytz-2018.9-py2.py3-none-any.whl
Collecting python-dateutil>=2.5.0 (from pandas)
  Using cached https://files.pythonhosted.org/packages/74/68/48749b36af36f44254a
8d512cbfc48369103a3b9e474be9bdfe536abfc45/python_dateutil-2.7.5-py2.py3-none-any
.whl
Collecting six>=1.5 (from python-dateutil>=2.5.0->pandas)
  Using cached https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecf
e898238ce23f502a221c0ac0ecfedb0e0d88c64e9/six-1.12.0-py2.py3-none-any.whl
Installing collected packages: numpy, pytz, six, python-dateutil, pandas
Successfully installed numpy-1.16.0 pandas-0.24.0 python-dateutil-2.7.5 pytz-201
8.9 six-1.12.0
```

Gambar 7.22 Install Pandas di CMD

- Apabila tampilan pada proses di komputer/laptop anda telah nampak seperti gambar yang dicontohkan maka prosesnya berhasil
- Dan apabila masih terjadi error maka silahkan lakukan kembali langkah-langkahnya, mungkin anda melewatkannya beberapa point pada tutorial ini
- Setelah pandasnya terpasang di komputer/laptop anda maka silahkan lanjutkan ke tutorial selanjutnya

Filenya yang akan dijadikan tutorial yaitu seperti pada listing 7.26:

```
1 from flask import Flask , request , jsonify
2 import pandas as pd
3
4 from load_data import load_data
5 from get_all_data import get_all_data
6 from get_column import get_column
7 from get_top_five import get_top_five
8 from sorting import sorting
9 from convert_json import convert_json
10 from insert_data import insert_data
11 from get_info import get_info
12 from get_row import get_row
13 from get_row_field import get_row_field
14 from get_row_json import get_row_json
15 from get_all_data_json import get_all_data_json
16 from jumlah_data import jumlah_data
17
18 app = Flask(__name__)
19
20 # deklarasi nama file csv
21 filename = 'Book1.csv'
22 dataframe = load_data(filename)
23 dataframe.index += 1
24
25 # mengupdate data ketika ada perubahan
26 def reload_data(fname , dframe):
27     global dataframe
28     dframe.to_csv(fname , index=False , sep=':')
29     dataframe = load_data(fname)
30
31 # endpoint localhost:5000/dataset
32 # endpoint merupakan istilah untuk penyebutan URL API
33 @app.route('/dataset' , methods=['GET'])
34 def dataset():
35
36 # endpoint localhost:5000/jumlah
37 @app.route('/dataset/jumlah' , methods=['GET'])
38 def jumlah():
39     args = request.args
40     res = None
41     if not len(args) is 0:
42         if 'from' in args:
43             res = jumlah_data(dataframe , args['from'])
44         else:
45
46 # endpoint localhost:5000/jumlah
47 # endpoint localhost:5000/dataset/id id berupa angka
48 @app.route('/dataset/<int:id>' , methods=['GET'])
49 def detail(id):
50     args = request.args
51     detailframe = get_row(get_all_data(dataframe) , id)
52     if request.method == 'GET':
53         if not len(args) is 0:
54             if 'field' in args:
55                 detailframe = get_row_field(detailframe , args['field']
56 )
```

```

56         if 'format' in args:
57             if args['format'] == 'json' and 'field' not in args:
58                 detailframe = jsonify(get_row_json(detailframe))
59             else:
60                 return jsonify(detailframe)
61             else:
62                 detailframe = str(detailframe)
63             else:
64                 detailframe = str(detailframe)
65             return detailframe
66
67 @app.route('/info', methods=['GET'])
68 def info():
69     return str(get_info(get_all_data(dataframe)))
70
71 if __name__ == "__main__":
72     app.run(debug=True)

```

Listing 7.26 File Main.py Python Flask

Penjelasan Lengkap :

1. Penjelasan I : Main.py

```

1 from flask import Flask, request, jsonify
2 import pandas as pd
3
4 from load_data import load_data
5 from get_all_data import get_all_data
6 from get_column import get_column
7 from get_top_five import get_top_five
8 from sorting import sorting
9 from convert_json import convert_json
10 from insert_data import insert_data
11 from get_info import get_info
12 from get_row import get_row
13 from get_row_field import get_row_field
14 from get_row_json import get_row_json
15 from get_all_data_json import get_all_data_json
16 from jumlah_data import jumlah_data

```

Listing 7.27 Penjelasan I : Main.py

Listing 7.27 merupakan codingan untuk mendefinisikan framework flask, beberapa module yang digunakan untuk mengeksekusi fungsi seperti request, jsonify dan juga ada library pandas yang digunakan untuk pengeksekusian data analitik seperti file csv yang digunakan. Pada code juga dilakukan pemanggilan file-file yang di dalamnya telah terdapat masing-masing fungsi yang telah dijelaskan sebelumnya sehingga semuanya dapat dieksekusi dalam 1 framework Flask Python. Langkah-langkah :

- Silahkan buka sublime anda
- Kemudian buatlah file baru dengan nama Main.py
- Kemudian silahkan masukkan perintah import

- Perintah import apa saja? Silahkan mengimportkan perintah/file sesuai dengan contoh pada gambar

2. Penjelasan II : Main.py

```

1 app = Flask(__name__)
2 filename = 'Book1.csv'
3 dataframe = load_data(filename)
4 dataframe.index += 1

```

Listing 7.28 Penjelasan II : Main.py

Listing 7.28 dijelaskan bahwa file yang dieksekusi ialah Book1.csv kemudian variabel dataframanya akan memanggil fungsi load_data yang di dalamnya juga didefinisikan variabel filename. Indexnya sendiri ialah 1. Untuk fungsi reload_data memiliki variabel fname dan dfname .

- Silahkan lanjutkan pada file main.py
- Kemudian masukkan perintah diatas
- Variabel Filenam disesuaikan dengan file yang akan diolah (misal Book1.csv)
- Kemudian jangan lupa masukkan variabel dataframe (tentunya untuk pengolahan data secara keseluruhan)
- Lalu, difungsikan juga reload_data (mengupdate data ketika ada perubahan)
- Jangan lupa untuk menambahkan dataframe.to_csv sehingga parameternya akan dipisahkan oleh perintah (sep = “ : ”) dengan index=false
- Selanjutnya silahkan return variabel dataframe

3. Penjelasan III : Main.py

Penjelasan ini terdiri dari penjelasan untuk setiap endpoint dari Method GETnya.

- Endpoint 1 (URL API : /dataset): Telah dijelaskan pada point contoh URL Get. Halaman 7-14
- Endpoint 2 (URL API : /dataset/jumlah): Telah dijelaskan pada point contoh URL Get. Halaman 7-14
- Endpoint 3 (URL API : /dataset/ $_id$): Telah dijelaskan pada point contoh URL Get. Halaman 7-14
- Endpoint 4 (URL API : /info): Telah dijelaskan pada point contoh URL Get. Halaman 7-14

4. Penjelasan IV : Main.py

```

1 if __name__ == "__main__":
2     app.run(debug=True)

```

Listing 7.29 Penjelasan IV : Main.py

Listing 7.29 difungsikan untuk mengeksekusi inputan yang telah dimasukkan kedalam flask yang menjadi body dari code tersebut yaitu Main.py . Kemudian ketika dijalankan hasilnya true maka aplikasinya akan jalan sesuai dengan yang seharusnya

7.5 Penanganan Error

7.5.1 Error 1

1. Silahkan jalankan kembali CMD anda
2. Kemudian masuk ke Directory tempat anda menyimpan file main.py
3. Kemudian masukkan perintah berikut : Env/scripts/activate
4. Maksud dari Code tersebut ialah untuk mengaktifkan scriptnya python3
5. Maka hasilnya akan nampak seperti pada gambar 7.23.

```
D:\Buat Praktek\done>env\scripts\activate
(env) D:\Buat Praktek\done>python3 main.py
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 201-749-500
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 7.23 Aktivasi main.py 2 pada contoh kasus 3

6. Apabila hasilnya seperti gambar diatas maka filenya telah berjalan dengan baik
7. Bisa dilihat, untuk URL APInya sudah muncul jadi silahkan copy dan buktikan menggunakan POSTMAN atau browser biasa.
8. Hasilnya seperti berikut, apabila dieksekusi sesuai endpoint.
9. Misalnya endpoint : dataset/|id|, seperti pada gambar 7.24.
10. Kemudian apabila muncul ERROR seperti pada gambar 7.25.
11. Maka tindakannya seperti berikut:
 - Perhatikan terlebih dahulu request yang diminta
 - Request yang diminta yaitu field dengan parameter kolom sign
 - Silahkan diperiksa dulu apakah pada file yang ingin di get, kolom tersebut masih ada.

Pretty Raw Preview HTML

i	1	raw_value	data
2	0	raw_value	-129.0
3	1	raw_value	-129.0
4	2	raw_value	-93.0
5	3	raw_value	-120.0
6	4	raw_value	-120.0
7	5	raw_value	-120.0
8	6	raw_value	-120.0
9	7	raw_value	-120.0
10	8	raw_value	-120.0
11	9	raw_value	-120.0
12	10	raw_value	-120.0
13	11	raw_value	-120.0
14	12	raw_value	-120.0
15	13	raw_value	-267.0
16	14	raw_value	-250.0
17	15	raw_value	-250.0
18	16	raw_value	-250.0

Gambar 7.24 Hasil pengujian 1 untuk error 9 pada contoh kasus 3

GET http://127.0.0.1:5000/dataset?field=sign| Params Send Save Code

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (5) Test Results Status: 500 INTERNAL SERVER ERROR Time: 281 ms

Pretty Raw Preview HTML

```

1 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"
2 "http://www.w3.org/TR/html4/loose.dtd">
3 
4 <html>
5   <head>
6     <title>KeyError: &quot;None of [["sign"]] are in the [columns]&quot; // Werkzeug Debugger</title>
7     <link rel="stylesheet" href="https://debugger._yes&cmd=resource&f=style.css">
8   <!-- We need to make sure this has a favicon so that the debugger does
9 not by accident trigger a request to /favicon.ico which might
10 change the application state. -->
11 <link rel="icon" href="https://debugger._yes&cmd=resource&f=console.png">
12 <script src="https://debugger._yes&cmd=resource&f=jquery.js"></script>
13 <script src="https://debugger._yes&cmd=resource&f=debugger.js"></script>
14 <script type="text/javascript">
15 <var TRACEBACK = 145575440,
16 DEBUGGER = true,
17 EVALEX = true,
18 EVALEX_TRUSTED = false,
19 >
```

Gambar 7.25 Error 1 untuk error 9 pada contoh kasus 3

- Apabila tidak ada, maka silahkan masukkan file baru yang ada kolom sign-nya
- Kemudian silahkan jalankan kembali requestnya
- Maka hasilnya akan nampak seperti pada gambar 7.26.

PrettyRawPreview

i	1	sign
2	1	0
3	2	0
4	3	0
5	4	0
6	5	0
7	6	0
8	7	0
9	8	0
10	9	0
11	10	0
12	11	0
13	12	0
14	13	0
15	14	0
16	15	0
17	16	0
18	17	0
19	18	0
20	19	0

Gambar 7.26 Hasil pengujian 2 untuk error 9 pada contoh kasus 3

7.5.2 Error 2

1. Silahkan jalankan kembali CMD anda
2. Kemudian masuk ke Directory tempat anda menyimpan file main.py
3. Kemudian masukkan perintah berikut : Env/scripts/activate
4. Maksud dari Code tersebut ialah untuk mengaktifkan scriptnya python3
5. Maka hasilnya akan nampak seperti pada gambar 7.27.

```
D:\Buat Praktek\done>env\scripts\activate
(env) D:\Buat Praktek\done>python3 main.py
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 201-749-500
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 7.27 Aktivasi main.py 3 pada contoh kasus 3

6. Apabila hasilnya seperti gambar diatas maka filenya telah berjalan dengan baik
7. Bisa dilihat, untuk URL APInya sudah muncul. Silahkan copy dan buktikan menggunakan POSTMAN atau browser biasa.
8. Hasilnya seperti berikut, apabila dieksekusi sesuai endpoint.
9. Misalnya endpoint : dataset seperti pada gambar 7.28.
10. Kemudian apabila muncul ERROR seperti pada gambar 7.29.
11. Maka tindakannya seperti berikut :
 - Perhatikan terlebih dahulu request yang diminta
 - Request yang diminta yaitu format
 - Silahkan diperiksa kembali apakah pada codingan API nya terdapat request global untuk parameter Format
 - Ada, namun harus disertai dengan parameter yang lain
 - Seperti ini : ?format=json
 - Ketika requestnya seperti itu, maka hasilnya akan muncul
 - Silahkan jalankan kembali requestnya
 - Maka hasilnya akan nampak seperti pada gambar 7.30.

Pretty Raw Preview HTML ↗

i	1	raw_value	data
2	0	raw_value	-129.0
3	1	raw_value	-129.0
4	2	raw_value	-93.0
5	3	raw_value	-120.0
6	4	raw_value	-120.0
7	5	raw_value	-120.0
8	6	raw_value	-120.0
9	7	raw_value	-120.0
10	8	raw_value	-120.0
11	9	raw_value	-120.0
12	10	raw_value	-120.0
13	11	raw_value	-120.0
14	12	raw_value	-120.0
15	13	raw_value	-267.0
16	14	raw_value	-250.0
17	15	raw_value	-250.0
18	16	raw_value	-250.0

Gambar 7.28 Hasil pengujian 1 untuk error 10 pada contoh kasus 3

GET http://127.0.0.1:5000/dataset?form=

Authorization Headers (1) Body Pre-request Script Tests Code

Type No Auth

Body Cookies Headers (5) Test Results Status: 500 INTERNAL SERVER ERROR Time: 106 ms

Pretty Raw Preview HTML ↗

```

1 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 4.01 Transitional//EN"
2 <html>
3   <head>
4     <title>TypeError: 'DataFrame' object is not callable
5   The view function did not return a valid response. The return type must be a string, tuple, Response instance, or WSGI callable, but it was a
6   DataFrame.
7   Detailed information about the error:
8     <link rel="stylesheet" href="7_>.debugger._yes&cmd=resource&f=style.css"
9     type="text/css">
10    <!-- We need to make sure this has a favicon so that the debugger does
11    not have to trigger a request to /favicon.ico which might
12    change the application state. -->
13    <link rel="shortcut icon" href="7_>.debugger._yes&cmd=resource&f=console.png"
14    type="image/png">
15    <script src="7_>.debugger._yes&cmd=resource&f=jquery.js"></script>
16    <script src="7_>.debugger._yes&cmd=resource&f=debugger.js"></script>
17    <script src="7_>.debugger._yes&cmd=resource&f=eval.js"></script>
18    var TRACEBACK = 1460332,
19      CONSOLE_NODE = false,
20      EVALEX = true,
21      EVALEX_TRUSTED = false,
```

Gambar 7.29 Error 1 untuk error 10 pada contoh kasus 3

The screenshot shows the Postman application interface. At the top, there is a header bar with 'GET' and the URL 'http://127.0.0.1:5000/dataset?format=json'. Below the header, there are tabs for 'Authorization', 'Headers (1)', 'Body', 'Pre-request Script', and 'Tests'. The 'Body' tab is selected, showing a 'Type' dropdown set to 'No Auth'. Under the 'Body' tab, there are tabs for 'Pretty', 'Raw', 'Preview', and 'JSON'. The 'JSON' tab is selected, displaying a JSON array with four elements. Each element is an object with three properties: 'data', 'raw_value', and 'sign'. The values for 'data' are -129, -129, -93, and -120 respectively. The 'raw_value' and 'sign' fields are both set to 'raw_value' and '0' respectively.

```
[{"1": {"data": -129, "raw_value": "raw_value", "sign": "0"}, {"2": {"data": -129, "raw_value": "raw_value", "sign": "0"}, {"3": {"data": -93, "raw_value": "raw_value", "sign": "0"}, {"4": {"data": -120, "raw_value": "raw_value", "sign": "0"}}]
```

Gambar 7.30 Hasil pengujian 2 untuk error 10 pada contoh kasus 3

7.5.3 Error 3

1. Silahkan jalankan kembali CMD anda
2. Kemudian masuk ke Directory tempat anda menyimpan file main.py
3. Kemudian masukkan perintah berikut : Env/scripts/activate
4. Maksud dari Code tersebut ialah untuk mengaktifkan scriptnya python3
5. Maka hasilnya akan nampak seperti pada gambar 7.31.

