

# **DASAR-DASAR PYTHON**



---

# DASAR-DASAR PYTHON

---

**Rolly M. Awangga**  
Informatics Research Center



**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



# CONTENTS IN BRIEF

---

<b>1</b>	<b>Python</b>	<b>1</b>
<b>2</b>	<i>Python</i>	<b>3</b>
<b>3</b>	<b>Judul Bagian Kedua</b>	<b>23</b>
<b>4</b>	<b>Fungsi dan Kelas</b>	<b>29</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 Python</b>	<b>1</b>
<b>2 Python</b>	<b>3</b>
2.1 Sejarah Python	3
2.2 Perbedaan Python 2.x dan Python 3.x	4
	<b>ix</b>

2.3	Installasi Python	11
2.3.1	Windows (Windows 10)	11
2.4	Instalasi Pip	16
2.4.1	Windows (Windows 10)	16
2.4.2	Linux (Ubuntu 19.04)	17
2.5	Setting Environment	18
2.5.1	Windows (Windows 10)	18
2.5.2	Linux (Ubuntu 19.04)	20
2.6	Command Line Interface/Interpreter	21
2.6.1	Windows (Windows 10)	21
2.6.2	Linux (Ubuntu 19.04)	22
<b>3</b>	<b>Judul Bagian Kedua</b>	<b>23</b>
3.1	Variabel	23
3.2	Input dan Output	24
3.3	Operasi Aritmatika	24
3.4	Perulangan	24
3.4.1	For	25
3.4.2	While	25
3.5	Kondisi	25
3.6	Error	27
3.7	Try Except	28
<b>4</b>	<b>Fungsi dan Kelas</b>	<b>29</b>
4.1	Teori	29
4.1.1	Fungsi	29
4.2	Package	30
4.3	Class, Object, Atribute, and Method	30
4.4	Pemanggilan Class	31
4.5	Pemakaian Package Fungsi Apabila File Didalam Folder	31
4.6	Pemakaian Package Kelas Apabila File didalam Folder	31
Daftar Pustaka		33

## DAFTAR GAMBAR

---

2.1	Gambar hasil print	4
2.2	Gambar perintah print	5
2.3	Gambar hasil pembagian	6
2.4	Gambar perintah pembagian	6
2.5	Gambar hasil error	7
2.6	Gambar perintah error	7
2.7	Gambar hasil looping	8
2.8	Gambar perintah looping	8
2.9	Gambar hasil unicode (bytes)	9
2.10	Gambar perintah unicode (bytes)	9
2.11	Gambar hasil unicode	10
2.12	Gambar perintah unicode	10
2.13	Run Setup Anaconda	11

2.14	Setup Loading	12
2.15	Welcome to Anaconda Setup	12
2.16	<i>License Agreement</i>	12
2.17	<i>Just Me(recomended)</i>	13
2.18	<i>Pilih lokasi</i>	13
2.19	<i>Centang Anaconda to my PATH</i>	14
2.20	<i>Installation Complete</i>	14
2.21	<i>Installation Complete</i>	15
2.22	<i>Anaconda+JetBrains</i>	15
2.23	<i>Thanks for install Anaconda</i>	16
2.24	<i>Install pip</i>	16
2.25	<i>Install pip Selesai</i>	17
2.26	<i>Melihat Versi pip</i>	17
2.27	Gambar instal pip	18
2.28	<i>Properties</i>	18
2.29	<i>Advanced system settings</i>	19
2.30	<i>Environment Variables</i>	19
2.31	<i>Path</i>	20
2.32	<i>Edit Environment Variable</i>	20
2.33	Gambar setpath	21
2.34	<i>CLI in Command Prompt</i>	22
2.35	Gambar running script dengan CLI	22

## DAFTAR TABEL

---



## Listings

---



# **FOREWORD**

---

Sepatah kata dari Kaprodi, Kabag Kemahasiswaan dan Mahasiswa



# KATA PENGANTAR

---

Buku ini diciptakan bagi yang awam dengan git sekalipun.

R. M. AWANGGA

*Bandung, Jawa Barat*

*Februari, 2019*



## ACKNOWLEDGMENTS

---

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

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

R. M. A.



## ACRONYMS

---

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



## GLOSSARY

---

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



# SYMBOLS

---

$A$  Amplitude

$\&$  Propositional logic symbol

$a$  Filter Coefficient

$B$  Number of Beats



# INTRODUCTION

---

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

Informatics Research Center  
Bandung, Jawa Barat, Indonesia

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

Rumusnya cuman sebagai contoh aja biar keren[1].

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



## **BAB 1**

---

# PYTHON

---



## BAB 2

---

# PYTHON

---

### 2.1 Sejarah *Python*

Python merupakan bahasa pemrograman tingkat tinggi yang dapat digunakan banyak hal, *Python* awalnya dirancang oleh **Guido van Rossum** pada tahun 1980 yang mana nama *Python* sebelum sebesar sekarang yaitu *ABC Programming Language* yang dijalankan di sistem operasi bernama *Amoeba Operating System*. Guido merasakan kehebatan dan kemampuan serta fitur yang berada pada bahasa pemrograman ABC ini sehingga Guido mengambil siktaks-sintaks yang berada pada bahasa pemrograman ABC ini, tentu saja banyak komplain yang berdatangan sehingga Guido terus melakukan perbaikan pada bahasa pemrograman yang sedang ia buat kala itu. Lalu, disinilah nama *Python* muncul dimuka bumi sebagai bahasa pemrograman, disaat Guido sedang menonton televisi dan menemukan kata '*Monty Python's Flying Circus*'.

