

**WHATSAPP TANPA DERITA**



---

# **WHATSAPP TANPA DERITA**

## **Dengan Bot**

---

**Dinda Majesty, Tri Angga Dio Simamora, Rolly M. Awangga, Syafrial  
Fachri Pane**  
Politeknik Pos Indonesia



**Kreatif Industri Nusantara**

***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

SYAFRIAL FACHRI PANE, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia

DINDA MAJESTY, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia

TRI ANGGA DIO SIMAMORA, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia



# CONTENTS IN BRIEF

---

<b>1 Mengenal Python</b>	<b>1</b>
<b>2 Instalasi Anaconda</b>	<b>39</b>
<b>3 Instalasi Pip dan Perintah CLI</b>	<b>53</b>
<b>4 Fungsi, Method, dan Kelas</b>	<b>57</b>
<b>5 Pengelolaan File Csv</b>	<b>65</b>
<b>6 Pengenalan Selenium</b>	<b>73</b>
<b>7 Chatbot</b>	<b>91</b>
<b>8 WhatsApp</b>	<b>93</b>
<b>9 Pengenalan Pemrograman Berorientasi Objek (OOP) dengan Python</b>	<b>95</b>
<b>10 Membuat Chatbot</b>	<b>109</b>



# DAFTAR ISI

---

Daftar Gambar	xi
Daftar Tabel	xiii
Foreword	xvii
Kata Pengantar	xix
Acknowledgments	xxi
Acronyms	xxiii
Glossary	xxv
List of Symbols	xxvii
Introduction	xxix
<i>Rolly Maulana Awangga, S.T., M.T.</i>	
<b>1 Mengenal Python</b>	<b>1</b>
1.1 Sejarah Python	1
1.1.1 Perbedaan Python 2 dan 3	2
1.1.2 Tipe Data	2
1.1.3 Konversi antar tipe data	10
	<b>ix</b>

1.1.4	Implementasi dan Penggunaan Python di Perusahaan Dunia	11
1.1.5	Jenis-Jenis Variabel	12
1.1.6	Input dan Output	13
1.1.7	Operator Aritmatika dan Konversi Tipe Data	13
1.1.8	Perulangan	14
1.1.9	Looping For	15
1.1.10	While	15
1.1.11	Perulangan Bertingkat	16
1.1.12	Break	17
1.1.13	IF Statement	18
1.1.14	ELSE statement	19
1.1.15	Elif	20
1.2	Praktek Pertama Python	21
1.2.1	Modulus	21
1.2.2	Hello NPM	21
1.2.3	Hello NPM (3 Digit Belakang)	21
1.2.4	Hello NPM (Digit ke-3)	22
1.2.5	Variabel Alfabet	22
1.2.6	Penjumlahan NPM	22
1.2.7	Perkalian NPM	23
1.2.8	Print Vertical	23
1.2.9	Digit Genap NPM	24
1.2.10	Digit Ganjil NPM	24
1.2.11	Bilangan Prima NPM	24
1.3	Penanganan Kesalahan (Error and Exception Handling)	25
1.3.1	Kesalahan Sintaksis	25
1.3.2	Pengecualian	26
1.3.3	Penanganan Pengecualian	27
1.3.4	Menghasilkan Pengecualian	29
1.3.5	Except	31
1.3.6	Run Script Hello World di Spyder	31
1.3.7	Pemakaian Variable Explorer	31
1.4	Indentasi	33
1.4.1	Penjelasan Indentasi	33
1.4.2	Jenis-Jenis Error Indentasi	33
1.4.3	Cara Membaca Error	34
1.4.4	Cara Menangani Error	35

1.4.5 Quiz: 1	36
<b>2 Instalasi Anaconda</b>	<b>39</b>
2.1 Pengenalan Anaconda	39
2.1.1 Instalasi Anaconda 3 Windows 10 x64	39
2.1.2 Update Anaconda dan Spyder	46
2.1.3 Instalasi Anaconda Ubuntu 19.04	46
2.1.4 Konfigurasi <i>Python</i>	50
2.1.5 Quiz 2	51
<b>3 Instalasi Pip dan Perintah CLI</b>	<b>53</b>
3.1 Pip	53
3.1.1 Instalasi Pip	53
3.1.2 Command Line Interface/Interpreter	55
<b>4 Fungsi, Method, dan Kelas</b>	<b>57</b>
4.1 Fungsi	57
4.2 Package	58
4.3 Kelas, Objek, Atribut, dan Method	58
4.3.1 Pemanggilan Library Kelas	59
4.3.2 Pemakaian Package Fungsi Apabila File Didalam Folder	59
4.3.3 Pemakaian Package Kelas Apabila File didalam Folder	60
4.4 Pretek Membuat Method dan Pemanggilan Method	60
4.4.1 Modulus	60
4.4.2 Hello NPM	61
4.4.3 Hello NPM (3 Digit Belakang)	61
4.4.4 Hello NPM (Digit ke-3)	61
4.4.5 Variabel Alfabet	61
4.4.6 Penjumlahan NPM	62
4.4.7 Perkalian NPM	62
4.4.8 Print Vertical	62
4.4.9 Digit Genap NPM	62
4.4.10 Digit Ganjil NPM	63
4.4.11 Bilangan Prima NPM	63
4.4.12 Pemanggilan Fungsi pada Main.py	63
<b>5 Pengelolaan File Csv</b>	<b>65</b>
5.1 Pengelolaan File CSV, Sejarah, dan Contoh	65

5.2	Aplikasi yang bisa Menciptakan File CSV	66
5.3	Menulis dan Membaca File CSV pada Ms.Excel	67
5.3.1	Sejarah Library CSV	68
5.3.2	Sejarah Library Pandas	68
5.3.3	Fungsi - fungsi yang terdapat di library CSV	68
5.4	Praktek CSV	69
5.4.1	CSV List	69
5.4.2	CSV Dictionary	70
5.4.3	Hello NPM (Pandas List)	70
5.4.4	Hello NPM (Pandas Dictionary)	70
5.4.5	Pandas Date	70
5.4.6	Pandas Ubah Index	71
5.4.7	Pandas Ubah Column	71
5.4.8	Main CSV	71
5.4.9	Main Pandas	71
<b>6</b>	<b>Pengenalan Selenium</b>	<b>73</b>
6.1	Sejarah <i>Selenium</i>	73
6.2	Jenis-Jenis <i>Selenium</i>	74
6.3	Geckodriver dan Chromedriver	75
6.3.1	Gechodriver untuk Mozilla Firefox	75
6.4	Penggunaan Selenium	76
6.4.1	Cara find element atau class	80
6.4.2	Mengambil element dari web siap.poltekpos.ac.id	81
6.5	WebDriver Exception : Can not connect to the service geckodriver	88
6.6	SessionNotCreatedException : Unable to find a matching set of capabilities	89
6.7	Module Notfound Error : no module named selenium	89
6.8	WebDriverExecution: cant load profile Possible version mismatch	89
<b>7</b>	<b>Chatbot</b>	<b>91</b>
7.1	Chatbot	91
7.1.1	Chatbot	91
7.1.2	NLP dan Rule Based	92
<b>8</b>	<b>WhatsApp</b>	<b>93</b>
8.1	WhatsApp	93
8.1.1	WhatsApp	93

<b>9 Pengenalan Pemrograman Berorientasi Objek (OOP) dengan Python</b>	<b>95</b>
9.1 Class	96
9.2 Menulis Method dan Kelas pada Python	96
9.2.1 Menulis Modul	97
9.2.2 Menambahkan variabel	97
9.2.3 Menambahkan kelas	98
9.2.4 Implementasi Kode	99
9.2.5 Mengakses Modul dari Folder Lain	100
9.3 Objek (object: an instance of a class)	101
9.4 Class' Constructor	101
9.5 Metode (Method)	102
9.6 Mekanisme Pewarisan (Inheritance)	105
9.7 Menimpa (Override) Metode dengan Nama yang Sama Dengan Kelas Dasar	107
9.8 Pemanggilan Metode Kelas Dasar dari Kelas Turunan dengan Sintaksis Super	107
9.9 Variabel Privat di Python	108
9.10 Pernak-Pernik Terkait Struktur Data	108
<b>10 Membuat Chatbot</b>	<b>109</b>
10.1 Installasi	109
10.1.1 Python / Anaconda	109
10.1.2 Chrome dan ChromeDriver	109
10.2 Kodingan Wanda	110



# DAFTAR GAMBAR

---

1.1	Perbedaan Python 2 dan Python 3	2
1.2	<i>Launch Spider</i>	32
1.3	<i>Print Hello World</i>	32
1.4	<i>Hello World</i>	32
1.5	<i>Variable Explorer</i>	33
1.6	<i>Indentasi</i>	34
1.7	<i>Error Indentasi</i>	34
1.8	<i>Syntax Error</i>	35
1.9	<i>Syntax yang Telah Diperbaiki</i>	36
2.1	Run Setup Anaconda	40
2.2	Setup Loading	40
2.3	Welcome to Anaconda Setup	41
2.4	<i>License Agreement</i>	41

2.5	<i>Just Me(recomended)</i>	42
2.6	<i>Pilih lokasi</i>	43
2.7	<i>Centang Anaconda to my PATH</i>	43
2.8	<i>Waiting Installation Complete</i>	44
2.9	<i>Installation Complete</i>	44
2.10	<i>Anaconda+JetBrains</i>	45
2.11	<i>Thanks for install Anaconda</i>	46
2.12	Gambar halaman download	47
2.13	Gambar install anaconda	47
2.14	Gambar eksekusi anaconda	48
2.15	Gambar anaconda license agreement	48
2.16	Gambar perintah yes or no	49
2.17	Gambar path anaconda	49
2.18	Gambar perintah install spyder3	50
2.19	Gambar setpath	51
3.1	<i>Install pip</i>	54
3.2	<i>Install pip Selesai</i>	54
3.3	<i>Melihat Versi pip</i>	55
3.4	<i>CLI in Command Prompt</i>	55
5.1	Contoh Penulisan CSV pada Excel	67
6.1	<i>Gechodriver</i>	76
6.2	<i>Tampilan awal spyder</i>	76
6.3	<i>Running spyder</i>	78
6.4	<i>Running spyder console</i>	79
6.5	<i>Running masih berjalan</i>	79
6.6	<i>Running selesai</i>	79
6.7	<i>Tampilan siap.poltekpos</i>	80
6.8	<i>Tampilan siap.poltekpos</i>	82

6.9	<i>inspect element by name</i>	82
6.10	<i>Tampilan loading login</i>	83
6.11	<i>Tampilan login</i>	83
6.12	<i>inspect element nilai mahasiswa</i>	84
6.13	<i>inspect element by xpath</i>	85
6.14	<i>Tampilan nilai semester mahasiswa</i>	85
6.15	<i>inspect element tahun akademik</i>	86
6.16	<i>inspect element by xpath semster genap</i>	86
6.17	<i>Tampilan nilai semester genap 2018/2019</i>	87
6.18	<i>inspect element cari</i>	87
6.19	<i>inspect element by class name cari</i>	87
10.1	Link About Chrome 1	110
10.2	Link About Chrome 2	110



## DAFTAR TABEL

---



# Listings

---

1.1	Input dan Output	13
1.2	While Loop	14
1.3	For Loop	14
1.4	Nested Loop	14
1.5	if Statement	18
1.6	Elif	18
1.7	Else	18
1.8	Nested If	19
1.9	Modulus	21
1.10	Hello NPM	21
1.11	3 Digit Belakang	22
1.12	Digit ke-3	22
1.13	Variabel Alfabet	22
1.14	Penjumlahan NPM	23
1.15	Perkalian NPM	23
1.16	Print Vertical	23
1.17	Digit Genap NPM	24
1.18	Digit Ganjil NPM	24

1.19 Bilangan Prima NPM	24
1.20 Try Except	31
4.1 Kelas	59
4.2 Modulus	60
4.3 Hello NPM	61
4.4 3 Digit Belakang	61
4.5 Digit ke-3	61
4.6 Variabel Alfabet	61
4.7 Penjumlahan NPM	62
4.8 Perkalian NPM	62
4.9 Print Vertical	62
4.10 Digit Genap NPM	63
4.11 Digit Ganjil NPM	63
4.12 Bilangan Prima NPM	63
4.13 Bilangan Prima NPM	63
5.1 Contoh CSV	65
5.2 Contoh Kode Python to Read CSV	66
5.3 Contoh Kode Python to Read CSV	66
5.4 Contoh CSV	67
src/fcsv.py	68
src/fcsv.py	68
src/fcsv.py	69
src/fcsv.py	69
5.5 CSV List	70
5.6 CSV Dictionary	70
5.7 Pandas List	70
5.8 Pandas Dictionary	70
5.9 Pandas Date	71
5.10 Pandas Ubah Index	71
5.11 Pandas Ubah Column	71
5.12 Main CSV	71
5.13 Digit Genap NPM	71
10.1 modul	110
10.2 getnilaiMahasiswa	111
10.3 cekjadwal sidang	111
10.4 saveProfile	112
10.5 splitString	112
10.6 waitLogin	112

10.7 typeAndSendMessage	112
10.8 deleteMessage	113



# **FOREWORD**

---

Sepatah kata dari Kaprodi, Kabag Kemahasiswaan dan Mahasiswa



# KATA PENGANTAR

---

Buku ini berisi pengantar serta tata cara yang harus dilakukan dalam pembuatan whatsapp chatbot. Buku ini diharapkan dapat membantu orang-orang memahami cara kerja chatbot, hal yang dibutuhkan dalam pembuatan aplikasi chatbot serta cara membuat chatbot.

TIM PENULIS

*Bandung, Jawa Barat*

*Januari, 2020*



## 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

---

chatbot	Merupakan sebuah teknologi layanan obrolan yang ditangani oleh sistem atau robot.
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.



# BAB 1

---

## MENGENAL PYTHON

---

Dalam praktek membuat whatsapp chatbot ini, hal pertama yang harus dilakukan adalah mengenal bahasa pemrograman Python dan mengetahui tentang software yang digunakan yaitu Anaconda3.

### 1.1 Sejarah Python

Nama python berasal dari acara televisi Monty python's flying circus. Python merupakan bahasa pemrograman yang dikembangkan oleh Guido Van Rossum pada tahun 1990 di CWI, Amsterdam. bahasa ini merupakan lanjutan dari bahasa pemrograman ABC. pada tahun 1995, Guido pindah ke CNRI dan mengeluarkan python versi 1.6. pada tahun 2000, Guido pindah ke BeOpen dan mengeluarkan python versi 2.0 setelah itu Guido dan tim PythonLabs pindah ke DigitalCreations. saat ini Guido dan Python Software Foundation terus melakukan perkembangan hingga python versi 2.6.1 dan python versi 3.0 Python Software Foundation merupakan sebuah organisasi yang memiliki hak atas bahasa pemrograman python, hal ini dilakukan untuk mencegah bahasa pemrograman python dimiliki oleh perusahaan komersial.

### 1.1.1 Perbedaan Python 2 dan 3

Perbedaan Python 2 dan Python 3 dapat dilihat pada gambar berikut.

No.	Perbedaan	Python 2	Python 3
1.	Syntax Print	Print tanpa kurung dan menggunakan kurung dapat dijalankan Print"tanpa kurung" Print("dengan kurung")	Print wajib menggunakan kurung, print tanpa kurung menyebabkan error Print("wajib pakai kurung")
2.	Syntax Input	Menggunakan raw_input Nama = raw_input("Masukkan nama anda: ") Menggunakan input akan menyebabkan error	Menggunakan input Nama = input("Masukkan nama anda") Menggunakan raw_input akan menyebabkan error
3.	Hasil Operator Pembagian	3/2 = 1 3//2 = 1 3/2.0 = 1.5 3//2.0 = 1.0 Apabila tipe data float maka akan menghasilkan float, jika tidak maka akan menghasilkan integer	3/2 = 1.5 3//2 = 1 3/2.0 = 1.5 3//2.0 = 1.0 Tipe data float ataupun integer tidak memiliki pengaruh kepada hasil.

**Gambar 1.1** Perbedaan Python 2 dan Python 3

### 1.1.2 Tipe Data

#### 1. Numbers

Tipe numerik pada Python dibagi menjadi 3: int, float, complex

```
a = 5
print(a, "is of type", type(a))
a = 2.0
print(a, "is of type", type(a))
a = 1+2j
print(a, "is complex number?", isinstance(1+2j, complex))
```

Output:

```
5 is of type <class 'int'>
2.0 is of type <class 'float'>
(1+2j) is complex number? True
```

Integer tidak dibatasi oleh angka atau panjang tertentu, namun dibatasi oleh memori yang tersedia. Sehingga Anda tidak perlu menggunakan variabel yang menampung big number misalnya long long (C/C++), biginteger, atau sejenisnya. Contoh kode untuk menunjukkan bahwa Python tidak membatasi output integer adalah pencarian bilangan ke-10.000 pada deret fibonacci (catatan: bilangan ke-10.000 pada deret fibonacci memiliki panjang 2.090 digit) sebagai berikut:

```
x=[0]*10005;                      #inisialisasi array 0 sebanyak 10005 elemen
x[1]=1;                            #x[1]=1

for j in range(2,10001):
    x[j]=x[j-1]+x[j-2]  # Fibonacci
print(x[10000])
```

Output:

336447648764317832666216120051075433103021484606800639065647699746800814

Perhatikan bagaimana Python melakukan pemotongan pada digit ke 16 pada variabel float b.Float atau bilangan pecahan dibatasi akurasinya pada 15 desimal. Yang membedakan Integer dan Float adalah titik (decimal points). Misalkan dalam penulisan angka 1 jenisnya Integer, tapi jika dituliskan sebagai 1.0 artinya berjenis Float atau pecahan.

```
b = 0.1234567890123456789
b
```

Output:

0.12345678901234568

Karena Python banyak digunakan juga oleh matematikawan, tipe bilangan di Python juga mendukung bilangan imajiner dan bilangan kompleks. Nilai bilangan kompleks (complex) dituliskan dalam formulasi  $x + yj$ , yakni bagian  $x$  adalah bilangan real, dan  $y$  adalah bilangan imajiner. Contohnya adalah sebagai berikut:

```
a = 1234567890123456789
a
```

Output:

1234567890123456789

```
c = 1+2j
c
```

Output:

(1+2j)

## 2. Strings

String adalah urutan dari karakter unicode. Dideklarasikan dengan petik tunggal atau ganda. String 1baris dapat ditandai dengan tiga petik tunggal atau ganda "" atau """.

```
s = "This is a string"
s = '''this is a multiline
next new line (1)
another new line (2)'''
```

Seperti list dan tuple, slicing operator [ ] dapat digunakan pada string. Sebuah string utuh bersifat mutable, namun elemennya bersifat immutable.

```
s = "Hello World!"
print(s[4])
print(s[6:11])
s[5] = "d"
```

Output:

```
'o'
'World'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

```
s = "Hello World!"
print(s)
s = "Try Python!"
print(s)
```

Output:

```
'Hello World!'
'Try Python!'
```

### 3. Bool/Boolean

Tipe data bool atau Boolean, merupakan turunan dari bilangan bulat (integer atau int) yang hanya punya dua nilai konstanta: True dan False.

#### Nilai Boolean

Nilai konstanta False dan True merepresentasikan nilai kebenaran (truth values), meskipun ada nilai-nilai lain yang juga dianggap benar atau salah. Di dalam konteks angka, misalnya digunakan sebagai argumen dari operator matematika aritmatika, kedua nilai ini berlaku seperti halnya bilangan bulat 0 dan 1, sesuai False dan True.

Ada fungsi bawaan `bool()` yang dapat mengubah nilai menjadi nilai Boolean, apabila nilai tersebut dapat direpresentasikan sebagai nilai kebenaran (truth values).

Nilai kebenaran adalah sebuah nilai yang dapat diuji sebagai benar atau salah, untuk digunakan di sintaksis kondisi if atau while atau sebagai operan dari operasi Boolean.

Berikut adalah objek bawaan yang didefinisikan bernilai salah dalam pengujian nilai kebenaran:

- (a) Konstanta yang sudah didefinisikan bernilai salah: `None` dan `False`
- (b) Angka nol dari semua tipe numeric: `0`, `0.0`, `0j`, `Decimal(0)`, `Fraction(0, 1)`
- (c) Urutan (sequence) dan koleksi (collection) yang kosong: `"`, `()`, `set()`, `range(0)`

Untuk objek yang didefinisikan sendiri, representasi nilai Boolean akan bergantung dari definisi metode (method) khusus bernama `__bool__(self)`. Jika metode ini mengembalikan True maka interpretasi nilai dari objeknya akan True, demikian juga sebaliknya. Operasi Boolean

Operasi dan fungsi bawaan yang memiliki hasil Boolean akan selalu mengembalikan 0 atau False untuk yang bernilai salah, serta 1 atau True untuk yang bernilai benar, kecuali dinyatakan berbeda dalam dokumentasi.

Operasi untuk tipe Boolean akan dijelaskan lebih lanjut di modul Operator, Operands, dan Expressions.

### 4. List

List adalah jenis kumpulan data terurut (ordered sequence), dan merupakan salah satu variabel yang sering digunakan pada Python. Serupa, namun tak sama dengan array pada bahasa pemrograman lainnya. Bedanya, elemen List pada Python tidak harus memiliki tipe data yang sama. Mendeklarasikan List cukup mudah dengan kurung siku dan elemen yang dipisahkan dengan koma. Setiap data didalamnya dapat diakses dengan indeks yang dimulai dari 0.

```
a = [1, 2.2, 'python']
```

Python mengenal slicing operator [] yang dapat melakukan ekstraksi sebuah item atau beberapa item yang berada dalam range tertentu pada tipe data urutan (sequences), misalnya list, string dan tuple. Beberapa tipe urutan juga mendukung "extended slicing" dengan parameter ketiga berupa "step".

x[0] artinya mengambil elemen paling awal, dengan index 0 dari List x

x[5] artinya mengambil elemen dengan index 5 dari List x

x[-1] artinya mengambil elemen dengan index paling belakang ke-1 dari List x

x[3:5] artinya membuat list dari anggota elemen List x dengan index 3 hingga sebelum index 5 (tidak termasuk elemen dengan index 5, dalam hal ini hanya index 3-4)

x[:5] artinya membuat list dari anggota elemen List x paling awal hingga sebelum index 5 (tidak termasuk elemen dengan index 5, dalam hal ini hanya index 0-4)

x[-3:] artinya membuat list dari anggota elemen List x mulai index ke-3 dari belakang hingga paling belakang

x[1:7:2] artinya membuat list dari anggota elemen List x dengan index 1 hingga sebelum index 5, dengan "step" 2 (dalam hal ini hanya index 1, 3, 5)

```
x = [5,10,15,20,25,30,35,40]
print(x[5])
print(x[-1])
print(x[3:5])
print(x[:5])
print(x[-3:])
print(x[1:7:2])
```

Output:

```
30
40
[20, 25]
[5, 10, 15, 20, 25]
[30, 35, 40]
[10, 20, 30]
```

Elemen pada list dapat diubah atau ditambahkan. Misalnya untuk melakukan perubahan kemudian penambahan:

```
x = [1, 2, 3]
x[2]=4
x
```

Output:

```
[1, 2, 4]
```

```
x.append(5)
x
```

Output:

```
[1, 2, 4, 5]
```

Untuk menghapus, gunakan fungsi del

```
spam = ['cat', 'bat', 'rat', 'elephant']
del spam[2]
spam
```

Output:

```
['cat', 'bat', 'elephant']
```

```
del spam[2]
spam
```

Output:

```
['cat', 'bat']
```

## 5. Tuple

Tuple adalah jenis dari list yang tidak dapat diubah elemennya. Umumnya tuple digunakan untuk data yang bersifat sekali tulis, dan dapat dieksekusi lebih cepat. Tuple didefinisikan dengan kurung dan elemen yang dipisahkan dengan koma.

```
t = (5,'program', 1+3j)
```

Seperti list, kita dapat melakukan slicing, namun pada tuple kita tidak dapat melakukan perubahan:

```
t = (5,'program', 1+3j)
t[1]
t[0:3]
t[0]=10
```

Output:

```
'program'
(5, 'program', (1+3j))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

## 6. Set

Set adalah kumpulan item bersifat unik dan tanpa urutan (unordered collection). Didefinisikan dengan kurawal dan elemennya dipisahkan dengan koma. Pada Set kita dapat melakukan union dan intersection, sekaligus otomatis melakukan penghapusan data duplikat.

```
a = {1,2,2,3,3,3}
a
```

Output:

```
{1, 2, 3}
```

Karena set bersifat unordered, maka kita tidak bisa mengambil sebagian data / elemen datanya menggunakan proses slicing.

```
a = {1,2,3}
a[1]
```

Output:

```
Traceback (most recent call last):
  File "<string>", line 301, in runcode
    File "<interactive input>", line 1, in <module>
TypeError: 'set' object does not support indexing
```

## 7. Dictionary

Dictionary pada Python adalah kumpulan pasangan kunci-nilai (pair of key-value) yang bersifat tidak berurutan. Dictionary dapat digunakan untuk menyimpan data kecil hingga besar. Untuk mengakses datanya, kita harus mengetahui kuncinya (key). Pada Python, dictionary didefinisikan dengan kurawal dan tambahan definisi berikut:

- (a) Setiap elemen pair key-value dipisahkan dengan koma (,)
- (b) Key dan Value dipisahkan dengan titik dua (:)
- (c) Key dan Value dapat berupa tipe variabel/obyek apapun

```
d = {1:'value', 'key':2}  
print(type(d))
```

Output:

```
<class 'dict'>  
  
d = {1:'value', 'key':2}  
print(type(d))  
print("d[1] = ", d[1]);  
print("d['key'] = ", d['key']);
```

Output:

```
<class 'dict'>  
d[1] = value  
d['key'] = 2
```

Dictionary bukan termasuk dalam implementasi urutan (sequences), sehingga tidak bisa dipanggil dengan urutan indeks. Misalnya dalam contoh berikut dicoba dengan indeks 2, tetapi menghasilkan error (KeyError) karena tidak ada kunci (key) 2:

```
d = {1:'value', 'key':2}  
print(type(d))  
print("d[1] = ", d[1]);  
print("d['key'] = ", d['key']);  
  
# Generates error  
print("d[2] = ", d[2]);
```

Output:

```
<class 'dict'>  
d[1] = value  
d['key'] = 2
```

---

```
KeyError Traceback (most recent call last)
<ipython-input-7-4b566e677ca2> in <module>()
 1 d = {1:'value','key':2}
 2 print("d[2] = ", d[2]);
```

KeyError: 2

### 1.1.3 Konversi antar tipe data

Kita dapat melakukan konversi tipe data bawaan dengan menggunakan fungsi konversi tipe bawaan (standard type) misalnya: int(), float(), str(), dll. Contoh:

float(5)

Output:

5.0

Konversi float ke int akan bersifat floor/truncating, menghilangkan nilai di belakang koma.