```
D:\Buat Praktek\done>env\scripts\activate
(env) D:\Buat Praktek\done>python3 main.py
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 201-749-500
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 7.31 Aktivasi main.py 4 pada contoh kasus 3

6. Apabila hasilnya seperti gambar diatas maka filenya telah berjalan dengan baik
7. Bisa dilihat, untuk URL APInya sudah muncul jadi silahkan copy dan buktikan menggunakan POSTMAN atau browser biasa.
8. Hasilnya seperti berikut, apabila dieksekusi sesuai endpoint.
9. Misalnya endpoint : dataset/*id*; seperti pada gambar 7.32.
10. Kemudian apabila muncul ERROR seperti pada gambar 7.33.
11. Maka tindakannya seperti berikut :
 - Perhatikan terlebih dahulu request yang diminta
 - Request yang diminta yaitu id 4000
 - Silahkan diperiksa kembali apakah pada filenya ada id 4000 yang bisa diubah
 - Tidak ada, maka dari itu silahkan coba diganti dengan id yang terdapat dalam file tersebut
 - Seperti ini : dataset/2000
 - Ketika requestnya seperti itu maka hasilnya akan muncul
 - Silahkan jalankan kembali requestnya
 - Maka hasilnya akan nampak seperti pada gambar 7.34.
12. Maka id yang ke POST atau Update yaitu id 2000 dengan isi yang dapat diliat seperti pada gambar diatas.

http http http http http http http http http

GET http://127.0.0.1:5000/dataset/1

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (4) Test Results

Pretty Raw Preview HTML

```
i 1 raw_value      raw_value
2   data          -129
3   sign          0
4   Name: 1, dtype: object
```

Gambar 7.32 Hasil pengujian 1 untuk error 11 pada contoh kasus 3

POST http://127.0.0.1:5000/dataset/4000

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

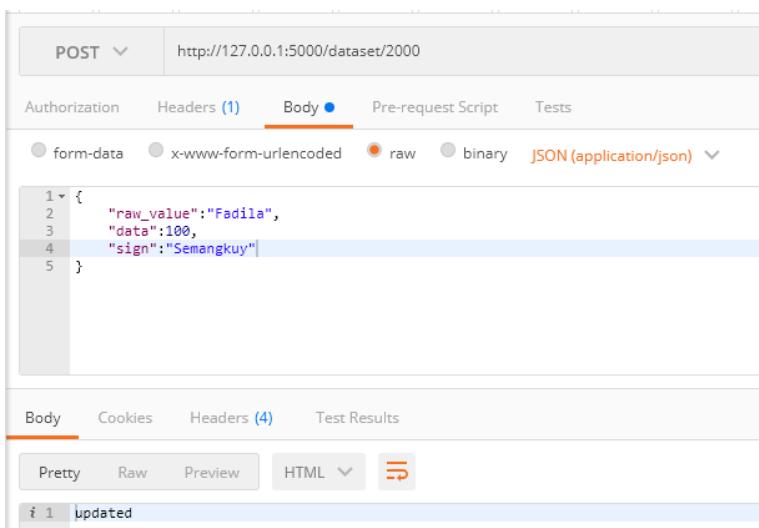
```
1 < {
2   "raw_value": "Fadila",
3   "data": 100,
4   "sign": "Semangkuy"
5 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview HTML

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 4.01 Transitional//EN"
<http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>KeyError: 'the label [4000] is not in the [index]' // Werkzeug Debugger</title>
<link rel="stylesheet" href="?__debugger__=yes&cmd=resource&f=style.css"
type="text/css">
<!-- We need to make sure this has a favicon so that the debugger does
not by accident trigger a request to /favicon.ico which might
change the application state. -->
```

Gambar 7.33 Error 1 untuk error 11 pada contoh kasus 3



Gambar 7.34 Hasil pengujian 2 untuk error 11 pada contoh kasus 3

BAB 8

HTTP POST, PUT, DELETE

8.1 Definisi HTTP Post Method

POST merupakan metode permintaan dari HTTP yang digunakan untuk mengirim data ke server untuk membuat / memperbarui sumber daya. Biasanya metode POST digunakan untuk menggugah file atau mengirimkan formulir web yang sudah diisi.

Metode POST digunakan untuk mengirimkan entitas ke sumber daya yang ditentukan, sering menyebabkan perubahan status atau efek samping pada server. Data akan dikirim ulang (browser harus memberi tahu pengguna bahwa data akan dikirim ulang). Metode POST tidak dapat di bookmark juga tidak terdapat cache. Jika menggunakan metode POST parameternya tidak akan tersimpan di browser history, ini membuat metode POST lebih aman untuk digunakan karena, informasi penting..rahasia tidak dapat diakses secara ilegal.

8.2 Mekanisme HTTP Post Method

Metode POST mentransfer informasi melalui HTTP. Informasi dikodekan dan dimasukkan ke dalam header yang disebut QUERY_STRING. Data untuk metode POST

sama seperti yang dikirimkan melalui formulir HTTP. Kemudian Agen internal mengekstrak bidang dari setiap formulir dan setiap aliran Post ke tabel aliran data. Namun data tidak akan ditampilkan pada header.

HTTP POST dapat terjadi sebagai hasil dari pengguna yang mengirimkan formulir HTML yang berisi bidang `<INPUT>`, atau POST dapat dibuat dengan menggunakan HTTP sebagai protokol komunikasinya.

8.3 Contoh URL HTTP Post Method

```

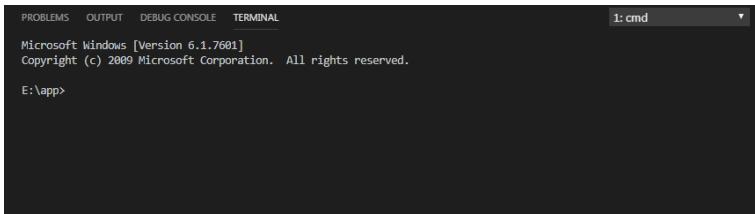
1 @app.route('/') ## Mendefinisikan URL
2 @app.route('/index', methods=['GET', 'POST'])
3 def show_index(): #membuat fungsi show_index
4     full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'brainwave.
      png') ##mengambil dan menampilkan gambar yang telah dipilih untuk
      ditampilkan
5     return render_template("matplotlib.html", user_image = full_filename)
      ## memanggil template html yang digunakan sebagai wadah untuk
      menampilkan gambar dari sintak sebelumnya pada web browser

```

Listing 8.1 Skrip Fungsi Contoh HTTP POST

Listing 8.1 merupakan contoh dari HTTP POST pada flask. Route disini berfungsi untuk menautkan URL ke suatu fungsi yang telah didefinisikan. `Show_index()` merupakan fungsi yang digunakan untuk menampilkan gambar kedalam template HTML yang dapat diakses lewat browser. Cara menjalankannya seperti berikut :

1. Buka terminal pada text editor yang dimiliki pada PC ataupun menggunakan CMD. Pada tutorial ini menggunakan terminal pada text editor Visual Code seperti pada gambar 8.1.



Gambar 8.1 Terminal Pada Visual Code

2. Buka Folder dimana file codingan tadi tersimpan seperti pada gambar 8.2.
3. Buka Folder tadi Di Terminal dengan mengetikan “cd E:/app” maka akan masuk ke folder tersebut.
4. Kemudian untuk menjalankan aplikasinya ketikan “python app.py” app.py merupakan file codingan python yang berisikan Skrip Fungsi diatas. Maka ketika dijalankan akan muncul hasil seperti pada gambar 8.3.

Computer > Engineering (E) > app >				
	Name	Date modified	Type	Size
	static	19/02/2019 22:27	File folder	
ads	templates	19/02/2019 22:27	File folder	
places	app	20/02/2019 3:02	PY File	3 KB
	cilok	19/01/2019 21:29	WinRAR archive	283 KB
	cilok	26/01/2019 15:46	PY File	1 KB
	coba	25/01/2019 13:31	Compiled Python ...	1 KB
nts	coba	06/01/2019 14:30	PY File	1 KB
ads	cobamatplotlib	26/01/2019 17:40	PY File	5 KB
	core	13/01/2019 12:13	Microsoft Excel C...	9 KB
	main	06/01/2019 14:24	PY File	1 KB
	main1	25/01/2019 13:28	PY File	1 KB
	mindwave	06/01/2019 14:24	PY File	11 KB
up	nyobian	20/02/2019 3:21	PY File	2 KB

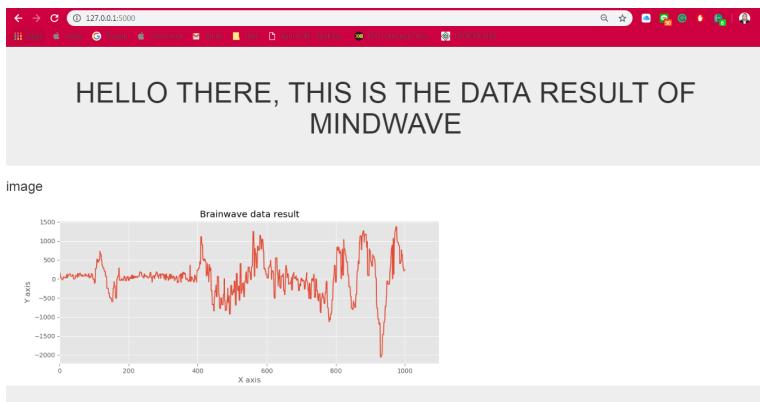
Gambar 8.2 Folder Lokasi Codingan

```
E:\app>python app.py
=====
0:00:00.001 - Start Program
=====

Wiendhyra, Tasya
Syarifatul, Nurgivani
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 8.3 Menjalankan Aplikasi

5. Terdapat link dari hasil menjalankan tadi, kemudian salin link tersebut ke web browser anda. Maka hasilnya akan seperti pada gambar 8.4.



Gambar 8.4 Contoh HTTP POST

8.4 Definisi HTTP Put

Metode permintaan HTTP PUT membuat resource baru atau menggantikan representasi resource target dengan muatan permintaan. Perbedaan antara PUT dan POST adalah bahwa PUT idempoten: menyebutnya sekali atau beberapa kali berturut-turut memiliki efek yang sama (yaitu tidak berpengaruh), di mana POST identik yang berurutan dapat memiliki efek tambahan, seperti mengirimkan pesan beberapa kali.

8.5 Mekanisme HTTP Put Method

Cara kerja dari PUT ini hampir sama dengan POST. Namun ketika pada saat mengirimkan request dan jika resource target tidak memiliki representasi saat ini dan permintaan PUT berhasil membuatnya, maka server asal harus memberi tahu agen pengguna dengan mengirimkan respons 201 (Created). Dan apabila request yang dikirimkan sudah ada akan dilakukan update data.

8.6 Contoh URL HTTP Put Method

```

1 @app.route('/lang/<string:name>', methods=['PUT'])
2 def editSatu(name):
3     langs = [language for language in languages if language['name'] ==
4             name]
5     langs[0]['name'] = request.json['name']

```

```
5 return jsonify({ 'language' : langs[0] })
```

Listing 8.2 Skrip Fungsi Contoh HTTP PUT

Listing 8.2 merupakan contoh dari HTTP PUT yang digunakan untuk mengupdate hasil dari fungsi yang telah didefinisikan menggunakan metode GET sebelumnya. Fungsinya seperti pada listing 8.3:

```
1 @app.route('/lang/<string:name>', methods=['GET'])
2 def Satu(name):
3     langs = [languages for language in languages if language['name']
4             == name]
5     return jsonify({ 'language' : langs[0] })
```

Listing 8.3 Skrip Fungsi Pada HTTP GET

8.7 Definisi HTTP Delete

Sesuai dengan namanya yaitu delete yang berarti menghapus resource yang dituju. Metode DELETE meminta server asal menghapus sumber yang diidentifikasi oleh Request-URI. Metode ini dapat ditimpa oleh intervensi manusia (atau cara lain) pada server asal. Klien tidak dapat menjamin bahwa operasi telah dilakukan, bahkan jika kode status dikembalikan dari server asal menunjukkan bahwa tindakan telah berhasil diselesaikan.

8.8 Mekanisme HTTP Delete Method

Ketika dilakukan request dengan menggunakan metode DELETE, maka Request-URI akan melakukan identifikasi dan meminta untuk menghapus sumber yang dituju. DELETE. Bisa saja terdapat respon pada saat penghapusan sumber/data dan bisa juga tidak ada respon namun data/sumber berhasil dihapus ketika dilakukan pengecekan.

8.9 Contoh URL HTTP Delete Method