Bahasa *Python* secara resmi dirilis pada tahun 1991, saat rilis pertama kali semua orang terkejut dengan sintaks yang dimiliki oleh *Python* ketika dibandingkan dengan bahasa lain seperti *Java*, *C*, *C++*, dan lain-lain pengekspresian bahasa ini cukup sederhana. Tujuan dari dibuatnya bahasa pemrograman ini adalah untuk memper-

mudah dalam membaca sebuah kode dari penulisan sintaks dan produktivitas dalam hal pengembangan tingkat lanjut.

## 2.2 Perbedaan Python 2.x dan Python 3.x

Banyak perbedaan yang akan kita temui jika kita dahulu pernah menggunakan *python* versi 2.x cukup lama sehingga berpindah ke versi 3.x, berikut contoh perbedaan pada *python* versi 2.x dan 3.x yang sangat penting untuk diketahui:

1. Perintah **print** Perbedaan perintah *print* pada dua versi ini adalah python 2.x tidak memakai kurung dan 3.x memakai kurung untuk perintah *print* bisa dilihat pada gambar 2.1 dan 2.2



```
Q                                     burger-man@burgerman-K46CB:~  
(base) burger-man@burgerman-K46CB:~$ python2 print.py  
Ga pake kurung  
(base) burger-man@burgerman-K46CB:~$ python3 print.py  
File "print.py", line 1  
    print "Ga pake kurung"  
          ^  
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("Ga pake kurung")?  
(base) burger-man@burgerman-K46CB:~$
```

Gambar 2.1 Gambar hasil print

Q burger-man@burgerman-K46CB:~

GNU nano 3.2 print.py

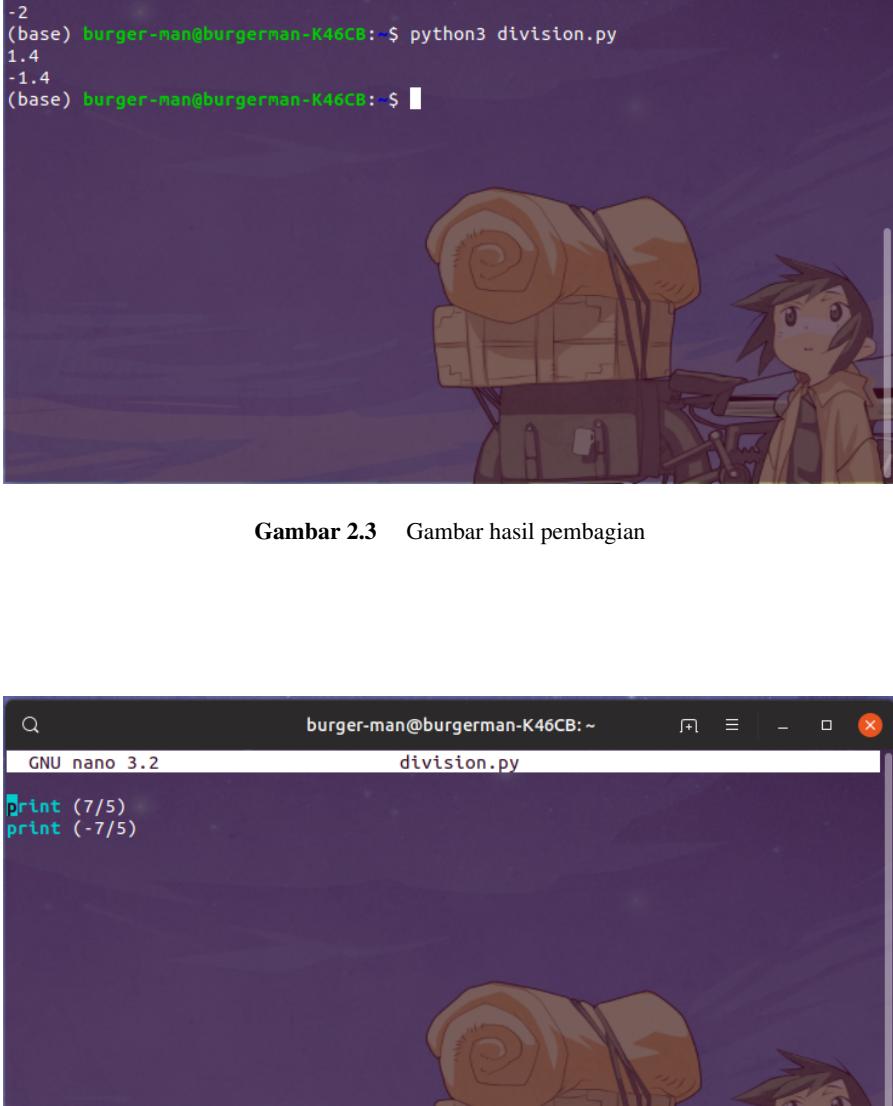
```
print "Ga pake kurung"
```

[ Read 1 line ]

**^G** Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos  
**^X** Exit **^R** Read File **^|** Replace **^U** Uncut Text **^T** To Spell **^** Go To Line