Contoh:

int(10.6)

Output:

10

int(-10.6)

Output:

-10

Konversi dari dan ke string akan melalui pengujian dan dipastikan validitasnya.

float('2.5')

Output:

2.5

str(25)

Output:

'25'

int('1p')

Output:

```
Traceback (most recent call last):
```

```
File "<string>", line 301, in runcode
```

```
File "<interactive input>", line 1, in <module>
```

```
ValueError: invalid literal for int() with base 10: '1p'
```

Anda juga dapat melakukan konversi kumpulan data (set, list, tuple).

set([1,2,3])

Output:

1, 2, 3

tuple(5,6,7)

Output:

```
(5, 6, 7)
```

```
list('hello')
```

Output:

```
[‘h’, ‘e’, ‘l’, ‘l’, ‘o’]
```

Untuk konversi ke dictionary, data harus memenuhi persyaratan key-value. Berikut adalah dua contoh konversi:

List dari beberapa List yang isinya pasangan nilai menjadi Dictionary.

Serta konversi List dari beberapa Tuple yang isinya pasangan nilai menjadi Dictionary.

```
dict([[1,2],[3,4]])
```

Output:

```
1: 2, 3: 4
```

```
dict([(3,26),(4,44)])
```

Output:

```
3: 26, 4: 44
```

#### 1.1.4 Implementasi dan Penggunaan Python di Perusahaan Dunia

##### 1. spotify

spotify adalah suatu layanan musik streaming yang menggunakan pemrograman python untuk analisis data dan backend. pada backend spotify berkomunikasi dengan 0MQ. 0MQ itu sendiri adalah suatu framework dan library open source untuk networking. untuk analisis data tersebut spotify menggunakan luigi, dan modul python yang sinkron dengan hadoop.

##### 2. Google

Google ini sudah menggunakan bahasa pemprograman python ini sudah sajak dari awal berdirinya. Dan pada saat ini bahasa pemprograman python merupakan salah satu bahasa pemprograman server-side resmi di google. Meskipun ada script yang ditulis untuk google menggunakan bahasa perl dan bash, maka nantinya script tersebut akan diubah ke python terlebih dahulu, karena kemudahan dalam perawatannya.

##### 3. Industrial Light and Magic

Industrial Light and Magic ini merupakan studio special efek yang dibutuhkan untuk film star wars saja. Karena infrastruktur awal industrial light and magic ini menggunakan C dan C++, maka akan lebih mudah mengintegrasikan bahasa pemprograman python ketimbang bahasa pemprograman lainnya. Dengan menggunakan bahasa pemprograman python ini industrial light and magic

dengan mudah membungkus komponen software dan dapat meningkatkan aplikasi grafisnya.

#### 4. Netflix

Netflix adalah suatu layanan pemutaran film yang dapat dilakukan oleh pengguna dimanapun dan kapanpun. Pada netfilx bahasa pemrograman yang digunakan adalah bahasa pemrograman python, bahasa pemrograman ini digunakan pada Central Alert Gateway yang akan me-reroute alert dan mengirimkannya pada individu yang akan melihatnya serta juga dapat secara otomatis reboot atau menghentikan proses yang dianggap bermasalah. Selain itu python juga digunakan untuk menelusuri riwayat dan perubahan pengaturan keamanan.

#### 5. instagram

Instagram adalah suatu aplikasi mobile berbasis IOS, android dan windows phone, dimana pengguna dapat berbagi foto dan video melalui instagram ini. Pada instagram ini menggunakan bahasa pemrograman python dalam task queuennya atau fitur dimana setiap pengguna dapat berbagi foto atau video ke beberapa social network lainnya seperti facebook, twitter, dan lain-lainnya.

### 1.1.5 Jenis-Jenis Variabel

Variabel merupakan tempat penyimpanan data. Tipe data merupakan jenis data yang tersimpan di dalam variabel. terdapat aturan dalam penulisan Variabel.

1. Nama variabel diawali dengan huruf atau garis bawah, contoh: nama, \_nama, namaKu, nama\_variabel.
2. Karakter selanjutnya dapat berupa huruf, garis bawah atau angka, contoh: \_\_nama, nama1, p1.
3. Karakter bersifat case-sensitive (huruf besar dan huruf kecil dibedakan), contoh: Nama dan NAMA keduanya adalah variabel yang berbeda.
4. Nama variabel tidak boleh menggunakan kata kunci yang ada pada bahasa pemrograman python, contoh: if, else, while
5. Nama variabel tidak boleh diawali dengan angka

Jenis-jenis tipe data pada python. Tipe Data Primitif, dibagi menjadi 3 yaitu:

1. Tipe data integer (angka), penulisannya tidak membutuhkan tanda petik, contoh: 10 atau 15
2. Tipe data string (teks), tipe data string ditandai dengan teks yang diapit oleh tanda petik (""), contoh: "nama saya adalah dinda majesty"
3. Tipe data boolean (memiliki dua nilai yaitu true dan false atau 0 dan 1)

Contoh penulisan variabel dan tipe datanya:

1. angka = 10, angka merupakan nama variabel sedangkan 10 adalah nilai dari variabel yang tipe datanya integer.
2. nama = "Dinda Majesty", nama merupakan nama variabel sedangkan "Dinda Majesty" merupakan nilai dari variabel yang tipe datanya string, ditandai dengan adanya petik ("").
3. makan = True , makan merupakan nama variabel sedangkan True merupakan nilai dari variabel yang tipe datanya boolean.

### 1.1.6 Input dan Output

Berikut kode untuk meminta inputan dari user.

```
1 #input output
2 print("masukkan nama anda : ")
3 nama = input()
4 print("nama saya adalah " + nama)
```

**Listing 1.1** Input dan Output

Perintah input() berguna untuk meminta inputan dari user, sehingga memungkinkan user untuk menginputkan data.

Perintah print() berguna untuk menampilkan output dari data yang diinputkan oleh user, sehingga data yang diinputkan user dapat ditampilkan ke layar.

### 1.1.7 Operator Aritmatika dan Konversi Tipe Data

Operator aritmatika

1. penjumlahan (+)
2. pengurangan (-)
3. perkalian (\*)
4. pembagian (/)
5. sisa bagi/modulus (%)
6. pemangkatan (\*\*)

Cara melakukan perubahan terhadap tipe data string menjadi integer, contoh: variabel = "10". Kita dapat mengubah string "10" menjadi angka 10 dengan menambahkan kode int(variabel), dengan begitu 10 yang awalnya bertipe data string akan dikonversikan menjadi integer.

Cara melakukan perubahan terhadap tipe data integer menjadi string, contoh: variabel = 150. Kita dapat mengubah integer 150 menjadi string "150" dengan menambahkan kode str(variabel), maka tipe data dari variabel akan dikonversikan menjadi string.

### 1.1.8 Perulangan

perulangan terdiri atas 3 kondisi.

1. While, apabila kondisinya True, maka perulangan akan terus berjalan hingga diperoleh kondisi False. Contoh penggunaan while:

```

1 #perulangan while
2 hitung = 0
3 while (hitung < 9):
4     print ('hitungan ke :', hitung)
5     hitung = hitung + 1
6
7 print ("Good bye!")

```

**Listing 1.2** While Loop

2. For, perulangan for bisa melakukan perulangan terhadap item apapun seperti list atau string. Contoh penggunaan For:

```

1 #perulangan for
2 minum = ["kopi", "susu", "teh"]
3 for minuman in minum:
4     print("Saya suka minum", minuman)

```

**Listing 1.3** For Loop

3. nested, perulangan ini memungkinkan adanya perulangan didalam perulangan. Contoh penggunaan nested:

```

1 #nested loop
2     i = 2
3 while(i < 100):
4     j = 2
5     while(j <= (i/j)):
6         if not(i%j): break
7         j = j + 1
8     if (j > i/j) : print(i, " is prime")
9     i = i + 1
10
11 print ("Good bye!")

```

**Listing 1.4** Nested Loop

Pada penulisan sintaks While dan For harus memperhatikan identasi (baris yang menjorok ke dalam), jika tidak diperhatikan dengan baik maka akan terjadi error terhadap identasi. Untuk menambahkan identasi dapat menggunakan spasi atau tab.

### 1.1.9 Looping For

Seperti di bahasa pemrograman lainnya, Python juga memiliki fungsi for. Bedanya di Python, For tidak hanya untuk perulangan dengan jumlah finite (terbatas), melainkan lebih ke fungsi yang dapat melakukan perulangan pada setiap jenis variabel berupa kumpulan atau urutan. Variabel yang dimaksud bisa berupa list, string, ataupun range. Jika sebuah list atau urutan berisi expression, maka ia akan dievaluasi terlebih dahulu. Kemudian item pertama pada urutan atau list akan diassign sebagai variabel iterating\_var. Setelahnya, blok statement akan dieksekusi, berlanjut ke item berikutnya, berulang, hingga seluruh urutan habis. contoh for:

```
for letter in 'Python': # First Example
    print('Current Letter: {}'.format(letter))
fruits = ['banana', 'apple', 'mango']
for fruit in fruits: # Second Example
    print('Current fruit: {}'.format(fruit))
```

Output:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango
```

Anda juga dapat melakukan perulangan berdasarkan indeks atau range dengan memanfaatkan fungsi len():

```
fruits = ['banana', 'apple', 'mango']
for index in range(len(fruits)):
    print('Current fruit: {}'.format(fruits[index]))
```

Output:

```
Current fruit : banana
Current fruit : apple
Current fruit : mango
```

### 1.1.10 While

While pada bahasa Python digunakan untuk mengeksekusi statement selama kondisi yang diberikan terpenuhi (True). Kondisi dapat berupa expression apapun, dan harap diingat bahwa True di Python termasuk semua nilai non-zero. Saat kondisi menjadi False, program akan melanjutkan ke baris setelah blok statement.

Seperti for dan semua statement percabangan, blok statement yang mengikuti kondisi while dan memiliki posisi indentasi yang sama, dianggap blok statement yang akan dieksekusi.

Contoh:

```
count = 0
while (count < 5):
    print('The count is: {}'.format(count))
    count = count + 1
```

Output:

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
```

Seperti pada bahasa lainnya, eksekusi statement while mungkin bersifat infinit / infinite loop saat sebuah kondisi tidak pernah bernilai False. Contohnya sebagai berikut:

```
var = 1
while var == 1: # This constructs an infinite loop
    num = input('Enter a number: ')
    print('You entered: {}'.format(num))
```

```
while True: # This constructs an infinite loop
    num = input('Enter a number: ')
    print('You entered: {}'.format(num))
```

Potongan kode di atas tidak akan pernah bernilai False karena nilai var tidak pernah berubah. Untuk menghentikan infinite loop, gunakan CTRL (atau CMD) - C untuk menghentikannya dan keluar dari program.

Anda juga dapat menyingkat penulisan blok statement While jika statement Anda cukup terwakili oleh satu baris.

```
while (var1): do_something()
```

### 1.1.11 Perulangan Bertingkat

Ada kalanya Anda perlu untuk melakukan perulangan bertingkat, misalnya untuk menghasilkan contoh print-out berikut:

```
*****
 ****
 ***
 **
 *
```

\*

Contoh:

```
for i in range(0, 5):
    for j in range(0, 5 - i):
        print('*', end='')
    print()
```

### 1.1.12 Break

Pernyataan break menghentikan perulangan kemudian keluar, dilanjutkan dengan mengeksekusi pernyataan (statement) setelah blok perulangan. Salah satu penggunaannya yang paling sering adalah sebuah kondisi eksternal yang membutuhkan program untuk keluar dari perulangan. Jika Anda memiliki perulangan bertingkat, break akan menghentikan perulangan sesuai dengan tingkatan atau di perulangan mana ia berada. Namun jika ia diletakkan di perulangan dengan kedalaman kedua misalnya, hanya perulangan itu saja yang berhenti, tidak dengan perulangan utama. Contoh 1:

```
for letter in 'Python':
    if letter == 'h':
        break
    print('Current letter: {}'.format(letter))
```

Output contoh 1:

```
Current Letter : P
Current Letter : y
Current Letter : t
```

Contoh 2:

```
var = 10
while var > 0:
    print('Current variable value: {}'.format(var))
    var = var - 1
    if var == 5:
        break
```

Output contoh 2:

```
Current variable value : 10
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
```

### 1.1.13 IF Statement

kondisi if dapat digunakan didalam looping dan dapat digunakan untuk memberikan kondisi tertentu dengan cara mengetikkan if lalu kondisi yang akan terjadi.

1. if hanya menjalankan satu kondisi dan menampilkan satu output. Contoh: kondisi dimana variabel a lebih besar dari variabel b, maka tampilkan hasil bahwa a lebih besar dari b.

```
1 #if statement
2 a = 330
3 b = 200
4 if a > a:
5     print("a lebih besar dari b")
```

**Listing 1.5** if Statement

2. elif digunakan apabila kondisi pertama tidak benar maka lakukan kondisi lain (alternatif). Contoh: kondisi dimana variabel a sama dengan variabel b, maka jika b lebih besar dari a, tampilkan hasil b lebih besar dari a, namun jika a dan b bernilai sama, maka tampilkan a sama dengan b

```
1 #elif
2 a = 33
3 b = 33
4 if b > a:
5     print("b lebih besar dari a")
6 elif a == b:
7     print("a sama dengan b")
```

**Listing 1.6** Elif

3. else digunakan apabila kondisi yang terjadi bernilai salah, maka lakukan else. Contoh: kondisi dimana variabel a lebih besar dari variabel b, maka jika b lebih besar dari a, tampilkan hasil b lebih besar dari a, jika a dan b bernilai sama, maka tampilkan a sama dengan b, jika salah maka tampilkan a lebih besar dari pada b

```
1 #else
2 a = 200
3 b = 33
4 if b > a:
5     print("b is greater than a")
6 elif a == b:
7     print("a and b are equal")
8 else:
9     print("a is greater than b")
```

**Listing 1.7** Else

4. Nested if merupakan if didalam if (if bersarang), terdapat dua if didalam satu kondisi. Contoh: variabel x sama dengan 41, kondisi pertama yaitu jika x besar

dari 10 maka tampilkan lebih besar dari 10, kondisi kedua yaitu jika x besar dari 20, maka tampilkan lebih besar dari 20, jika salah maka tampilkan tidak melebihi 20.

```
1 #nested if
2 x = 41
3
4 if x > 10:
5     print("lebih besar dari 10")
6     if x > 20:
7         print("lebih besar dari 20!")
8     else:
9         print("tidak melebihi 20.")
```

**Listing 1.8** Nested If

contoh if statement:

```
var1 = 100
if var1:
    print ("1 - Got a true expression value")
    print (var1)
var2 = 0
if var2:
    print ("2 - Got a true expression value")
    print (var2)
```

Output:

```
1 - Got a true expression value
100
```

### 1.1.14 ELSE statement

Statement Else dapat dikombinasikan dengan IF Statement, sebagai jalan keluar saat kondisi / hasil evaluasi bernilai False. Else bersifat opsional dan tunggal. contoh else statement:

```
amount = int(input("Enter amount: "))
if amount<1000:
    discount = amount*0.05
    print ("Discount",discount)
else:
    discount = amount*0.10
    print ("Discount",discount)

print ("Net payable:",amount-discount)
```

Output true:

```
Enter amount: 600
Discount 30.0
Net payable: 570.0
```

Output false:

```
Enter amount: 1200
Discount 120.0
Net payable: 1080.0
```

### 1.1.15 Elif

Alternatif untuk Switch/Case dan IF bertingkat di python. Elif adalah kependekan dari else if, dan merupakan alternatif untuk if bertingkat atau switch/case di beberapa bahasa pemrograman lain. Sebuah IF Statement dapat diikuti satu atau lebih statement elif (opsional dan tidak dibatasi). contoh elif:

```
amount = int(input("Enter amount: "))
if amount<1000:
    discount = amount*0.05
    print ("Discount",discount)
elif amount<5000:
    discount = amount*0.10
    print ("Discount",discount)
else:
    discount = amount*0.15
    print ("Discount",discount)
print ("Net payable:",amount-discount)
```

Output true if:

```
Enter amount: 600
Discount 30.0
Net payable: 570.0
```

Output true elif:

```
Enter amount: 3000
Discount 300.0
Net payable: 2700.0
```

Output false:

```
Enter amount: 6000
Discount 900.0
Net payable: 5100.0
```

## 1.2 Praktek Pertama Python

### 1.2.1 Modulus

Praktek kali ini kita akan mencoba praktek menggunakan modulus atau sisa bagi, kita membuat inputan terlebih dahulu menggunakan perintah input. Kemudian buatlah variabel untuk menampung hasil sisa bagi dari jumlah yang diinputkan. Misalnya, teman-teman menginputkan nilai yaitu 1184011, maka hasil modulus 1184011 mod 3 adalah 1 maka print 1184011 menggunakan tanda pagar. jika hasil modulus adalah 0 maka print 1184011 menggunakan tanda bintang.

```

1 #Modulus
2 print("Soal no 1")
3
4 print("Masukkan NPM anda: ")
5 NPM = input()
6
7 npm = int(NPM) % 3
8 print(npm)
9
10 print("###    ###    #####    ###    ###    #####    ###    ###")
11 print("###    ###    #####    ###    ###    #####    ###    ###")
12 print("###    ###    ###    ###    ###    ###    ###    ###")
13 print("###    ###    ###    ###    ###    ###    ###    ###")
14 print("###    ###    #####    #####    ###    ###    ###")
15 print("###    ###    ###    ###    ###    ###    ###    ###")
16 print("###    ###    ###    ###    ###    ###    ###    ###")
17 print("###    ###    #####    ###    #####    ###    ###")
18 print("###    ###    #####    ###    #####    ###    ###")

```

**Listing 1.9** Modulus

### 1.2.2 Hello NPM

Setelah selesai praktek modulus kita akan menampilkan output berupa kalimat "Hello 1184011 Apa Kabar" sebanyak 2 digit belakang angka, yaitu angka 11, maka akan berulang sebanyak 11 kali.

```

1 #Hello NPM
2 print("Soal No 2")
3
4 Loop = NPM[5:7]
5
6 for x in range(int(Loop)):
7     print("Hallo " + NPM + " Apa Kabar?")

```

**Listing 1.10** Hello NPM

### 1.2.3 Hello NPM (3 Digit Belakang)

Jika telah selesai, maka selanjutnya kita akan melakukan praktek untuk menampilkan output berupa kalimat "Hallo 011 Apa Kabar?" sebanyak angka keenam ditambah

angka ketujuh atau 1 ditambah 1, sehingga kalimat tersebut akan berulang sebanyak 2 kali.

```

1 #3 digit belakang NPM
2 print("Soal No 2")
3
4 Loop = NPM[4:7]
5
6 total = int(NPM[5]) + int(NPM[6])
7
8 for x in range(total):
9     print("Hallo " + Loop + " Apa Kabar?")

```

**Listing 1.11** 3 Digit Belakang

### 1.2.4 Hello NPM (Digit ke-3)

Kemudian kita akan melakukan praktek untuk menampilkan output berupa kalimat "Hallo 0 Apa Kabar?".

```

1 #digit ke 3
2 print("Soal No 3")
3
4 Loop = NPM[4]
5 print("Hello " + Loop + " Apa Kabar?")

```

**Listing 1.12** Digit ke-3

### 1.2.5 Variabel Alfabet

Teman-teman, sekarang mari kita coba menambahkan variabel pada angka 1184011, tambahkan abcdefg kedalam sebuah variabel dan tambahkan variabel index dengan nilai 0, buatlah perulangan agar variabel huruf menyesuaikan dengan variabel angka sehingga menjadi a=1, b=1, c=8, d=4, e=0, f=1 ,g=1.

```

1 #variabel alfabet
2 print("Soal no 5")
3
4 var = "abcdefg"
5 index = 0
6
7 for i in var:
8     print(i + " = " + NPM[index])
9     index += 1

```

**Listing 1.13** Variabel Alfabet

### 1.2.6 Penjumlahan NPM

Praktek selanjutnya adalah menjumlahkan angka 1184011 dengan menerapkan perulangan dan penambahan. Apabila nilai 1 telah didapatkan maka akan ditambahkan

dengan nilai 1 sehingga menjadi 2, kemudian nilai 2 ditambahkan lagi dengan nilai selanjutnya yaitu 8 sehingga menjadi 10, begitu seterusnya.

```
1 #penjumlahan NPM
2 print("Soal no 6")
3
4 index = 0
5 angka = 0
6
7 for i in NPM:
8     jumlah = int(NPM[index]) + int(angka)
9     angka = jumlah
10    index += 1
11
12 print(jumlah)
```

**Listing 1.14** Penjumlahan NPM

### 1.2.7 Perkalian NPM

Praktek selanjutnya adalah mengalikan angka 1184011 dengan menerapkan perulangan dan perkalian. Apabila nilai 1 telah didapatkan maka akan dikalikan dengan nilai 1 sehingga menjadi 1, kemudian nilai 1 dikalikan lagi dengan nilai selanjutnya yaitu 8 sehingga menjadi 8, begitu seterusnya.

```
1 #perkalian NPM
2 print("Soal no 7")
3
4 index = 0
5 angka = 0
6
7 for i in NPM:
8     jumlah = int(NPM[index]) * int(angka)
9     angka = jumlah
10    index += 1
11
12 print(jumlah)
```

**Listing 1.15** Perkalian NPM

### 1.2.8 Print Vertical

Melakukan print secara vertikal hanya perlu menambahkan perulangan terhadap nilai 1184011.

```
1 #print vertical
2 print("Soal no 8")
3
4 for i in NPM:
5     print(i)
```

**Listing 1.16** Print Vertical

### 1.2.9 Digit Genap NPM

Selanjutnya, mari kita lakukan print hanya terhadap digit genap pada angka 1184011 dengan memanfaatkan perulangan, if statement dan operator logika. Logika yang akan diterapkan adalah masing-masing angka 1184011 akan di cek terlebih dahulu, apakah angka tersebut memiliki angka yang apabila dibagi 2 akan menghasilkan sisa bagi sama dengan 0 dan angka tersebut tidak boleh sama dengan 0, karena angka 0 bukan merupakan angka ganjil maupun genap.

```

1 #digit genap NPM
2 print("Soal no 9")
3
4 index = 0
5 for i in NPM:
6     if (int(NPM[index])%2 == 0) & (int(NPM[index]) != 0):
7         print(NPM[index])
8     index += 1

```

**Listing 1.17** Digit Genap NPM

### 1.2.10 Digit Ganjil NPM

Selanjutnya, mari kita lakukan print hanya terhadap digit ganjil pada angka 1184011 dengan memanfaatkan perulangan, if statement dan operator logika. Logika yang akan diterapkan adalah masing-masing angka 1184011 akan di cek terlebih dahulu, apakah angka tersebut memiliki angka yang apabila dibagi 2 akan menghasilkan sisa bagi tidak sama dengan 0 dan angka tersebut tidak boleh sama dengan 0, karena angka 0 bukan merupakan angka ganjil maupun genap.

```

1 #digit ganjil NPM
2 print("Soal no 10")
3
4 index = 0
5 for i in NPM:
6     if (int(NPM[index])%2 != 0) & (int(NPM[index]) != 0):
7         print(NPM[index])
8     index += 1

```

**Listing 1.18** Digit Ganjil NPM

### 1.2.11 Bilangan Prima NPM

Praktek terakhir adalah menampilkan hasil dari bilangan prima angka 1184011 dengan cara apabila angka 1184011 memiliki angka yang merupakan angka prima maka akan ditampilkan, logika yang akan diterapkan adalah apabila angka kecil sama dengan angka 1 maka angka tersebut bukan bilangan prima. Jika angka 1184011 dibagi 2 setelah itu dibagi dengan angka itu sendiri dan menghasilkan sisa bagi sama dengan 0, maka angka tersebut bukan bilangan prima

```

1 #bilangan prima NPM
2 print("Soal no 11")

```

```

3 index = 0
4
5 for i in NPM:
6     prima = True
7     var=int(NPM[index])
8     if(var<=1):
9         prima=False
10    for i in range(2,var):
11        if(var%i==0):
12            prima=False
13    if(prima==True):
14        print(var,"Prima")
15    else:
16        print(var,"bukan prima")
17 index += 1

```

**Listing 1.19** Bilangan Prima NPM

### 1.3 Penanganan Kesalahan (Error and Exception Handling)

Ada setidaknya dua jenis kesalahan berdasarkan kejadiannya:

1. Kesalahan sintaksis (syntax errors) atau sering disebut kesalahan penguraian (parsing errors)
2. Pengecualian (exceptions) atau sering disebut kesalahan saat beroperasi (run-time errors)

Kesalahan sintaksis terjadi ketika Python tidak dapat mengerti apa yang Anda perintahkan. Sedangkan pengecualian (kesalahan saat beroperasi) terjadi ketika Python mengerti apa yang Anda perintahkan tetapi mendapatkan masalah saat mengikuti yang Anda perintahkan (terjadi saat aplikasi sudah mulai beroperasi).