```
1 @app.route('/lang/<string:name>', methods=['DELETE'])
2 def hapus(name):
3     langs = [languages for language in languages if language['name'] ==
4             name]
4     languages.remove(langs[0])
5     return jsonify({ 'languages' : languages })
```

Listing 8.4 Skrip Fungsi Pada HTTP DELETE

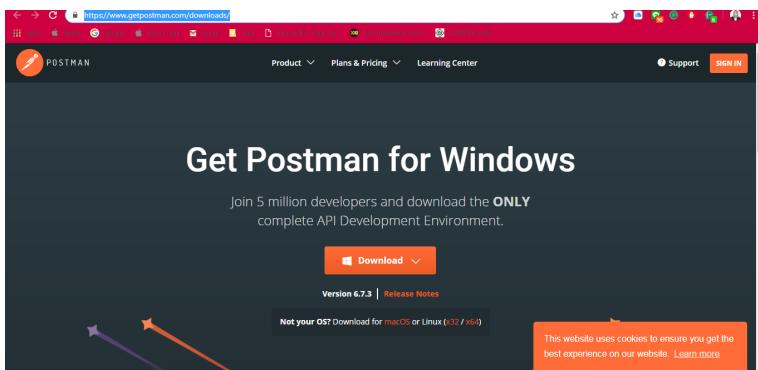
Listing 8.4 akan menghapus data yang terdapat pada fungsi yang sebelumnya.

8.10 Instalasi Dan Tata Cara Penggunaan Postman Untuk HTTP Post, Put Dan Delete

Postman merupakan klien HTTP yang digunakan untuk melakukan tes terhadap web service. Postman sendiri dapat diinstal langsung pada PC ataupun diinstal pada web browser yang mendukung. Kali ini akan dibuat sebuah tutorial cara menginstal Postman di PC serta pada web browser.

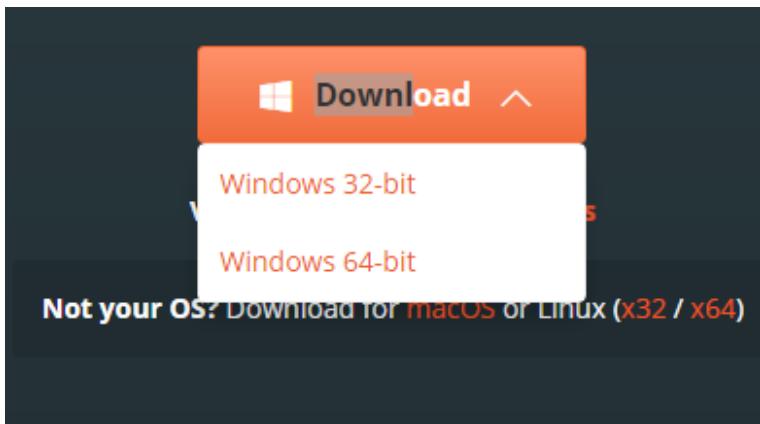
8.10.1 Instalasi Postman Di PC

1. Untuk menginstal Postman, berkasnya dapat diunduh di <https://www.getpostman.com/downloads>. unduh sesuai dengan Sistem Operasi yang anda gunakan, untuk tutorial ini menggunakan Windows seperti pada gambar 8.5.

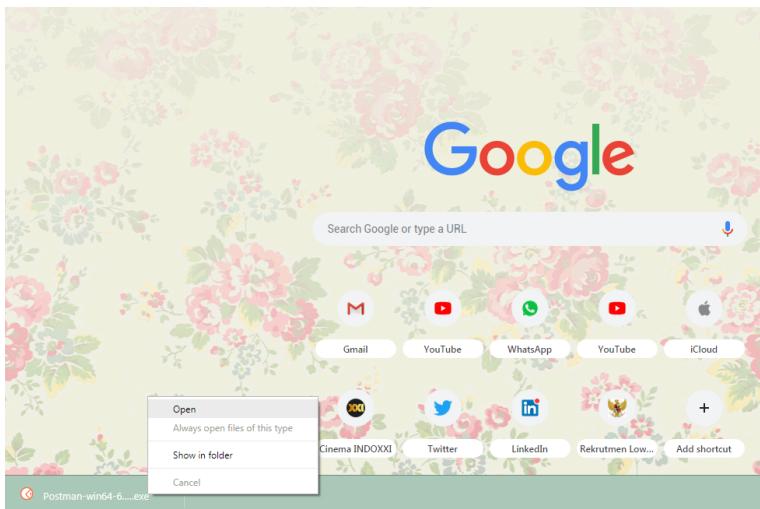


Gambar 8.5 Website Postman

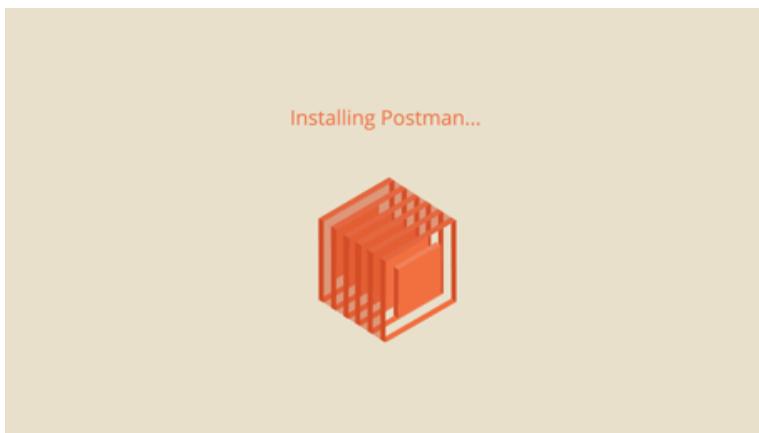
2. Unduh versi Windows 64 bit, karena pada tutorial ini menggunakan Windows 64 bit. Jika Anda memiliki Windows 32 bit, Anda dapat mengunduh Postman untuk Windows 64 bit seperti yang ditunjukkan seperti pada gambar 8.6.
3. Setelah diunduh, Anda dapat menemukan file yang diunduh di lokasi unduhan default sistem Anda. Jika Anda menggunakan browser Chrome, file yang diunduh akan muncul di bagian bawah browser seperti yang ditunjukkan seperti pada gambar 8.7.
4. Buka file exe Postman Windows 64 bit untuk instalasi pada sistem anda. Dan lakukan instalasi seperti pada gambar 8.8.
5. Tunggu sampai proses instalasi selesai.
6. Setelah proses instalasi selesai, akan diminta untuk membuat akun seperti pada gambar 8.9.



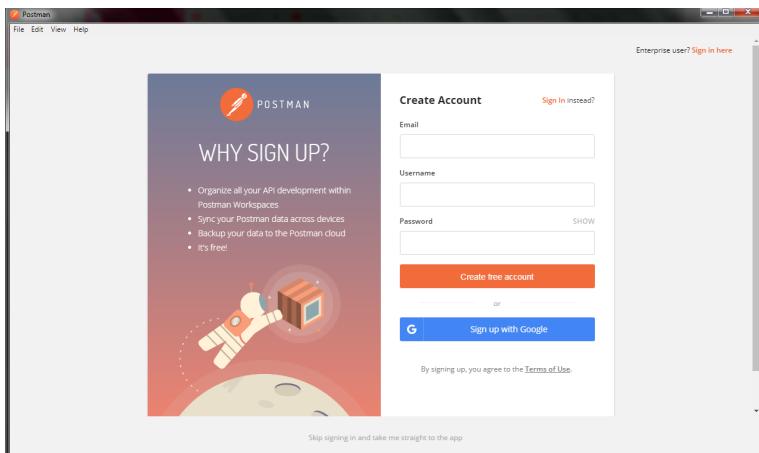
Gambar 8.6 Postman Versi Windows 64-bit



Gambar 8.7 Buka File Instalasi

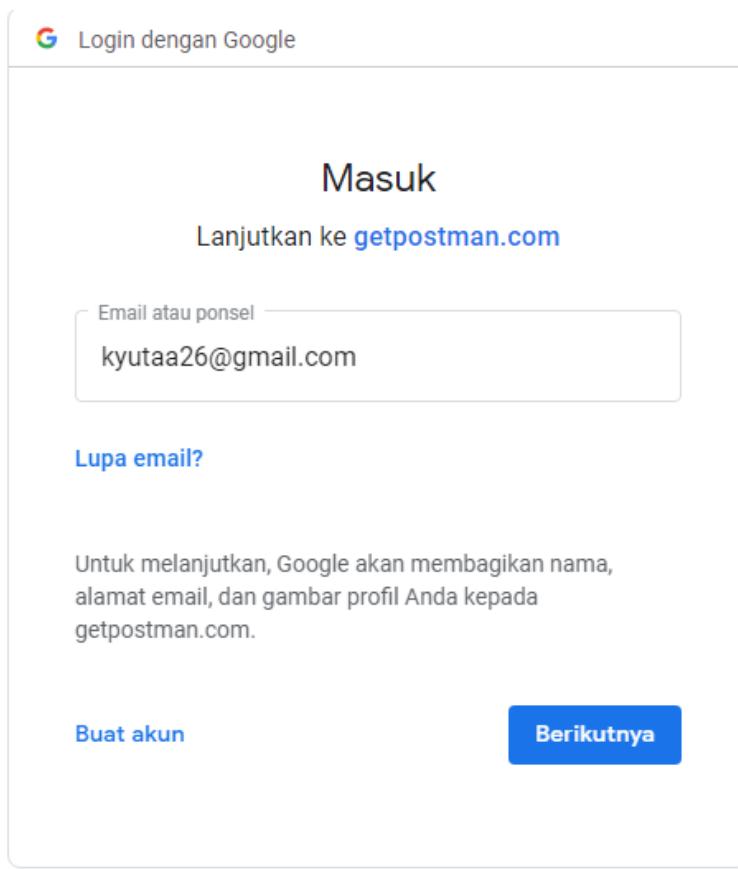


Gambar 8.8 Instalasi Postman



Gambar 8.9 Halaman Membuat Akun Postman

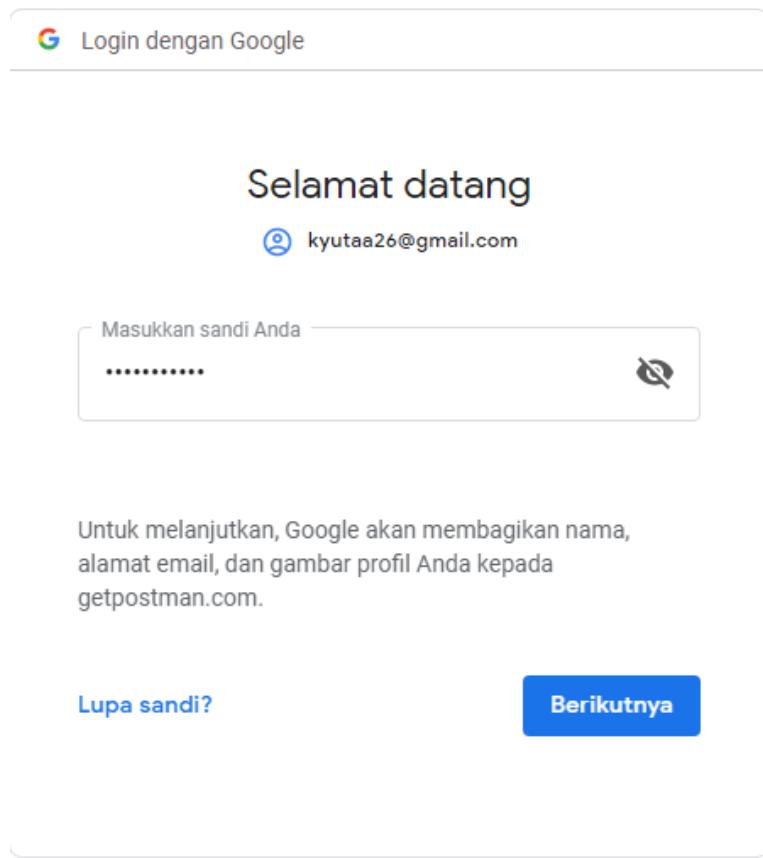
7. Pilihlah membuat akun dengan akun Google anda yaitu dengan klik tombol “Sign Up With Google”. Anda akan diminta untuk login ke akun google anda. Silahkan isi kolom email dan password sesuai dengan akun google yang anda miliki seperti pada gambar 8.10.



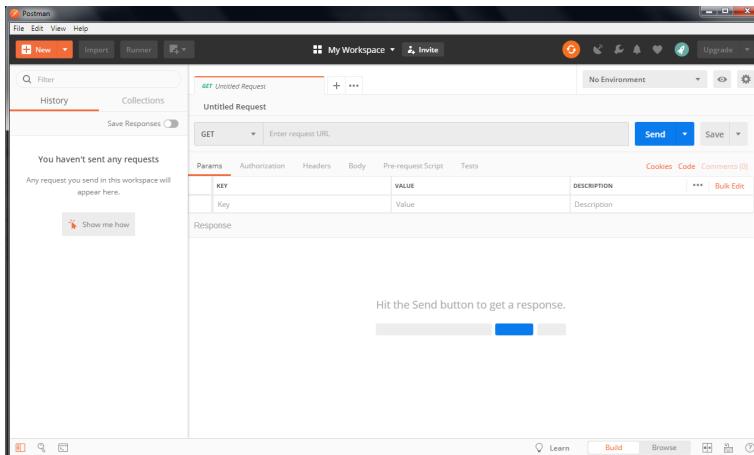
Gambar 8.10 Login Postman Dengan Akun Google

Password seperti pada gambar 8.11.

8. Klik berikutnya untuk menyelesaikan proses login. Ini akan secara otomatis membuka Alat Postman. Setelah instalasi dan pendaftaran berhasil (daftar dengan Gmail Anda), Anda bisa melihat toolkit Postman seperti pada gambar 8.12.



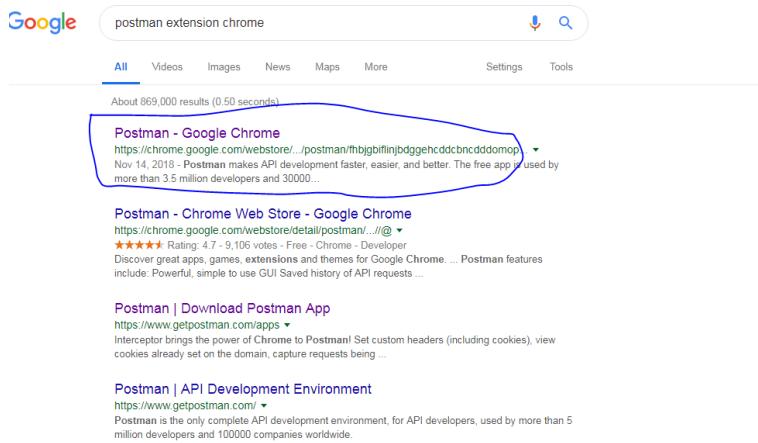
Gambar 8.11 Password Untuk Login Postman Dengan Akun Google



Gambar 8.12 Tampilan Postman

8.10.2 Instalasi Ekstensi Postman Pada Web Browser Chrome

1. Buka Web store di Chrome dan ketikan Postman. Atau anda dapat melakukan cara lain yaitu dengan melakukan pencarian dengan kata kunci “Postman extension chrome” seperti pada gambar 8.13.



Gambar 8.13 Pencarian Ekstensi Postman Untuk Chrome

2. Klik tautan yang dilingkari biru dan akan muncul halaman berikut untuk mengunduh file ekstensinya. Dan klik Add To Chrome untuk memulai pengunduhan seperti pada gambar 8.14.

[Home](#) > [Apps](#) > Postman



Postman

Offered by: www.getpostman.com

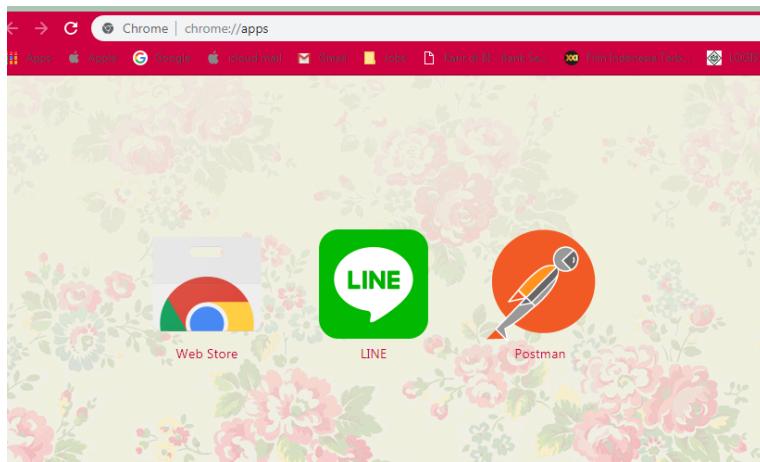
★★★★★ 9,100 | [Extensions](#) | 3,869,285 users

Runs offline

[Add to Chrome](#)

Gambar 8.14 Menambahkan Postman ke Ekstensi Chrome

- Setelah proses instalasi berhasil, anda akan diarahkan ke laman Apps pada Chrome dan terdapat Ikon Postman yang menandakan bahwa Postman telah terinstal seperti pada gambar 8.15.



Gambar 8.15 Ikon Postman Pada Laman App Chrome

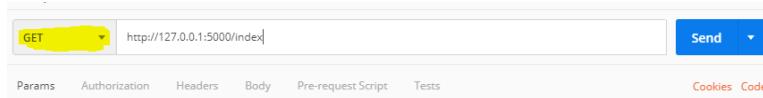
8.10.3 Cara Penggunaan Postman Untuk Pengujian HTTP POST,PUT, Dan DELETE

- Pada cmd, jalankan file dari aplikasi yang berisi metode POST,PUT, dan DELETE seperti pada gambar 8.16.
- Didapatkan Link yang akan berfungsi untuk menjalankan/ pengujian HTTP Methods yang telah didefinisikan tadi.
- Buka Postman lalu masukan URL request yang didapatkan tadi dengan menambahkan URL yang telah ditautkan pad fungsi yang telah didefinisikan seperti pada gambar 8.17.

```
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

::\app>python app.py
=====
:00:00.002 - Start Program
=====

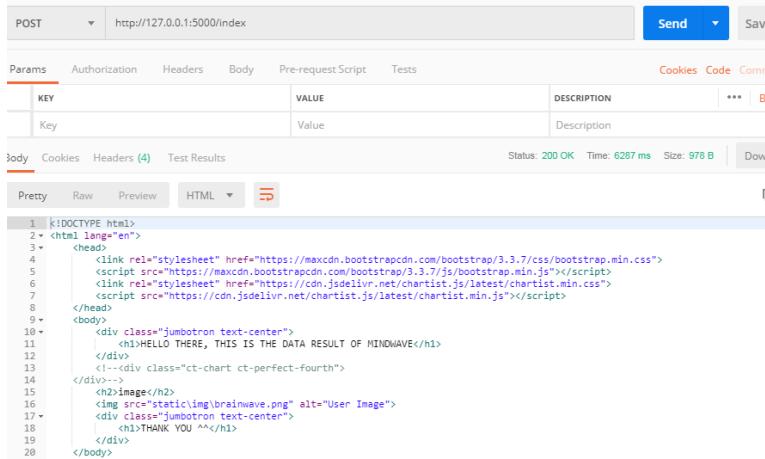
Hiendhyra, Tasya
yarifatul, Nurgivani
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
27.0.0.1 - - [20/Feb/2019 00:36:40] "GET /lang HTTP/1.1" 404 -
27.0.0.1 - - [20/Feb/2019 00:37:02] "GET /%20lang HTTP/1.1" 404 -
|
```

Gambar 8.16 Menjalankan Aplikasi Pada CMD**Gambar 8.17** Menambahkan URL Untuk POST Method

- Pada gambar yang diberi tanda highlight diubah sesuai dengan metode HTTP yang digunakan. Kemudian diubah menjadi POST untuk melakukan pengujian terhadap metode HTTP POST seperti pada gambar 8.18.

**Gambar 8.18** Mengubah Request Menjadi POST

- Kemudian klik send dan akan muncul seperti ini yang menandakan bahwa metodenya berjalan seperti pada gambar 8.19.
 - Klik Preview, maka akan tampil laman sesuai dengan yang ada pada web broser, namun gambarnya tidak akan tampil seperti pada gambar 8.20.
 - Sekarang akan dilakukan pengujian pada metode PUT. Yang dimana akan meng Update data.
 - Pertama – tama jalankan GET Request untuk mendapatkan data yang ingin di update seperti pada gambar 8.21.
- Hasilnya seperti pada gambar 8.22.

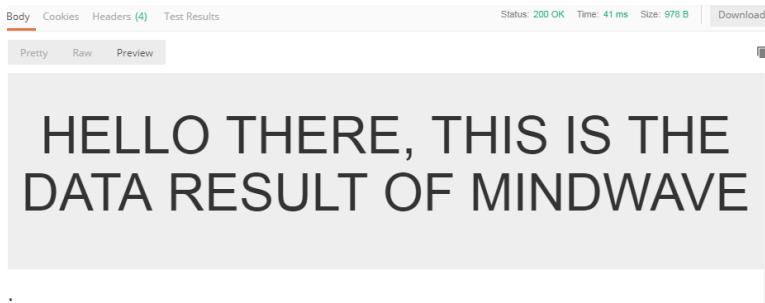