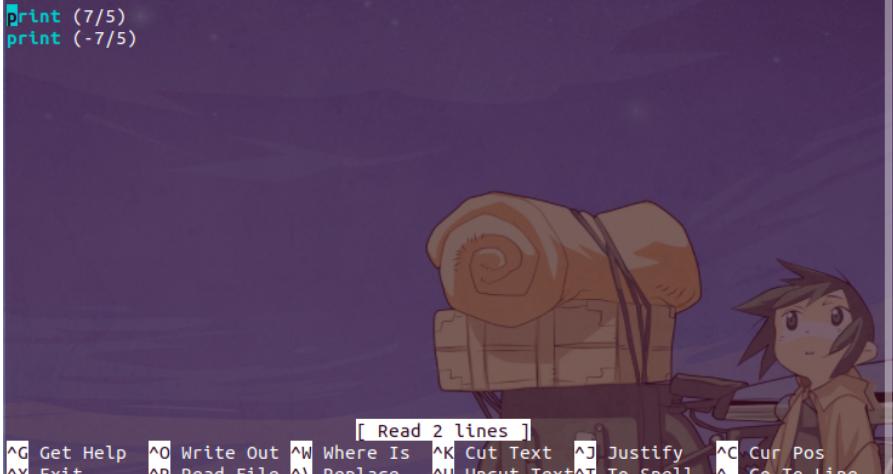
**Gambar 2.2** Gambar perintah print

2. Perintah pembagian **integer** Hasil dari perintah pembagian cukup jelas berbeda yang mana versi 2.x tidak secara mendetail untuk hasilnya sehingga angka yang dihasilkan bilangan **integer** sedangkan versi 3.x bertipe **float** perbedaannya bisa dilihat pada gambar 2.3 dan 2.4.



```
Q                                burger-man@burgerman-K46CB:~      ⌂  ⌄  ⌁  ⌂  ⌁  ⌁  ⌁  ⌁  ⌁  ⌁
(base) burger-man@burgerman-K46CB:~$ python2 division.py
1
-2
(base) burger-man@burgerman-K46CB:~$ python3 division.py
1.4
-1.4
(base) burger-man@burgerman-K46CB:~$
```

Gambar 2.3 Gambar hasil pembagian



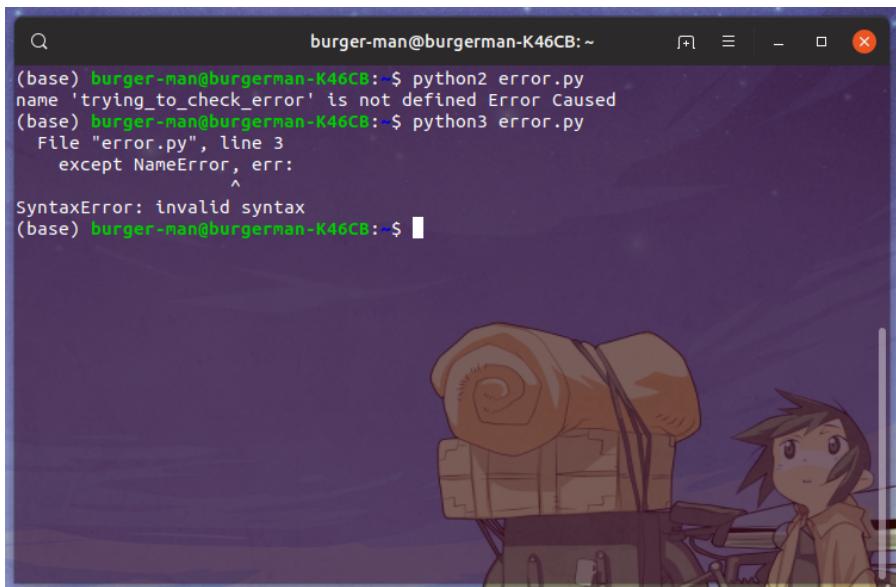
```
Q                                burger-man@burgerman-K46CB:~      ⌂  ⌄  ⌁  ⌂  ⌁  ⌁  ⌁  ⌁  ⌁  ⌁
GNU nano 3.2                         division.py
```

```
print (7/5)
print (-7/5)
```

```
[ Read 2 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit     ^R Read File  ^A Replace   ^U Uncut Text ^T To Spell  ^  Go To Line
```