#### 1.3.1 Kesalahan Sintaksis

Kesalahan sintaksis biasanya sering terjadi saat Anda masih baru memulai belajar Python, misalnya contoh berikut adalah penempatan indentasi (spasi di awal) yang tidak sesuai.

```

print('salah indentasi')
File "<stdin>", line 1
    print('salah indentasi')
    ^
IndentationError: unexpected indent

```

Contoh berikut ini menampilkan kesalahan sintaksis, dimana setelah kondisi dari perintah while diharuskan ada tanda titik dua (:).

```
while True print('Hello world')
File "<stdin>", line 1
    while True print('Hello world')
^
SyntaxError: invalid syntax
```

ada kesalahan sintaksis, baris dimana kesalahan terdeteksi dimunculkan kembali, kemudian terdapat tanda panah yang menunjukkan titik awal dari kesalahan.

Kedua contoh di atas memiliki kelompok (tipe) kesalahan yang berbeda, yang pertama adalah `IndentationError` dan yang kedua adalah `SyntaxError`. Kemudian setelah penyebutannya, ada pesan detail kesalahan (keterangan), misalnya indentasi yang tidak diharapkan (`unexpected`).

Jika Anda menggunakan mode pemanggilan skrip, nama file skrip dan nomor baris dimana terjadi kesalahan akan dimunculkan. Sedangkan untuk mode interaktif pada dua contoh di atas, nama file muncul sebagai “`<stdin>`”. Berikut adalah contoh pada pemanggilan skrip bernama `contoh_salah_sintaksis.py` dimana terjadi kesalahan pada baris 2.

```
python contoh_salah_sintaksis.py
File "contoh_salah_sintaksis.py", line 2
    if True print('salah sintaksis')
^
SyntaxError: invalid syntax
```

### 1.3.2 Pengecualian

Meski pernyataan atau ekspresi dari Python sudah Anda tulis dengan benar, ada kemungkinan terjadi kesalahan ketika perintah tersebut dieksekusi. Kesalahan yang terjadi saat proses sedang berlangsung disebut pengecualian (exceptions) dan akan berakibat fatal jika tidak ditangani. Kebanyakan pengecualian di Python tidak ditangani oleh aplikasi, sehingga aplikasi terhenti kemudian muncul pesan kesalahan seperti contoh berikut.

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

Misalkan Anda lupa memberikan nilai pada variabel `angka`, tetapi Anda langsung memanggil variabel tersebut. Secara sintaksis sudah sesuai, tapi muncul pengecualian dengan kelompok (tipe) kesalahan `NameError` dan pesan detail kesalahan yang menyatakan bahwa variabel `angka` tidak terdefinisi.

Contoh lain terkait pengecualian yang sering juga terjadi adalah operasi dari variabel yang jenisnya tidak sesuai, misalnya contoh berikut.

```
bukan_angka = '1'  
bukan_angka + 2  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: can only concatenate str (not "int") to str
```

Pada contoh tersebut, variabel bukan\_angka berjenis string, sehingga saat mengoperasikan variabel tersebut dengan angka (berjenis integer), meskipun secara sintaksis sudah sesuai, muncul pengecualian dengan kelompok (tipe) kesalahan TypeError dan pesan detail kesalahan yang menyatakan bahwa operasi penambahan untuk string (concatetation) hanya bisa dilakukan jika kedua operannya adalah string (dan bukan integer).

Seperti terlihat bahwa pada saat terjadi pengecualian, informasi yang muncul seperti saat terjadi kesalahan (errors), termasuk juga informasi nama file dan nomor baris dimana kesalahan terjadi.

### 1.3.3 Penanganan Pengecualian

Pada aplikasi Python yang Anda buat bisa dilengkapi dengan penanganan terhadap pengecualian (exceptions handling) dari kelompok (tipe) kesalahan yang Anda tentukan. Proses penanganan pengecualian menggunakan pernyataan try yang berpasangan dengan except.

Misalnya kita ingin menangani pengecualian yang terjadi jika ada pembagian angka dengan nilai nol (0).

```
z = 0  
1 / z  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
ZeroDivisionError: division by zero  
  
try:  
... x = 1 / z  
... print(x)  
... except ZeroDivisionError:  
...     print('tidak bisa membagi angka dengan nilai nol')  
...  
tidak bisa membagi angka dengan nilai nol
```

Perhatikan bahwa operasi aplikasi berhenti di  $x = 1 / z$ , sedangkan bagian `print(x)` tidak sempat dioperasikan, karena aplikasi sudah mengalami pengecualian, sehingga yang tercetak adalah operasi `print('tidak bisa membagi angka dengan nilai nol')`.

Pada operasi yang dicontohkan di atas, penanganan pengecualian untuk `ZeroDivisionError` dilakukan sehingga aplikasi tidak lagi keluar dari eksekusi karena kesalahan, tapi digantikan dengan mencetak pesan ke layar. Pada contoh ini kita fokus pada penanganan pengecualian, meskipun ada cara lain untuk menyelesaiakannya, misal menggunakan kondisi (percabangan) untuk menghindari nilai nol.

Pernyataan `except` dilanjutkan dengan kelompok (tipe) kesalahan yang ingin ditangani, atau bisa juga berupa tuple dari satu atau lebih tipe kesalahan yang akan ditangani. Di contoh berikut, menangani `FileNotFoundException` sebagai tuple satu elemen, jangan lupa dalam menuliskan tuple satu elemen harus tetap diakhiri dengan koma.

```
>>> try:  
...     with open('contoh_tidak_ada.py') as file:  
...         print(file.read())  
... except (FileNotFoundException, ):  
...     print('file tidak ditemukan')  
...  
file tidak ditemukan
```

Pada operasi di atas, aplikasi akan membuka dan mengakses file bernama `contoh_tidak_ada.py`, tetapi file tersebut tidak ada di direktori dimana aplikasi Python tersebut berada, selanjutnya akan terjadi pengecualian (exceptions) tetapi ditangani, dalam pasangan pernyataan `try` dan `except`, sehingga aplikasi tidak terhenti tetapi tercetak di layar bahwa file tidak ditemukan.

Dalam aplikasi yang lebih kompleks, penanganan pengecualian dapat menggunakan pernyataan `except` lebih dari satu. Di contoh berikutnya akan menggunakan pernyataan `except` lebih dari satu (untuk satu pernyataan `try`), maupun menggunakan satu pernyataan `except` yang menangani lebih dari satu tipe kesalahan yang digabung dalam sebuah tuple.

```
>>> d = {'ratarata': '10.0'}  
>>> try:  
...     print('rata-rata: {}'.format(d['rata_rata']))  
... except KeyError:  
...     print('kunci tidak ditemukan di dictionary')  
... except ValueError:  
...     print('nilai tidak sesuai')  
...  
...
```

```
kunci tidak ditemukan di dictionary
>>> try:
...     print('rata-rata: {}'.format(d['ratarata']/3))
... except KeyError:
...     print('kunci tidak ditemukan di dictionary')
... except (ValueError, TypeError):
...     print('nilai atau tipe tidak sesuai')
...
nilai atau tipe tidak sesuai
>>> try:
...     print('pembulatan rata-rata: {}'.format(int(d['ratarata'])))
... except (ValueError, TypeError) as e:
...     print('penangan kesalahan: {}'.format(e))
...
penangan kesalahan: invalid literal for int() with base 10: '10'
```

Pada contoh tersebut, yang paling awal terjadi pengecualian untuk tipe kesalahan `KeyError` karena dalam `dictionary d` tidak memiliki kunci (`key`) `rata_rata`, yang ada adalah kunci `ratarata`.

Kemudian contoh selanjutnya terjadi pengecualian untuk tipe kesalahan `TypeError` karena nilai `d['ratarata']` memiliki tipe string, sehingga tidak dapat dibagi dengan integer (angka) 3. Dalam penanganan kesalahannya, satu buah pernyataan `except` menangani tipe kesalahan `ValueError` atau `TypeError`, sehingga cocok salah satunya akan menampilkan ke layar bahwa nilai atau tipe tidak sesuai.

Di bagian paling akhir contoh, terjadi pengecualian untuk tipe kesalahan `ValueError` karena berusaha melakukan konversi (casting) dari sebuah string ke integer dengan format yang tidak sesuai (bilangan bulat seharusnya tidak memiliki titik dalam penulisannya). Dalam penulisan penanganan kesalahannya digunakan variasi lain untuk mendapatkan pesan kesalahan sebagai variabel `e` untuk kemudian variabel tersebut dicetak dalam pesan yang ditampilkan ke layar.

### 1.3.4 Menghasilkan Pengecualian

Dalam membuat aplikasi, ada kemungkinan Anda butuh untuk menghasilkan pengecualian (`raise exceptions`), salah satu caranya bisa dengan menggunakan pengecualian yang sudah ada, hanya ditambahkan informasi detailnya saja.

Misalnya dalam contoh berikut, Anda wajibkan sebuah `dictionary` memiliki kunci (`key`) `total`.

```
>>> d = {'ratarata': '10.0'}
```

```
>>> if 'total' not in d:
...     raise KeyError('harus memiliki total')
...
Traceback (most recent call last):
File "<stdin>", line 2, in <module>
KeyError: 'harus memiliki total'
```

1. NameError, terjadi apabila kode mengeksekusi nama yang tidak terdefenisikan. Contoh:

```
nama = "Dinda Majesty"
print(Nama)
```

Maka akan menghasilkan output NameError: name 'Nama' is not defined. error ini dapat diatasi dengan mengubah variabel yang di print sesuai dengan variabel yang didefinisikan, karena penulisan pada python bersifat case-sensitive

2. SyntaxError, terjadi apabila kode python mengalami kesalahan saat penulisan. Contoh: menuliskan variabel yang didahului angka (1nama = "Dinda Majesty") maka akan muncul eror SyntaxError: invalid syntax. error ini dapat diatasi dengan memperhatikan tata cara penulisan kode pada bahasa pemrograman python.
3. TypeError, terjadi apabila kode melakukan operasi atau fungsi terhadap tipe data yang tidak sesuai. Contoh: melakukan penjumlahan terhadap tipe data string dan integer. eror ini dapat diatasi dengan mengubah tipe data string menjadi integer.

```
a = "10"
b = 5

print(a + b)
```

Maka akan menghasilkan output eror TypeError: can only concatenate str (not "int") to str

4. IndentationError, terjadi apabila kode perulangan atau pengkondisian tidak menjorok kedalam (tidak menggunakan identasi), error ini dapat diatasi dengan menambahkan tab atau spasi. Contoh:

```
a = 200
b = 330
if b > a:
    print("b lebih besar dari a")
```

Maka akan menghasilkan output eror IndentationError: expected an indented block

### 1.3.5 Except

Try Except merupakan perintah yang bisa digunakan dalam penanganan error pada bahasa pemrograman python. perintah ini biasanya digunakan saat penanganan error input/output, operasi database, pengaksesan indeks suatu list atau dictionary dan berbagai kasus lainnya.

Contoh sederhana penggunaan Try-Except saat menangani NameError

```
1 #try except
2 try :
3     print(x)
4 except NameError:
5     print("Variable x tidak ada")
6 except:
7     print("ada sesuatu yang salah nih")
```

**Listing 1.20** Try Except

Pada contoh diatas, try except akan menghasilkan output Variabel x tidak ada, karena kita tidak mendefinisikan variabel x sebelum kita menampilkan output x ke layar. Dengan menggunakan try except kode yang kita buat terhindar dari error dan kita bisa mengetahui kesalahan yang terjadi ketika terdapat error pada kode kita.

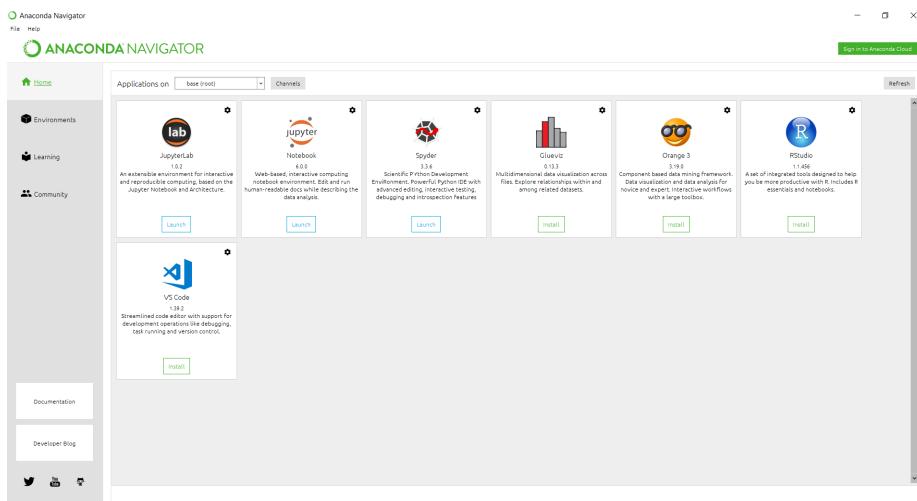
### 1.3.6 Run Script Hello World di Spyder

Jika teman-teman telah berhasil melakukan instalasi anaconda3 maka teman-teman akan memiliki software spyder, pada software ini teman-teman akan mengetikkan kode program yang akan teman-teman buat nantinya. Sekarang ayo kita coba menjalankan program untuk mencetak kata *hello world*.

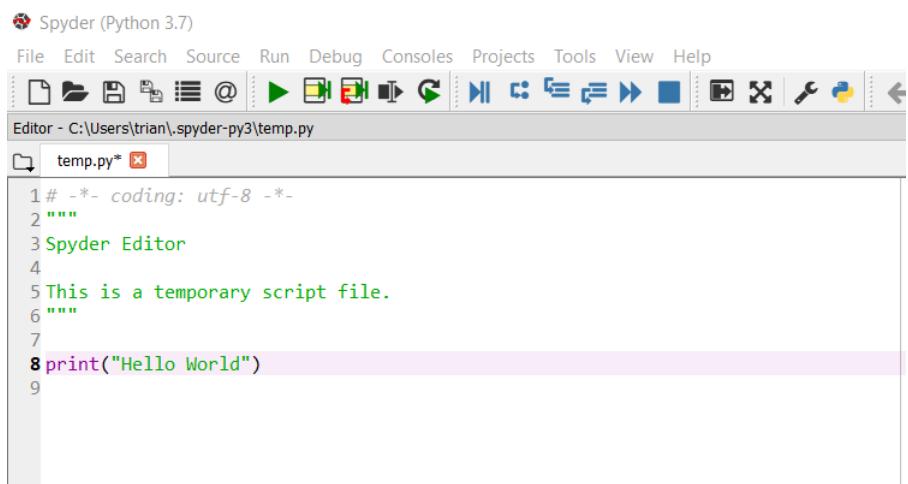
1. Buka anaconda navigator, lalu klik launch seperti pada gambar 1.2.
2. ketikkan `print("Hello World")` dan run spyder dengan cara mengklik tombol berwarna hijau yang terletak ditengah toolbar, untuk lebih jelas dapat dilihat pada gambar 1.3.
3. hasilnya akan tampak seperti pada gambar 1.4

### 1.3.7 Pemakaian Variable Explorer

Variable explorer akan secara otomatis terisi ketika kita membuat variable, pada variable explorer kita bisa melihat nama variable, tipe data, length, dan value dari variable tersebut. Contoh penggunaan variabel explorer dapat teman-teman lihat pada gambar 1.5.



**Gambar 1.2** *Launch Spyder*



**Gambar 1.3** *Print Hello World*

---

```

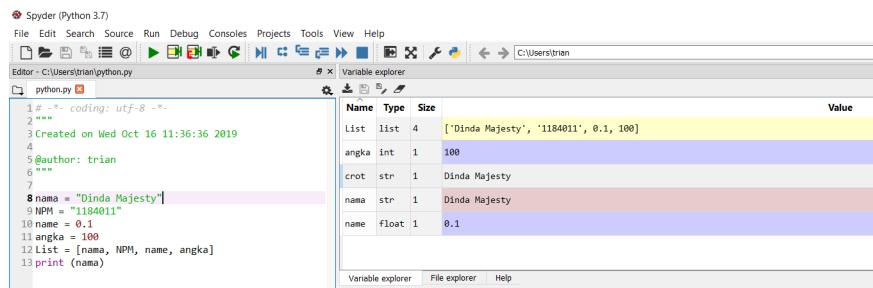
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.6.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/trian/.spyder-py3/temp.py', wdir='C:/Users/trian/.spyder-py3')
Hello World

```

**Gambar 1.4** *Hello World*



**Gambar 1.5 Variable Explorer**

## 1.4 Indentasi

### 1.4.1 Penjelasan Indentasi

Indentasi adalah bagian paragraf yang menjorok ke dalam pada baris-baris paragraf. Mengatur indentasi dapat menggunakan tab atau spasi. Indentasi digunakan oleh bahasa pemrograman python sebagai pengganti briket () untuk membuka dan menutup fungsi. Error indentasi dapat terjadi apabila syntax tidak menggunakan tab atau space. Contoh yang benar (menggunakan tab/spasi sebagai indentasi):

```
# blok percabangan if
if username == 'petanikode':
    print("Selamat Datang Admin")
    print("Silahkan ambil tempat duduk")

# blok percabangan for
for i in range(10):
    print i
```

Contoh yang salah (tidak menggunakan tab/spasi):

```
# blok percabangan if
if username == 'petanikode':
print("Selamat Datang Admin")
print("Silahkan ambil tempat duduk")

# blok percabangan for
for i in range(10):
print i
```

### 1.4.2 Jenis-Jenis Error Indentasi

IndentationError: unexpected indent. Error diatas terjadi apabila syntax kekurangan tab atau spasi. Contoh error indentasi dapat dilihat pada gambar 1.6. Apabila di

The screenshot shows the Spyder Python 3.7 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. Below the menu is a toolbar with various icons. The main area is titled 'Editor - C:\Users\trian\send\_whatsapp\coba.py'. It contains two tabs: 'WA.py' and 'coba.py'. The code in 'coba.py' is as follows:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Oct 16 18:13:23 2019
4
5 @author: trian
6 """
7 username = input()
8 # blok percabangan if
9 if username == 'petanikode':
10    print("Selamat Datang Admin")
11    print("Silahkan ambil tempat duduk")
12
13 # blok percabangan for
14 for i in range(10):
15    print(i)

```

Line 10, which contains the opening brace of the if block, is highlighted in yellow.

**Gambar 1.6 Indentasi**

running akan memunculkan error seperti pada gambar 1.7.

```

In [6]: runfile('C:/Users/trian/send_whatsapp/coba.py', wdir='C:/Users/trian/
send_whatsapp')
Traceback (most recent call last):

  File "C:\Users\trian\Anaconda3\lib\site-packages\IPython\core
\interactiveshell.py", line 3325, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)

  File "<ipython-input-6-0a8945c5cd6>", line 1, in <module>
    runfile('C:/Users/trian/send_whatsapp/coba.py', wdir='C:/Users/trian/
send_whatsapp')

  File "C:\Users\trian\Anaconda3\lib\site-packages\spyder_kernels\customize
\spydercustomize.py", line 827, in runfile
    execfile(filename, namespace)

  File "C:\Users\trian\Anaconda3\lib\site-packages\spyder_kernels\customize
\spydercustomize.py", line 110, in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "C:/Users/trian/send_whatsapp/coba.py", line 10
    print("Selamat Datang Admin")
           ^
IndentationError: expected an indented block

```

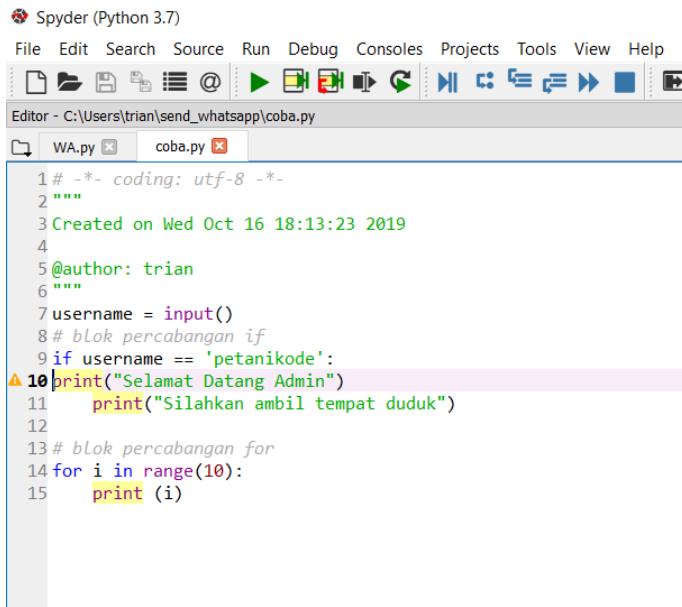
**Gambar 1.7 Error Indentasi**

### 1.4.3 Cara Membaca Error

Jika terjadi error maka cari di line berapa error terjadi, pada gambar 1.7 terdapat error indentasi pada line 10.

#### 1.4.4 Cara Menangani Error

Menangani error indentasi dapat dilakukan dengan cara menambahkan tab atau space pada line yang error. Untuk lebih jelasnya dapat teman-teman lihat pada gambar 1.8.



The screenshot shows the Spyder Python 3.7 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar has various icons for file operations like Open, Save, and Run. The status bar at the bottom shows 'Editor - C:\Users\trian\send\_whatsapp\coba.py'. The code editor window displays a Python script named 'coba.py' with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Oct 16 18:13:23 2019
4
5 @author: trian
6 """
7 username = input()
8 # blok percabangan if
9 if username == 'petanikode':
10|print("Selamat Datang Admin")
11     print("Silahkan ambil tempat duduk")
12
13# blok percabangan for
14for i in range(10):
15    print (i)
```

A syntax error is highlighted in line 10, where the opening parenthesis of the print statement is missing. The line '10|print("Selamat Datang Admin")' is shown with a yellow background and a red warning triangle icon to its left.