```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
5     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
6     <link rel="stylesheet" href="https://cdn.jsdelivr.net/chartist.js/latest/chartist.min.css">
7     <script src="https://cdn.jsdelivr.net/chartist.js/latest/chartist.min.js"></script>
8   </head>
9   <body>
10    <div class="jumbotron text-center">
11      <h1>HELLO THERE, THIS IS THE DATA RESULT OF MINDWAVE</h1>
12    </div>
13    <!--<div class="ct-chart ct-perfect-fourth">
14    </div>-->
15    <h2>image</h2>
16    
17    <div class="jumbotron text-center">
18      <h1>THANK YOU ^~^</h1>
19    </div>
20  </body>

```

Gambar 8.19 Hasil Pengujian Post Method



Gambar 8.20 Preview Pengujian Post Method



Gambar 8.21 Get Request

```

1 "languages": [
2   {
3     "name": "Contoh"
4   },
5   {
6     "name": "HTTP"
7   },
8   {
9     "name": "PUT"
10  }
11 ]
12
13

```

Gambar 8.22 Hasil Pengujian Get Request

9. Pada Step ini akan mengupdate data “Contoh” menjadi kata lain. Pada bagian URL tambahkan /Contoh yang artinya akan mengedit data tersebut, dan ubah Method Requestnya menjadi PUT. Namun jangan di klik Send seperti pada gambar 8.23.

Params	Authorization	Headers	Body	Pre-request Script	Tests	Cookies	Code
KEY			VALUE			DESCRIPTION	*
Key			Value			Description	

Body Cookies Headers (4) Test Results Status: 200 OK Time: 25 ms Size: 277 B

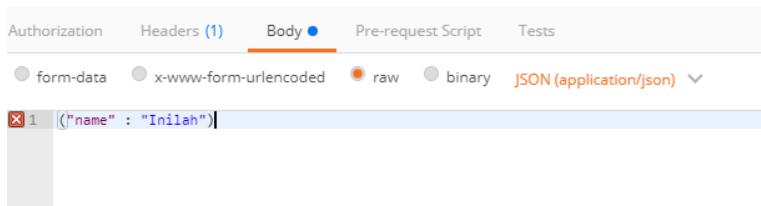
```

1 "languages": [
2   {
3     "name": "Contoh"
4   },
5   {
6     "name": "HTTP"
7   },
8   {
9     "name": "PUT"
10  }
11 ]
12
13

```

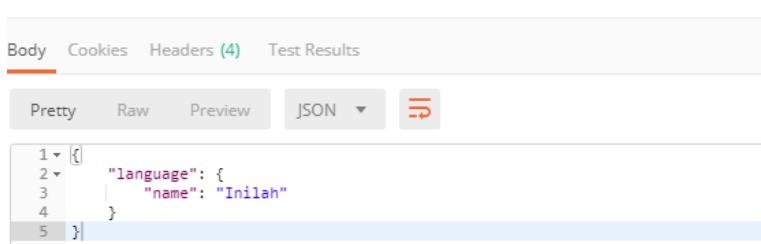
Gambar 8.23 Contoh Put Method

10. Pilih Body kemudian pilih Raw dan ketikan seperti dibawah ini. Dan setelah itu klik send. Pastikan Requestnya sudah diganti dengan JSON(application/json) seperti pada gambar 8.24.



Gambar 8.24 Edit Data Contoh

11. Apabila berhasil maka akan muncul seperti ini yang menandakan bahwa data berhasil di update seperti pada gambar 8.25.



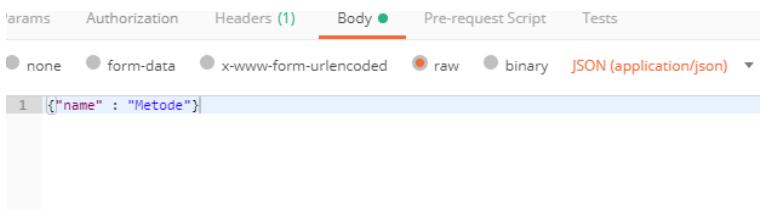
Gambar 8.25 Data Contoh Terupdate

12. Untuk memastikan apakah data benar benar ter update maka jalankan kembali URL <http://127.0.0.1:5000/lang> pada Postman dengan metode GET. Maka akan terlihat Data yang sebelumnya “Contoh” berubah menjadi “Inilah” seperti pada gambar 8.26.
13. Selanjutnya akan mencoba mengupdate data “HTTP” menjadi kata lain. Untuk itu Pilih Body kemudian pilih Raw dan ketikan seperti dibawah ini. Dan setelah itu klik send seperti pada gambar 8.27.
14. Apabila berhasil maka akan muncul seperti ini yang menandakan bahwa data berhasil di update seperti pada gambar 8.28.
15. Untuk memastikan apakah data benar benar ter update maka jalankan kembali URL <http://127.0.0.1:5000/lang> pada Postman dengan metode GET. Maka akan terlihat Data yang sebelumnya “HTTP” berubah menjadi “Metode” seperti pada gambar 8.29.
16. Berikutnya pengujian terhadap HTTP DELETE.

The screenshot shows the Postman interface with the 'Body' tab selected. The 'JSON' preview pane displays the following JSON data:

```
1 ✘ [
2 ✘   "languages": [
3 ✘     {
4 ✘       "name": "Inilah"
5 ✘     },
6 ✘     {
7 ✘       "name": "HTTP"
8 ✘     },
9 ✘     {
10      "name": "PUT"
11    }
12  ]
13 ]
```

Gambar 8.26 Data Contoh Yang Terupdate



Gambar 8.27 Edit Data HTTP

The screenshot shows the Postman interface with the 'Body' tab selected. The 'JSON' preview pane displays the following JSON data, indicating an update:

```
1 ✘ [
2 ✘   "language": {
3 ✘     "name": "Metode"
4 ✘   }
5 ✘ ]
```

Gambar 8.28 Data HTTP Terupdate

```

1  [
2   "languages": [
3     {
4       "name": "Contoh"
5     },
6     {
7       "name": "HTTP"
8     },
9     {
10      "name": "PUT"
11    }
12  ]
13 ]

```

Gambar 8.29 Data HTTP Yang Terupdate

17. Pada Postman ketikan URL `http://127.0.0.1:5000/lang/Inilah` . Artinya data Inilah dipilih untuk dihapus dari resource seperti pada gambar 8.30.

The screenshot shows the Postman interface with the following details:

- URL:** `http://127.0.0.1:5000/lang/Inilah`
- Method:** DELETE (not explicitly shown in the UI, but implied by the context)
- Headers:** None
- Body:** None
- Pre-request Script:** None
- Tests:** None
- Status:** 200 OK
- Preview:** Shows a JSON response with the key `languages` and its value `["Contoh"]`.
- JSON:** Selected view mode

Gambar 8.30 Mengubah URL Untuk HTTP Delete

18. Ubah metodenya menjadi DELETE seperti pada gambar 8.31.
19. Klik send, dan jika berhasil maka akan seperti ini seperti pada gambar 8.32.
20. Untuk memastikan data telah terhapus atau tidak. Lakukan Penghapusan data lagi. Dan akan mencoba menghapus data “HTTP”.
21. Pada Postman ketikan URL `http://127.0.0.1:5000/lang/HTTP` seperti pada gambar 8.33.

The screenshot shows the Postman interface with a DELETE request to the URL `http://127.0.0.1:5000/lang/Inilah`. The 'Params' tab is selected, showing a key-value pair where 'Key' is 'Value'. The 'Send' button is highlighted. Below the request, the status bar shows 'Status: 200 OK' and 'Time: 56 ms'. The 'Body' tab is also visible.

Gambar 8.31 Mengubah Metode Menjadi Delete

The screenshot shows the Postman interface with the 'Body' tab selected. The JSON response is displayed as follows:

```
1 [  
2   "languages": [  
3     {  
4       "name": "HTTP"  
5     },  
6     {  
7       "name": "PUT"  
8     }  
9   ]  
10 ]
```

Gambar 8.32 Data Inilah Terhapus

The screenshot shows the Postman interface with a DELETE request to the URL `http://127.0.0.1:5000/`. The 'Authorization' tab is selected. The status bar at the bottom shows 'Status: 200 OK'.

Gambar 8.33 Mengubah URL Untuk Delete Data HTTP

22. Klik send, dan jika berhasil maka akan seperti ini seperti pada gambar 8.34.

The screenshot shows the Postman interface with the following details:

- Body:** Contains the JSON data: {"languages": [{"name": "PUT"}]}
- Cookies:** None
- Headers:** (4) - This tab is selected, showing the following headers:

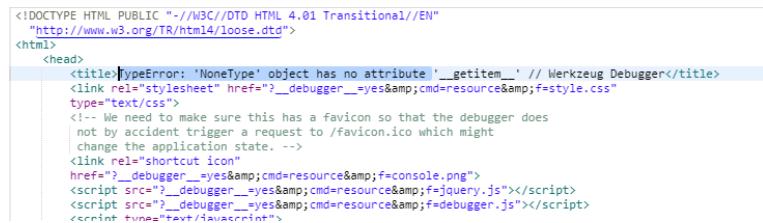
Content-Type	application/json
Content-Length	18
Date	Mon, 10 Dec 2018 10:10:30 GMT
Connection	keep-alive
- Test Results:** None
- JSON View:** Shows the JSON structure: 1 [2 {"languages": [3 { 4 "name": "PUT" 5 } 6] 7 }]
- Buttons:** Pretty, Raw, Preview, JSON, and a copy icon.

Gambar 8.34 Data HTTP Terhapus

8.11 Penanganan Error

8.11.1 TypeError: 'NoneType' Object Has No Attribute

Errornya seperti pada gambar 8.35.



The screenshot shows a browser error page with the following details:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>TypeError: 'NoneType' object has no attribute '_getitem_' // Werkzeug Debugger</title>
    <link rel="stylesheet" href=?_debugger_=yes&cmd=resource&f=style.css"
    type="text/css">
    <!-- We need to make sure this has a favicon so that the debugger does
        not by accident trigger a request to /favicon.ico which might
        change the application state. -->
    <link rel="shortcut icon"
    href=?_debugger_=yes&cmd=resource&f=console.png">
    <script src=?_debugger_=yes&cmd=resource&f=jquery.js"></script>
    <script src=?_debugger_=yes&cmd=resource&f=debugger.js"></script>
    <script type="text/javascript">

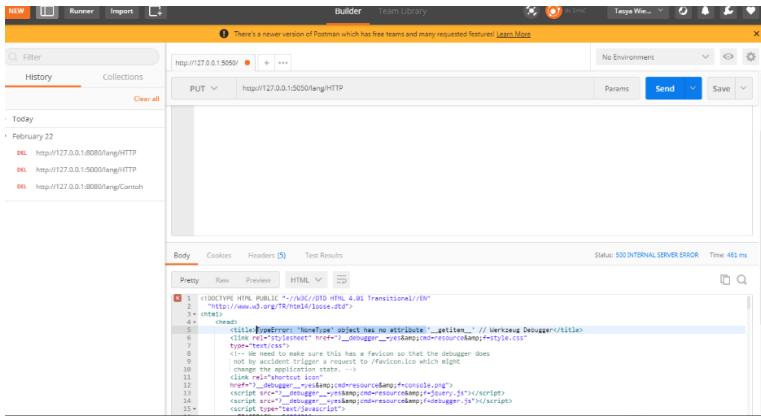
```

Gambar 8.35 TypeError: 'NoneType' Object Has No Attribute

Error ini biasanya terjadi pada saat mengedit Body untuk menambahkan atau memperbarui data. Eror ini menandakan bahwa NoneType berarti bahwa alih-alih sebuah instance dari Kelas atau Obyek apa pun yang Anda pikir sedang dikerjakan, sebenarnya tidak anda miliki satupun. Itu biasanya berarti bahwa tugas atau fungsi panggilan di atas gagal atau mengembalikan hasil yang tidak terduga.

Cara Mengatasi Error:

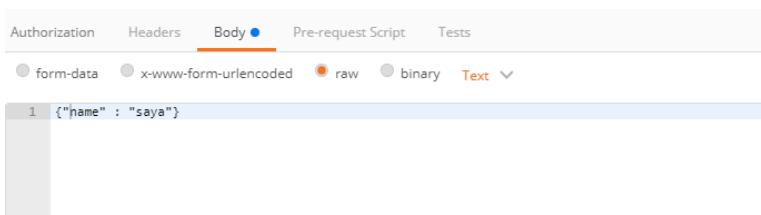
1. Buka Postman seperti pada gambar 8.36.
2. Ubah Objek metode menjadi ‘PUT’ dan pilih Data yang akan diupdate atau menambahkan data baru seperti pada gambar 8.37.
3. Setelah itu, pilih Body seperti pada gambar 8.38.



Gambar 8.36 Menjalankan Postman

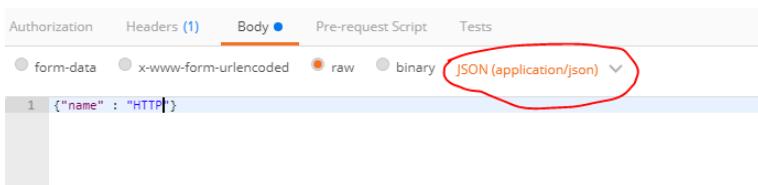


Gambar 8.37 Mengubah Metode



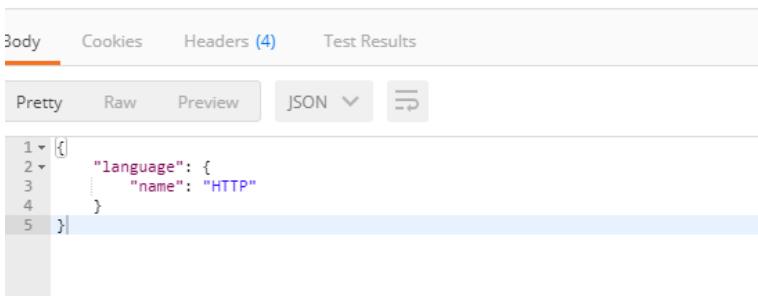
Gambar 8.38 Memilih Body Pada Postman

4. Dari gambar diatas, diketahui bahwa objek type nya masih ‘TEXT’. Maka dari itu klik pada bagian tersebut, dan ubah menjadi json, seperti pada gambar 8.39.



Gambar 8.39 Mengubah Object Type

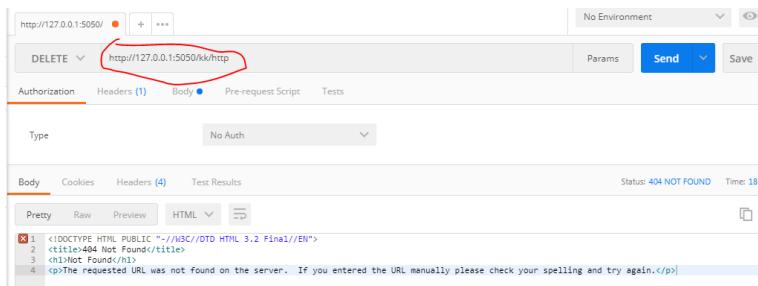
5. Ubah juga isi dari body seperti gambar diatas. Maka ketika dijalankan, data Saya akan berubah menjadi ‘HTTP’ seperti pada gambar 8.40.



Gambar 8.40 Pengujian

8.11.2 DELETE Error Code 404

Delete Error Code 404 seperti pada gambar 8.41.

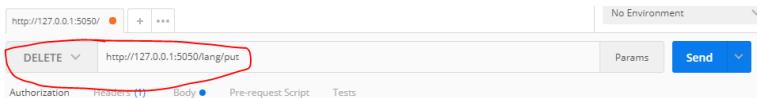


Gambar 8.41 DELETE Error Code 404

Pada tutorial ini menerima 404 karena / kk/ HTTP bukan URI yang menunjuk ke sumber daya di sistem. Idempotensi juga dipertahankan karena berapa kali Anda mengirim permintaan HTTP DELETE ini, perubahan tambahan ke kondisi server tidak akan terjadi karena sumber daya sudah dihapus. Jadi, permintaan HTTP DELETE tambahan tidak akan melakukan apa-apa.