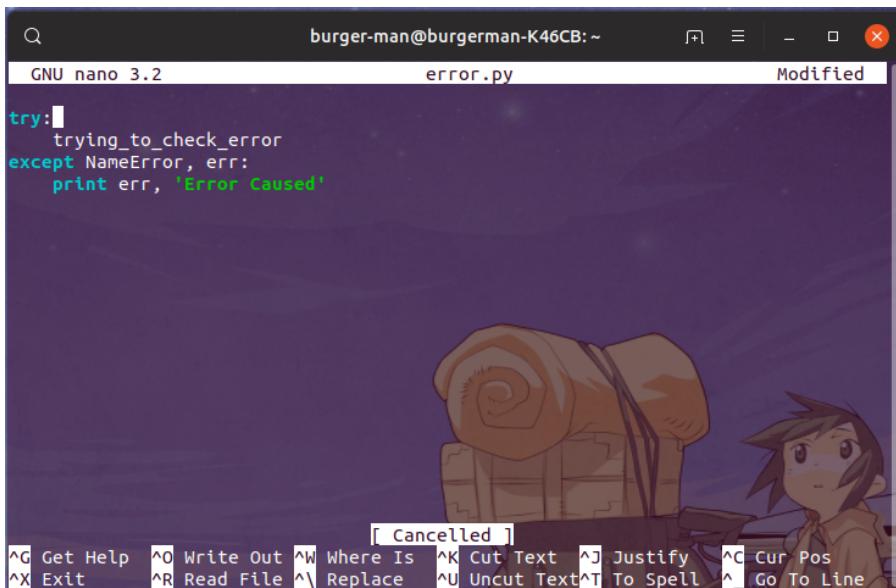
Gambar 2.4 Gambar perintah pembagian

3. **Try and Except** Perbedaan pada *try and except* hanya berbeda di penggunaan , untuk versi 2.x dan as untuk versi 3.x.



```
burger-man@burgerman-K46CB:~$ python2 error.py
name 'trying_to_check_error' is not defined Error Caused
(base) burger-man@burgerman-K46CB:~$ python3 error.py
File "error.py", line 3
    except NameError, err:
                       ^
SyntaxError: invalid syntax
(base) burger-man@burgerman-K46CB:~$
```

**Gambar 2.5** Gambar hasil error



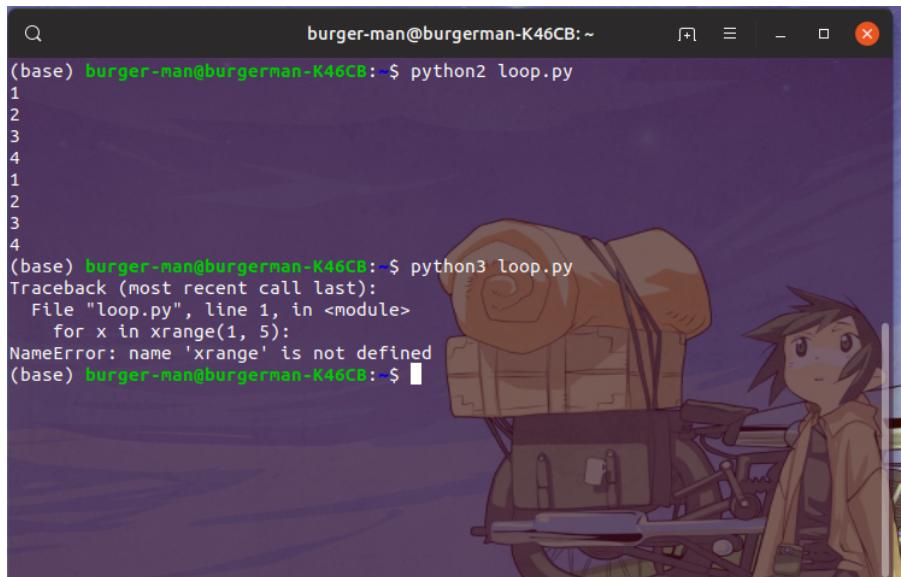
```
GNU nano 3.2          error.py          Modified
try:
    trying_to_check_error
except NameError, err:
    print err, 'Error Caused'
```

[ Cancelled ]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos  
^X Exit ^R Read File ^N Replace ^U Uncut Text ^T To Spell ^L Go To Line

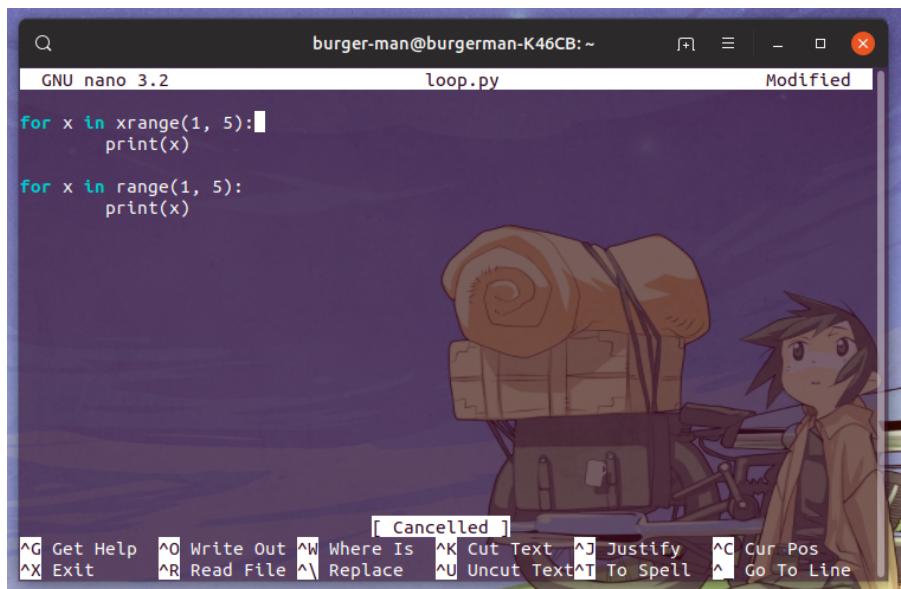
**Gambar 2.6** Gambar perintah error

4. **Looping** Perbedaan pada looping hanya saja versi 3.x tidak bisa menggunakan sintaks **xrange** lagi.



```
Q burger-man@burgerman-K46CB:~$ python2 loop.py
1
2
3
4
1
2
3
4
(base) burger-man@burgerman-K46CB:~$ python3 loop.py
Traceback (most recent call last):
  File "loop.py", line 1, in <module>
    for x in xrange(1, 5):
NameError: name 'xrange' is not defined
(base) burger-man@burgerman-K46CB:~$
```