Gambar 1.8 Syntax Error

Penulisan syntax identasi yang benar dapat dilihat pada gambar 1.9.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Oct 16 18:13:23 2019
4
5 @author: trian
6 """
7 username = input()
8 # blok percabangan if
9 if username == 'petanikode':
10     print("Selamat Datang Admin")
11     print("Silahkan ambil tempat duduk")
12
13 # blok percabangan for
14 for i in range(10):
15     print(i)

```

**Gambar 1.9** Syntax yang Telah Diperbaiki

#### 1.4.5 Quiz: 1

1. Siapakah pencipta Python?
  - a) Guido van Rossum
  - b) Guido van Persie
  - c) Guido van Linguini
  - d) Guido van Laptop
  
2. Pada tahun berapa python dikembangkan?
  - a) 1999
  - b) 1995
  - c) 1945
  - d) 1990
  
3. Apa itu indentasi?
  - a) Bagian paragraf yang menjorok ke dalam
  - b) Bagian paragraf yang menjorok ke sungai
  - c) Bagian paragraf yang menjorok ke laut
  - d) Bagian paragraf yang menjorok ke hati
  
4. Apa itu try except?
  - a) Perintah untuk penanganan masalah hidup
  - b) Perintah untuk penanganan masalah error
  - c) Perintah untuk penanganan masalah galau
  - d) Perintah untuk penanganan masalah sakit perut

5. Apa itu if statement?

- a) Kondisi yang digunakan untuk percabangan logika
- b) Kondisi yang digunakan untuk percabangan pohon
- c) Kondisi yang digunakan untuk percabangan tunas
- d) Kondisi yang digunakan untuk percabangan akar



## BAB 2

---

# ANACONDA

---

### 2.1 Pengenalan Anaconda

Anaconda merupakan sebuah software yang mendistribusikan bahasa pemrograman python dan R untuk keperluan komputasi ilmiah seperti data science, machine learning, data processing skala-luas, analisis prediksi, dan lain sebagainya. Anaconda memiliki anaconda navigator yang didalamnya terdapat software-software yang dapat digunakan untuk membuat program python dan R. Salah satu software yang akan kita gunakan yaitu Spyder. Sebelum kita membuat program kita harus menginstall anaconda terlebih dahulu. Instalasi pada kali ini menggunakan 2 sistem operasi yaitu Windows 10 x64 dan Ubuntu 19.04. Hal yang dibutuhkan adalah laptop dengan os windows atau ubuntu dan internet.

#### 2.1.1 Instalasi Anaconda 3 Windows 10 x64

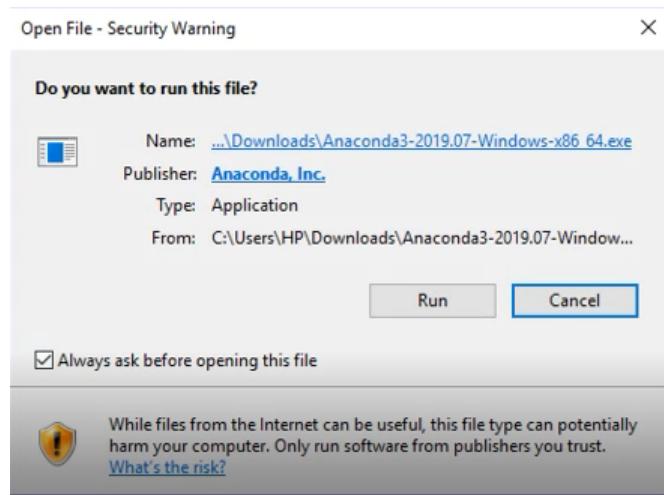
Hal yang harus diperhatikan sebelum melakukan instalasi *Anaconda Python*

1. *Download Anaconda Python* <https://www.anaconda.com/distribution/>
2. Perhatikan versi dari sistem operasi yang digunakan (versi 32bit atau 64bit)

3. Download file anaconda yang sesuai dengan versi sistem operasi (32bit atau 64bit)

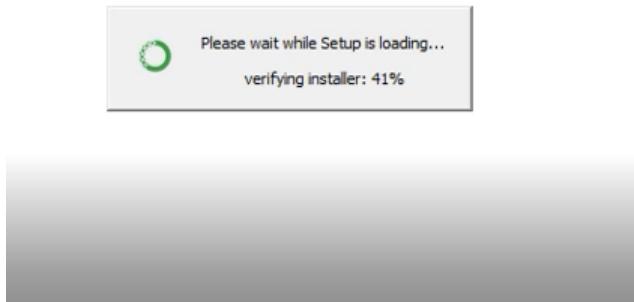
Berikut langkah-langkah instalasi anaconda.

1. Buka aplikasi *installer Anaconda* yang telah didownload lalu akan muncul gambar 2.1, lalu pilih run untuk menjalankan proses instalasi.



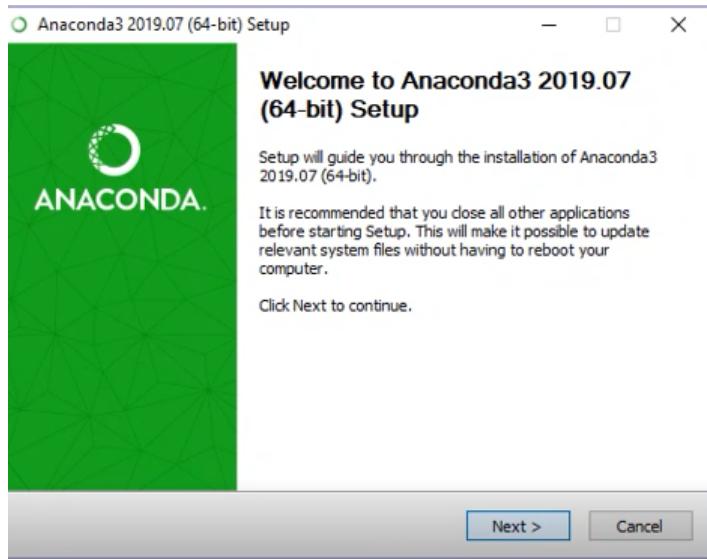
**Gambar 2.1** Run Setup Anaconda

2. Tunggu hingga *setup loading* selesai seperti pada gambar 2.2.



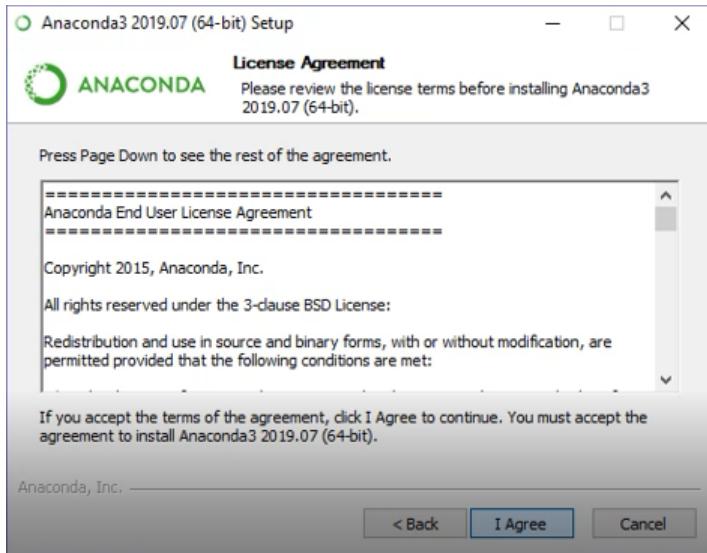
**Gambar 2.2** Setup Loading

3. Jika *setup loading* telah selesai, maka klik *next* seperti pada gambar 2.3.



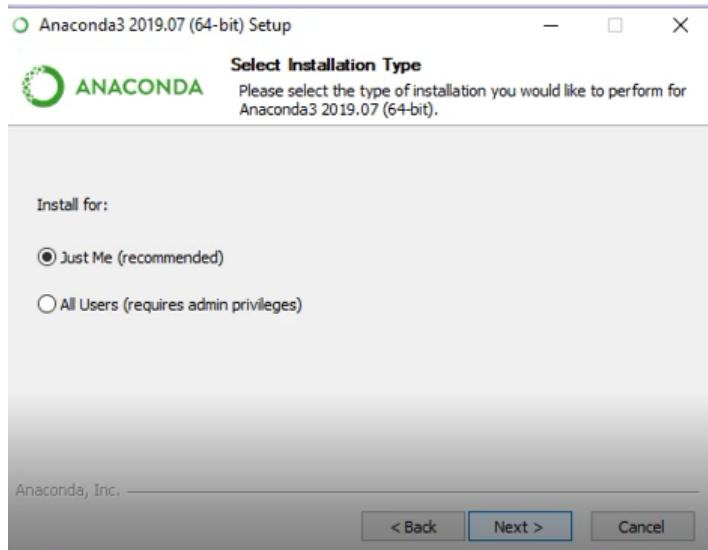
**Gambar 2.3** Welcome to Anaconda Setup

4. Pada *License Agreement* klik *I Agree* karena jika teman-teman tidak menyetujui lisensi anaconda maka teman-teman tidak akan bisa melanjutkan proses instalasi. lakukan langkah ini seperti pada gambar 2.4



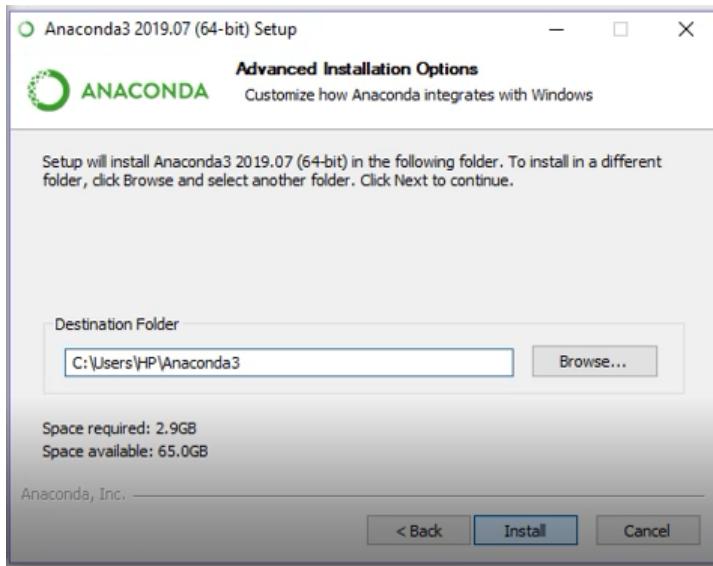
**Gambar 2.4** License Agreement

5. Kemudian pilih *Just Me(Recomended)* agar sesuai dengan komputer yang digunakan, kemudian klik *next* seperti pada gambar 2.5.



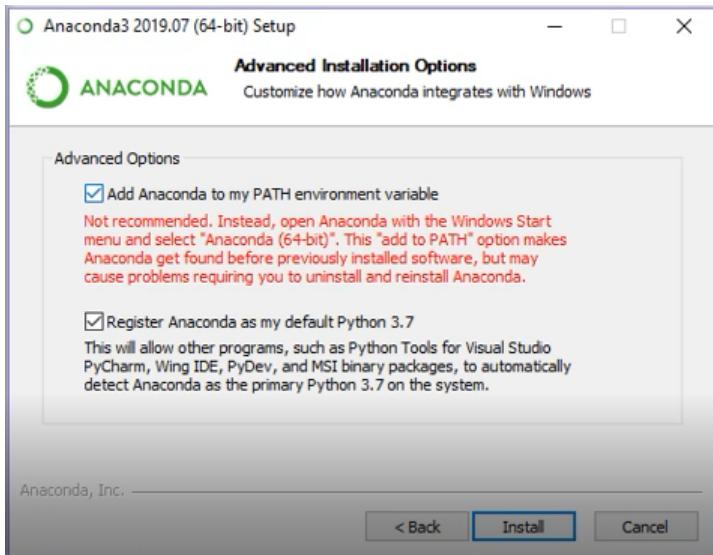
**Gambar 2.5** *Just Me(recomended)*

6. Kemudian pilih direktori tempat kita akan *menginstall anaconda* seperti pada gambar 2.6



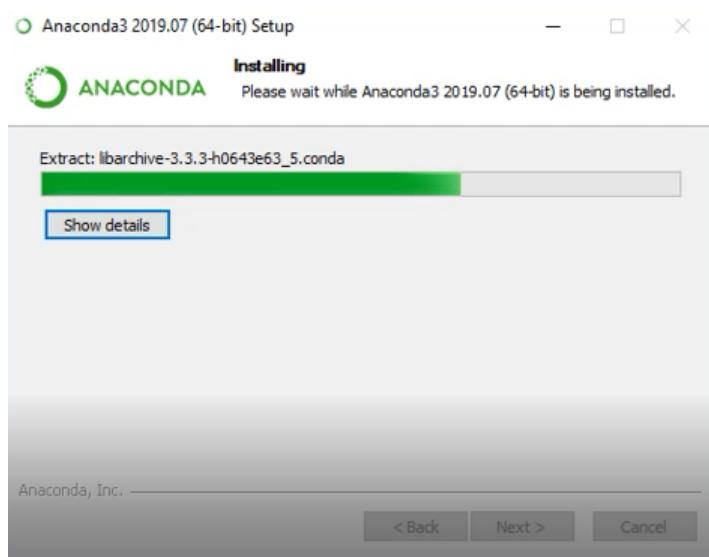
Gambar 2.6 Pilih lokasi

7. Kemudian centang *Add Anaconda to my Path environment variable*, agar saat melakukan instalasi package anaconda, package tersebut akan langsung tertuju ke path anaconda tidak ke aplikasi yang lain. kemudian Klik *install* seperti pada gambar 2.7.



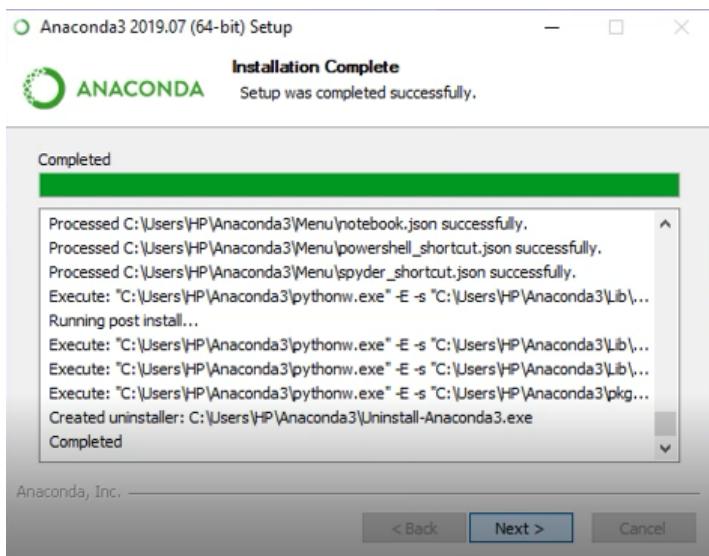
Gambar 2.7 Centang Anaconda to my PATH

8. Tunggu sampai proses *installasi* selesai seperti pada gambar 2.8.



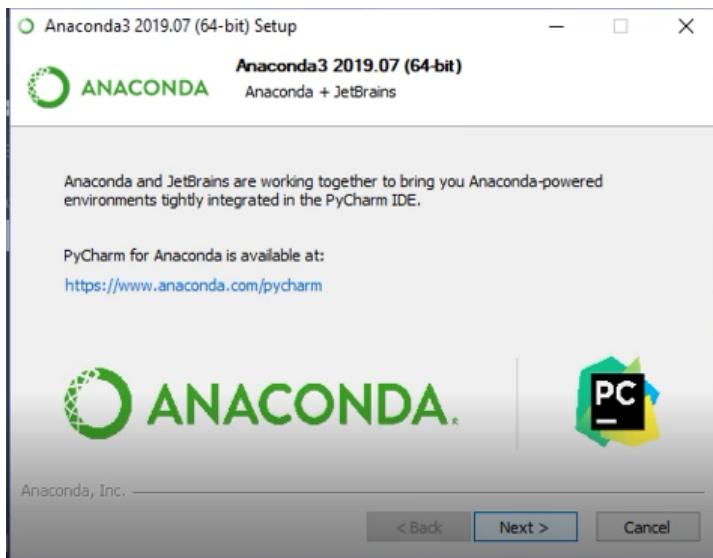
Gambar 2.8 Waiting Installation Complete

9. Apabila instalasi telah selesai maka akan terlihat seperti gambar 2.9, kemudian klik *next*



Gambar 2.9 Installation Complete

10. apabila muncul gambar 2.10, maka klik *next*



**Gambar 2.10**    *Anaconda+JetBrains*

11. Jika instalasi telah selesai maka akan ada ucapan terima kasih telah menginstall anaconda 3 seperti pada gambar 2.11, hal ini menandakan bahwa teman-teman telah selesai dan berhasil melakukan instalasi anaconda. Kemudian klik *finish* untuk mengakhiri instalasi.



**Gambar 2.11** *Thanks for install Anaconda*

### 2.1.2 Update Anaconda dan Spyder

Kenapa kita harus melakukan update anaconda dan spyder? melakukan update diperlukan agar software yang kita gunakan merupakan software yang terbaru, karena versi lama dan versi baru akan memiliki banyak perbedaan dan akan menjadi masalah nantinya ketika kita membuat program atau mengimportkan modul-modul python yang digunakan. Berikut cara mengupdate Spyder:

1. Buka anaconda prompt, lalu ketikkan perintah conda update spyder
2. Konfirmasi update dengan mengetikkan y, lalu tekan enter
3. Tunggu hingga installan selesai

Berikut cara mengupdate Anaconda:

1. Buka anaconda prompt, lalu ketikkan perintah conda update anaconda
2. Konfirmasi update anaconda dengan mengetikkan y dan kemudian tekan enter
3. Tunggu hingga installan selesai

### 2.1.3 Instalasi Anaconda Ubuntu 19.04

Untuk instalasi python pada Ubuntu 19.04 dibutuhkan sebagai berikut:

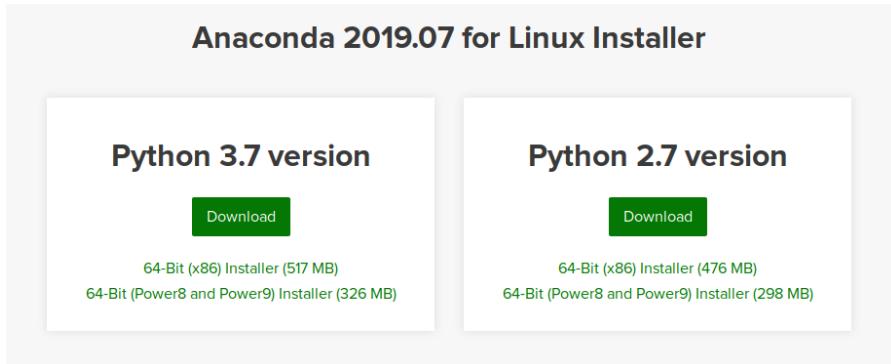
1. Internet

2. Anaconda installer (64bit or 32bit)

3. enter, dan yes atau no

Ikuti langkah berikut:

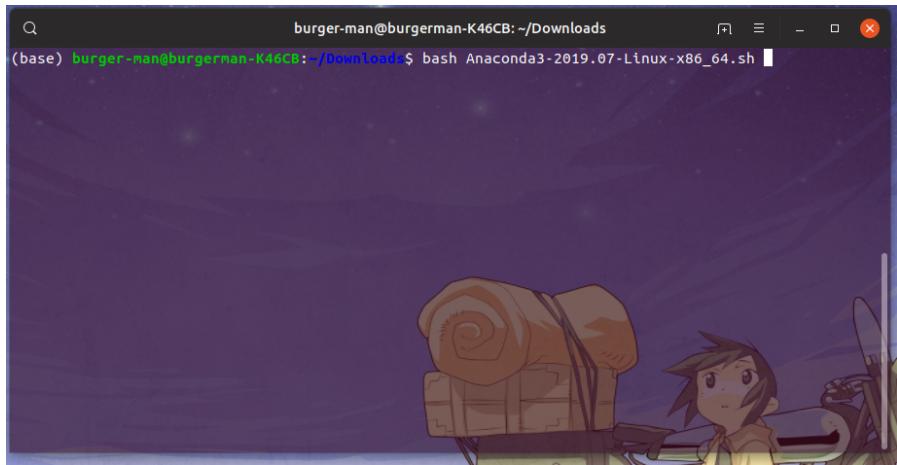
1. Pertama kita kunjungi situs <https://www.anaconda.com/distribution/#download-section> seperti gambar 2.12 dan pilih **64-Bit (x86) Installer (517 MB)**



**Gambar 2.12** Gambar halaman download

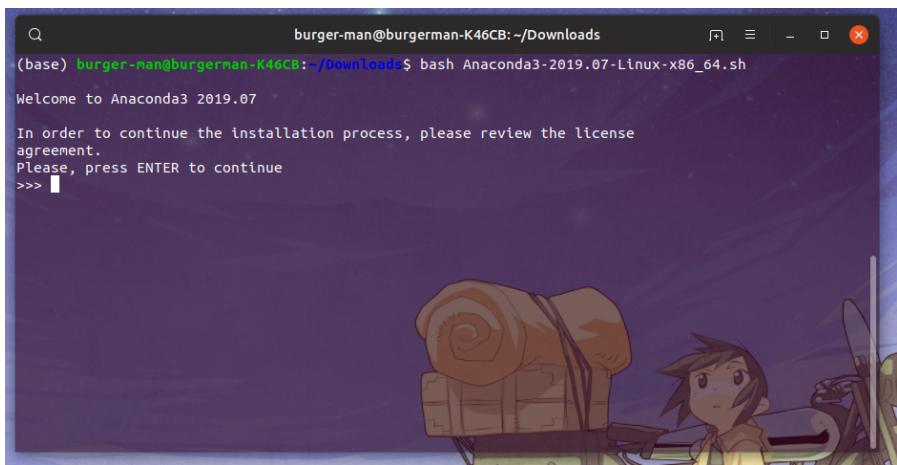
2. Kedua kita buka **terminal** kita lalu arahkan ke direktori kita menyimpan file download anaconda

3. Ketiga kita ketikkan sebagai berikut **bash namafileanaconda.sh** lalu enter, contoh seberti gambar 2.13



**Gambar 2.13** Gambar install anaconda

- Setelah itu, tekan **ENTER** saja seperti gambar 2.14



**Gambar 2.14** Gambar eksekusi anaconda

- Lalu akan muncul sebuah tulisan **End User License Agreement** seperti gambar 2.15, tekan **ENTER** dan tahan hingga seperti gambar



**Gambar 2.15** Gambar anaconda license agreement

**Gambar 2.16** Gambar perintah yes or no

6. Lalu setelah muncul perintah '**yes**' or '**no**' ketik **yes** lalu enter
  7. Setelah itu muncul path direktori instalasi anaconda kita seperti gambar 2.17 lalu tekan enter

```
Q burger-man@burgerman-K46CB: ~/Downloads
Please answer 'yes' or 'no':'
>>>
Please answer 'yes' or 'no':'
>>> yes

Anaconda3 will now be installed into this location:
/home/burger-man/anaconda3

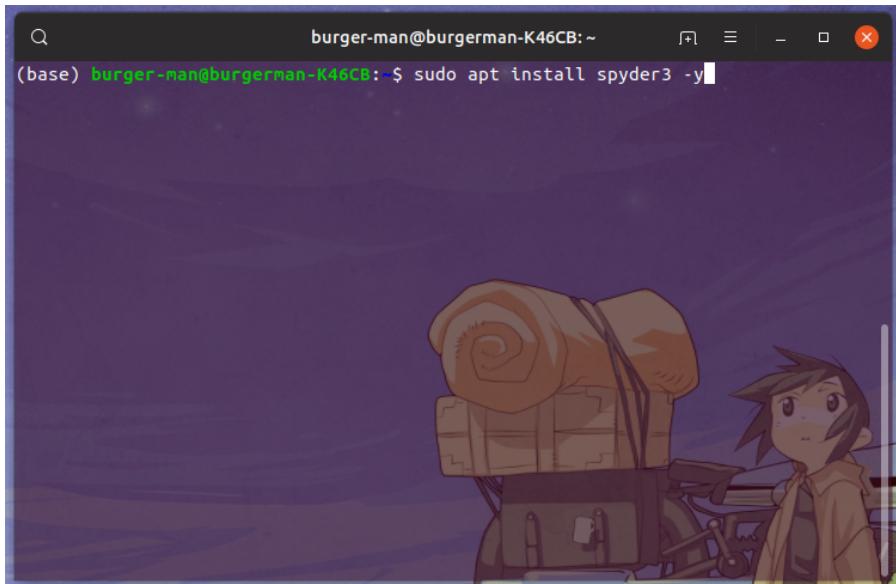
- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

[home/burger-man/anaconda3] >>> |
```

**Gambar 2.17** Gambar path anaconda

Setelah kita selesai instalasi anaconda jangan lupa juga untuk menginstal spyder ide, caranya seperti berikut:

- (a) ketikkan perintah `sudo apt install spyder3 -y` seperti gambar 2.18



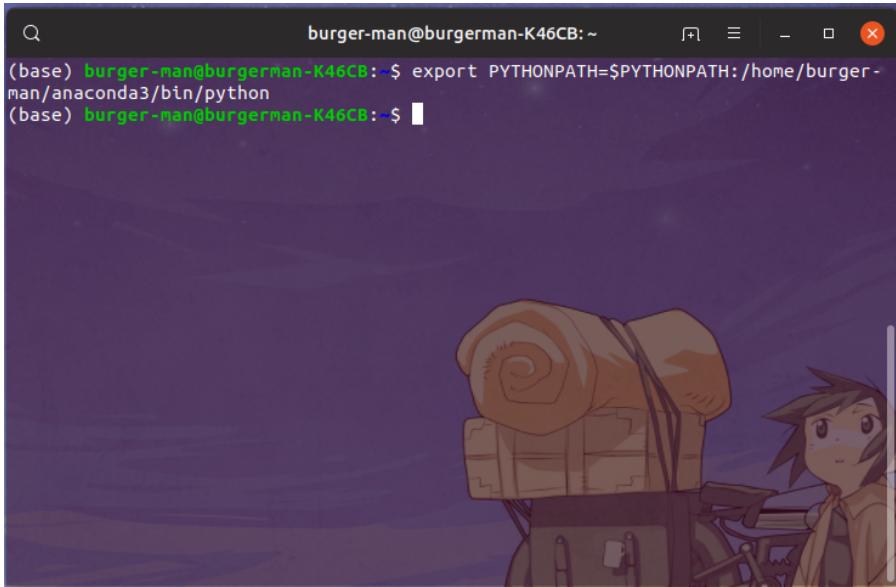
**Gambar 2.18** Gambar perintah install spyder3

(b) lalu jalankan dengan perintah **spyder** atau **spyder3**

#### 2.1.4 Konfigurasi *Python*

Setelah kita selesai instal Anaconda dan Spyder, selanjutnya kita akan mempelajari bagaimana cara setting environments python kita? caranya sebagai berikut

1. pertama kita buka terminal kita lalu ketikkan perintah **export PYTHONPATH=\$PYTHONPATH:./** contoh seperti gambar 2.19, lalu enter



**Gambar 2.19** Gambar setpath

### 2.1.5 Quiz 2

1. Sistem operasi apa yang digunakan untuk instalasi pada buku ini?
  - a) Ubuntu dan Windows
  - b) Usus buntu dan Jendela
  - c) Penguin dan Kaca
  - d) Hewan dan Benda
2. Apa yang harus diperhatikan ketika ingin download Anaconda?
  - a) Kamu
  - b) Dia
  - c) Aku
  - d) Versi Sistem Operasi
3. Ubuntu versi berapa pada instalasi buku ini??
  - a) 19.04
  - b) 19.10
  - c) 19.99
  - d) 19.90
4. Apakah berbeda instalasi Anaconda pada windows dan linux? a) Ya  
b) Tidak

5. Apa yang kita memerlukan internet untuk instal Anaconda?
  - a) Ya
  - b) Tidak

# BAB 3

---

## PIP DAN PERINTAH CLI

---

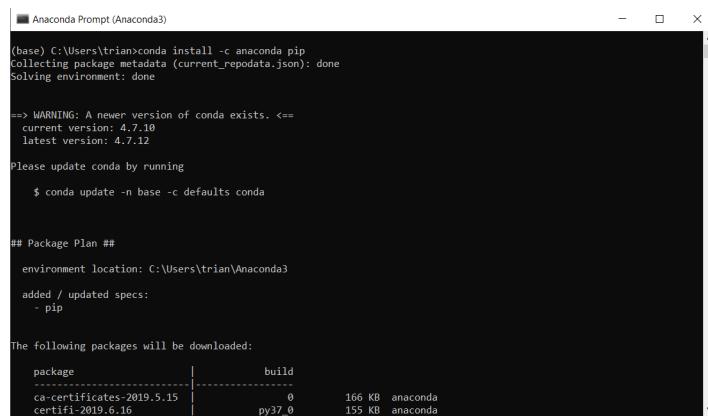
### 3.1 Pip

Pip merupakan modul atau paket yang harus kita miliki apabila kita menggunakan bahasa pemrograman python. Pip digunakan untuk menginstal package-package python yang akan kita gunakan dalam pembuatan kode program.

#### 3.1.1 Instalasi Pip

Cara melakukan instalasi pip pada anaconda CLI:

1. buka anaconda prompt (Start -> Anaconda Prompt)
2. ketikkan conda install -c anaconda pip seperti pada gambar 3.1.