Cara Mengatasi Error:

1. Untuk eror ini sangat mudah dalam penyelesaiannya. Cukup dengan mengubah URI ke sumber daya di sistem.
2. URI pada sistem tutorial ini seperti pada gambar 8.42.



Gambar 8.42 Mengubah URI Sesuai Dengan Sistem

3. Setelah diubah, klik Send. Maka data ‘PUT’ akan terhapus seperti pada gambar 8.43.

A screenshot of a REST client interface showing the "Body" tab selected. The "JSON" dropdown is set to "Pretty". The JSON data is displayed as follows:

```
1  {
2    "languages": [
3      {
4        "name": "Contoh"
5      },
6      {
7        "name": "PUT"
8      }
9    ]
10 }
```

The line "name": "Contoh" is highlighted with a blue selection bar.

Gambar 8.43 Data Sebelum Dihapus

Data setelah dihapus seperti pada gambar 8.44.

Type No Auth

Body Cookies Headers (4) Test Results

Pretty Raw Preview JSON ↻

```
1 | {}  
2 | "languages": [  
3 | | {  
4 | | | "name": "Contoh"  
5 | | }  
6 | ]  
7 | }
```

Gambar 8.44 Data Setelah Dihapus

BAB 9

JSON

9.1 Membuat Kembalian Web Service Dalam Bentuk JSON Pada HTTP GET, POST, PUT, dan DELETE

9.1.1 Definisi JSON

JavaScript Object Nation atau biasa dikenal dengan JSON ialah sebuah format untuk berbagi data. Sesuai namanya, JSON merupakan turunan dari JavaScript, tetapi format ini tersedia untuk beberapa bahasa pemrograman lain, di antaranya: Python, PHP, Ruby, dan Java. JSON biasanya dilafalkan menjadi ‘Jason’. JSON memiliki format file .json saat ia bersiri sendiri. Akan tetapi jika didefinisikan ke dalam format lain semisal .html, ia dapat ditampilkan dalam tanda petik sebagai JSON string, atau JSON ini juga bisa dimasukkan ke dalam sebuah variabel. Format ini sangat mudah untuk ditransfer antar server web dengan klien atau browser.

Karena sangat ringan dan mudah dibaca, JSON bisa memberikan alternatif lebih baik dari XML dan tidak membutuhkan formatting yang tidak banyak. Panduan ini dapat membantu pembaca untuk memahami apa itu JSON, bagaimana menggunakan data di file JSON, dan struktur serta sintaks dari format ini. JSON juga merupakan format yang ringan dan mempermudah segala sesuatu supaya dapat membagi, meny-

impar dan bekerja dengan data. Sebagi sebuah format, JSON telah mendapatkan dukungan yang makin meningkat dalam bentuk API, misalnya di Twitter API.

Karena mungkin akan jarang membuat file .json sendiri namun mengunduhnya dari sumber lain, pikirkan bagaimana menggunakan JSON dengan baik diprogram daripada memikirkan strukturnya.

9.1.2 Sintaks dan Struktur

Objek dari JSON adalah format data key-value yang biasanya dirender di dalam kurung kurawal. Saat mengerjakan file JSON, mungkin akan sering melihat objek JSON disimpan sebagai objek JSON atau string di dalam sebuah program. Contoh objek JSON seperti pada listing 9.1:

```

1 {
2   "first_name" : "Nurgivani",
3   "last_name" : "Husna",
4   "location" : "Bandung",
5   "online" : true,
6   "followers" : 1041
7 }
```

Listing 9.1 Contoh Objek JSON

Walaupun contoh di atas merupakan contoh singkat dan JSON dapat memiliki isi yang sangat banyak. Secara umum, contoh di atas menggambarkan dua kurung kurawal ({}) di awal dan akhir dengan pasangan key-value di antara kedua tanda kurang. Sebagian data yang dipakai di JSON dienkapsulasi di dalam sebuah objek JSON.

Yang di sebut Key JSON yaitu berada di sebelah kiri titik dua. Mereka perlu dilengkapi oleh tanda petik dua seperti ini: “key”. Key dapat berupa string apa pun dengan syarat string valid. Dalam sebuah objek, key harus bersifat unik. Key juga bisa berupa spasi seperti “first name”, namun disarankan jangan menambahkan spasi karena ini akan membuat repot saat menjalankan kode, lebih baik menggunakan underscore seperti ini: “first_name”.

1. Value JSON terletak di sebelah kanan setelah titik dua. Ada enam tipe data yang dapat dipakai, yaitu: Strings, numbers, objects, arrays, booleans (true/false), null.
2. Setiap data tipe yang dimasukkan sebagai value ke dalam JSON akan mengingat sintaksnya. Jadi string akan diberikan tanpa petik, namun tidak dengan angka.
3. JSON bukan hanya bisa ditulis secara beberapa baris, tetapi juga bisa ditulis dalam satu baris seperti pada listing 9.2:

```

1 {"first_name" : "Nurgivani", "last_name" : "Husna", "location" :
  "Bandung", "online" : true, "followers" : 1041}
```

Listing 9.2 Contoh Objek JSON

4. Satu hal yang perlu diingat, walaupun JSON bisa dimasuki fungsi JavaScript, tetapi saja tidak akan bisa menggunakannya sebagai value di JSON. JSON memiliki kelebihan akan kemudahan dan kesiapannya untuk ditransfer antar bahasa pemrograman. Dan satu hal lagi, objek JavaScript hanya bisa digunakan dalam bahasa pemrograman JavaScript saja.

9.1.3 Perbandingan Dengan XML

Extensible Markup Language atau biasa disebut XML adalah sebuah cara untuk menyimpan data yang dapat dibaca baik oleh manusia maupun mesin. Format XML tersedia secara luas bagi banyak bahasa pemrograman. XML hampir mirip dengan JSON, hanya saja lebih membutuhkan lebih banyak teks sehingga isinya panjang dan lama untuk dibaca serta ditulis. XML harus dibaca dengan XML parser, namun JSON bisa dibaca dengan fungsi standar. Layaknya JSON, XML tidak dapat menggunakan array.

JSON jauh lebih ringkas daripada XML. JSON juga tidak perlu ada tag penutup layaknya XML. Terakhir, XML sama sekali tidak menggunakan array. Jika pembaca sudah tahu tentang HTML, maka pembaca akan melihat kemiripan dari penggunaan tag-nya. Walaupun JSON lebih simpel dan mudah serta dapat digunakan diberbagai situasi termasuk aplikasi AJAX, tetapi tentunya tetap harus memahami proyek apa yang sedang dikerjakan sebelum menentukan struktur data apa yang akan digunakan.

9.1.4 Sumber-sumber Lain

JSON merupakan format natural untuk JavaScript dan memiliki banyak implementasi yang dapat langsung dipakai diberbagai bahasa pemrograman populer. Pembaca dapat melihat bahasa yang didukung pada situs “Introducing JSON”, dan jQuery library juga memiliki kemampuan untuk membaca format ini. Seringkali, tidak perlu menulis JSON utuh namun mengambilnya dari suatu sumber data atau mendapatnya dari mengubah jenis data lain ke JSON.

CSV dapat diubah atau data yang terpisah dengan tab ke dalam JSON menggunakan aplikasi open source Mr. Data Converter. XML juga dapat diubah ke JSON atau sebaliknya dengan aplikasi berlisensi Creative Commons ini utilities-online.info site. Saat mengubah tipe data lain ke JSON, atau membuat tipe data sendiri (dengan objek bersarang), anda dapat memvalidasi JSON dengan JSONLint dan dapat mengejek file JSON dalam konteks web development dengan JSFiddle.

9.1.5 RESTful Web Service

Representational State Transfer atau biasa disebut REST adalah arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Biasanya REST menggunakan Hypertext Transfer Protocol (HTTP) sebagai protokol untuk komunikasi data. REST pertama kali diperkenalkan tahun 2000 oleh Roy Fielding. REST server menyediakan sumber daya atau data (resource). REST client akan mengakses data tersebut dan menampilkan data untuk penggunaan se-

lanjutnya. Setiap resource diidentifikasi oleh Universal Resource Identifiers (URIs) atau biasa disebut Global ID. Resource tersebut direpresentasikan dalam format teks, JSON, atau XML. Biasanya format yang dipakai adalah JSON dan XML.

Web Service juga merupakan standar yang biasa digunakan untuk melakukan pertukaran data antar aplikasi atau sistem karena aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada platform yang berbeda. Contohnya adalah SOAP dan REST.

1. Keuntungan RESTful Web Service

Pada REST, bahasa dan platform agnostic. Juga, REST lebih sederhana untuk dikembangkan daripada SOAP. REST juga mudah dipelajari dan tidak bergantung pada tools. REST lebih ringkas dan tidak membutuhkan layer pertukaran pesan tambahan. Terakhir, desain dan filosofi REST lebih dekat dengan WEB.

2. Kekurangan RESTful Web Service

Kurangnya dukungan standar keamanan, kebijakan, keandalan pesan, dan banyak lagi, sehingga layanan yang mempunyai persyaratan lebih canggih dan lebih sulit untuk dikembangkan. REST berkaitan dengan model transport HTTP.

9.1.6 Metode HTTP yang biasa digunakan dalam arsitektur REST

GET, metode ini menyediakan akses baca pada resource. PUT, biasanya metode ini digunakan untuk membuat resource baru. DELETE, berfungsi untuk menghapus resource. POST, metode ini biasanya digunakan untuk memperbarui resource yang sudah ada atau membuat resource baru. OPTION, biasanya digunakan untuk mendapatkan operasi yang didukung pada resource.

9.1.7 Cara kerja RESTful Web Service

Client mengirimkan request melalui HTTP request dan kemudian server akan merespons melalui HTTP Response. Adapun komponen HTTP request ialah: Verb, HTTP method yang digunakan misalnya GET, POST, DELETE, PUT, dan yang lainnya, URI untuk mengidentifikasi lokasi resource pada server, HTTP version untuk menunjukkan versi dari HTTP yang digunakan (contoh: HTTP v1.1.), Request Header berisi metadata untuk HTTP Request (contohnya type client/browser, format yang didukung oleh client, format dari body pesan, setting cache, dll), Request Body yang merupakan konten dari data.

Adapun yang disebut dengan HTTP response ialah: Status atau response code yang berfungsi mengindikasikan status server terhadap resource yang di request (404, artinya resource tidak ditemukan. 200, response OK), sebagian besar hampir sama dengan HTTP request.

9.1.8 Cara implementasi JSON pada Web Service

Sebagai contoh, kami akan mengimplementasikannya pada Brainwave Signal Web Service yang di mana Web Service ini memiliki fungsi untuk menampilkan sinyal

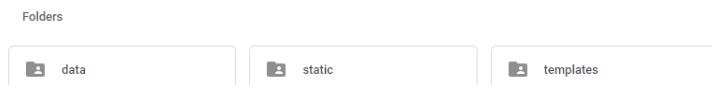
otak dari EEG menggunakan JSON sebagai template. Hal pertama yang harus dilakukan adalah membuat folder yang disimpan di c:/users/asus/, kemudian dalam folder asus ini dibuat lagi folder yang namanya bisa dimodifikasi sesuai keinginan. Untuk contoh, saya membuat folder Zroygiv. Di dalam folder Zroygiv, buat lagi folder example2 yang mana di dalamnya akan berisi folder lain.

1. Untuk lebih jelas mengenai folder, bisa dilihat seperti pada gambar 9.1.

This PC > OS (C:) > Users > asus > Zroygiv > example2				
	Name	Date modified	Type	Size
ss	app	15/02/2019 1:55	File folder	
js	Include	16/01/2019 13:50	File folder	
ts	Lib	16/01/2019 13:54	File folder	
	Scripts	16/01/2019 13:54	File folder	
	tcl	16/01/2019 13:56	File folder	

Gambar 9.1 Folder Users

2. Dalam folder app, terdapat dua folder yang bernama static dan template seperti pada gambar 9.2.



Gambar 9.2 Folder App

3. Kedua folder itu nantinya akan diisi oleh template dan juga data dari Brainwave Sinyal Web Service.
4. Di dalam folder app, buatlah file python yang bernama flasklivechart.py, lalu isikan file tersebut dengan kode seperti pada listing 9.3:

```
1 from flask import Flask, render_template, make_response,
2     send_file, url_for
3 import json, time
4 import Tkinter
5 import tkMessageBox
6 import atexit
7 from time import clock
8 import request
9 import os
10
11 app = Flask(__name__)
12
13
14 #
15 #
```

```

16 #
17 #
18 #
19
20 def img():
21     PEOPLE_FOLDER = os.path.join('static', 'img')
22     app.config['UPLOAD_FOLDER'] = PEOPLE_FOLDER
23
24 @app.route('/')
25 def show_index():
26     img()
27     full_filename = os.path.join(app.config['UPLOAD_FOLDER'],
28                                 'EEG.jpg') #nampilin gambar
29     return render_template("index.html", user_image =
30                           full_filename)
31
32 def popup():
33     tkMessageBox.showinfo("Information","Created in Python.")
34
35 @app.route('/live')
36     #dekorator flask , hubungan antara url
37 def live():
38     popup()
39     img()
40     full_filename = os.path.join(app.config['UPLOAD_FOLDER'],
41                                 'pict.jpg')
42     return render_template('live.html', user = full_filename)
43     #fungsi untuk merender file
44     #html didalam templates yang nanti akan dipanggil ke browser
45
46 dataset = [] #menampung data sementara
47
48 def myline():
49     f = open('tmp','r+')
50     mode read
51     line = f.readlines()
52     for x in line:
53         dataset.insert(0,x)
54     return
55
56 def fromDataset():
57     for x in dataset:
58         return int(x) #return dataset (convert
59         to integer)
60
61 @app.route('/live-data')
62 def live_data(): #fungsi
63     livedata
64     myline() #fungsi
65     myline dipanggil agar data refresh per/detik
66     data = [time.time() * 500, fromDataset()] #waktu
67     runing nampilin data yang di ambil
68     print time.ctime(time.time())
69     response = make_response(json.dumps(data))
70     response.content_type = 'application/json'
71     return response

```

```
61
62 @app.route('/live-tmp', methods=['GET'])
63 def livetmp():
64     f = open('tmp', 'r')                                     #buka
65     file tmp dengan mode read
66     line = f.readlines()                                    #baca
67     file per line
68     for x in line:                                         #loping
69         data perline
70         response = make_response(json.dumps(x))
71         response.content_type = 'application/json'
72     return response
73
74 @app.route('/data', defaults={'req_path':''}, methods=['GET'])
75 @app.route('/<path:req_path>')
76 def simpanjson(req_path):
77     BASE_DIR = '/Users/PC 10/Zroyek/example2/LIVE/data'      #
78     data_path = os.path.join(BASE_DIR, req_path)               #menyambungkan base dir dan path yang di request
79
80     if os.path.isfile(data_path):                             #
81         check path dan mengembalikan data
82         return send_file(data_path)
83     files = os.listdir(data_path)
84
85     return render_template('files.html', files=files)        #menampilkan konten dari direktori
86
87 #
88 #
89 #
90 #
91 def secondsToStr(t):#fungsi waktu
92     return "%d:%02d:%02d.%03d" % \
93         reduce(lambda ll,b : divmod(ll[0],b) + ll[1:], [ (t*1000,) , 1000,60,60])           #ngubah supaya formatnya H:MM:SS.SSS
94
95 def sapa(nama):
96     """Fungsi ini untuk menyapa seseorang sesuai nama yang dimasukkan sebagai parameter"""
97     print("Hi, " + nama + ". Apa kabar?")
98 sapa('teman')
99
100 line = "="*40
101 def log(s, elapsed=None):                                #menghitung waktu
102     yang berlalu
103     print line
104     print secondsToStr(clock()), '-', s                  # nampilin waktu
105     yg berlalu
106     if elapsed:
107         print "Elapsed time:", elapsed
108     print line
```

```

107     print
108
109 def date():
110     date = time.time
111     print time.ctime(time.time())
112
113
114 def endlog():
115     end = clock()
116     elapsed = end-start #ngitung waktu awal sampai akhir
117     log("End Program", secondsToStr(elapsed)) #nampilkan waktu
118     akhir saat program di hentikan
119     date()
120     print ('Proyek dua')
121
122 def now():
123     return secondsToStr(clock())
124
125
126 start = clock()
127 atexit.register(endlog)
128 log("Start Program") #nmpilin waktu
awal bngt
129
130 def print_info( nama, Npm ):
131     print ("Nama: ", nama)
132     print ("Npm: ", Npm)
133 print_info( Npm=1164035, nama = "Eko Cahyono Putro" )
134 print_info( Npm=1164049, nama = "Nur Arkhamia Batubara" )
135
136
137 #
138 #
139 #
140 #
141 #
142 if __name__ == '__main__':
143     app.run(debug=True, host='127.0.0.1', port=8080) #url
untuk masuk browser

```

Listing 9.3 Flask Live Chart

9.1.9 Sebelum membuat file JSON, buatlah folder js dan img di dalam folder static

Seperti pada gambar 9.3.