Gambar 2.7 Gambar hasil looping

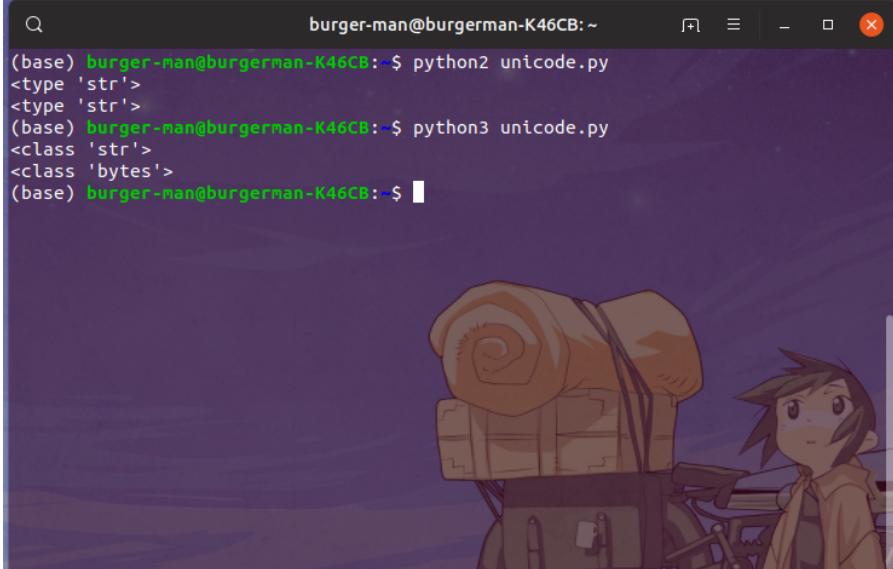


```
Q burger-man@burgerman-K46CB:~$ GNU nano 3.2          loop.py          Modified
for x in xrange(1, 5):
    print(x)

for x in range(1, 5):
    print(x)
```

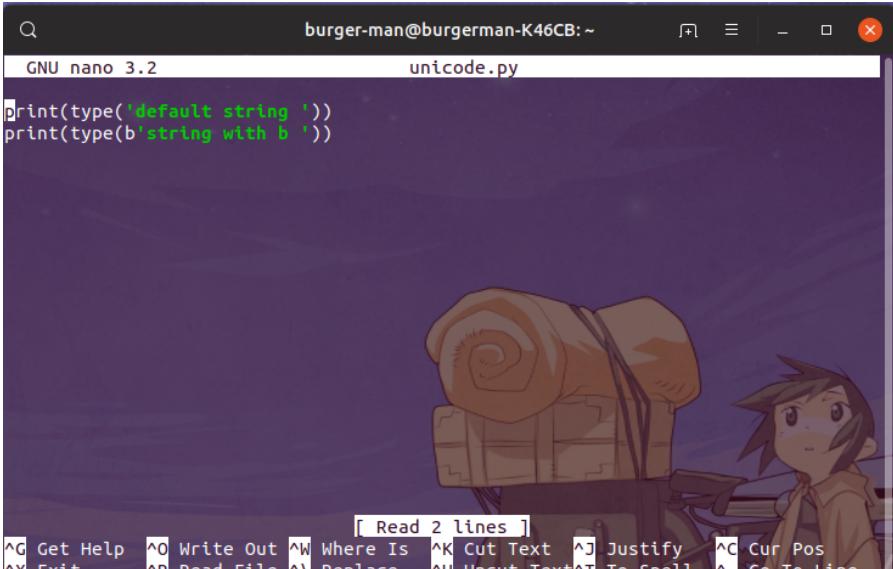
Gambar 2.8 Gambar perintah looping

5. **Unicode** Unicode ini cukup penting karena kita mengetahui bagaimana setiap versi merespons setiap unicode yang diberikan.



```
Q burger-man@burgerman-K46CB:~$ python2 unicode.py
<type 'str'>
<type 'str'>
(base) burger-man@burgerman-K46CB:~$ python3 unicode.py
<class 'str'>
<class 'bytes'>
(base) burger-man@burgerman-K46CB:~$
```

**Gambar 2.9** Gambar hasil unicode (bytes)



```
Q burger-man@burgerman-K46CB:~$ GNU nano 3.2          unicode.py
print(type('default string '))
print(type(b'string with b '))
```

[ Read 2 lines ]
<b>^G</b> Get Help <b>^O</b> Write Out <b>^W</b> Where Is <b>^K</b> Cut Text <b>^J</b> Justify <b>^C</b> Cur Pos
<b>^X</b> Exit <b>^R</b> Read File <b>^V</b> Replace <b>^U</b> Uncut Text <b>^T</b> To Spell <b>^L</b> Go To Line