```
[base] C:\Users\trian>conda install -c anaconda pip
Collecting package metadata (current_repodata.json): done
Solving environment: done

=> WARNING: A newer version of conda exists. <=>
  current version: 4.7.10
  latest version: 4.7.12

Please update conda by running

$ conda update -n base -c defaults conda

## Package Plan ##

environment location: C:\Users\trian\Anaconda3

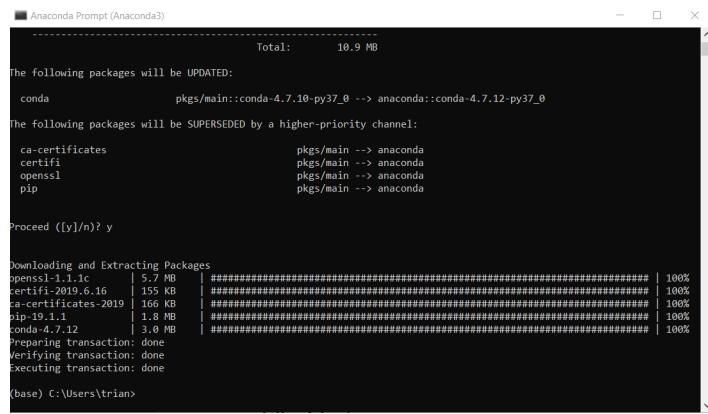
added / updated specs:
- pip

The following packages will be downloaded:

  package          |           build
ca-certificates-2019.5.15 |             0    166 KB  anaconda
certifi-2019.6.16        | py37_0      155 KB  anaconda
```

**Gambar 3.1** *Install pip*

3. ketik y, lalu enter. Tunggu hingga proses instalasi selesai seperti pada gambar 3.2.



```
[base] C:\Users\trian>
Total:      10.9 MB

The following packages will be UPDATED:
  conda          pkgs/main::conda-4.7.10-py37_0 --> anaconda:::conda-4.7.12-py37_0

The following packages will be SUPERSEDED by a higher-priority channel:
  ca-certificates          pkgs/main --> anaconda
  certifi                  pkgs/main --> anaconda
  openssl                 pkgs/main --> anaconda
  pip                      pkgs/main --> anaconda

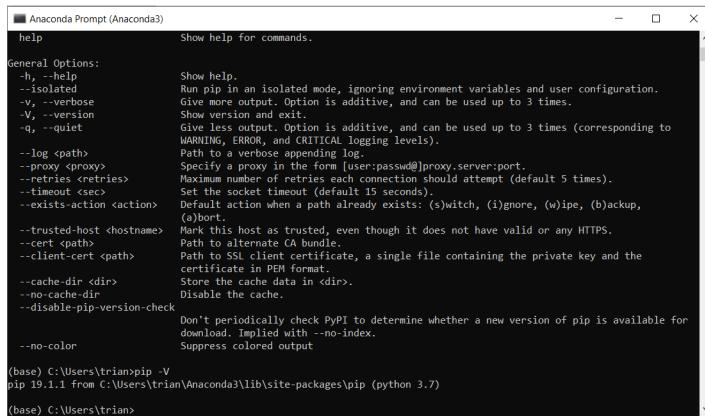
Proceed ([y]/n)? y

Downloading and Extracting Packages
openssl-1.1.1c          | 5.7 MB | #####| #####| #####| #####| 100%
certifi-2019.6.16        | 155 KB | #####| #####| #####| #####| 100%
ca-certificates-2019     | 166 KB | #####| #####| #####| #####| 100%
pip-19.1.1               | 1.8 MB | #####| #####| #####| #####| 100%
conda-4.7.12              | 3.0 MB | #####| #####| #####| #####| 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(base) C:\Users\trian>
```

**Gambar 3.2** *Install pip Selesai*

4. jika telah selesai, lakukan pengecekan versi pip dengan mengetikkan pip -V seperti pada gambar 3.3.



```

Anaconda Prompt (Anaconda3)
help           Show help for commands.

General Options:
-h, --help      Show help.
--isolated     Run pip in an isolated mode, ignoring environment variables and user configuration.
-v, --verbose   Give more output. Option is additive, and can be used up to 3 times.
-V, --version   Shows version and exits.
-q, --quiet    Give less output. Option is additive, and can be used up to 3 times (corresponding to
               WARNING, ERROR, and CRITICAL logging levels).
--log <path>  Path to a verbose appending log.
--proxy <proxy> Specify a proxy in the form [user:password@]proxy.server:port.
--retries <retries> Maximum number of retries each connection should attempt (default 5 times).
--timeout <sec> Set the socket timeout (default 5 seconds).
--exists-action <action> Default action when a path already exists: (s)witch, (i)gnore, (w)ipe, (b)ackup,
(a)bort.
--trusted-host <hostname> Mark this host as trusted, even though it does not have valid or any HTTPS.
--cert <path>  Path to alternative CA bundle.
--client-cert <path> Path to SSL client certificate, a single file containing the private key and the
certificate in PEM format.
--cache-dir <dir> Store the cache data in <dir>.
--no-cache-dir Disable the cache.
--disable-pip-version-check Don't periodically check PyPI to determine whether a new version of pip is available for
download. Implied with --no-index.
--no-color     Suppress colored output

(base) C:\Users\trian>pip -V
pip 19.1.1 from C:\Users\trian\Anaconda3\lib\site-packages\pip (python 3.7)

(base) C:\Users\trian>

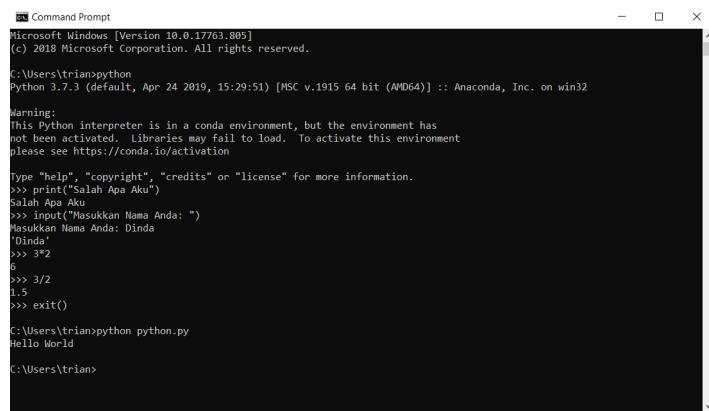
```

**Gambar 3.3 Melihat Versi pip**

### 3.1.2 Command Line Interface/Interpreter

Command line interpreter atau yang biasa teman-teman ketahui sebagai command prompt memungkinkan kita untuk menuliskan baris perintah yang akan dijalankan oleh komputer. Command line interpreter hanya berupa script atau tulisan kode program, berbeda dengan graphic user interface atau GUI yang memungkinkan kita memerintahkan sesuatu kepada komputer dengan hanya menggunakan tombol-tombol dan tampilan yang mudah dipahami. Cara menjalankan program python pada CLI:

1. Buka command prompt lalu ketikkan python seperti pada gambar 6.19.
2. Buatlah perintah print, input, perkalian, dan pembagian seperti pada gambar 6.19.
3. Bisa juga menjalankan file .py yang telah dibuat di IDE dengan cara python namafile.py, lalu klik enter seperti pada gambar 6.19.



The screenshot shows a Microsoft Windows Command Prompt window titled "Command Prompt". The window displays the following text:

```
Microsoft Windows [Version 10.0.17763.805]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\trian>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.

>>> print("Salah Apa Aku")
Salah Apa Aku
>>> input("Masukkan Nama Anda: ")
Masukkan Nama Anda: Dinda
'Dinda'
>>> 3**2
9
>>> 3/2
1.5
>>> exit()

C:\Users\trian>python python.py
Hello World

C:\Users\trian>
```

**Gambar 3.4** *CLI in Command Prompt*

## BAB 4

---

# FUNGSI, METHOD, DAN KELAS

---

### 4.1 Fungsi

Fungsi adalah sebuah blok kode yang memiliki nama fungsi dan kode program di dalamnya. Fungsi dapat dipanggil berkali-kali sesuai dengan nama fungsi yang telah didefinisikan. Fungsi memiliki nilai kembalian (return).

contoh fungsi:

```
def my_biodata(nama, umur) :  
    bio = "nama saya " + nama + " umur saya " + umur  
    return bio
```

Inputan pada fungsi berada di dalam (). Contoh (str), ini merupakan inputan yang terdapat pada fungsi. Return merupakan kembalian dari fungsi. Misalnya return nama, maka program akan mengembalikan string yang terdapat dalam variabel nama yaitu "Nama Saya Dinda Majesty".

## 4.2 Package

Package merupakan sekumpulan modul yang dikemas oleh programmer dengan tujuan agar mempermudah dalam pembuatan kode program. Kita dapat membuat sebuah kode program atau fungsi didalamnya dan dapat secara mudah menggunakan kode program itu dengan cara memanggilnya pada kode program lainnya atau import package.

Contoh package:

```
def my_biodata(nama, umur):
    bio = "nama saya " + nama + " umur saya " + umur
    return bio
def my_study(kampus, prodi):
    study = "saya berkuliah di " + kampus + " program studi " + prodi
    return study
```

Kode diatas merupakan isi dari file fungsi.py, sedangkan saya ingin menjalankan program fungsi.py pada main.py sehingga kode program pada file main.py akan dituliskan seperti berikut:

```
import fungsi

nama = "Dinda Majesty"
umur = "19 Tahun"
biodata = my_biodata(nama, umur)
print(biodata)

kampus = "Politeknik Pos Indonesia"
prodi = "D4-Teknik Informatika"
kuliah = my_study(kampus, prodi)
print(kuliah)
```

Kode program pada file main.py akan mengimport kode program yang ada pada file fungsi.py, sehingga dengan adanya fungsi dan package kita dapat dengan mudah melakukan pemanggilan fungsi yang telah kita deskripsikan sebelumnya, walaupun berada pada file python yang berbeda.

## 4.3 Kelas, Objek, Atribut, dan Method

Kelas merupakan blueprint dari sebuah objek atau kode program yang berisi fungsi dan dibuat untuk mendefenisikan objek dengan atribut yang sesuai dengan kelas yang telah dibuat.

Objek merupakan wujud dari kelas. Sebuah kelas harus memiliki objek yang nantinya akan di kodekan sesuai dengan fungsi yang telah dibuat pada kelas, tanpa adanya objek sebuah kelas tidak akan bisa menjalankan fungsi-fungsi didalamnya.

Atribut berisi variabel yang memiliki tipe data dan dapat kita berikan pada objek.

Method merupakan kode program yang berisi tindakan atau perintah untuk menjalankan objek.

contoh pembuatan kelas, objek, atribut, dan method:

```
1 class Kelas:
2     def __init__(self, NPM):
3         self.NPM = NPM
4
5     def Modulus(self):
6         #Modulus
7         print("Soal no 1")
8
9         npm = int(self.NPM) % 3
10        print(npm)
11
12        print("###    ###    #####    ###    ###    #####    ###")
13        print("###    ###    #####    ###    ###    #####    ###")
14        print("###    ###    ###    ###    ###    ###    ###    ###")
15        print("###    ###    ###    ###    ###    ###    ###    ###")
16        print("###    ###    #####    #####    ###    ###    ###")
17        print("###    ###    ###    ###    ###    ###    ###    ###")
18        print("###    ###    ###    ###    ###    ###    ###    ###")
19        print("###    ###    #####    #####    ###    #####    ###")
20        print("###    ###    #####    #####    ###    #####    ###")
```

**Listing 4.1** Kelas

### 4.3.1 Pemanggilan Library Kelas

Pemanggilan library kelas dapat dilakukan dengan cara import dan membuat objek dari kelas tersebut.

Contohnya, kita memiliki file python yang diberi nama ngitung dan didalamnya terdapat class Ngitung yang memiliki banyak fungsi didalamnya. Untuk melakukan pemanggilan class maka kita bisa mengetikkan kode seperti berikut.

```
import ngitung

hitung = ngitung.Ngitung
```

### 4.3.2 Pemakaian Package Fungsi Apabila File Didalam Folder

Pemakaian Package fungsi apabila file terdapat didalam sebuah folder maka kita bisa menggunakan from folder import file dan from file import fungsi. Contohnya, kita

memiliki folder src yang didalamnya terdapat file fungsi.py dan didalam fungsi.py terdapat fungsi Berhitung, untuk mengimportkan fungsi maka kita dapat mengetikkan kode seperti berikut.

```
from src import fungsi  
from fungsi import Berhitung
```

#### 4.3.3 Pemakaian Package Kelas Apabila File didalam Folder

Pemakaian package kelas apabila file terdapat didalam sebuah folder maka kita bisa menggunakan from folder import file dan from file import kelas. Contohnya, kita memiliki folder src yang didalamnya terdapat file fungsi.py dan didalam fungsi.py terdapat kelas Ngitung, maka untuk melakukan import kelas kita dapat mengetikkan kode sebagai berikut.

```
from src import fungsi  
Kelas = fungsi.Nqitung(a,b)
```

#### **4.4 Pretek Membuat Method dan Pemanggilan Method**

Cara membuat method hanya dengan menambahkan def Nama Method, self atau variabel inputan, dan indentation pada source code yang telah dipraktekkan sebelumnya. Pembuatan method merupakan hal yang wajib saat kita membuat sebuah program yang berorientasi objek (OOP).

#### 4.4.1 Modulus

```
1 def Modulus(self):
2     #Modulus
3     print("Soal no 1")
4
5     npm = int(self.NPM) % 3
6     print(npm)
7
8     print("###    ###    #####    ###    ###    #####    ###")
9     print("###    ###    #####    ###    ###    #####    ###")
10    print("###    ###    ###    ###    ###    ###    ###    ###")
11    print("###    ###    ###    ###    ###    ###    ###    ###")
12    print("###    ###    #####    #####    ###    ###    ###")
13    print("###    ###    ###    ###        ###    ###    ###    ###")
14    print("###    ###    ###    ###        ###    ###    ###    ###")
```

```

15     print("###    ###    #####    ###    #####    ###")
16     print("###    ###    #####    ###    #####    ###")

```

**Listing 4.2** Modulus

#### 4.4.2 Hello NPM

```

1 def Hello_NPM(self):
2     #Hello NPM
3     print("Soal No 2")
4
5     Loop = self.NPM[5:7]
6
7     for x in range(int(Loop)):
8         print("Hallo " + self.NPM + " Apa Kabar?")

```

**Listing 4.3** Hello NPM

#### 4.4.3 Hello NPM (3 Digit Belakang)

```

1 def NPM_DigitBelakang(self):
2     #3 digit belakang NPM
3     print("Soal No 3")
4
5     Loop = self.NPM[4:7]
6
7     total = int(self.NPM[5]) + int(self.NPM[6])
8
9     for x in range(total):
10        print("Hallo " + Loop + " Apa Kabar?")

```

**Listing 4.4** 3 Digit Belakang

#### 4.4.4 Hello NPM (Digit ke-3)

```

1 def NPM_DigitKetiga(self):
2     #digit ke 3
3     print("Soal No 4")
4
5     Loop = self.NPM[4]
6     print("Hello " + Loop + " Apa Kabar?")

```

**Listing 4.5** Digit ke-3

#### 4.4.5 Variabel Alfabet

```

1 def Variabel_Alfabet(self):
2     #variabel alfabet
3     print("Soal no 5")
4
5     var = "abcdefg"
6     index = 0

```

```

8     for i in var:
9         print(i + " = " + self.NPM[index])
10        index += 1

```

**Listing 4.6** Variabel Alfabet

#### 4.4.6 Penjumlahan NPM

```

1 def Penjumlahan_NPM(self):
2     #penjumlahan NPM
3     print("Soal no 6")
4
5     index = 0
6     angka = 0
7
8     for i in self.NPM:
9         jumlah = int(self.NPM[index]) + int(angka)
10        angka = jumlah
11        index += 1
12
13     print(jumlah)

```

**Listing 4.7** Penjumlahan NPM

#### 4.4.7 Perkalian NPM

```

1 def Perkalian_NPM(self):
2     #perkalian NPM
3     print("Soal no 7")
4
5     index = 0
6     angka = 0
7
8     for i in self.NPM:
9         jumlah = int(self.NPM[index]) * int(angka)
10        angka = jumlah
11        index += 1
12
13     print(jumlah)

```

**Listing 4.8** Perkalian NPM

#### 4.4.8 Print Vertical

```

1 def Print_Vertical(self):
2     #print vertical
3     print("Soal no 8")
4
5     for i in self.NPM:
6         print(i)

```

**Listing 4.9** Print Vertical

#### 4.4.9 Digit Genap NPM

```

1 def DigitGenap(self):
2     #digit genap NPM
3     print("Soal no 9")
4
5     index = 0
6     for i in self.NPM:
7         if (int(self.NPM[index])%2 == 0) & (int(self.NPM[index]) != 0):
8             print(self.NPM[index])
9             index += 1

```

**Listing 4.10** Digit Genap NPM

#### 4.4.10 Digit Ganjil NPM

```

1 def DigitGanjil(self):
2     #digit ganjil NPM
3     print("Soal no 10")
4
5     index = 0
6     for i in self.NPM:
7         if (int(self.NPM[index])%2 != 0) & (int(self.NPM[index]) != 0):
8             print(self.NPM[index])
9             index += 1

```

**Listing 4.11** Digit Ganjil NPM

#### 4.4.11 Bilangan Prima NPM

```

1 def PrimaNPM(self):
2     #bilangan prima NPM
3     print("Soal no 11")
4     index = 0
5
6     for i in self.NPM:
7         prima = True
8         var=int(self.NPM[index])
9         if(var<=1):
10             prima=False
11         for i in range(2,var):
12             if(var%i==0):
13                 prima=False
14         if(prima==True):
15             print(var,"Prima")
16         else:
17             print(var,"bukan prima")
18         index += 1

```

**Listing 4.12** Bilangan Prima NPM

#### 4.4.12 Pemanggilan Fungsi pada Main.py

```

1 import fungsi
2
3 print("Masukkan NPM anda: ")

```

```
4 NPM = input()
5 crot = fungsi.Kelas(NPM)
6
7 soal1 = crot.Modulus()
8 soal2 = crot.Hello_NPM()
9 soal3 = crot.NPM_DigitBelakang()
10 soal4 = crot.NPM_DigitKetiga()
11 soal5 = crot.Variabel_Alfabet()
12 soal6 = crot.Penjumlahan_NPM()
13 soal7 = crot.Perkalian_NPM()
14 soal8 = crot.Print_Vertical()
15 soal9 = crot.DigitGenap()
16 soal10 = crot.DigitGanjil()
17 soal11 = crot.PrimaNPM()
```

**Listing 4.13** Bilangan Prima NPM

## BAB 5

---

# PENGELOLAAN FILE CSV

---

### 5.1 Pengelolaan File CSV, Sejarah, dan Contoh

File csv (comma separated value) sering digunakan dalam dunia pemrograman untuk menampilkan data. file csv memiliki format yang sangat sederhana, setiap baris dipisahkan oleh enter dan setiap kolom dipisahkan oleh tanda koma.

Kegunaan file csv dibandingkan file dengan format lainnya adalah dari segi kompatibilitas karena file csv dapat diolah, dimodifikasi, digunakan, import/export menggunakan berbagai software dan bahasa pemrograman, salah satunya python.

Contoh file csv:

```
1 nomor , nama klub , jumlah main , poin , tanggal  
2 1 , Manchester City ,8 ,19 ,1/2/1999  
3 2 , Arsenal ,8 ,18 ,1/3/1999  
4 3 , Tottenham Hotspurs ,8 ,18 ,1/4/1999  
5 4 , Liverpool ,8 ,17 ,1/5/1999  
6 5 , Chelsea ,8 ,16 ,1/6/1999  
7 6 , Everton ,8 ,15 ,1/7/1999  
8 7 , Manchester United ,8 ,14 ,1/8/1999  
9 8 , Southampton ,8 ,12 ,1/9/1999  
10 9 , AFC Bournemouth ,8 ,12 ,1/10/1999
```

11 10, Crystal Palace ,8,11,1/11/1999

### **Listing 5.1 Contoh CSV**

Contoh kode program python untuk membaca file csv.

```
1 #CSV Reader
2 import csv
3
4 with open('FCSV.csv') as csvfile:
5     readCSV = csv.reader(csvfile, delimiter=',')
6     for row in readCSV:
```

### **Listing 5.2 Contoh Kode Python to Read CSV**

Contoh kode program python untuk membaca file csv menggunakan library pandas.

```
1 import pandas as pd
2 df1=pd.read_csv("FCSV.csv")
3 print(df1)
```

### **Listing 5.3 Contoh Kode Python to Read CSV**

Sejarah CSV.

CSV sudah digunakan sejak tahun 1972, CSV dapat dikompilasi pada bahasa pemrograman IBM Fortran. Data yang dipisahkan oleh koma apabila terdapat spasi di dalamnya maka harus diberi tanda petik di awal dan akhir isi dari data tersebut. Nama CSV digunakan pada tahun 1983. Pada panduan dari Osborne Executive Computer terdapat kutipan yang membolehkan isi karakter memiliki koma. Pada tahun 2005 dengan RFC4180, CSV didefinisikan sebagai MIME Content Type. Lalu pada tahun 2013, defisiensi dari RFC4180 dipecahkan oleh rekomendasi dari W3C. Pada tahun 2014, IETF mempublikasi RFC7111 yang mendeskripsikan pecahan Uniform Resource Identifier(URI) ke dokumen CSV. RFC7111 menjelaskan tentang bagaimana baris, kolom dapat digunakan dalam dokumen CSV menggunakan indeks posisi. Pada Tahun 2015, W3C mempublikasikan draft rekomendasi untuk CSV-metadata standard yang dimulai pada bulan Desember 2015.

## **5.2 Aplikasi yang bisa Menciptakan File CSV**

Aplikasi yang dapat kita gunakan untuk membuat file csv ada banyak, diantaranya:

### 1. Spreadsheet

Spreadsheet merupakan aplikasi pembuatan file CSV dengan cara memasukan data sesuai baris dan kolom yang diinginkan. Contoh spreadsheet seperti Google Spreadsheet, Microsoft Excel, dan aplikasi lainnya.

### 2. Bahasa

Bahasa pemrograman merupakan media untuk membuat aplikasi yang dapat

Pemrograman

digunakan untuk membuat file CSV khusus dengan bahasa pemrograman tertentu yang support dengan pembuatan file CSV. Seperti Python, C Sharp, dan lain sebagainya.

3. Notepad atau Text Editor
- Text editor juga dapat membuat file CSV, cukup dengan membuat file sesuai format CSV dan save file tersebut dengan ekstensi .csv.

### 5.3 Menulis dan Membaca File CSV pada Ms.Excel

Membuat file csv melalui ms.excel sangatlah mudah, isikan nomor pada kolom, kemudian isikan variabel sebagai judul pada baris, lalu isikan data sesuai dengan variabel yang telah ditentukan, contoh:

nomor	nama klub	jumlah main	poin
1	Manchester	8	19
2	Arsenal	8	18
3	Tottenham	8	18
4	Liverpool	8	17
5	Chelsea	8	16
6	Everton	8	15
7	7 Manchester	8	14
9	8 Southampton	8	13
10	AFC Bourn	8	12
11	Crystal Pa	8	11

Gambar 5.1 Contoh Penulisan CSV pada Excel

Setelah mengetikkan data pada excel, lalu pilih menu file, pilih menu export, pilih menu change file type, pilih csv, lalu klik tombol save as. Maka file excel akan menjadi file csv.