1. Dalam folder js, buatlah file yang bernama highcharts.js, kemudian isikan kode seperti pada listing 9.4:

```

1 var chart;
2
3 /**

```

Folders

**Gambar 9.3** Folder Static

```
4 * Request data from the server, add it to the graph and set a
5   timeout
6 * to request again
7 */
8 function requestData() {
9     $.ajax({
10         url: '/live-data',
11         success: function(point) {
12             var series = chart.series[0],
13                 shift = series.data.length > 70; // shift if the
14             series is
15                                         // longer than
16             // add the point
17             chart.series[0].addPoint(point, true, shift);
18             // call it again after one second
19             setTimeout(requestData, 700);
20         },
21         cache: false
22     });
23 }
24
25
26 $(document).ready(function() {
27     chart = new Highcharts.Chart({
28         chart: {
29             renderTo: 'data-container',
30             defaultSeriesType: 'line',
31             events: {
32                 load: requestData
33             }
34         },
35         title: {
36             text: 'Live data'
37         },
38         xAxis: {
39             type: 'datetime',
40             tickPixelInterval: 150,
41             maxZoom: 20 * 1000
42         },
43         yAxis: {
44             minPadding: 0.2,
```

```

45         maxPadding: 0.2 ,
46         title: {
47             text: 'Range',
48             margin: 10
49         }
50     },
51     plotOptions: {
52         line: {
53             dataLabels: {
54                 enabled: true
55             },
56             enableMouseTracking: false
57         }
58     },
59     series: [
60         {
61             name: 'Data',
62             data: []
63         }
64     });

```

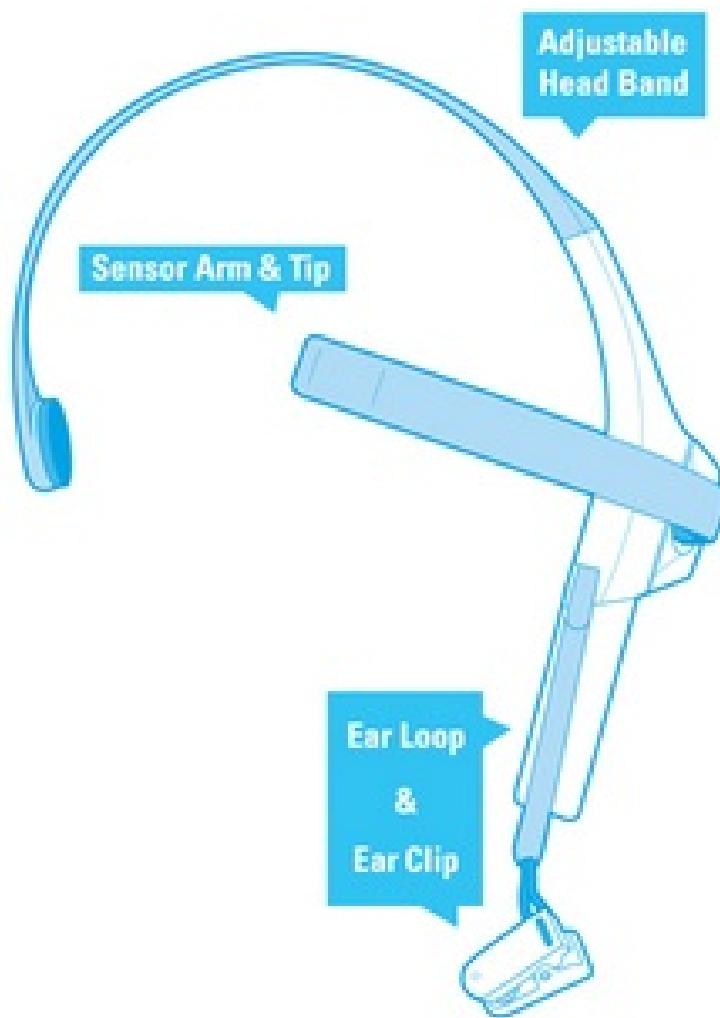
Listing 9.4 Highcharts.js

2. Dan untuk folder img, masukan gambar untuk yang nantinya akan ditampilkan pada template HTML. Adapun gambar yang dimasukkan adalah seperti pada gambar 9.4.
3. Untuk menjalankan aplikasi, dibutuhkan template html untuk menjalankannya. Semua file HTML disimpan di folder templates. Terdapat empat buat file HTML yang nantinya akan dijadikan templates.
4. Template pertama, buatlah file files.html dengan kode seperti pada listing 9.5:

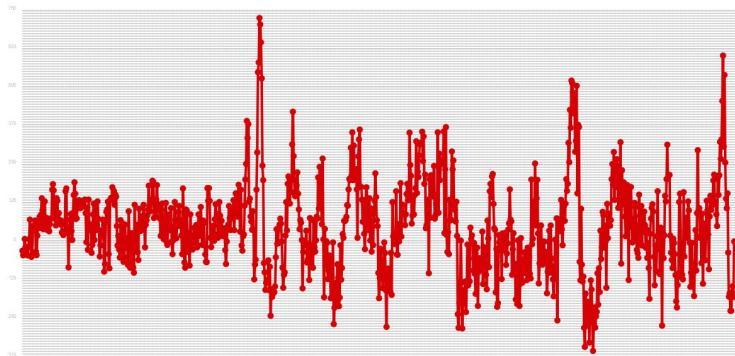
```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>Bootstrap Example</title>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-
       scale=1">
7     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
       bootstrap/3.4.0/css/bootstrap.min.css">
8     <script src="https://ajax.googleapis.com/ajax/libs/jquery/
       3.3.1/jquery.min.js"></script>
9     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/
       bootstrap.min.js"></script>
10 </head>
11 <body>
12
13 <nav class="navbar navbar-inverse">
14     <div class="container-fluid">
15         <div class="navbar-header">
16             <a class="navbar-brand" href="#">WebSiteName</a>
17         </div>
18         <ul class="nav navbar-nav navbar-right">

```



Gambar 9.4 EEG

**Gambar 9.5** Charts

```

19     <a href="/" >
20         <button type="button" class="btn btn-success">Home</
21             button>
22         </a>
23         <a href="/live" >
24             <button type="button" class="btn btn-info">Back</button>
25         </a>
26     </ul>
27 </div>
28 </nav>
29 <div class="container">
30     <ul>
31         {% for file in files %}
32             <li><a href="http://127.0.0.1:8080/{{ file }}">{{ file }}</a>
33         {% endfor %}
34     </ul>
35 </div>
36 </body>
37 </html>
38 \end{enumerate}
39
40 \subsection{Untuk file kedua buatlah file yang bernama index.html
41 kemudian isikan kode berikut:}
42 Table 11.6. Index.html
43 <!DOCTYPE html>
44 <html lang="en">
45
46 <head>
47     <meta charset="utf-8">
48     <meta http-equiv="X-UA-Compatible" content="IE=edge">
49     <meta name="viewport" content="width=device-width, initial-
50         scale=1">
51     <title>Flask Live Data</title >

```

```
51   <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
52   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">
53   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
54   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>
55
56 </head>
57
58 <body>
59   <div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
60     <div class="container-fluid">
61       <div class="navbar-header">
62         <a class="navbar-brand" href="/" align="center">Flask-
63           Live-Chart </a>
64       </div>
65     </div>
66   </div>
67
68   <div class="jumbotron text-center">
69     <div class="container-fluid">
70       <br />
71       <p>Live data using Flask and Highcharts.</p>
72     </div>
73   </div>
74
75   <div class="container">
76     <div class="row">
77       <div class="col-sm-4">
78         <h3>EEG Neurosky Mindwave</h3>
79         
80       </div>
81       <div class="col-sm-6">
82         <h3>Description </h3>
83         <p>NeuroSky technology enables product developers and
84           company service providers to create solutions that can
85           capture,
86           measure, and uncover the widest range of personal
87           health and fitness metrics. Easy to understand and use,
88           the advanced biometrics made possible by NeuroSky
89           technology provides consumers with actionable information
90           to help them evaluate their fitness – and if
91           necessary, change their behavior to take the path to a
92           healthier life.</p>
93         <p>NeuroSky's mission is to make innovative biosensor
94           technology available on a mass market scale.
95           Our leadership in electroencephalogram (EEG)
96           biosensors has produced hundreds of breakthroughs in
97           monitoring consumer-level brain waves. And our latest
98           advances in biosensors and electrocardiogram
99             (ECG) biometric algorithms now enable OEMs, ODMs and
100            service providers to develop solutions that can
```

```

90     monitor and analyze a variety of bio-cardio signals
91     that were not previously available on products
92         that consumers can wear. Our ECG Biosensor Cardiochip
93         " and our biometric roadmap algorithm will
94         accelerate innovative differentiation for our
95     customers by enabling them to develop the types of
96         Health solutions demanded by consumers today.</p>
97         <br><br>
98     </div>
99
100    </div>
101
102    <div class="jumbotron text-right">
103        <div class='container-fluid'>
104            <a href="/live" target="">
105                <button type="button" class="btn btn-info">Live data
106                using Flask and Highcharts.</button>
107            </a>
108        </div>
109    </div>
110
111    <footer>
112        <center>
113            <p>&copy; Proyek Dua</p>
114        </center>
115    </footer>
116    </div> <!-- /container -->
117
118    <script src="/ static /js /highcharts.js"></script>
119    <script src="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js /
120        bootstrap.min.js"></script>
121 </body>
122 </html>
123     f. Untuk file ketiga buatlah file yang bernama live-data.html
124     lalu isikan kode berikut:
125 Table 11.7. live-data.html
126 {{ data }}
127
128 <p>{{ data }}</p>
```

Listing 9.5 Files.html

9.1.10 Dan file keempat, buatlah file HTML yang bernama live.html, kemudian isikan kode seperti pada listing 9.6:

```

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="utf-8">
6     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <meta name="viewport" content="width=device-width, initial-scale=1"
8         >
```

```
8 <title>Flask Live Data</title>
9
10 <!-- Bootstrap -->
11 <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap
12 /3.2.0/css/bootstrap.min.css">
13
14 <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and
15 media queries -->
16 <!-- WARNING: Respond.js doesn't work if you view the page via file
17 :// -->
18 <!--[if lt IE 9]>
19     <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.
20 min.js"></script>
21     <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.
22 js"></script>
23 <![endif]-->
24 </head>
25
26 <body>
27     <div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
28         <div class="container-fluid">
29             <div class="navbar-header">
30                 <a class="navbar-brand" href="/">Flask-Live-Chart </a>
31             </div>
32             <!-- Collect the nav links, forms, and other content for
33 toggling -->
34             {# <div class="collapse navbar-collapse" id="bs-example-navbar-
35 collapse-1">#}
36             {# <ul class="nav navbar-nav">#}
37             {# <li><a href="#">A link </a></li>#}
38             {# </ul>#}
39             {# </div><!-- /.navbar-collapse -->#}
40         </div>
41     </div>
42     <!-- Main jumbotron for a primary marketing message or call to
43 action -->
44     <div class="jumbotron">
45         <div class="container-fluid">
46             </div>
47     </div>
48     <hr>
49
50     <div class="jumbotron">
51         <div class="container-fluid">
52             
```

```

54      <ul class="nav navbar-nav navbar-right">
55          <a href="/" >
56              <button type="button" class="btn btn-success">Home</
57          button>
58          </a>
59          <a href="/data" target="">
60              <button class="btn btn-info">Show Data Json </button>
61          </a>
62      </ul>
63  </div>
64  </div>
65  <center> <p>&copy; Proyek Dua</p></center>
66  </center>
67 </div> <!-- / container -->
68
69 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.3/
    jquery.min.js"></script>
70 <script src="https://code.highcharts.com/highcharts.js"></script>
71 <script src="https://code.highcharts.com/modules/exporting.js"></
    script>
72 <script src="https://code.highcharts.com/modules/export-data.js"></
    script>
73
74 <script src="/static/js/highcharts.js"></script>
75
76 <!-- Latest compiled and minified JavaScript -->
77 <script src="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap
    .min.js"></script>
78 </body>
79
80 </html>
```

Listing 9.6 live.html

1. Jika semua urusan kode sudah selesai, maka saatnya untuk menjalankan aplikasi. Untuk menjalankannya bukalah CMD, masuk ke python, kemudian masuk ke direktori tempat menyimpan semua file yang telah dibuat

2. Masukkan perintah seperti pada listing 9.7:

```
| Python flasklivechart.py
```

Listing 9.7 Perintah

3. Kemudian enter.

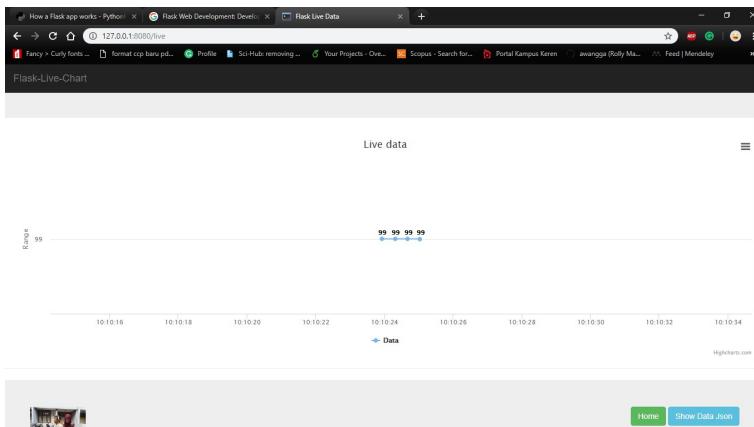
4. Jika muncul berupa link <http://localhost:5000/>, salin link tersebut dan tempel di browser. Maka hasilnya akan seperti pada gambar 9.6.

Hasilnya seperti pada gambar 9.7.

5. Pada gambar 9.6, data yang muncul tidak berupa gelombang dikarenakan tidak adanya inputan data yang masuk. Data akan muncul jika aplikasi dijalankan sambil memakai EEG.



Gambar 9.6 Halaman Awal Aplikasi



Gambar 9.7 Hasil Data

BAB 10

TEMPLATE

10.1 Cara Mengimplementasikan Jinja pada Flask

Flask memanfaatkan Jinja2 sebagai mesin template. Di sinilah orang dapat mengambil keuntungan dari mesin template Jinja2, di mana Flask didasarkan. Istilah 'sistem templating web merujuk pada merancang skrip HTML tempat data variabel dapat dimasukkan secara dinamis. Template web berisi sintaks HTML yang diselingi placeholder untuk variabel dan ekspresi (dalam hal ini ekspresi Python) yang merupakan nilai yang diganti saat template diberikan.

Jinja2 adalah mesin templat yang ditulis dengan Python murni. Ini kecil tapi cepat, selain menjadi mesin template mandiri yang simple dan mudah digunakan. Flask adalah kerangka kerja web mikro berbasis Python yang memungkinkan Anda menulis aplikasi web dengan cepat dan efisien. Dasar-dasar Jinja2 templating dari perspektif Flask menata templat dalam aplikasi berbasis Flask dalam desain modular dan dapat diperluas.

1. Pastinya anda sudah memiliki pemahaman dasar tentang Flask dan pengaturan lingkungan praktik terbaik menggunakan virtualenv yang harus diikuti saat mengembangkan aplikasi Python. Sebelumnya anda sudah mempelajari cara in-

stalasi micro-framework flask dan membuat aplikasi sederhana. Secara umum framework menyediakan template engine bawaan, dalam hal ini flask dengan jinja2.

- Untuk memahami konsep dan cara kerja template contoh dibawah akan menunjukkan sebuah aplikasi sederhana dengan tag html dan tipe data python dictionary pada sebuah route flask yang sudah dibuat sebelumnya.