**Gambar 2.10** Gambar perintah unicode (bytes)

Pada gambar 2.9 terlihat jelas bahwa perintah *bytes* hanya direspon pada versi 3.x sedangkan versi 2.x merespon *string*



```
Q burger-man@burgerman-K46CB:~$ python2 unicode.py
<type 'str'>
<type 'unicode'>
(base) burger-man@burgerman-K46CB:~$ python3 unicode.py
<class 'str'>
<class 'str'>
(base) burger-man@burgerman-K46CB:~$
```

Gambar 2.11 Gambar hasil unicode



```
Q burger-man@burgerman-K46CB:~$ GNU nano 3.2          unicode.py
^C Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit      ^R Read File ^A Replace   ^U Uncut Text ^T To Spell ^L Go To Line
[ Read 2 lines ]
```

Gambar 2.12 Gambar perintah unicode

## 2.3 Installasi Python

Untuk installasi kali ini akan bagi menjadi dua sistem operasi yaitu Windows (Windows 10), dan Linux (Ubuntu 19.04). Installasi menggunakan *environment* Anaconda sebagai installasi *python*. Anaconda merupakan *environment open-source* untuk bahasa pemrograman *Python*, dan *R* berfungsi untuk memanajemen penggunaan *package* pada *python* dan *R*.

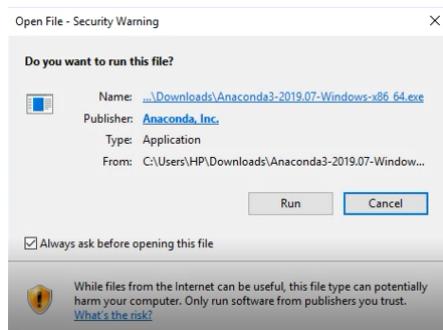
### 2.3.1 Windows (Windows 10)

Hal yang harus diperhatikan sebelum melakukan instalasi *Anaconda Python*

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

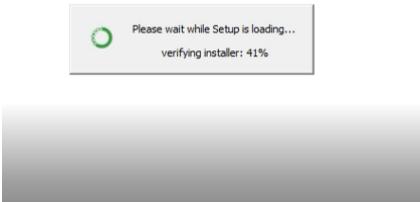
Berikut langkah-langkah instalasi anaconda.

1. Buka aplikasi *installer Anaconda* tersebut lalu akan muncul gambar *installer anaconda*.



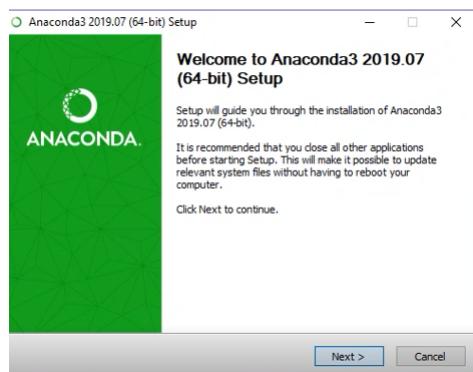
**Gambar 2.13** Run Setup Anaconda

2. Tunggu hingga *setup loading* selesai



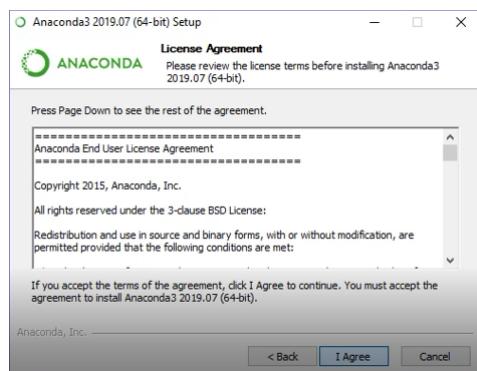
**Gambar 2.14** Setup Loading

3. Jika *setup loading* telah selesai, maka klik *next*



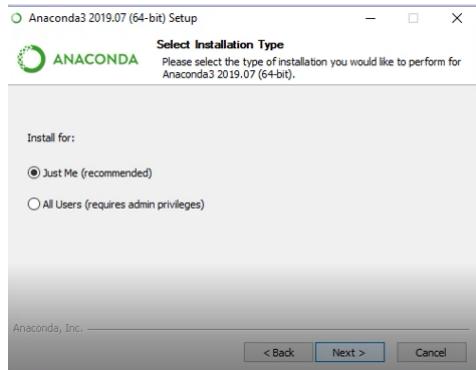
**Gambar 2.15** Welcome to Anaconda Setup

4. Pada *License Agreement* klik *I Agree* gambar *License Agreement*.



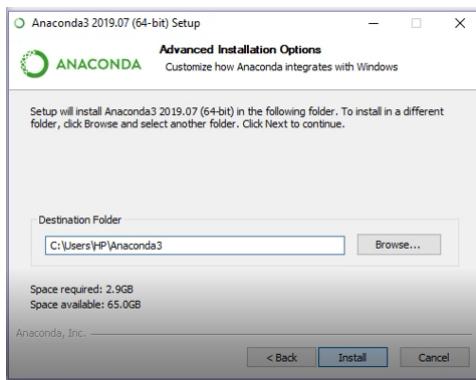
**Gambar 2.16** License Agreement

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



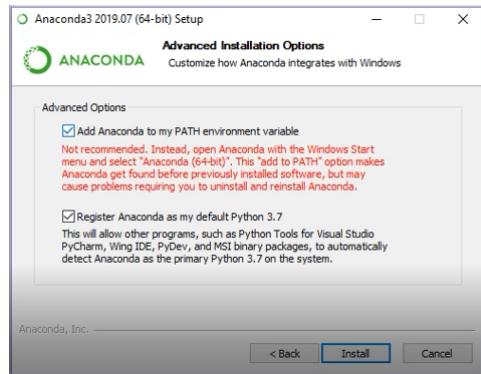
Gambar 2.17 Just Me(*recomended*)