```

1 nomor , nama klub , jumlah main , poin , tanggal
2 1 , Manchester City ,8 ,19 ,1/2/1999
3 2 , Arsenal ,8 ,18 ,1/3/1999
4 3 , Tottenham Hotspurs ,8 ,18 ,1/4/1999
5 4 , Liverpool ,8 ,17 ,1/5/1999
6 5 , Chelsea ,8 ,16 ,1/6/1999
7 6 , Everton ,8 ,15 ,1/7/1999
8 7 , Manchester United ,8 ,14 ,1/8/1999
9 8 , Southampton ,8 ,12 ,1/9/1999

```

```
10 9 , AFC Bournemouth ,8 ,12 ,1/10/1999
11 10, Crystal Palace ,8 ,11 ,1/11/1999
```

#### Listing 5.4 Contoh CSV

### 5.3.1 Sejarah Library CSV

Library CSV pada python merupakan library yang paling umum untuk import export data pada spreadsheet dan basis data dengan format sesuai dengan standarisasi RFC4180. Seiring dengan lahirnya bahasa pemrograman python, library mulai dibuat dan dikembangkan sampai akhirnya pada tahun 2003, Kevin Altis dan lainnya telah merilis versi final untuk library Python CSV.

### 5.3.2 Sejarah Library Pandas

Pandas (Python Data Analysis Library) adalah library open source yang digunakan untuk melakukan data manajemen dan data analysis. Pandas diciptakan pada tahun 2008 oleh Wes McKinney dan diperbaharui oleh Sien Chang pada tahun 2010. Inspirasi dari pembuatan pandas muncul pada komunitas yang membutuhkan library khusus untuk analisis data.

### 5.3.3 Fungsi - fungsi yang terdapat di library CSV

Berikut fungsi-fungsi yang terdapat pada library csv.

1. `csv.reader(csvfile, dialect='excel', **fmparms)`

Untuk mengembalikan object reader yang akan mengambil setiap line pada csv. Data setiap baris diambil saat next() dipanggil. Berikut contohnya :

```
1 #CSV Reader
2 import csv
3
4 with open('FCSV.csv') as csvfile:
5     readCSV = csv.reader(csvfile, delimiter=',')
6     for row in readCSV:
7         print(row)
```

2. `csv.writer(csvfile, dialect='excel', **fmparms)`

Mengembalikan file pembuat object untuk dapat mengkonversi data pada python ke file CSV yang akan dibuat. Berikut contoh penggunaan csv.writer :

```
1 #CSV Writer
2 import csv
3
4 with open('asal.csv', mode='w') as csvfile:
5     fieldnames = ['nomor', 'nama klub', 'jumlah main', 'poin']
6     csv_writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
7
8     csv_writer.writeheader()
```

```
9     csv_writer.writerow({ 'nomor': '1', 'nama klub': 'Asal-Asalan' }  
10    , 'jumlah main': '8', 'poin': '10'})  
11    csv_writer.writerow({ 'nomor': '2', 'nama klub': 'Asal-Asalan2' }  
12    , 'jumlah main': '8', 'poin': '12'})
```

3. `csv.register_dialect(name[, dialect[, **fmtparams]])`  
Mengasosiasikan dialek dengan nama, nama yang dimasukkan harus berupa karakter.
4. `csv.unregister_dialect(name)`  
Menghapus asosiasi dialek dengan nama pada registry dialek.
5. `csv.get_dialect(name)`  
Mengambil dialek yang telah diasosiasikan dengan nama.
6. `csv.list_dialects()`  
Mengembalikan dialek yang telah diregistrasi.
7. `csv.field_size_limit([new_limit])`  
Mengembalikan maksimal kolom data yang diperbolehkan oleh pembaca.

#### 5.3.3.1 Fungsi - fungsi yang terdapat di library Pandas

1. `pandas.read_csv(filepath_or_buffer[, sep, ...])`  
Untuk membaca file CSV dan menyimpannya ke DataFrame. Contohnya:

```
1 import pandas as pd  
2 df1=pd.read_csv("FCSV.csv")  
3 print(df1)
```
2. `pandas.read_excel(io[, sheet_name, header, names, ...])`  
Membaca file excel dan menyimpannya ke DataFrame. Contohnya:

```
1 import pandas as pds  
2 df2=pds.read_excel("FCSV.xlsx", index_col=None, header=None)  
3 print(df2)
```
3. `to_csv([path, index, sep, na_rep, ...])`  
Untuk membuat file CSV dari data yang ada

## 5.4 Praktek CSV

### 5.4.1 CSV List

```

1 import csv
2
3 def CsvList():
4     with open('FCSV.csv') as csv_file:
5         csv_reader = csv.reader(csv_file, delimiter=',')
6         line_count = 0
7         for row in csv_reader:
8             if line_count == 0:
9                 print(f'Nama Kolom adalah {" ".join(row)}')
10                line_count += 1
11            else:
12                print(f'\t Nomor : {row[0]}, Nama Klub : {row[1]},'
13 Jumlah Main : {row[2]}, Poin : {row[3]}')
14                line_count += 1
15                print(f'Jumlah Total {line_count} lines.')

```

**Listing 5.5** CSV List

#### 5.4.2 CSV Dictionary

```

1 def CsvDict():
2     with open('dictionary.csv', mode='r') as csv_file:
3         csv_reader = csv.DictReader(csv_file)
4         line_count = 0
5         for row in csv_reader:
6             if line_count == 0:
7                 print(f'Nama Kolom adalah {" ".join(row)}')
8                 line_count += 1
9             else:
10                print(f'\t First Name : {row["first_name"]} Last Name'
11 : {row["last_name"]}')
12                line_count += 1
13                print(f'Jumlah Total {line_count} lines.')

```

**Listing 5.6** CSV Dictionary

#### 5.4.3 Hello NPM (Pandas List)

```

1 import pandas
2
3 def pandasList():
4     pd = pandas.read_csv("FCSV.csv")
5     print(pd)

```

**Listing 5.7** Pandas List

#### 5.4.4 Hello NPM (Pandas Dictionary)

```

1 def pandasDict():
2     pds = pandas.read_csv("dictionary.csv")
3     crot = pandas.DataFrame.from_dict(pds)
4     print(crot)

```

**Listing 5.8** Pandas Dictionary

#### 5.4.5 Pandas Date

```

1 def Tanggal():
2     tanggal = pandas.read_csv("FCSV.csv")
3     tanggal['tanggal']=pandas.to_datetime(tanggal['tanggal'])
4     print(tanggal)

```

**Listing 5.9** Pandas Date

#### 5.4.6 Pandas Ubah Index

```

1 def ubahIndex():
2     baris = pandas.read_csv("FCSV.csv")
3     barisindex = [ '1', '2', '0', '3', '5', '4', '6', '7', '9', '8' ]
4     oke = baris.reindex(barisindex)
5     print(oke)

```

**Listing 5.10** Pandas Ubah Index

#### 5.4.7 Pandas Ubah Column

```

1 def ubahColumn():
2     ubah = pandas.read_csv("FCSV.csv")
3     okey = ubah.rename(columns={"nomor" : "number"})
4     print(okey)

```

**Listing 5.11** Pandas Ubah Column

#### 5.4.8 Main CSV

```

1 import D1184011_csv
2
3 csvList = D1184011_csv.CsvList()
4 csvDict = D1184011_csv.CsvDict()

```

**Listing 5.12** Main CSV

#### 5.4.9 Main Pandas

```

1 import D1184011_pandas
2
3 pandasList = D1184011_pandas.pandasList()
4 pandasDict = D1184011_pandas.pandasDict()
5 pandasDate = D1184011_pandas.Tanggal()
6 pandasUbah = D1184011_pandas.ubahIndex()
7 pandasUbahCol = D1184011_pandas.ubahColumn()

```

**Listing 5.13** Digit Genap NPM



## BAB 6

---

# SELENIUM

---

### 6.1 Sejarah *Selenium*

Kisah ini dimulai pada 2004 di ThoughtWorks di Chicago, dengan Jason Huggins membangun mode Inti sebagai JavaScriptTestRunner untuk pengujian aplikasi waktu dan Pengeluaran internal (Python, Plone). Pengujian otomatis terhadap aplikasi apa pun adalah inti dari gaya ThoughtWork, mengingat kecenderungan Agile dari konsultasi ini. Dia mendapat bantuan dari Paul Gross dan Jie Tina Wang. Bagi mereka, ini adalah pekerjaan harian.

Jason mulai memperagakan alat uji ke berbagai rekan. Banyak yang senang dengan umpan balik visual yang langsung dan intuitif, serta potensinya untuk tumbuh sebagai kerangka pengujian yang dapat digunakan kembali untuk aplikasi web lainnya.

Segara setelah tahun 2004 sesama ThoughtWorker Paul Hammant melihat demo, dan memulai diskusi tentang sumber terbuka Selenium, serta mendefinisikan mode 'didorong' Selenium di mana Anda bisa menggunakan Selenium melalui kabel dari bahasa pilihan Anda. , yang akan menyiasati 'kebijakan asal yang sama'. Rekan-rekan (saat itu) lainnya, Aslak Hellesoy dan Mike Melia, bereksperimen dengan berbagai ide untuk karya 'server', termasuk penulisan ulang halaman untuk men-

dapatkan kebijakan asal yang sama. Paul menulis karya server asli di Jawa, dan Aslak dan Obie Fernandez mengangkut driver klien ke Ruby, menetapkan fondasi untuk driver dalam lebih banyak bahasa.

Pekerja Pikir di berbagai kantor di seluruh dunia mengambil Selenium untuk proyek komersial, dan berkontribusi kembali ke Selenium dari pelajaran yang dipetik dari proyek ini. Mike Williams, Darrell Deboer, dan Darren Cotterill semuanya membantu meningkatkan kemampuan dan ketahanannya.

## 6.2 Jenis-Jenis *Selenium*

Selenium adalah perangkat lunak yang berfungsi untuk mendukung pengembangan otomatisasi uji berbasis web aplikasi. Selenium menyediakan pengujian khusus terhadap domain bahasa, untuk melakukan tes menulis pada beberapa bahasa pemrograman yang populer, termasuk C, Groovy, Java, Pearl, PHP, Python, Ruby, dan juga Scala. Pengujian dapat berjalan melalui browser web apa saja dan dapat dilakukan melalui Sistem Operasi di Windows, Linux, dan Platform OS X. Selenium Python Bindings menyediakan API yang sederhana untuk menulis uji fungsional menggunakan Selenium WebDriver, dan juga dapat mengakses semua fungsi Selenium WebDriver secara intuitif. Selenium Python Bindings menyediakan API yang cukup nyaman untuk melakukan suatu akses Selenium WebDrivers seperti di Firefox, Internet Explorer, Chrome, dll

Jenis-Jenis Selenium Sebagai Berikut :

### 1. *IDE selenium*

Selenium IDE (Lingkungan Pengembangan Terpadu) adalah sumber terbuka alat rekam dan putar untuk menghasilkan skrip Selenium, yang terintegrasi dengan browser web Firefox sebagai ekstensi. Ini adalah tes UI berbasis web yang terkenal alat otomatisasi yang mengekstrak segala jenis locator dari halaman web. Atau banyak yang bisa baik berbasis atribut atau berbasis struktur, dan termasuk ID, nama, tautan, XPath, CSS, dan DOM. IDE memiliki seluruh Selenium Core, yang memungkinkan pengguna mencatat 10, memutar, mengedit, dan men-debug tes secara manual di browser. Tindakan pengguna di web halaman dapat direkam dan diekspor dalam bahasa apa saja yang paling populer, seperti Java, C , Ruby, dan Python, Selenium Builder adalah alat open source alternatif untuk dicatat oleh Selenium IDE dan pemutaran aplikasi web. Ini adalah ekstensi dari web browser Firefox, Yang mirip dengan Selenium IDE, tetapi, ia memiliki beberapa fitur unik yaitu Selenium IDE tidak mendukung. Selenium Builder adalah alat standar dari Sauce Labs yang menjalankan tes Sauce Cloud dari antarmuka Selenium Builder itu sendiri.

### 2. *Selenium WebDriver*

Selenium webdriver adalah versi terbaru dari selenium IDE dan selenium Remote Control (RC). Ini juga dinamai selenium 2.0. Ini memungkinkan skrip uji yang dirancang untuk berkomunikasi dengan browser secara langsung dengan bantuan metode asli. Ini mendukung pengujian aplikasi web pada desktop

serta pada perangkat seluler seperti Android dan iOS perangkat. Biaya proyek berkurang dengan bantuan ini alat karena itu adalah alat open-source. Waktu yang diperlukan untuk mengeksekusi skrip pengujian di webdriver kurang jika dibandingkan ke selenium IDE dan Selenium RC. Ini memungkinkan skrip pengujian dirancang untuk berbagai browser seperti Internet explorer, Firefox, Mac safari dan Chrome. Script pengujian dapat dikembangkan menggunakan bahasa seperti Java, C , Ruby, Perl, Python.

### *3. Remote Control Selenium*

Remote Control Selenium (RC) adalah selenium utama yang digunakan untuk memproyeksikan waktu yang lama. Selenium RC lebih lambat daripada selenium webdriver karena menggunakan program java script yang disebut sebagai suatu inti dari selenium. Selenium RC harus memulai server sebelum menjalankan suatu skrip pengujian, dan itu tidak mendukung untuk aplikasi Ajax. Cara menghindari keterbatasan Selenium RC, aitu dengan selenium Web Driver.

### *4. Selenium Grid*

Server yang memungkinkan pengujian untuk menggunakan instace browser web yang sedang berjalan di mesin jarak jauh. Dengan selenium grid, satu server bertindak sebagai hub. Tes hubungi hub untuk mendapatkan akses ke instance browser karena hub memiliki daftar server yang menyediakan akses ke insntance browser (node WebDriver), dan memungkinkan pengujian menggunakan instance ini. Selenium Grid memiliki kemampuan untuk menjalankan tes pada instance browser jarak jauh yang berguna untuk menyebarluaskan beban pengujian di beberapa mesin, dan untuk menjalankan tes di browser yang berjalan pada platform atau sistem operasi yang berbeda. Yang terakhir ini sangat berguna dalam kasus di mana tidak semua browser yang akan digunakan untuk pengujian dapat berjalan pada platform yang sama.

### *5. TestNG*

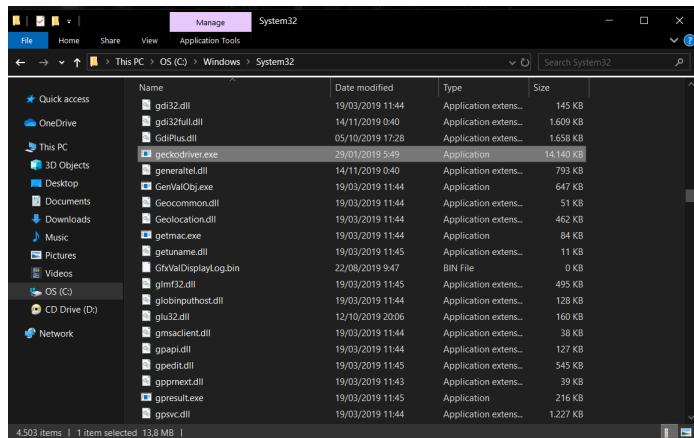
TestNG adalah kerangka pengujian yang digunakan untuk pengujian otomatisasi bersama dengan selenium 2.0. Ini mendukung berbagai tingkat pengujian seperti unit, integrasi, sistem dan pengujian penerimaan pengguna (UAT). Biasanya disebut sebagai "Uji Generasi Baru".

## **6.3 Geckodriver dan Chromedriver**

### **6.3.1 Gechodriver untuk Mozilla Firefox**

Download Geckodriver pada link ini <https://github.com/mozilla/geckodriver/releases>. sebelum mendownload harus menyamakan versi mozilla dengan versi Geckodriver misal versi Mozilla firefox versi 32bit Geckodrivernya pun harus 32bit jika tidak maka akan terjadi kesalahan.

jika sudah mendownload Geckodriver pindahkan file tersebut ke system32

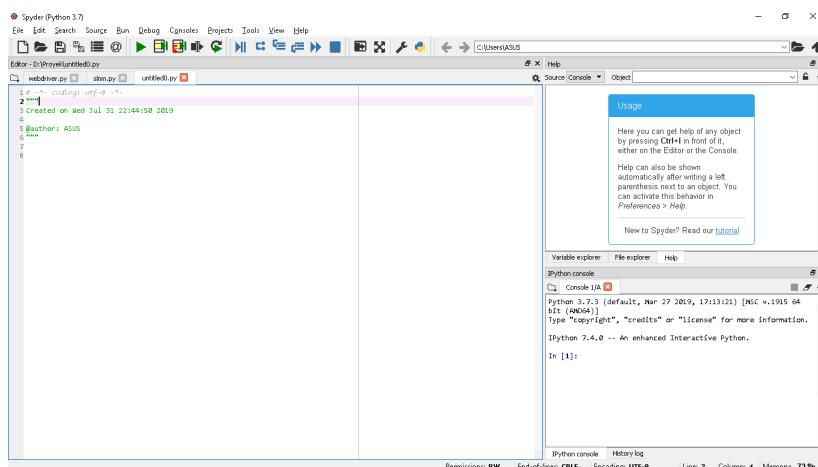


**Gambar 6.1 Gecodriver**

## 6.4 Penggunaan Selenium

Disini kami mencoba menjalankan otomasi *web testing* menggunakan *python* dan dengan menggunakan *IDE Spyder*. Langkah-langkahnya yaitu :

1. buka *spyder* dan tampilan awalnya seperti ini :



**Gambar 6.2 Tampilan awal spyder**

2. kemudian ketikkan codingan:

```

1 from selenium.webdriver import Firefox
2 from selenium.webdriver.firefox.options import Options
3 from selenium.webdriver.common.desired_capabilities import
DesiredCapabilities
4 from selenium.webdriver.firefox.firefox_binary import
FirefoxBinary
5
6 print("Masukkan Npm Anda:")
7 npm = input()
8 print("Masukkan Password SIAP Anda:")
9 paswd = input('`')
10
11 opsi = Options()
12
13 opsi.headless = False
14 binary = FirefoxBinary("C:\\\\Program Files\\\\Mozilla Firefox\\\\
firefox.exe")
15 cap = DesiredCapabilities().FIREFOX
16 cap['marionette'] = True
17
18 browser=Firefox(executable_path='geckodriver.exe',
19 options=opsi, capabilities=cap, firefox_binary=binary)
20 browser.get('http://siap.poltekpos.ac.id/siap/besan.depan.php')
21
22 name = browser.find_element_by_name('user_name')
23 word = browser.find_element_by_name('user_pass')
24 login = browser.find_element_by_name('login')
25
26
27 name.send_keys(npm)
28 word.send_keys(paswd)
29 login.click()

```

### Penjelasan Codingan :

```
1 from selenium.webdriver import Firefox
```

Yaitu Modul selenium webdriver mengimplementasikan kelas yang mendukung berbagai browser termasuk Firefox, Chrome, Internet Explorer, Safari, yang lain, dan RemoteWebDri ver juga untuk menguji pada browser yang tersedia di mesin jarak jauh. Kita perlu mengimpor webdriver dari paket Selenium untuk menggunakan metode Selenium WebDriver.

```
1 from selenium.webdriver.firefox.options import Options
```

Yaitu Opsi kelas dalam paket webdriver selenium firefox. opts adalah turunan dari kelas Opsi yang dipakai untuk program.

```
1 from selenium.webdriver.common.desired_capabilities import
DesiredCapabilities
```

Webdriver.common.desired\_capabilities import DesiredCapabilities() sebagai titik awal untuk membuat objek kemampuan yang diinginkan untuk meminta driver web jarak jauh untuk terhubung ke server selenium.

```
1 from selenium.webdriver.firefox.firefox_binary import  
  FirefoxBinary
```

Yaitu untuk mengimport FirefoxBinary atau lokasi dari si firefox.

```
1 print("Masukkan Npm Anda:")  
2 npm = input()  
3 print("Masukkan Password SIAP Anda:")  
4 paswd = input('')
```

ini merupakan inputan *user*

```
1 browser.get('http://siap.poltekpos.ac.id/siap/besan.depan.php')
```

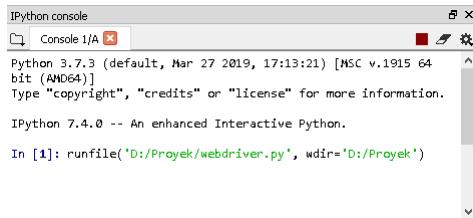
Browser.get metode akan menavigasi ke halaman yang diberikan oleh URL. WebDriver akan menunggu hingga halaman dimuat penuh (yaitu, "download" telah diaktifkan) sebelum mengembalikan kontrol ke tes atau skrip.

3. Setelah membuat Tambahan Codingan seperti diatas untuk merunning program anda tekan run pada bar diatas.



Gambar 6.3 *Running spyder*

4. Pada saat di run akan terlihat pada IPython console seperti gambar



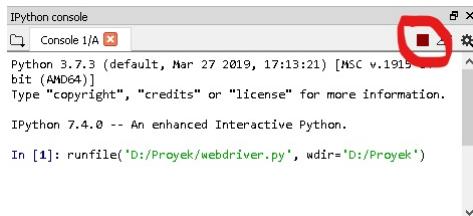
```
IPython console
Console 1/A
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64
bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.4.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/Proyek/webdriver.py', wdir='D:/Proyek')
```

**Gambar 6.4** *Running spyder console*

5. Saat kotak yang ditandai pada gambar dibawah, berwarna merah artinya proses running program tersebut masih berjalan.



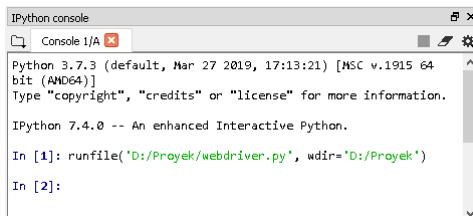
```
IPython console
Console 1/A
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64
bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.4.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/Proyek/webdriver.py', wdir='D:/Proyek')
```

**Gambar 6.5** *Running masih berjalan*

6. Jika proses *running* sudah selesai tampilannya akan seperti ini. Berarti Tambahan Codingan tersebut berhasil di *running* dan tidak terdapat *error*.



```
IPython console
Console 1/A
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64
bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.4.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/Proyek/webdriver.py', wdir='D:/Proyek')
In [2]:
```

**Gambar 6.6** *Running selesai*

7. Setelah program di run akan otomatis membuka Mozilla Firefox dan akan langsung membuka website siap.poltekpos.ac.id secara otomatis.

The screenshot shows a web browser window for 'SIAP Politek Pos Indonesia'. The main content area displays a 'Statistik Perwalian Semester Ganjil 2019/2020' table. The table has two main sections: 'Pengisian KRS' (top) and 'Perwalian' (bottom). The 'Pengisian KRS' section includes columns for Program Studi, Hari Ini, Sebelumnya, Jumlah, and Akhir. The 'Perwalian' section includes columns for Hari Ini, Sebelumnya, Jumlah, and Akhir. The table contains data for various programs like D3 Teknik Informatika, D4 Teknik Informatika, D3 Manajemen Informatika, D3 Akuntansi, D4 Akuntansi Keuangan, D3 Manajemen Pemasaran, D4 Manajemen Perusahaan, D3 Logistik Bisnis, and D4 Logistik Bisnis. The bottom right corner of the screenshot shows a sidebar with a 'Logout' button and a QR code.

Program Studi	Pengisian KRS					Perwalian				
	Hari Ini	Sebelumnya	Jumlah	Akhir	Mks	Hari Ini	Sebelumnya	Jumlah	Akhir	Mks
D3 Teknik Informatika	0	0	104	857	113	0	0	104	857	113
D4 Teknik Informatika	0	0	294	1035	293	204	1035	0	0	204
D3 Manajemen Informatika	0	0	66	519	67	66	519	0	0	66
D3 Akuntansi	0	0	104	939	104	104	939	0	0	104
D4 Akuntansi Keuangan	0	0	211	1649	215	211	1649	0	0	211
D3 Manajemen Pemasaran	0	0	96	837	96	97	837	0	0	95
D4 Manajemen Perusahaan	0	0	346	2220	346	2220	0	0	346	2220
D3 Logistik Bisnis	0	0	357	5175	364	357	5175	0	0	357
D4 Logistik Bisnis	0	0	729	5382	729	5299	5382	0	0	728
Jumlah	0	0	2297	17313	2356	2297	17313	0	0	2295

Gambar 6.7 Tampilan siap.poltekpos

#### 6.4.1 Cara find element atau class

Selanjutnya kami akan mencoba *Find Element* yang terdapat pada website tersebut. Sebelumnya anda harus mengetahui apa saja jenis-jenis *element* dan berikut adalah contoh dari *element*.

##### 1. find\_element\_by\_id

Gunakan ini ketika Anda tahu atribut id suatu elemen. Dengan strategi ini, elemen pertama dengan nilai atribut id yang cocok dengan lokasi akan dikembalikan. contoh :

```
1 <form id="login">
  2   login = Browser.find_element_by_id('login')
```

##### 2. find\_element\_by\_name

Gunakan ini ketika Anda tahu atribut nama elemen. Dengan strategi ini, elemen pertama dengan nilai atribut nama yang cocok dengan lokasi akan dikembalikan. contoh :

```
1 <input name ="username" type="text" />
  2   username = Browser.find_element_by_name('username')
```

##### 3. find\_element\_by\_xpath

XPath adalah bahasa yang digunakan untuk menemukan node dalam dokumen XML. Karena HTML dapat menjadi implementasi XML (XHTML), pengguna Selenium dapat memanfaatkan bahasa yang kuat ini untuk menargetkan elemen dalam aplikasi web mereka. Dan cara mendapatkan xpath adalah inspect website tersebut dan klik kanan pada element yang ingin di cari dan klik copy dan disana ada copy Xpath. contoh :

```
1 """ /html/body/table/tbody/tr[5]/td/table[3]/tbody/tr[1]/td[2]/p  
2 [1]/table/tbody/tr/td[3]/select ).click ()  
3 browser.find_element_by_xpath('/html/body/table/tbody/tr[5]/td/  
4 table[3]/tbody/tr[1]/td[2]/p[1]/table/tbody/tr/td[3]/select')  
5 .click ()
```

#### 4. find\_element\_by\_link\_text

Gunakan ini ketika Anda tahu teks tautan yang digunakan dalam tag jangkar. Dengan strategi ini, elemen pertama dengan nilai teks tautan yang cocok dengan lokasi akan dikembalikan. contoh :

```
1 <a href="continue.html">Continue</a>  
2 Continue = Browser.find_element_by_link_text('Continue')
```

#### 5. find\_element\_by\_tag\_name

Gunakan ini ketika Anda ingin mencari elemen dengan nama tag. Dengan strategi ini, elemen pertama dengan nama tag yang diberikan akan dikembalikan. contoh :

```
1 <strong>Hello </strong>  
2 Strong = Browser.find_element_by_tag_name('strong')
```

#### 6. find\_element\_by\_class\_name

Gunakan ini ketika Anda ingin mencari elemen dengan nama atribut kelas. Dengan strategi ini, elemen pertama dengan nama atribut kelas yang cocok akan dikembalikan. contoh :

```
1 <p class="body">Halo.</p>  
2 body = Browser.find_element_by_class_name('body')
```

#### 7. find\_element\_by\_css\_selector

Gunakan ini ketika Anda ingin mencari elemen dengan sintaks pemilih CSS. Dengan strategi ini, elemen pertama dengan pemilih CSS yang cocok akan dikembalikan. contoh :