Dalam Flask, anda dapat menulis aplikasi web yang lengkap tanpa perlu mesin templating pihak ketiga. Mari Anda lihat aplikasi app.py seperti pada listing 10.1:

```

1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6
7 def hello_world():
8
9     orang = {'nama': 'MIA', 'umur': '20thn'}
10
11    return '''
12 <html>
13     <head>
14         <title>Beranda Flask</title>
15     </head>
16
17     <body>
18         <h1>Hello, ''' + orang['nama'] + ''' Umur Anda, ''' +
19         orang['umur'] + '''!</h1>
20     </body>
21
22 </html>'''
23
24 if __name__ == "__main__":
25     app.run(debug=True)

```

Listing 10.1 File app.py

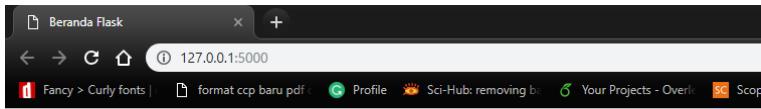
- Simpan kode diatas dengan nama app.py kemudian jalankan program python diatas di cmd seperti pada gambar 10.1.
- Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
- Jalankan program di <http://127.0.0.1:5000/> maka akan muncul halaman web seperti pada gambar 10.2.
- “orang = {'nama': 'MIA', 'umur': '20thn'}” ini adalah dictionary yang dibuat dan akan digunakan dan di masukan kedalam aplikasi flask yang akan di buat.
- Nama MIA ditampilkan di browser karna ‘mia’ ada di dalam dictionary yang sudah dibuat pada kode diatas begitupul dengan umur yang di tampilkan di browser seperti pada listing 10.2:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

```
C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 988-858-628
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 10.1 Pengujian app.py



Hello, MIA Umur Anda, 20thn!

Gambar 10.2 Hasil Pengujian File app.py

```
1 <body>
2     <h1>Hello ,  ' ' + orang[ 'nama' ] + ' ' Umur Anda ,  ' ' +
3         orang[ 'umur' ] + ' '!</h1>
```

Listing 10.2 File app.py edit

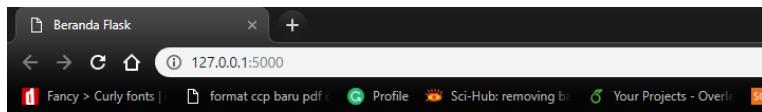
8. Kemudian ini merupakan body atau isi yang sudah dibuat yang kemudian di tampilkan di web browser. ['nama'] akan dipanggil sesuai dengan dictionary yang sudah dibuat begitu juga dengan umur, jika nama dan umur Anda ubah maka apa yang di tampilkan di browser juga terubah. Contoh :
9. Misalnya dictionary “orang = {‘nama’: ‘MIA’, ‘umur’:’20thn’}” diubah menjadi “orang = {‘nama’: ‘Novi BB’, ‘umur’:’19thn’}”
10. kemudian jalankan kembali program yang sudah diubah.
11. jalankan program python diatas di cmd seperti pada gambar 10.3.
12. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
13. Jalankan program di <http://127.0.0.1:5000/> maka akan muncul halaman web seperti pada gambar 10.4.
14. Jika nama di ganti didalam dictionary yang sudah dibuat maka tampilan web yang akan muncul sesuai dengan nama yang ada di dalam dictionary.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 988-858-620
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 10.3 Pengujian File app.py edit



Hello, Novi BB Umur Anda, 19thn!

Gambar 10.4 Hasil Pengujian File app.py edit

Contoh diatas kurang efektif untuk membuat suatu aplikasi yang nantinya akan memiliki banyak function dan kode lainnya. Aplikasi juga akan memiliki lebih banyak view function dengan semakin banyaknya URL yang akan dibuat, jadi bayangkan jika suatu hari nanti Anda akan mengubah salah satu layout, maka Anda harus memperbarui HTML di view function lainnya. Maka dari itu flask menyediakan jinja2 sebagai template untuk mempermudah dalam memodifikasi aplikasi yang akan teman teman bangun.

Jinja2 sudah include pada saat Anda install flask, sehingga Anda tidak perlu lagi melakukan konfigurasi /modifikasi apapun di terminal. Memang flask mengizinkan Anda untuk menggunakan template engine lainnya akan tetapi yang paling popular digunakan pada flask adalah jinja2.

Templates membantu mencapai pemisahan antara presentation dan business logic. Di Flask, template ditulis sebagai file yang berbeda, disimpan di folder templates yang ada di dalam application package. Jadi setelah memastikan Anda berada di direktori microblog, buat direktori untuk menyimpan templates:

1. Buat folder baru bernama templates didalam package tempat anda menyimpan program.
2. Kemudian buat file baru bernama index.html dalam folder template sehingga struktur projek yang akan dibuat seperti pada gambar 10.5.
3. Sekarang mari Anda modifikasi file yang Anda buat sebelumnya.

**Gambar 10.5** Struktur Projek

4. File index.html akan di modifikasi seperti pada listing 10.3:

```

1 <!doctype html>
2
3 <html>
4
5     <head>
6         <title >{{ title }} – Flask Blog</title >
7     </head>
8
9     <body>
10        <h1>Hello {{ orang.nama }}, Umur Anda {{ orang.umur }}!<
11        h1>
12    </body>
13 </html>
```

Listing 10.3 File index.html

5. Simpan scrip html dengan nama index.html didalam folder template.

6. Kode html diatas sangat standar satu-satunya kode yang perlu anda ketahui mengenai jinja adalah kode yang diapit kurung keriting {{ ... }} didalam tanda tersebut Anda letakkan variabel python.

7. Kemudian edit juga file app.py yang sudah dibuat sebelumnya seperti pada listing 10.4:

```

1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4 @app.route('/')
5
6 def hello_world():
7
8     orang = {'nama': 'MIA', 'umur': '20thn'}
9
10    return render_template('index.html', title='Beranda', orang=
11        orang)
12
13 if __name__ == "__main__":
14     app.run(debug=True)
```

Listing 10.4 File app.py edit 2

8. Operasi yang mengubah sebuah template menjadi halaman HTML utuh disebut dengan rendering.
9. Untuk me-render template Anda harus mengimpor sebuah fungsi Flask bernama render_template().
10. Fungsi ini meminta nama file template dan daftar variabel python yang diapit kurung keriting {{ ... }} yang ada di file index.html dalam folder templates tersebut.
11. Fungsi render_template() juga memanggil template engine bernama Jinja2 yang sudah ada bersama dengan framework Flask.
12. Jinja2 mengganti blok {{ ... }} menjadi nilai yang diinginkan menggunakan argumen yang Anda tulis di pemanggilan render_template().
13. Selanjutnya Anda akan mencoba menjalankan program yang sudah di modifikasi
14. Simpan kode file app.py kemudian jalankan program python diatas di cmd seperti pada gambar 10.6.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

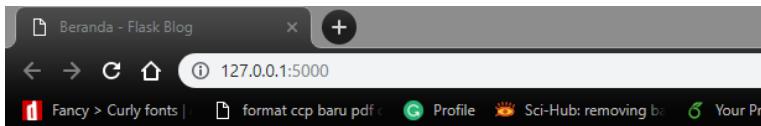
C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 988-858-620
- * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

Gambar 10.6 Pengujian file app.py edit 2

15. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
16. Jalankan program di <http://127.0.0.1:5000/> maka akan muncul halaman web seperti pada gambar 10.7.

Flask akan merender file index.html didalam folder templates dengan memanggil modul render_template secara otomatis jinja2 akan memanggil file index.html dalam folder templates dan menampilkannya di browser dengan URL <http://127.0.0.1:5000> yang telah disediakan oleh flask. Hampir sama dengan contoh sebelumnya yang mana kode html yang disatukan dengan kode flask dan menggunakan dictionary yang dibuat untuk mendefinisikan sebuah variable nama dan umur yang akan ditampilkan di web browser.



Hello MIA, Umur Anda 20thn!

Gambar 10.7 Hasil Pengujian file app.py edit 2

Bedanya kode html dipisahkan kedalam folder templates yang dibuat secara dinamis dan di dalam kode flask dipanggil menggunakan modul render_template(). Kemudian flask akan memanfaatkan jinja2 untuk memanggil file index.html yang ada pada folder templates. Di dalam kode index.html digunakan sebuah variable yang di masukan didalam kurung capit {{...}} agar jinja2 dapat mengetahui apa yang akan di panggil dan di tampilkan ke browser. Dan di dalam kode flask app.py masih harus menggunakan dictionary agar dapat mendefinisikan sebuah variael yang sudah dibuat. Kuncinya ada di dalam dictionary yang dibuat pada file app.py \orang = \{'nama': 'MIA', 'umur': '20thn'\}'

17. Dictionary di letakan di dalam fungsi yang dibuat
18. Dan didalam file index.html seperti pada listing 10.5:

```

1 <body>
2     <h1>Hello {{ orang.nama }}, Umur Anda {{ orang.umur }}!</
3 </body>
```

Listing 10.5 File index.html

19. Ini merupakan isi dari apa yang akan di tampilkan di browser, pada browser akan menampilkan kalimat “Hello MIA, Umur Anda 20 thn”
20. Dan menggunakan heading 1 agar kalimat yang ditampilkan di browser jelas.
21. {{orang.nama}} variable ini menampilkan nama orang yang ada di dictionary ‘orang’ yang sudah dibuat.
22. Begitupula dengan {{orang.umur}} akan menampilkan umur yang sudah di setting di dalam dictionary yang sudah dibuat.
23. Kemudian didalam file app.py tentunya menggunakan modull render_template().
24. Dan mengembalikan fungsi yang sudah dibuat dengan menambahkan kode
25. ‘return render_template(‘index.html’, title=’Beranda’, orang=orang)’
26. Kode tersebut akan menampilkan file index.html dengan cara merender dari folder template dengan menggunakan modul render_template().

27. Kemudian mengatur tilte dengan kalimat ‘beranda’, dan menampilkan kalimat yang sudah di seting di file index.html
28. Dengan memanggil dictionary yang sudah si buat di file app.py
29. Kemudian flask akan memanfaatkan jinja2 yang bekerja sebagai render template file index.html.
30. File index.html bisa Anda edit agar tampilan yang ada dibroeser menjadi lebih menarik untuk dilihat
31. Anda bisa menambahkan css untuk mengedit tampilan pada web browser tapi dengan output yang sama
32. Bisa juga dengan menggunakan bootstrap agar tampilan web lebih menarik untuk di pandang.
33. Sekarang teman teman bisa menambahkan file css yang nanti akan Anda buat untuk mempercantik tulisan yang akan di tampilkan di web browser.
34. Edit file index.html didalam folder templates dan kemudian tambahkan contoh code css untuk mengubah tampilan yang akan di tampilkan di browser seperti pada listing 10.6:

```
1 <!DOCTYPE html>
2
3 <html>
4
5     <head>
6
7         <title>{{ title }} – Flask Blog</title>
8
9         <style>
10            body {
11
12                font-family: 'Times New Roman', Times, serif;
13
14                background-color: rgb(11, 229, 245);
15
16                h1 {
17                    color: rgb(79, 4, 94);
18                }
19
20            }
21
22        </head>
23
24        <body>
25            <h1>Hello {{ orang.nama }}, Umur Anda {{ orang.umur }}!<
26            h1>
27        </body>
28
29    </html>
```

Listing 10.6 File index.html edit 1

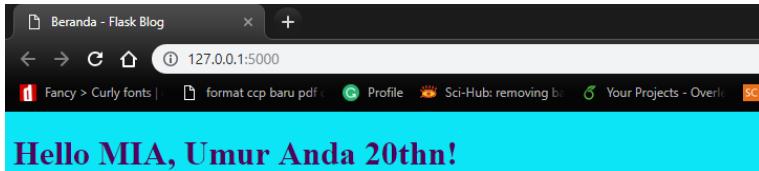
35. Simpan code index.html diatas, kemudian untuk file app.py tidak ada yang dimodifikasi.
36. Kemudian jalankan flask seperti biasa di cmd dengan menjalankan file app.py dengan perintah python app.py seperti pada gambar 10.8.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 988-858-620
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 10.8 Pengujian file app.py edit 2

37. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
38. Jalankan program di <http://127.0.0.1:5000/> maka akan muncul halaman web seperti pada gambar 10.9.



Gambar 10.9 Hasil Pengujian file app.py edit 2

39. Tampilan pada file index.html didalam folder template dapat Anda modifikasi sesuai selera Anda, dan pada saat program dijalankan tentunya tidak akan jadi masalah dengan program yang sudah Anda buat selama ketika memodifikasi file index.html tidak ada salah kata atau kuran tanda dalam kode yang Anda modifikasi.
40. Flask akan menggunakan jinja2 dalam hal merender template dan memanggil file index.html.
41. Selama variable dalam file index.html tidak ada yang berubah dan tetap mendefinisikan variable yang ada pada dictionary flask.
42. Kalaupun ada yang bermasalah (Error) flask akan memberi tahu dimana letak kesalahan yang terjadi. Contoh seperti pada listing 10.7:

```

1 <!doctype html>
2 <html>
3
4     <head>
5         <title>{{ title }} – Flask Blog</title>
6
7         <style>
8             body {
9
10             font-family: 'Times New Roman', Times, serif;
11
12             background-color: rgb(11, 229, 245);}
13
14             h1 {
15                 color: rgb(79, 4, 94);}
16         </style>
17
18     </head>
19
20     <body>
21         <h1>Hello {{ orang.nama }}, Umur Anda {{ orang.umur }}!</
22         h1>
23     </body>
24 </html>
```

Listing 10.7 File index.html edit 2

43. Anda akan sedikit merubah kode di bagian yang dijadikan variable
44. <h1>Hello \{\{\ orang.nama\}\}, Umur Anda \{\{\ orang.,umur \}\}\}
45. Simpan code index.html diatas Kemudian jalankan flask seperti biasa di cmd dengan menjalankan file app.py dengan perintah python app.py seperti pada gambar 10.10.



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 988-858-620
 - Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 10.10 Pengujian file app.py edit 2

46. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.

47. Jalankan program di <http://127.0.0.1:5000/> maka akan muncul halaman web seperti pada gambar 10.11.

The screenshot shows a browser window with the title "TemplateSyntaxError: unexpected '"'. The page content displays the following error message:

```

jinja2.exceptions.TemplateSyntaxError
TemplateSyntaxError: unexpected '"'

Traceback (most recent call last):
  File "C:\Python27\lib\site-packages\flask\app.py", line 2309, in __call__
    return self._wsgi_app(environ, start_response)
  File "C:\Python27\lib\site-packages\flask\app.py", line 2295, in wsgl_app
    response = self.handle_exception(e)
  File "C:\Python27\lib\site-packages\flask\app.py", line 1741, in handle_exception
    reraise(exc_type, exc_value, tb)
  File "C:\Python27\lib\site-packages\flask\app.py", line 2292, in wsgl_app
    response = self.full_dispatch_request()
  File "C:\Python27\lib\site-packages\flask\app.py", line 1815, in full_dispatch_request
    rv = self.handle_user_exception(e)
  File "C:\Python27\lib\site-packages\flask\app.py", line 1718, in handle_user_exception
    reraise(exc_type, exc_value, tb)
  File "C:\Python27\lib\site-packages\flask\app.py", line 1713, in full_dispatch_request
    rv = self.dispatch_request()
  File "C:\Python27\lib\site-packages\flask\app.py", line 1799, in dispatch_request
    return self.view_functions[rule.endpoint](**req.view_args)
File "C:\Users\PC-10\Zroyek\example1\venv\app\app.py", line 15, in hello_world

```

Gambar 10.11 Hasil Pengujian file app.py edit 2

48. Maka terjadilah eror seperti gambar diatas.
49. Jinja2 pada flask memberitahu bahwa ada syntax eror dalam file template yang di render
50. Syntax yang eror adalah sebagai berikut `TemplateSyntaxError: unexpected '`
51. Jinja2 flask memberitahu bahwa di dalam file index.html terjadi hal tak terduga atau bisa dibilang salah satu kode yang mendefinisikan variable pada flask tidak ada.
52. Sehingga jinja2 tidak bisa merender file index.html di dalam folder template dengan benar.
53. Anda akan memperbaiki kesalahan yang Anda buat, ini hanya sekedar contoh saja seperti pada listing 10.8:

```

1 <!doctype html>
2
3 <html>
4
5   <head>
6
7     <title>{{ title }} – Flask Blog</title>
8
9     <style>
10    body {
11      font-family: 'Times New Roman', Times, serif;
12
13

```

```

14         background-color: rgb(11, 229, 245);}
15
16     h1 {
17         color: rgb(79, 4, 94);}
18     </style>
19
20 </head>
21
22 <body>
23     <h1>Hello {{ orang.nama }}, Umur Anda {{ orang.umur }}!<
24     h1>
25 </body>
26 </html>
```

Listing 10.8 File index.html edit 3

54. Menambahkan kurung kurung keriting untuk menjadikannya sebuah variable kembali.
55. Simpan code index.html diatas Kemudian jalankan flask seperti biasa di cmd dengan menjalankan file app.py dengan perintah python app.py seperti pada gambar 10.12.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

```
C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 988-858-620
 _* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 10.12 Pengujian file app.py edit 2

56. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
57. Jalankan program di <http://127.0.0.1:5000/> maka akan muncul halaman web seperti pada gambar 10.13.
58. Selain mengubah tampilan dengan memasukan kode css kedalam file index.html seperti yang dilakukan sebelumnya, file index.html bisa anda tambahkan dengan bootstrap agar tampilan pada web browser lebih menarik untuk dipandang.
59. Teman teman bisa mengunjungi situs yang menyediakan bootstrap untuk tampilan web sederhana.