6. Kemudian pilih lokasi tempat *menginstall anaconda* gambar *Pilih lokasi*.



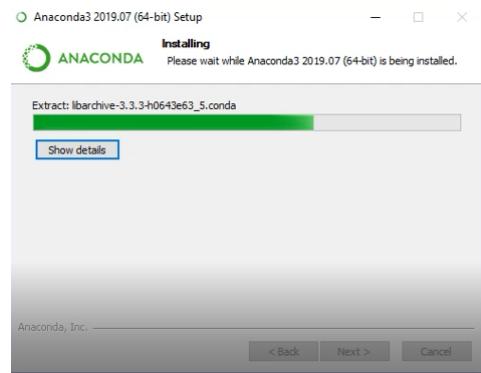
Gambar 2.18 Pilih lokasi

7. Kemudian centang *Add Anaconda to my Path environment variable*, agar saat *menginstall selenium* langsung ke *path anaconda* tidak ke aplikasi yang lain. Klik *install* gambar *Centang Anaconda to my PATH*.



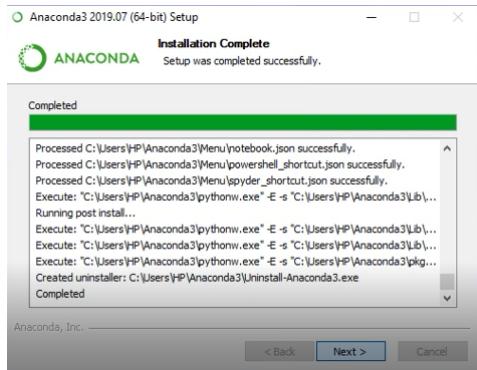
**Gambar 2.19** Centang Anaconda to my PATH

8. Tunggu sampai proses *installasi* selesai gambar *Installation Complete*.



**Gambar 2.20** Installation Complete

9. Apabila instalasi telah selesai klik *next*



Gambar 2.21 Installation Complete

10. klik next



Gambar 2.22 Anaconda+JetBrains

11. Jika sudah klik finish gambar Thanks fo install Anaconda.



Gambar 2.23 Thanks for install Anaconda

## 2.4 Instalasi Pip

### 2.4.1 Windows (Windows 10)

1. buka anaconda prompt
2. ketikkan conda install -c anaconda pip

```
(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
```

A screenshot of the Anaconda Prompt window titled "Anaconda Prompt (Anaconda3)". It shows the command "conda install -c anaconda pip" being run. The output indicates that a newer version of conda exists (4.7.12) and provides instructions to update. It then lists the packages to be downloaded: ca-certificates and certifi. A table at the bottom shows the package names, builds, and sizes.

Gambar 2.24 Install pip

3. ketik y, lalu enter. Tunggu hingga proses instalasi selesai.

```
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 2.25** *Install pip Selesai*

4. jika telah selesai, lakukan pengecekan versi pip dengan mengetikkan pip -V

```
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   Show version and exit.
-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:passwd@]proxy.server:port.
--retries <retries> Maximum number of retries each connection should attempt (default 5 times).
--timeout <sec> Set the socket timeout (default 15 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 alternate 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

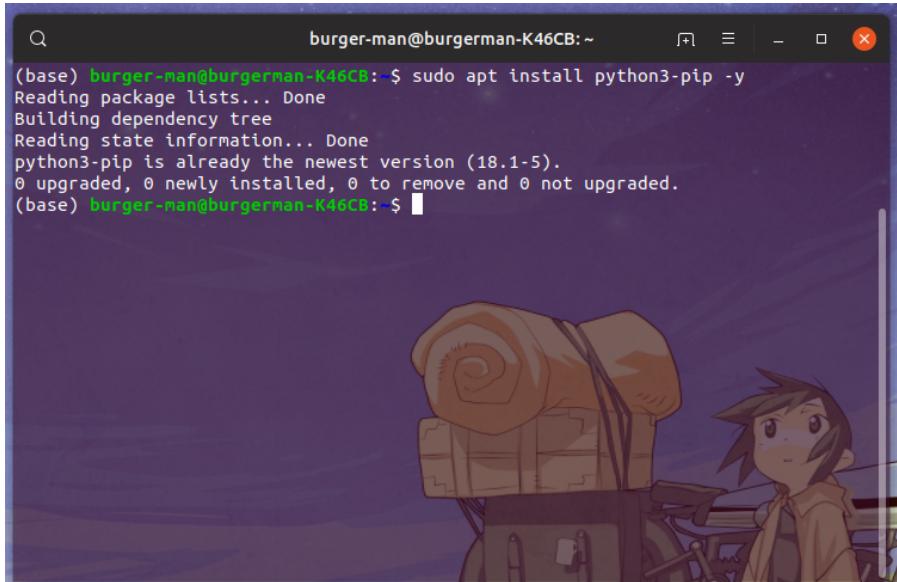
(base) C:\Users\trian>pip -V
pip 19.1.1 from C:\Users\trian\Anaconda3\lib\site-packages\pip (python 3.7)