```
1 <p class="body">Halo.</p>  
2 body = Browser.find_element_by_class_name('p.body')
```

### 6.4.2 Mengambil element dari web siap.poltekpos.ac.id

Setelah mengenal tentang element mari kita mencoba mencari element pada website siap.poltekpos.ac.id

1. Disini kami mencoba untuk mengisi data *user* pada *login*.

Statistik Perwalian Semester Ganjil 2019/2020												
Program Studi	Pengisian KRS						Perwalian					
	Hari Ini	Mks	Mk	Sebelumnya	Jumlah	Aktif	Hari Ini	Mks	Mk	Sebelumnya	Jumlah	Aktif
D3 Teknik Informatika	0	0	104	857	113	104	857	0	0	104	857	
D4 Teknik Informatika	0	0	204	1805	293	204	1805	0	0	204	1779	
D3 Manajemen Informatika	0	0	66	519	67	66	519	0	0	66	517	
D3 Akuntansi	0	0	104	939	104	104	939	0	0	104	931	
D4 Akuntansi Keuangan	0	0	211	1649	215	211	1649	0	0	211	1649	
D3 Manajemen Pemasaran	0	0	96	837	98	96	837	0	0	95	835	
D4 Manajemen Perusahaan	0	0	346	2220	346	2220	0	0	346	2207	350	
D3 Logistik Bisnis	0	0	357	3175	364	357	3175	0	0	357	3173	
D4 Logistik Bisnis	0	0	729	5262	752	729	5262	0	0	728	5239	
Jumlah	0	0	2397	17313	2356	2397	17313	0	0	2295	17167	

Log Pengisian KRS dan Perwalian Semester Ganjil 2019/2020											
Pengisian KRS						Perwalian					
No	N.P.M	Nama	Jml MK	Waktu	No	N.P.M	Nama	Jml MK	Waktu		
1	1173051	Bobby Setiawan	8	29-11-2019 (09:30:01)	1	1173036	Elo Permai Prayitno	8	29-11-2019 (08:49:16)		
2	1173018	Chandra Rohman Sholahang	8	29-11-2019 (09:30:09)	2	1173018	Chandra Rohman Sholahang	8	29-11-2019 (08:49:03)		
3	1194071	Raza Neufal Perez Idam	8	26-11-2019 (15:05:05)	3	1164084	Pudji Hemri	1	27-11-2019 (08:29:45)		

Gambar 6.8 Tampilan siap.poltekpos

2. Untuk mencari elementnya arahkan cursor ke *login* pengguna, kata sandi, dan login. lalu klik kanan dan *inspect*, disini kami menggunakan element\_by\_name.

Statistik Perwalian Semester Ganjil 2019/2020												
Program Studi	Pengisian KRS						Perwalian					
	Hari Ini	Mks	Mk	Sebelumnya	Jumlah	Aktif	Hari Ini	Mks	Mk	Sebelumnya	Jumlah	Aktif
D3 Teknik Informatika	0	0	104	857	113	104	857	0	0	204	1779	
D4 Teknik Informatika	0	0	204	1805	293	204	1805	0	0	204	1779	
D3 Manajemen Informatika	0	0	66	519	67	66	519	0	0	66	517	
D3 Akuntansi	0	0	104	939	104	104	939	0	0	104	931	
D4 Akuntansi Keuangan	0	0	211	1649	215	211	1649	0	0	211	1649	
D3 Manajemen Pemasaran	0	0	96	837	98	96	837	0	0	95	835	
D4 Manajemen Perusahaan	0	0	346	2220	346	2220	0	0	346	2207	350	
D3 Manajemen Perusahaan	0	0	357	3175	364	357	3175	0	0	357	3173	
D4 Logistik Bisnis	0	0	729	5262	752	729	5262	0	0	728	5239	
Jumlah	0	0	2397	17313	2356	2397	17313	0	0	2295	17167	

Log Pengisian KRS dan Perwalian Semester Ganjil 2019/2020											
Pengisian KRS						Perwalian					
No	N.P.M	Nama	Jml MK	Waktu	No	N.P.M	Nama	Jml MK	Waktu		
1	1173051	Bobby Setiawan	8	29-11-2019 (09:30:01)	1	1173036	Elo Permai Prayitno	8	29-11-2019 (08:49:16)		
2	1173018	Chandra Rohman Sholahang	8	29-11-2019 (09:30:09)	2	1173018	Chandra Rohman Sholahang	8	29-11-2019 (08:49:03)		
3	1194071	Raza Neufal Perez Idam	8	26-11-2019 (15:05:05)	3	1164084	Pudji Hemri	1	27-11-2019 (08:29:45)		

Gambar 6.9 inspect element by name

Tambahan Codingan :

```
1 name = browser.find_element_by_name('user_name')
2 word = browser.find_element_by_name('user_pass')
3 login = browser.find_element_by_name('login')
```

Hasil :

Program Studi	Pengisian KRS						Perwalian							
	Hari Ini		Sebelumnya		Jumlah		Hari Ini		Sebelumnya		Jumlah			
Mhs	MK	Mhs	MK	Aktif	Mhs	MK	Mhs	MK	Aktif	Mhs	MK			
D3 Teknik Informatika	0	0	104	857	113	104	857	0	0	104	857	113	104	857
D4 Teknik Informatika	0	0	284	1853	293	284	1853	0	0	284	1779	293	284	1779
D3 Manajemen Informatika	0	0	66	519	67	66	519	0	0	66	517	67	66	517
D3 Akuntansi	0	0	104	939	104	104	939	0	0	104	931	104	104	931
D4 Akuntansi Keuangan	0	0	211	1649	215	211	1649	0	0	211	1649	215	211	1649
D3 Manajemen Pemasaran	0	0	96	837	96	96	837	0	0	95	805	96	95	837
D3 Manajemen Perusahaan	0	0	346	2250	346	346	2250	0	0	346	2207	346	346	2207
D3 Logistik Bisnis	0	0	357	3175	364	359	3175	0	0	357	3173	364	357	3173
D4 Logistik Bisnis	0	0	729	5262	732	729	5262	0	0	728	5239	732	728	5239
Jumlah	0	0	2297	17133	2356	2297	17133	0	0	2295	17187	2356	2295	17187

Log Pengisian KRS dan Perwalian Semester Ganjil 2019/2020											
No	N P M	Pengisian KRS				Perwalian				Jml MK	Waktu
		No	N P M	Nama	Jml MK	Waktu	No	N P M	Nama		
1	1173051	Bobby Setiawan	8	29-11-2019 (09:34:01)	1	1173036	Elo Penel Partahan	8	29-11-2019 (09:49:16)		
2	1173018	Chando Rokman Stohang	8	29-11-2019 (09:32:09)	2	1173018	Chando Rokman Stohang	8	29-11-2019 (09:49:03)		
3	1194071	Raja Naufal Faraz Islam	8	26-11-2019 (15:35:05)	3	1164094	Puad Hendri	1	27-11-2019 (09:28:45)		
4	1193012	Kurnadi Barisa Adityaya	10	25-11-2019 (15:53:10)	4	1194071	Raja Naufal Faraz Islam	8	26-11-2019 (15:35:33)		
			10	25-11-2019 (15:49:00)	5	1183028	Fikri Ahmad Shidq	7	26-11-2019 (08:24:28)		

Gambar 6.10 Tampilan loading login

Hasil :

Program Studi	Pengisian KRS						Perwalian							
	Hari Ini		Sebelumnya		Jumlah		Hari Ini		Sebelumnya		Jumlah			
Mhs	MK	Mhs	MK	Aktif	Mhs	MK	Mhs	MK	Aktif	Mhs	MK			
D3 Teknik Informatika	0	0	104	857	113	104	857	0	0	104	857	113	104	857
D4 Teknik Informatika	0	0	284	1853	293	284	1853	0	0	284	1779	293	284	1779
D3 Manajemen Informatika	0	0	66	519	67	66	519	0	0	66	517	67	66	517
D3 Akuntansi	0	0	104	939	104	104	939	0	0	104	931	104	104	931
D4 Akuntansi Keuangan	0	0	211	1649	215	211	1649	0	0	211	1649	215	211	1649
D3 Manajemen Pemasaran	0	0	96	837	98	96	837	0	0	95	835	98	95	835
D3 Manajemen Perusahaan	0	0	346	2250	350	346	2250	0	0	346	2207	350	346	2207
D3 Logistik Bisnis	0	0	357	3175	364	357	3175	0	0	357	3173	364	357	3173
D4 Logistik Bisnis	0	0	729	5262	752	729	5262	0	0	728	5239	752	728	5239
Jumlah	0	0	2297	17133	2356	2297	17133	0	0	2295	17187	2356	2295	17187

Log Pengisian KRS dan Perwalian Semester Ganjil 2019/2020											
No	N P M	Pengisian KRS				Perwalian				Jml MK	Waktu
		No	N P M	Nama	Jml MK	Waktu	No	N P M	Nama		
1	1173051	Bobby Setiawan	8	29-11-2019 (09:34:01)	1	1173036	Elo Penel Partahan	8	29-11-2019 (09:49:16)		
2	1173018	Chando Rokman Stohang	8	29-11-2019 (09:32:09)	2	1173018	Chando Rokman Stohang	8	29-11-2019 (09:49:03)		
3	1194071	Raja Naufal Faraz Islam	8	26-11-2019 (15:35:05)	3	1164094	Puad Hendri	1	27-11-2019 (09:28:45)		

Gambar 6.11 Tampilan login

- Pada layanan mahasiswa, kami mencoba untuk melihat nilai mahasiswa secara otomatis. Dengan cara yaitu klik kanan pada nilai mahasiswa, kemudian pilih inspect.

The screenshot shows a web-based application interface for a college. At the top, there is a logo consisting of a globe icon and the word "SIAP". Below the logo, the text "POLTEKPOS ::" is displayed. On the left side, there is a sidebar menu titled "Layanan Mahasiswa" which includes options like "Ubah Profil", "Lihat Profil", "Pengisian KRS", "Jadwal Kuliah", and "Nilai". Under "Nilai", there are links for "Mata Kuliah", "Ke Akhir", and "Akhir Semester". A context menu is open over the "Nilai" link, containing options: "Buka link di tab baru", "Buka link di jendela baru", "Buka link di jendela penyamaran", "Simpan tautan sebagai...", "Salin alamat link", "Fair AdBlock by STANDS", and "Inspeksi". The main content area displays a table titled "Statistik Perwalian Semester Ganjil 2019/2020". The table has three columns: "Program Studi", "Hari Ini", and "Semester". The "Hari Ini" column has sub-columns "Mhs", "MK", and "Ml". The table data is as follows:

Program Studi	Hari Ini	Semester	
	Mhs	MK	Ml
D3 Teknik Informatika	0	0	10
D4 Teknik Informatika	0	0	21
D3 Manajemen Informatika	0	0	6
D3 Akuntansi	0	0	11
	0	2	
	0	9	
	0	3	
	0	3	
	0	7	
	0	22	

Gambar 6.12 inspect element nilai mahasiswa

Disini kami mengambil *element by xpath*

Gambar 6.13 inspect element by xpath

Tambahan codingan :

```
nilai= browser.find_element_by_xpath("//html/body/table/tbody/tr[5]/td/table[1]/tbody/tr/td[1]/table[2]/tbody/tr[1]/td[2]/a[5]").click()
```

Hasil :

Gambar 6.14 Tampilan nilai semester mahasiswa

4. Kemudian pada kolom tahun akademik, klik kanan dan pilih *inspect*

Gambar 6.15 *inspect element* tahun akademik

Disini kami mengambil *element by xpath* pada semester genap 2018/2019

Gambar 6.16 *inspect element by xpath* semster genap

Tambahan codingan :

```
nilai.semester.genap = browser.find_element_by_xpath('/html/body/table/tbody/tr[5]/td/table[3]/tbody/tr[1]/td[2]/p[1]/table/tbody/tr[3]/select/option[4]').click()
```

Hasil :

#	Kode	Notakuliah	Kelas	SKS	Jenis	Tugas	Presensi	UTS	UAS	Akhir	Grade	Bobot	Status				
1	PP11102	GENERAL ENGLISH II	B	3	Kuliah	80.00	81.00	0.00	0.00	100.00	79.00	67.00	B	3.00	Aifz		
2	TP42204	BASIS DATA I	B	4	Kuliah	80.00	80.00	80.00	80.00	93.00	80.00	78.00	B	94.20	B	3.00	Aifz

Gambar 6.17 Tampilan nilai semester genap 2018/2019

5. kemudian klik find cari dengan cara klik kanan pilih inspect

Gambar 6.18 inspect element cari

Disini kami mengambil element by class name, class name yaitu button.

Gambar 6.19 inspect element by class name cari

Tambahan codingan :

```
1 cari = browser.find_element_by_class_name('button').click()
```

6. codingan keseluruhan :

```
1 from selenium.webdriver import Firefox
2 from selenium.webdriver.firefox.options import Options
3 from selenium.webdriver.common.desired_capabilities import
4     DesiredCapabilities
5 from selenium.webdriver.firefox.firefox_binary import
6     FirefoxBinary
7
8 print("Masukkan Npm Anda:")
9 npm = input()
10 print("Masukkan Password SIAP Anda:")
11 paswd = input('')
12
13 opsi = Options()
14
15 opsi.headless = False
16 binary = FirefoxBinary("C:\\\\Program Files\\\\Mozilla Firefox\\\\
17     firefox.exe")
18 cap = DesiredCapabilities().FIREFOX
19 cap['marionette'] = True
20
21 browser=Firefox(executable_path='geckodriver.exe',
22 options=opsi, capabilities=cap, firefox_binary=binary)
23 browser.get('http://siap.poltekpos.ac.id/siap/besan.depan.php')
24
25
26 name = browser.find_element_by_name('user_name')
27 word = browser.find_element_by_name('user_pass')
28 login = browser.find_element_by_name('login')
29
30
31 name.send_keys(npm)
32 word.send_keys(paswd)
33 login.click()
34
35 nilai = browser.find_element_by_xpath("//html/body/table/tbody/tr
36     [5]/td/table[1]/tbody/tr/td[1]/table[2]/tbody/tr[1]/td[2]/a
37     [5]").click()
38 semester1 = browser.find_element_by_xpath('/html/body/table/tbody
39     /tr[5]/td/table[3]/tbody/tr[1]/td[2]/p[1]/table/tbody/tr/td
40     [3]/select/option[4]').click()
41 cari = browser.find_element_by_class_name('button').click()
```

## 6.5 WebDriver Exception : Can not connect to the service geckodriver

Cara mengatasinya yaitu dengan menyamakan versi python, mozilla firefox, gecko-driver , misalnya versi 64bit. Jika masih tidak bisa, silahkan buka localhost pada laptop atau komputer anda. Lokasi localhost berada di (C: Windows System32 drivers etc) kemudian buka file hosts dan edit menggunakan notepad++ kemungkinan error karena di hosts 127.0.0.1

## 6.6 SessionNotCreatedException : Unable to find a matching set of capabilities

Cara mengatasinya yaitu dengan pada desktop klik kanan pada mozilla firefox, pilih properties dan copy pada bagian lokasi direktori Jika sudah paste kedalam kode program pada binnary mozilla firefox

## 6.7 Module Notfound Error : no module named selenium

Cara mengatasinya:

1. uninstal aplikasi berbahasa pemrograman python, selain anaconda karena nanti akan terjadi konflik
2. restart komputer anda
3. setting environtment kembali
4. buka cmd
5. install kembali selenium
6. selesai

## 6.8 WebDriverException: cant load profile Possible version mismatch

Cara mengatasinya cukup mengupdate versi Mozilla Firefox atau Chrome agar bisa compatible dengan drivernya.



## BAB 7

---

# CHATBOT

---

### 7.1 Chatbot

#### 7.1.1 Chatbot

Okeee.., kali ini kita akan mulai memperkenalkan apa itu chatbot dan potensi yang akan didapatkan jika kita membangun sebuah chatbot. Chatbot merupakan kata gabungan dari kata dasar berbahasa inggris yaitu *chat* dan *bot*, arti kata chat yaitu perbincangan atau komunikasi antara 2 orang atau lebih, chat terbagi menjadi 2 yaitu group chat atau perbincangan yang dilakukan secara berkelompok, atau personal chat atau chat yang dilakukan hanya 2 orang saja dan tidak lebih. Bot merupakan potongan kata dari kata robot yang artinya adalah sebuah system yang dibangun untuk membantu manusia dalam sebuah pekerjaan. Lalu arti chatbot adalah sebuah robot/system yang dapat berinteraksi dengan manusia melalui perbincangan/komunikasi 2 arah bisa dalam group chat maupun personal chat. Sehingga chat yang dilakukan adalah orang dengan robot tetapi seakan-akan yang membalas pesan tersebut seperti manusia, padahal yang membalas itu adalah robot yang sudah dirancang sedemikian rupa dengan aturan yang sudah ditentukan oleh *programmer* atau orang yang membangun otak dari robot chat itu sendiri.

## 7.1.2 NLP dan Rule Based

**7.1.2.1 NLP** Natural Language Processing (NLP) adalah sebuah cabang keilmuan yang ada pada Artificial Intelligence (AI). NLP adalah sebuah keilmuan yang digunakan untuk interaksi antara manusia dengan mesin atau robot yang melewati pertama lingustik atau kata-kata. Beberapa hal yang akan dilakukan oleh NLP adalah membaca kalimat, mengekstrak kalimat menjadi kata-kata, mengartikan kata demi kata, memahaminya apa yang dimaksud, dan akan menghasilkan sebuah respon.

**7.1.2.2 Rule Based** Rule Based adalah sebuah kata yang telah ditentukan oleh programmer seperti apa aturan yang harus ditetapkan agar chatbot akan merespon dengan aturan yang sudah ditetapkan, contoh: ketika programmer menetapkan ketika user chat "halo" maka chatbot harus merespon "halo, juga", ketika user mengetikkan "hai" maka chatbot tidak akan merespon hal tersebut karena yang sudah di program adalah kata "halo" bukan kata "hai", maka berbeda dengan NLP yang mana setiap kata yang dikirim oleh user akan dipahaminya lalu akan diberikan sebuah respon oleh chatbot.

## **BAB 8**

---

# **WHATSAPP**

---

### **8.1 WhatsApp**

#### **8.1.1 WhatsApp**

WhatsApp adalah sebuah platform chat dengan penggunaanya mencapai 1.5 Milliar pengguna diseluruh dunia. Tidak heran mengapa banyak orang menggunakan whatsapp, dikarenakan penggunaan yang mudah mulai dari mendaftar hingga penggunaannya pun cukup mudah. Ditambah lagi whatsapp terintegrasi dengan nomor handphone yang mana pendaftarannya cukup mudah. WhatsApp yang sekarang pada 3 februari 2020 ini dimiliki oleh facebook memiliki fitur yang tidak kalah baik dibanding pesaingnya seperti LINE, dan lain-lain. Kini whatsapp bisa mengirim sticker dengan sticker yang sudah kita desain sedemikian rupa/edit, dan itu gratis tanpa dipungut biaya.



## BAB 9

---

# PENGENALAN PEMROGRAMAN BERORIENTASI OBJEK (OOP) DENGAN PYTHON

---

Kelas atau dalam bahasa Inggris disebut class, merupakan sebuah konsep yang menyediakan sarana untuk menyatukan data dan fungsionalitas secara satu kesatuan. Membuat sebuah kelas artinya membuat sebuah tipe baru, kemudian dengan membuat instance dari kelas tersebut akan menghasilkan objek baru dari tipe tersebut. Setiap objek (hasil instance dari kelas tersebut) dapat memiliki atribut untuk mengelola status dari objek tersebut, juga dapat memiliki metode untuk mengubah status atau informasinya.

Catatan:

Kata objek adalah terjemahan bahasa Inggris dari kata object.

Kata metode adalah terjemahan bahasa Inggris dari kata method. Selanjutnya kita akan mempelajari secara mendalam implementasi kelas dan fitur-fitur terkait di bahasa pemrograman Python.

## 9.1 Class

Class merupakan sintaksis di Python yang menyediakan semua fitur-fitur standar dari Pemrograman Berorientasi Objek atau dalam bahasa Inggris disebut dengan Object Oriented Programming (OOP).

Definisi dari kelas menggunakan sintaksis class seperti hanya definisi fungsi yang menggunakan sintaksis def, kemudian perlu dipanggil (dieksekusi) dahulu sebelum dapat digunakan dan memiliki efek pada program.

```
1 class NamaKelas:
2     pass # gantikan dengan pernyataan-pernyataan , misal: atribut
          atau metode
```

Pada pemanggilan sintaksis class tersebut, setelah seluruh pernyataan-pernyataan semuanya selesai diproses (didaktarkan sebagai atribut ataupun metode), maka kelas sudah dibuat dan dapat digunakan.

Sebuah kelas sendiri mendukung dua macam operasi:

1. Mengacu pada atribut
2. Pembuatan instance atau dalam bahasa Inggris disebut instantiation

Agar lebih jelas, kita akan membahas menggunakan contoh berikut.

```
1 class Kalkulator:
2     """contoh kelas kalkulator sederhana"""
3     i = 12345
4
5     def f(self):
6         return 'hello world'
```

Dari pembuatan class Kalkulator di atas, di dalamnya ada definisi atribut i dan definisi fungsi f.

Proses mengacu atribut yaitu Kalkulator.i dan Kalkulator.f sesuai definisi akan mengembalikan nilai integer dan fungsi. Pada proses mengacu atribut tersebut juga dapat mengubah nilainya, misalnya dengan memberikan bilangan bulat lain ke Kalkulator.i akan mengubah nilai yang ada saat ini.

```
1 Kalkulator.i = 1024 # maka nilai atribut i dalam Kalkulator berubah
                      dari 12345 menjadi 1024
```

## 9.2 Menulis Method dan Kelas pada Python

Module Python adalah berkas teks berekstensi .py yang berisikan kode Python. Anda dapat mereferensi berkas .py apa pun sebagai modul. Modul-modul umum yang

disediakan oleh Python Standard Library dan mungkin sudah terinstal secara default pada instalasi Python Anda. PIP juga dapat dimanfaatkan untuk menginstal modul atau library berikut dengan dependensi yang dibutuhkannya. Anda pun dapat membuat dan menghasilkan modul Python Anda sendiri.

### 9.2.1 Menulis Modul

Menuliskan modul pada bahasa Python dapat dimulai dengan menuliskan definisi fungsi, kelas, dan variabel yang dapat digunakan kembali pada program lainnya. Misalkan saja kita membuat berkas hello.py yang akan kita panggil di berkas lain.

hello.py

```
1 # Define a function
2 def world():
3     print("Hello , World!")
```

Jika hello.py dijalankan, maka program tidak akan menjalankan apapun karena world() hanya berupa definisi fungsi, kita belum memanggilnya. Jika kita biasanya memanggil sebuah fungsi dari berkas yang sama di bagian main, kali ini kita akan membuat berkas lain main.py yang seolah mengimpor hello.py. Pastikan hello.py dan main.py berada dalam satu direktori agar dapat diimpor dan dipanggil.

main.py

```
1 #impor
2 import hello
3 #panggil
4 hello.world()
```

Saat memanggil sebuah fungsi dari modul yang impor, jangan lupa untuk menambahkan nama modulnya diikuti tanda titik, baru fungsi yang akan kita panggil. Dalam hal ini karena kita mengimpor hello.py, maka cukup kita tulis import hello, dan saat memanggilnya dengan hello.world(). Selain itu, kita juga dapat menggunakan from ... import ..., dalam hal ini adalah from hello import world dan memanggil fungsinya langsung yakni world().

Sekarang, saat memanggil main.py, maka akan menghasilkan:

Hello, World!

### 9.2.2 Menambahkan variabel

Menambahkan variabel pada modul hello, tambahkan variabel nama, misalnya Di-coding.

hello.py

```

1 def world():
2     print("Hello, World!")
3 nama = "Dicoding"

```

Berikutnya, kita coba cetak variabel nama.

main.py

```

1 #impor
2 import hello
3
4 #panggil
5 hello.world()
6 #cetak
7 print(hello.nama)

```

Saat Dijalankan:

Hello, World!

Dicoding

### 9.2.3 Menambahkan kelas

Contoh yang lain, mari tambahkan kelas di modul hello. Kita akan membuat kelas Reviewer dengan atribut nama dan kelas, serta fungsi review() yang akan mencetak atribut yang telah didefinisikan.

hello.py

```

1 def world():
2     print("Hello, World!")
3 nama = "Dicoding"
4 class Reviewer:
5     def __init__(self, nama, kelas):
6         self.nama = nama
7         self.kelas = kelas
8     def review(self):
9         print("Reviewer " + self.nama + " bertanggung jawab di kelas "
" + self.kelas)

```

Tambahkan kelas pada main.py

main.py

```

1 #impor
2 import hello
3 #panggil
4 hello.world()
5 #cetak
6 print(hello.nama)
7 #review
8 diko = hello.Reviewer("Diko", "Python")
9 diko.review()

```

Seperti umumnya kelas pada bahasa pemrograman lainnya, Fungsi dan Atributnya dapat diakses setelah kita melakukan instansiasi. Fungsi Review adalah fungsi yang melekat pada kelas Reviewer. Kita juga dapat memanggil diko>Nama atau diko.Kelas sebagai atribut yang melekat di kelas tersebut.

Output:

```
Hello, World!  
Dicoding  
Reviewer Diko bertanggung jawab di kelas Python
```

Lihat kedua variabel "nama" yang dapat menghasilkan dua nilai berbeda, karena nama yang pertama (hello.nama) melekat pada modul, sementara diko. Nama adalah atribut nama pada kelas Reviewer. Anda harus cukup berhati-hati dalam memastikan variabel seperti pada pembahasan fungsi yang lalu.

#### 9.2.4 Implementasi Kode

Seringkali, modul dimanfaatkan untuk dapat memisahkan antara definisi dan implementasi kode. Namun modul juga dapat berfungsi selayaknya program pada umumnya, yang juga langsung mengeksekusi dalam modul itu sendiri. Contohnya, kita buat hello2.py seperti berikut:

hello2.py

```
1 # Definisi  
2 def world():  
3     print("Hello, World!")  
4 # Panggil disini  
5 world()
```

Kemudian bersihkan main.py hingga menyisakan import hello saja.

main\_program.py

```
1 import hello
```

Saat main\_program dijalankan, langsung muncul:

Hello, World!

Sehingga modul dapat digunakan dengan berbagai metode pemanggilan, bergantung pada definisi, maupun implementasi.

## 9.2.5 Mengakses Modul dari Folder Lain

Jika Anda bekerja dengan beberapa proyek secara paralel, berikut adalah opsi untuk mengakses modul dari folder lain:

### 1. Menambahkan path folder

Opsi ini dipilih umumnya di tahap awal pengembangan, sebagai solusi temporer. Untuk mengetahui path pemanggilan utama, Anda perlu bantuan dari modul sys yang sudah tersedia. Impor modul sys di main program dan gunakan fungsi sys.path.append. Berikut contoh jika berkas hello.py kita berada di direktori /home/dicoding/ dan main.py di direktori lainnya, pada Anda akan menambahkan path /home/dicoding pada main.py dengan cara:

```
1 import sys  
2 sys.path.append('/home/dicoding')  
3 import hello
```

### 2. Menambahkan modul pada Python Path

Alternatif ini dapat dipilih saat Anda melakukan pemanggilan modul `1x`. Pada intinya pilihan ini akan menambahkan modul yang Anda buat pada Path yang diperiksa oleh Python sebagai modul dan paket-paket bawaan. Anda dapat memanfaatkan sys.path kembali untuk mengetahui posisi Anda saat ini.