Gambar 10.13 Hasil Pengujian file app.py edit 2

60. Sebagai contoh disini akan menambahkan bootstrap sederhana dari situs yang menyediakan bootstrap tersebut
61. Anda modifikasi file index.html seperti pada listing 10.9:

```
1 <!DOCTYPE html>
2
3 <html lang="en">
4
5 <head>
6
7 <title>Bootstrap Example</title>
8 <meta charset="utf-8">
9 <meta name="viewport" content="width=device-width, initial-scale
   =1">
10 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
      bootstrap/3.4.0/css/bootstrap.min.css">
11 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/
      jquery.min.js"></script>
12 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/
      bootstrap.min.js"></script>
13
14 </head>
15
16 <body>
17
18 <div class="jumbotron text-center">
19 <h1>My First Bootstrap Page</h1>
20 <p>Resize this responsive page to see the effect!</p>
21 </div>
22 <div class="row">
23 <div class="col-md-4">
24 <div class="thumbnail">
25 </div>
26 </div>
27 <div class="col-md-4">
28 <div class="thumbnail">
29 </div>
30 </div>
31
32 <div class="col-md-4">
33 <div class="thumbnail">
34 </div>
```

```

36  </div>
37  </div>
38
39 <div class="jumbotron text-center">
40 <h1>Hello {{ orang.nama }}, Umur Anda {{ orang.umur }}!</h1>
41 </div>
42
43 </body>
44
45 </html>

```

Listing 10.9 File index.html edit 4

62. Simpan file index.html yang telah dimodifikasi dan ditambahkan dengan bootstrap sederhana.
63. Kemudian jalankan flask seperti biasa di cmd dengan menjalankan file app.py dengan perintah python app.py seperti pada gambar 10.14.



```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 988-858-620
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

Gambar 10.14 Pengujian file app.py edit 2

64. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
65. Jalankan program di <http://127.0.0.1:5000/> maka akan muncul halaman web seperti pada gambar 10.15.
66. Ini adalah tampilan dari file yang sudah dimodifikasi sebelumnya, jinja2 flask tetap merender file html yang di modifikasi menampilkan kalimat sapaan kepada user MIA.
67. Selama penggunaan variable pada file html dilakukan dengan benar maka jinja2 flask akan melakukan tugasnya dengan benar:

```

<div class="jumbotron text-center">
<h1>Hello \{\{ orang.nama \}\}, Umur Anda \{\{ orang.umur \}\}

```



Gambar 10.15 Hasil Pengujian file app.py edit 2

</div>

68. Bukan hanya nama MIA saja yang bisa di tampilkan di web browser Anda juga mengubah nama Anda dengan sesuka hati
69. Caranya hanya dengan mengubah nama di file app.py di bagian dictionary yang dibuat untuk mendefinisikan nama yang akan di tampilkan.
Contoh seperti pada listing ??:
70. Anda akan mengubah nama MIA menjadi nama Orang lain orang = \{'nama': 'O
71. Simpan kode app.py yang sudah di modifikasi kemudian jalankan program python diatas di cmd seperti pada gambar 10.16.

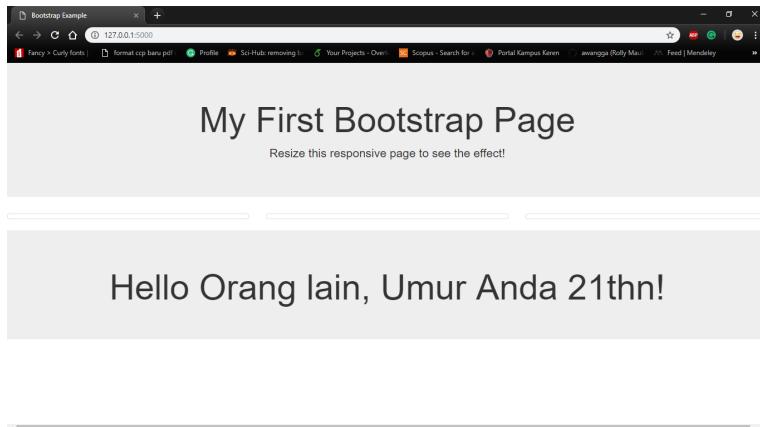
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PC 10\Zroyek\example1\venv\app>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 988-858-620
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 10.16 Pengujian file app.py edit 3

72. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.

73. Jalankan program di <http://127.0.0.1:5000/> maka akan muncul halaman web seperti pada gambar 10.17.



Gambar 10.17 Hasil Pengujian file app.py edit 3

74. Itulah tadi cara mengimplementasikan Jinja2 di flask.

10.2 Penanganan Error

Flask adalah sebuah Micro Web Framework yang ditulis menggunakan Bahasa Pemrograman Python. Flask sendiri berlisensi BSD. Untuk mengetahui atau anda ingin mempelajari Flask bisa mengunjungi halaman web doc dari flask <http://flask.pocoo.org>. Disana telah disediakan semua tentang Flask. Mulai dari instalasi Flask sampai menggunakan Modul yang tersedia pada Flask.

Kode dibawah merupakan kode program yang sudah dibuat. Program yang sudah dibuat merupakan program yang menggunakan bahas pemrograman python dan menggunakan framework flask. Sebagai contoh eror pada program yang sudah dibuat seperti pada listing 10.10:

```

1 from flask import Flask , render_template , make_response , send_file ,
2     url_for
3 import json , time
4 import Tkinter
5 import tkMessageBox
6 import atexit
7 from time import clock
8 import request
9 import os
10
11 app = Flask(__name__)
12

```

```
13  
14 #  
15 #  
16 #  
17 #  
18 #  
19 def img():  
20     PEOPLE_FOLDER = os.path.join('static', 'img')  
21     app.config['UPLOAD_FOLDER'] = PEOPLE_FOLDER  
22  
23 @app.route('/')  
24 def show_index():  
25     img()  
26     full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'EEG.jpg')  
27     return render_template("index.html", user_image = full_filename)  
28  
29  
30 def popup():  
31     tkMessageBox.showinfo("Information","Created in Python.")  
32  
33 @app.route('/live')  
34 def live():  
35     popup()  
36     img()  
37     full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'pict.jpg')  
38     return render_template('live.html', user = full_filename)  
39  
40 dataset = []  
41  
42 def myline():  
43     f = open('tmp', 'r+')[br/>44     line = f.readlines()  
45     for x in line:  
46  
47         dataset.insert(0,x)  
48     return  
49  
50  
51 def fromDataset():  
52     for x in dataset:  
53         return int(x)  
54  
55 @app.route('/live-data')  
56 def live_data():  
57     myline()  
58     data = [time.time() * 500, fromDataset()]  
59     print time.ctime(time.time())  
60     response = make_response(json.dumps(data))  
61     response.content_type = 'application/json'  
62     return response  
63  
64 @app.route('/live-tmp', methods=['GET'])  
65 def livetmp():
```

```

66     f = open('tmp', 'r')
67     line = f.readlines()
68     for x in line:
69         response = make_response(json.dumps(x))
70         response.content_type = 'application/json'
71         return response
72
73
74 @app.route('/data', defaults={'req_path': ''}, methods=['GET'])
75 @app.route('/<path:req_path>')
76
77 def simpanjson(req_path):
78     BASE_DIR = '/Users/PC 10/Zroyek/example2/LIVE/data'
79     data_path = os.path.join(BASE_DIR, req_path)
80     if os.path.isfile(data_path):
81         return send_file(data_path)
82     files = os.listdir(data_path)
83     return render_template('files.html', files=files)
84 #
85 #
86 #
87 #
88 #
89 def secondsToStr(t):#fungsi waktu
90     return "%d:%02d:%02d.%03d" % \
91             reduce(lambda ll ,b : divmod(ll[0] ,b) + ll[1:], [(t*1000,),1000,60,60])
92
93 def sapa(nama):
94     print("Hi, " + nama + ". Apa kabar?")
95 sapa('teman')
96
97 line = "="*40
98 def log(s, elapsed=None):
99     print line
100    print secondsToStr(clock()), '_', s
101    if elapsed:
102        print "Elapsed time:", elapsed
103    print line
104    print
105
106 def date():
107     date =time.time
108     print time.ctime(time.time())
109
110 def endlog():
111     end = clock()
112     elapsed = end-start
113     log("End Program", secondsToStr(elapsed))
114     date()
115     print ('Proyek dua')
116
117
118 def now():
119     return secondsToStr(clock())
120

```

```

121 start = clock()
122 atexit.register(endlog)
123 log("Start Program")
124 def print_info( nama , Npm ):
125     print ("Nama: " , nama)
126     print ("Npm: " , Npm)
127 print_info( Npm=1164035 , nama = "Eko Cahyono Putro" )
128 print_info( Npm=1164049 , nama = "Nur Arkhamia Batubara" )
129
130 if __name__ == '__main__':
131     app.run(debug=True, host='127.0.0.1', port=8080)

```

Listing 10.10 File flasklivechart.py

- Simpan skrip kode diatas kemudian namakan file tersebut menjadi flasklivechart.py, jalankan file tersebut menggunakan cmd dengan menuliskan perintah python flasklivechart.py seperti pada gambar 10.18.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: python

('Nama: ', 'Eko Cahyono Putro')
('Npm: ', 1164035)
('Nama: ', 'Nur Arkhamia Batubara')
('Npm: ', 1164049)
* Serving Flask app "flasklivechart" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
Hi, teman. Apa kabar?
=====
0:00:00.000 - Start Program
=====

('Nama: ', 'Eko Cahyono Putro')
('Npm: ', 1164035)
('Nama: ', 'Nur Arkhamia Batubara')
('Npm: ', 1164049)
* Debugger is active!
* Debugger PIN: 988-858-628
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)

```

Gambar 10.18 Pengujian file flasklivechart.py

- Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
- Jalankan program di <http://127.0.0.1:8080> kemudian akan muncul tampilan seperti pada gambar 10.19.
- Maka terjadilah eror seperti gambar diatas.
- Jinja2 pada flask memberitahu bahwa ada syntax eror dalam file template yang di render. Syntax yang eror adalah sebagai berikut: `TemplateSyntaxError: unexpected token`
- Jinja2 flask memberitahu bahwa di dalam file index.html terjadi hal tak terduga atau bisa dibilang salah satu kode yang mendefinisikan variable pada flask tidak ada. Sehingga jinja2 tidak bisa merender file index.html di dalam folder template dengan benar.

```

TemplateSyntaxError: unexpected EOF
Traceback (most recent call last):
File "C:\Python27\lib\site-packages\flask\app.py", line 2309, in __call__
    return self._wsgl_app(environ, start_response)
File "C:\Python27\lib\site-packages\flask\app.py", line 2295, in wsgl_app
    response = self._handle_exception(e)
File "C:\Python27\lib\site-packages\flask\app.py", line 1741, in handle_exception
    reraise(exc_type, exc_value, tb)
File "C:\Python27\lib\site-packages\flask\app.py", line 2292, in wsgl_app
    response = self.full_dispatch_request()
File "C:\Python27\lib\site-packages\flask\app.py", line 1815, in full_dispatch_request
    rv = self._handle_user_exception(e)
File "C:\Python27\lib\site-packages\flask\app.py", line 1778, in handle_user_exception
    reraise(exc_type, exc_value, tb)
File "C:\Python27\lib\site-packages\flask\app.py", line 1813, in full_dispatch_request
    rv = self.dispatch_request()
File "C:\Python27\lib\site-packages\flask\app.py", line 1799, in dispatch_request
    return self.view_functions[rule.endpoint](*req.view_args)
File "C:\Users\PC10\Zoyek\example\venv\app\app.py", line 15, in hello_world

```

Gambar 10.19 Hasil Pengujian file flasklivechart.py

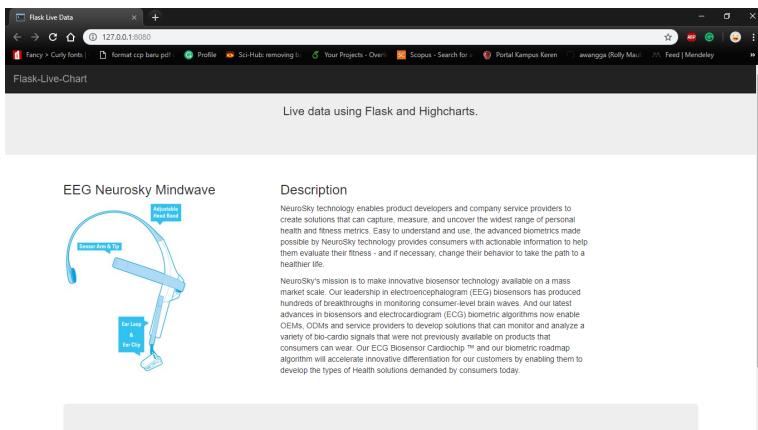
7. Dan untuk file indek.html adalah seperti pada listing ??:
8. Di dalam kode index.html digunakan sebuah variable yang di masukan didalam kurung capit {{....}} agar jinja2 dapat mengetahui apa yang akan di panggil dan di tampilkan ke browser.
9. Kita akan coba memperbaiki kesalahan yang terjadi pada index.html:
10. Tambahkan kurung kurawa tutup pada script kode kemudian save file yang sudah dimodifikasi
11. Setelah itu jalankan kembali program tersebut seperti pada gambar 10.20.
12. Setelah itu akan muncul pemberitahuan bahwa Flask telah berjalan di localhost.
13. Jalankan program di <http://127.0.0.1:8080> kemudian akan muncul tampilan seperti pada gambar 10.21.
14. Akhirnya program program sudah bisa di jalankan kembali. Kesalahan yang terjadi sebelumnya kenapa sampai bisa terjadi eror adalah pada file indek.html yang dijadikan sebagai tampilan untuk halaman web terdapat suatu kesalahan dimana variable yang di gunakan kurang lengkap. Flask akan merender file index.html didalam folder templates dengan memanggil modul render_template secara otomatis jinja2 akan memanggil file index.html dalam folder templates dan menampilkannya di browser. Halaman web yang ditampilkan seperti pada gambar diatas merupakan halaman web dari program yang sudah dibuat. Dan pada halaman web terdapat sebuah button yang akan mengarah ke dalam program yang akan dijalankan.

```
C:\Users\PC 10\Zroyek\example2\LIVE>python flasklivechart.py
Hi, teman. Apa kabar?
=====
0:00:00.000 - Start Program
=====

('Nama: ', 'Eko Cahyono Putro')
('Npm: ', 1164035)
('Nama: ', 'Nur Arkhamia Batubara')
('Npm: ', 1164049)
* Serving Flask app "flasklivechart" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
Hi, teman. Apa kabar?
=====
0:00:00.000 - Start Program
=====

('Nama: ', 'Eko Cahyono Putro')
('Npm: ', 1164035)
('Nama: ', 'Nur Arkhamia Batubara')
('Npm: ', 1164049)
* Debugger is active!
* Debugger PIN: 988-858-620
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
```

Gambar 10.20 Pengujian file flasklivechart.py



Gambar 10.21 Hasil Pengujian file flasklivechart.py

DAFTAR PUSTAKA

1. R. Awangga, “Sampeu: Servicing web map tile service over web map service to increase computation performance,” in *IOP Conference Series: Earth and Environmental Science*, vol. 145, no. 1. IOP Publishing, 2018, p. 012057.
2. M. I. Fadhil and G. M. Plexindo, “.net microframework untuk pemula.”
3. A. L. Ramdani and H. B. Firmansyah, “Clustering application for ukt determination using pillar k-means clustering algorithm and flask web framework,” *Indonesian Journal of Artificial Intelligence and Data Mining*, vol. 1, no. 2, pp. 53–59, 2018.
4. P. M. Panjaitan, “Sistem monitoring cuaca dan identifikasi keadaan cuaca menggunakan teknik web scraping,” 2018.
5. S. S. Azzahra, “Karya tulis,” 2017.
6. N. R. Wyler, *Aggressive network self-defense*. Elsevier, 2005.

Index

disruptif, xlvii
modern, xlvii