```
print(sys.path)
```

Anda mungkin akan menerima output seperti berikut, sangat bergantung dengan jenis environment Anda, tapi pilihlah (atau cobalah satu per satu) jika ada beberapa output.

```
'/home/dicoding/my_env/lib/python3.5/site-packages'
```

Pindahkan hello.py pada direktori di atas. Maka ia akan dikenali sebagai sebuah modul yang dapat diimpor oleh siapa saja dalam environment tersebut.

Pada main\_program.py cukup impor.

```
import hello
```

Pastikan path yang Anda assign tepat untuk menghasilkan pemanggilan yang tepat. Modul yang tersebar pada beberapa folder mungkin akan menghasilkan galat. Usahakan peletakan yang se-sederhana mungkin.

### 9.3 Objek (object: an instance of a class)

embahasan berikutnya adalah instantiation dari sebuah class, menggunakan notasi fungsi yaitu dengan kurung buka-kurung tutup, akan menghasilkan sebuah objek. Kemudian hasil instantiation ini biasanya disimpan dalam sebuah variabel dengan nama yang representatif.

Berikut ini adalah contoh membuat instance dari class Kalkulator menghasilkan sebuah objek.

```
k = Kalkulator() # membuat instance dari kelas jadi objek, kem
```

Sebagai hasil instance sebuah class, suatu objek memiliki atribut dan metode yang didapatkan dari class. Sebuah metode atau dalam bahasa Inggris disebut method, adalah sebuah fungsi khusus yang menjadi "milik" suatu objek.

Untuk memanggil metode f dari objek k, hasil instance dari class Kalkulator di atas sebagai berikut.

```
k.f() # akan mencetak hello world ke layar
```

Kenapa metode adalah sebuah fungsi khusus?

Jika diperhatikan kembali fungsi f dalam definisi class Kalkulator memiliki satu argumen bernama self, sedangkan dalam pemanggilan metode dari objek k di atas tidak menggunakan argumen. Apabila f adalah fungsi biasa pada Python tentu pemanggilan ini akan mengembalikan kesalahan (error). Lebih detail mengenai konvensi ini akan dibahas pada bagian metode dari class.

Pembahasan lebih lanjut mengenai metode dari class ada di bagian selanjutnya.

### 9.4 Class' Constructor

Kembali membahas proses instantiation dari class, sering ditemui kebutuhan menge-set nilai awal atau kondisi awal dari atribut yang dimiliki oleh class tersebut, sehingga untuk kebutuhan ini digunakan sebuah fungsi khusus yang biasa disebut sebagai pembangun atau dalam bahasa Inggris disebut constructor. Di Python, fungsi khusus atau metode sebagai constructor ini bernama `__init__` atau biasa diucapkan sebagai "double underscore init". Pada saat dilakukan instantiation dari class, metode `__init__` ini secara otomatis akan dipanggil di terlebih dahulu.

Berikut adalah definisi class Kalkulator di atas jika diubah dengan menggunakan constructor.

```
1 class Kalkulator:
2     """contoh kelas kalkulator sederhana"""
3
```

```

4     def __init__(self):
5         self.i = 12345
6
7     def f(self):
8         return 'hello world'

```

Nilai dari atribut i tidak terdefinisi pada awal definisi Kalkulator, setelah dilakukan instantiation maka nilai atribut i akan bernilai 12345. Meskipun bisa mendefinisikan variabel i sebagai atribut dari class Kalkulator, tetapi sebaiknya berhati-hati mengenai variabel yang akan terbagi (shared) untuk semua instance dari class, terutama untuk tipe yang dapat berubah (mutable), misalnya list dan dictionary.

```

1 class KeranjangBelanja:
2     """contoh tidak baik dilakukan dengan definisi variabel terbagi
3     """
4     isi = [] # menggunakan list di sini akan terbagi untuk semua
5     instance. JANGAN DILAKUKAN

```

Lanjut pembahasan constructor, dengan dilengkapi constructor pun proses instantiation tidak berubah dari sebelumnya.

```

1 k = Kalkulator() # membuat instance dari kelas jadi objek, kemudian
2 disimpan pada variabel k

```

Lebih lanjut tentang constructor, tentu saja untuk mendukung aplikasi yang lebih dinamis maka constructor dapat memiliki parameter yang bisa dikirimkan saat proses instantiation, bahkan parameternya bisa lebih dari satu jika diperlukan.

Pada contoh berikut ini, constructor memiliki parameter i yang bersifat opsional, apabila dalam proses instantiation tidak dikirimkan parameter, secara otomatis i akan diisi nilai bawaan 12345.

```

1 class Kalkulator:
2     """contoh kelas kalkulator sederhana"""
3
4     def __init__(self, i=12345):
5         self.i = i # i adalah variabel pada constructor, self.i
6         adalah variabel dari class
7
8     def f(self):
9         return 'hello world'

```

Dengan contoh pemanggilan berikut.

```

1 k = Kalkulator(i=1024) # melakukan instantiation sekaligus mengisi
2     atribut i jadi 1024
3 print(k.i)             # mencetak atribut i dari objek k dengan
4     keluaran nilai 1024

```

## 9.5 Metode (Method)

Pembahasan lebih detail mengenai metode, selain yang dibahas sebelumnya, kita akan membahas 3 jenis metode:

1. Metode dari objek (object method)
2. Metode dari class (class method)
3. Metode secara static (static method)

Pertama kita membahas metode dari objek, seperti yang sempat dijelaskan secara singkat di atas mengenai metode, atau dalam bahasa Inggris disebut method, secara umum metode adalah sebuah fungsi khusus yang menjadi “milik” suatu objek, yakni hasil instantiation dari class.

Salah satu hal khusus yang dimiliki oleh metode dengan adanya argumen bernama self, Anda tentu bertanya-tanya tentang argumen self pada metode-metode dalam kelas tersebut sebetulnya apa? Argumen pertama dari metode-metode dalam class, biasa diberikan nama self sebagai suatu konvensi atau standar penamaan, meskipun Anda bisa juga menggunakan nama lain. Bahkan dalam Python tidak ada arti khusus tentang sintaksis self ini, namun sangat disarankan menggunakan konversi ini agar program Python yang Anda buat akan lebih mudah dimengerti oleh pemrogram lainnya.

Seperti yang Anda sudah perkirakan, untuk sebuah metode, sebetulnya dikirimkan objek (hasil instance dari class) sebagai argumen pertamanya, dalam hal ini bernama self.

Misalnya menggunakan contoh di atas, jika k adalah objek hasil instance dari class Kalkulator, saat melakukan pemanggilan metode f.

```
| k . f ()
```

ekuivalen dengan

```
| Kalkulator . f (k)
```

Argumen self pada metode f akan diisi dengan objek hasil instance dari class Kalkulator.

Sebelum kita membahas yang kedua dan ketiga, yakni metode dari class dan metode secara static, Anda tentu mengingat bahwa sebelumnya sudah belajar fungsi-fungsi bawaan (built-in) dari Python, antara lain: open, sorted, int, str, dan sejumlah lainnya. Terkait metode, ada dua fungsi bawaan yang akan kita bahas, yakni classmethod dan staticmethod.

Classmethod adalah sebuah fungsi yang mengubah metode menjadi metode dari class (class method). Dalam penggunaannya, fungsi ini dijadikan sebagai fungsi decorator @classmethod, kemudian pemanggilannya bisa langsung dari class yang terdefinisi ataupun melalui objek. Metode dari class (class method) menerima masukan class secara implisit sebagai argumen pertama yang secara konvensi diberikan nama cls.

Berdasar contoh yang sama dengan class sebelumnya, berikut adalah metode dari class.

```

1 class Kalkulator:
2     """contoh kelas kalkulator sederhana"""
3
4     def f(self):
5         return 'hello world'
6
7     @classmethod
8     def tambah_angka(cls, angka1, angka2):
9         return '{} + {} = {}'.format(angka1, angka2, angka1 + angka2)

```

Nampak pada kode, sesuai konvensi ada metode yang menggunakan argumen pertama self, sedangkan untuk class method menggunakan konvensi argumen pertama cls.

Untuk melakukan pemanggilan dari class, dilakukan seperti berikut, dimana argumen pertama cls sudah mendapatkan masukan class Kalkulator.

```

1 Kalkulator.tambah_angka(1, 2) # tanpa perlu memberikan masukan untuk
                                argumen cls

```

Metode dari class (class method) juga dapat dipanggil dari objek, hasil instantiation dari class Kalkulator, contohnya mirip seperti pemanggilan metode dari objek (object method).

```

1 k = Kalkulator()
2 k.tambah_angka(1, 2)

```

staticmethod adalah sebuah fungsi yang mengubah metode menjadi metode statis (static method). Dalam penggunaannya, fungsi ini dijadikan sebagai fungsi decorator @staticmethod, kemudian pemanggilannya bisa langsung dari class yang terdefinisi ataupun melalui objek.

Metode statis (static method) tidak menerima masukan argumen pertama secara implisit.

Untuk Anda yang pernah memrogram Java atau C++, metode statis ini mirip seperti yang ada di bahasa pemrograman tersebut.

Berdasar contoh yang sama dengan class sebelumnya, berikut adalah metode statis.

```

1 class Kalkulator:
2     """contoh kelas kalkulator sederhana"""
3
4     def f(self):
5         return 'hello world'
6
7     @staticmethod

```

```
8     def kali_angka(angka1, angka2):  
9         return '{} x {} = {}'.format(angka1, angka2, angka1 * angka2)
```

Nampak pada kode, tidak ada argumen pertama yang implisit seperti halnya pada dua metode sebelumnya.

Pemanggilan dari class seperti halnya pemanggilan fungsi biasa.

```
1 a = Kalkulator.kali_angka(2, 3)  
2 print(a)
```

Metode statis (static method) juga dapat dipanggil dari objek, hasil instantiation dari class Kalkulator, mirip seperti pemanggilan fungsi biasa meskipun dipanggil dari objek.

```
1 k = Kalkulator()  
2 a = k.kali_angka(2, 3)  
3 print(a)
```

## 9.6 Mekanisme Pewarisan (Inheritance)

Paradigma Pemrograman Berorientasi Objek memiliki konsep pewarisan atau dalam bahasa Inggris disebut inheritance, tentunya di Python mendukung fitur ini.

Suatu kelas B dapat mewarisi kelas A, sehingga secara otomatis memiliki semua fitur yang dimiliki oleh kelas A, dalam hal ini atribut-atribut dan metode-metode.

Dalam contoh ini, kelas A disebut sebagai kelas dasar, yakni kelas yang memberikan warisan atau biasa juga disebut kelas yang diturunkan.

Kemudian kelas B disebut sebagai kelas turunan, yakni kelas yang mendapatkan warisan. Jika di kelas B memiliki metode dengan nama yang sama dengan yang dimiliki kelas A, maka metode tersebut akan menimpa metode yang diwariskan dari kelas A.

Catatan:

Frasa kelas dasar adalah terjemahan bahasa Inggris dari frasa base class.

Frasa kelas turunan adalah terjemahan bahasa Inggris dari frasa derived class.

Frasa menimpa metode adalah terjemahan bahasa Inggris dari frasa method override.

Di Python, mekanisme pewarisan memungkinkan untuk memiliki lebih dari satu kelas dasar (kelas orang tua, yang diwarisi)

Kita akan mengembangkan aplikasi yang sudah dimiliki di atas, class Kalkulator sebagai kelas dasar yang mempunyai fungsi melakukan penambahan melalui metode tambah\_angka.

```

1 class Kalkulator:
2     """contoh kelas kalkulator sederhana. anggap kelas ini tidak
3         boleh diubah!"""
4
5     def __init__(self, nilai=0):
6         self.nilai = nilai
7
8     def tambah_angka(self, angka1, angka2):
9         self.nilai = angka1 + angka2
10        if self.nilai > 9: # kalkulator sederhana hanya memroses
11            sampai 9
12            print('kalkulator sederhana melebihi batas angka: {}'.format(
13                self.nilai))
14        return self.nilai

```

Kemudian kita punya kebutuhan membuat sebuah kelas yang punya fitur perkalian tapi juga punya fitur penambahan, dalam contoh ini misalnya kita tidak boleh men-gubuh kalkulator yang sudah ada. Dibandingkan dengan membuat kelas baru kemudian menuliskan kembali implementasi penambahan angka, maka mewarisi kelas yang sudah ada akan lebih efisien.

Dari situ, kita membuat class KalkulatorKali yang mewarisi class Kalkulator.

```

1 class KalkulatorKali(Kalkulator):
2     """contoh mewarisi kelas kalkulator sederhana"""
3
4     def kali_angka(self, angka1, angka2):
5         self.nilai = angka1 * angka2
6         return self.nilai

```

Dengan pemanggilan class KalkulatorKali sebagai berikut.

```

1 kk = KalkulatorKali()
2 a = kk.kali_angka(2, 3) # sesuai dengan definisi class memiliki
3         fitur kali_angka
4 print(a)
5 b = kk.tambah_angka(5, 6) # memiliki fitur tambah_angka karena
6         mewarisi dari Kalkulator
print(b)

```

Dengan melakukan pewarisan, Anda dengan mudah bisa menambahkan (extend) kemampuan dari suatu class dengan fitur yang ingin Anda buat sendiri. Hal tersebut akan sangat berguna jika Anda ingin membuat aplikasi yang mudah diguna-ulang (reusable).

## 9.7 Menimpa (Override) Metode dengan Nama yang Sama Dengan Kelas Dasar

Anda melihat bahwa kalkulator yang anda operasikan mendapatkan peringatan melebihi batas angka yang bisa diproses? Lalu bagaimana kalau Anda ingin mengubah keterbatasan itu? Meski dalam contoh ini anggap tetap tidak boleh mengubah class Kalkulator yang sudah ada.

Dalam proses pewarisan, kita bisa menimpa (override) definisi metode yang dimiliki oleh kelas dasar (kelas orang tua, yang diwarisi) dengan nama metode yang sama. Misalnya kita menimpa metode tambah\_angka untuk menghilangkan batasan yang dimiliki.

```

1 class KalkulatorKali(Kalkulator):
2     """contoh mewarisi kelas kalkulator sederhana"""
3
4     def kali_angka(self , angkal , angka2):
5         self.nilai = angkal * angka2
6         return self.nilai
7
8     def tambah_angka(self , angkal , angka2):
9         self.nilai = angkal + angka2
10        return self.nilai

```

Kemudian kita coba kembali, apakah batasan yang dimiliki sudah hilang?

```

1 kk = KalkulatorKali()
2
3 b = kk.tambah_angka(5 , 6) # fitur tambah_angka yang dipanggil milik
   KalkulatorKali
4 print(b)

```

## 9.8 Pemanggilan Metode Kelas Dasar dari Kelas Turunan dengan Sintaksis Super

Anggaplah fungsi tambah\_angka adalah sebuah fungsi yang rumit, dimana kita sebaiknya gunakan saja kemampuan yang sudah ada di kelas dasar, kemudian kita hanya ubah sebagian fiturnya saja dengan yang kita inginkan.

```

1 class KalkulatorTambah(Kalkulator):
2     """contoh mewarisi kelas kalkulator sederhana"""
3
4     def tambah_angka(self , angkal , angka2):
5         if angkal + angka2 <= 9: # fitur ini sudah oke di kelas
           dasar , gunakan yang ada saja
6             super().tambah_angka(angkal , angka2) # panggil fungsi
           dari Kalkulator lalu isi nilai
7         else: # ini adalah fitur baru yang ingin diperbaiki dari
           keterbatasan kelas dasar
8             self.nilai = angkal + angka2

```

```
9     return self.nilai
```

## 9.9 Variabel Privat di Python

Jika Anda sebelumnya pernah belajar bahasa pemrograman yang memiliki variabel privat, dimana variabel tersebut tidak dapat diakses kecuali dari objek yang bersangkutan, di Python hal tersebut tidak ada.

Terkait variabel privat tersebut, di Python ada konvensi dimana penggunaan nama yang diawali dengan garis bawah (underscore), baik itu fungsi, metode, maupun anggota data, akan dianggap sebagai non-publik.

## 9.10 Pernak-Pernik Terkait Struktur Data

Buat Anda yang pernah membuat program dengan menggunakan bahasa pemrograman C atau Pascal, Anda mungkin tertarik untuk membuat sebuah struktur data seperti halnya struct pada C atau record pada Pascal, bertujuan menyatukan sejumlah penamaan item data menjadi satu.

Dalam Python, dimana Anda sebelumnya pernah mempelajari mengenai duck typing, maka Anda cukup mendefinisikan saja sebuah class kosong, selanjutnya penamaan item data dapat secara langsung didefinisikan dan diisi saat sudah instantiation.

```
1 class Pegawai:
2     pass # definisi class kosong
3
4 don = Pegawai() # membuat Pegawai baru menjadi objek bernama don
5
6 # tambahkan item data pada objek sebagai record
7 don.nama = 'Don Doo'
8 don.bagian = 'IT'
9 don.gaji = 999
```

# BAB 10

---

## MEMBUAT CHATBOT

---

### 10.1 Installasi

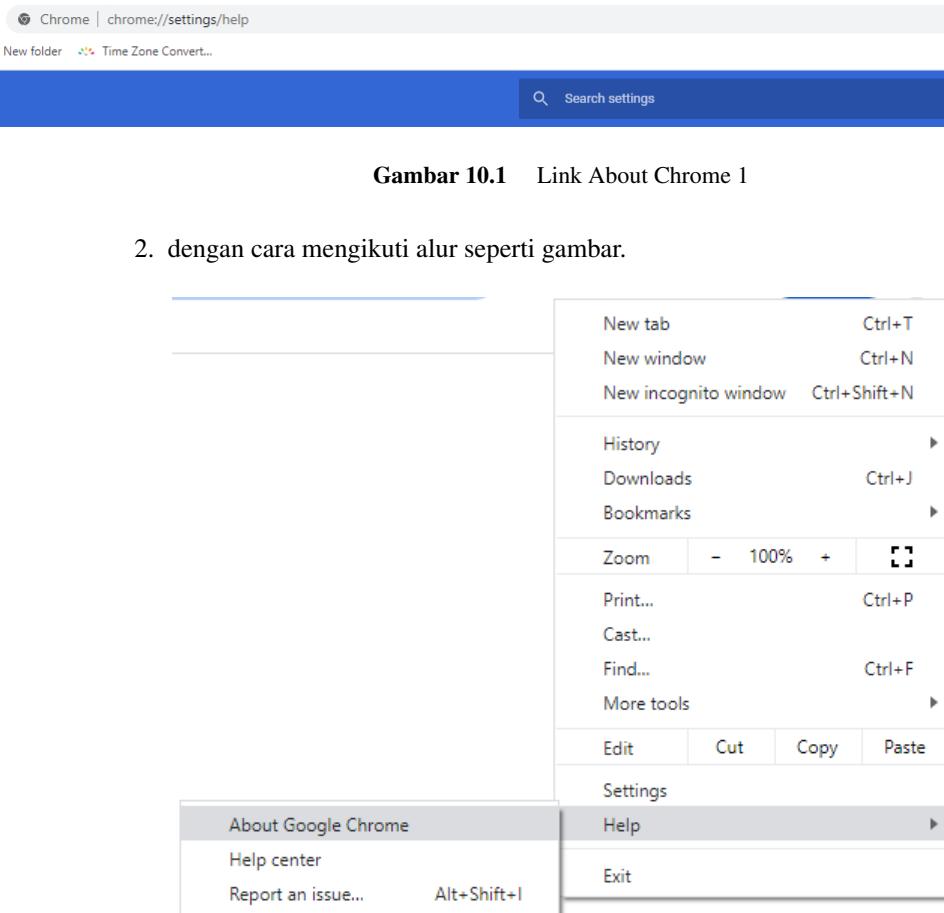
#### 10.1.1 Python / Anaconda

Installasi ini pastikan sudah dikerjakan pada modul bagian Peneganalan Anaconda karena disitu sudah lengkap tutorial cara installasi dengan os ubuntu 19.04 dan windows 10.

#### 10.1.2 Chrome dan ChromeDriver

Pastikan sudah menginstall chrome dengan versi terbaru jika sudah install periksa update dari chrome dengan cara:

1. mengetikkan link about chrome seperti berikut.



Gambar 10.2 Link About Chrome 2

## 10.2 Kodingan Wanda

```

1 from selenium import webdriver
2 from selenium.webdriver.common.keys import Keys
3 from selenium.webdriver.support.ui import WebDriverWait
4 from selenium.webdriver.support import expected_conditions as EC
5 from selenium.webdriver.common.by import By
6 from oauth2client.service_account import ServiceAccountCredentials
7 from dateutil.parser import parse
8 from time import sleep
9 from googletrans import Translator
10 import datetime
11 import face_recognition
12 import cv2

```

```

13 import numpy as np
14 import gspread
15 import os
16 import dawet
17 import random
18 import subprocess

```

### Listing 10.1 modul

```

1 def getNilaiMahasiswa(self, npm, pertemuan):
2     if npm[:3] == "118":
3         db = dawet.Dawet("BukuProyek2")
4     elif npm[:3] == "117":
5         db = dawet.Dawet("BukuProyek3")
6     else:
7         db = dawet.Dawet("BukuInternship1")
8
9     nilai = db.getData(npm, pertemuan, 0)
10
11    if nilai == "not_found":
12        return "invalid"
13    elif nilai == "pertemuan_not_found":
14        return "pertemuan_invalid"
15    else:
16        nama_mahasiswa = db.getData(npm, "nama", 0)
17        nilai_rata = db.getData(npm, "rata_rata", 0)
18
19        hasil = []
20        hasil.append(nilai)
21        hasil.append(nama_mahasiswa)
22        hasil.append(nilai_rata)
23
24    return hasil

```

### Listing 10.2 getnilaiMahasiswa

```

1 def cekJadwalSidang(self, pilihan):
2     db = dawet.Dawet("Jadwal_Sidang_Proyek_2")
3
4     allData = db.getAllData(1)
5
6     sekarang = datetime.datetime.now().date()
7     besok = sekarang + datetime.timedelta(days=1)
8     kemaren = sekarang - datetime.timedelta(days=1)
9     lusa = sekarang + datetime.timedelta(days=2)
10
11    if pilihan == "sekarang":
12        self.pilihanTanggal = sekarang
13        runnerVariable = 1
14    if pilihan == "besok":
15        self.pilihanTanggal = besok
16        runnerVariable = 1
17    if pilihan == "kemarin":
18        self.pilihanTanggal = kemaren
19        runnerVariable = 1
20    if pilihan == "lusa":
21        self.pilihanTanggal = lusa

```

```

22     runnerVariable = 1
23
24     if runnerVariable == 1:
25         for data in allData:
26             try:
27                 tanggal = parse(data[0]).date()
28                 print(self.pilihanTanggal)
29
30                 print(tanggal)
31
32                 if self.pilihanTanggal == tanggal:
33                     getIndex = allData.index(data)
34
35                     nextData = allData[getIndex:]
36
37                     for nextdata in nextData:
38                         if nextdata[0] == ':':
39                             getIndexNull = nextData.index(
40                                 nextdata)
41
42                             result = nextData[:getIndexNull]
43
44                             return result
45
46                         except:
47                             print("beda")
48             else:
49                 return "no_pilihan"

```

**Listing 10.3** cekjadwal sidang

```

1 def saveProfile(self):
2     self.options = webdriver.ChromeOptions()
3     self.options.add_argument('--user-data-dir=./user_data')
4     self.driver = webdriver.Chrome(chrome_options=self.options)

```

**Listing 10.4** saveProfile

```

1 def splitString(self, string):
2     li = list(string.split(" "))
3     return li

```

**Listing 10.5** splitString

```

1 def waitLogin(self):
2     self.target = '_3RWII'
3     self.x_arg = '//div[contains(@class, ' + self.target + ')]'
4     self.wait = WebDriverWait(self.driver, 600)
5     self.wait.until(EC.presence_of_element_located((By.XPATH,
self.x_arg)))

```

**Listing 10.6** waitLogin

```

1 def typeAndSendMessage(self, message):
2     self.message_target = self.driver.find_elements_by_xpath(
3         '//*[@id="main"]/footer/div[1]/div[2]/div/div[2]')[0]
4     self.message_target.send_keys(message)

```

```
4     self.sendbutton = self.driver.find_elements_by_xpath('//*[@id  
5     ="main"]/footer/div[1]/div[3]/button')[0]  
      self.sendbutton.click()
```

**Listing 10.7** typeAndSendMessage

```
1 def deleteMessage(self):  
2     self.driver.find_elements_by_class_name('_3j8Pd')[-1].click()  
3     sleep(1)  
4  
5     value_name = self.driver.find_elements_by_class_name('_3zy-4')  
6     sleep(1)  
7  
8     if 'Exit group' in value_name[4].text:  
9         print('group')  
10        value_name[3].click()  
11        self.driver.find_elements_by_class_name('_2eK7W')[1].  
12        click()  
13    else:  
14        print('personal')  
15        value_name[4].click()  
16        self.driver.find_elements_by_class_name('_2eK7W')[1].  
17        click()
```

**Listing 10.8** deleteMessage