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

**Gambar 2.26** *Melihat Versi pip*

#### 2.4.2 Linux (Ubuntu 19.04)

1. pertama kita buka terminal kita lalu ketikkan perintah **sudo apt install python3-pip -y** untuk pip3 dan **sudo apt install python-pip -y** untuk pip contoh seperti gambar 2.27, lalu enter



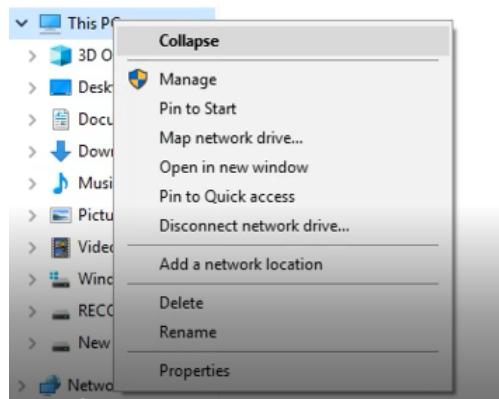
```
(base) burger-man@burgerman-K46CB:~$ sudo apt install python3-pip -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-pip is already the newest version (18.1-5).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
(base) burger-man@burgerman-K46CB:~$
```

Gambar 2.27 Gambar instal pip

## 2.5 Setting Environment

### 2.5.1 Windows (Windows 10)

1. Buka file explorer
2. Klik kanan pada This pc, lalu pilih properties



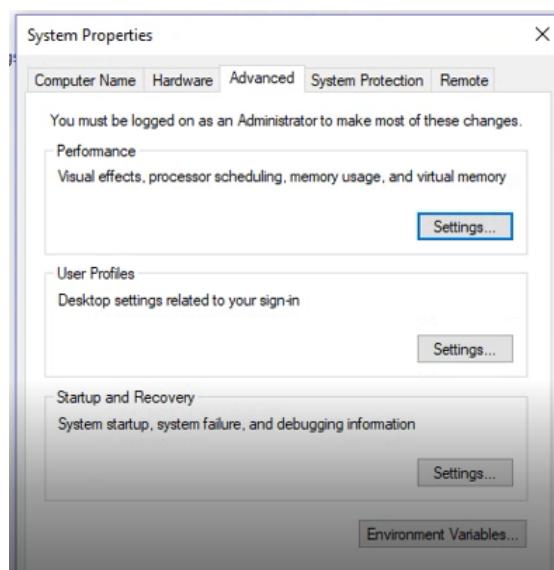
Gambar 2.28 Properties

### 3. Pilih menu Advanced system settings



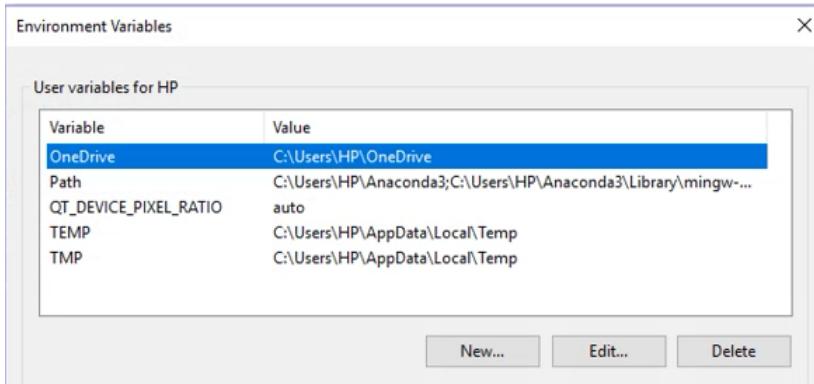
**Gambar 2.29** Advanced system settings

### 4. Pilih Environment Variables

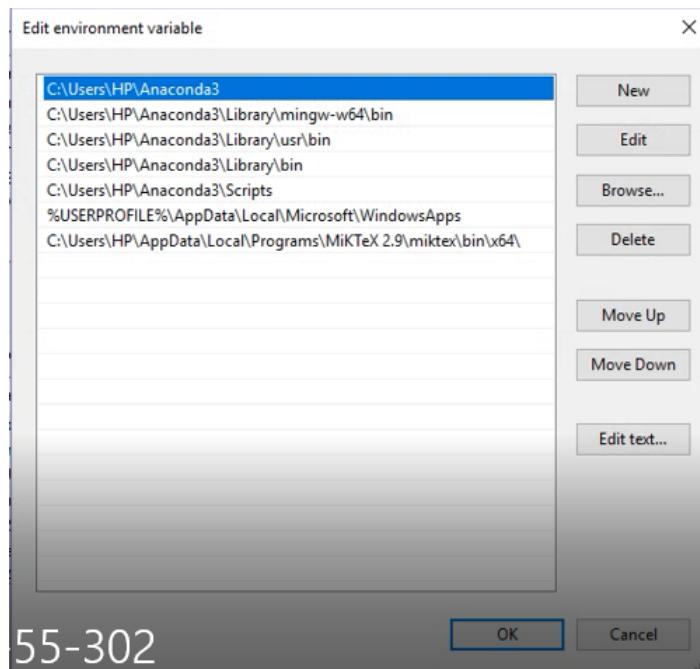


**Gambar 2.30** Environment Variables

### 5. Pilih Path

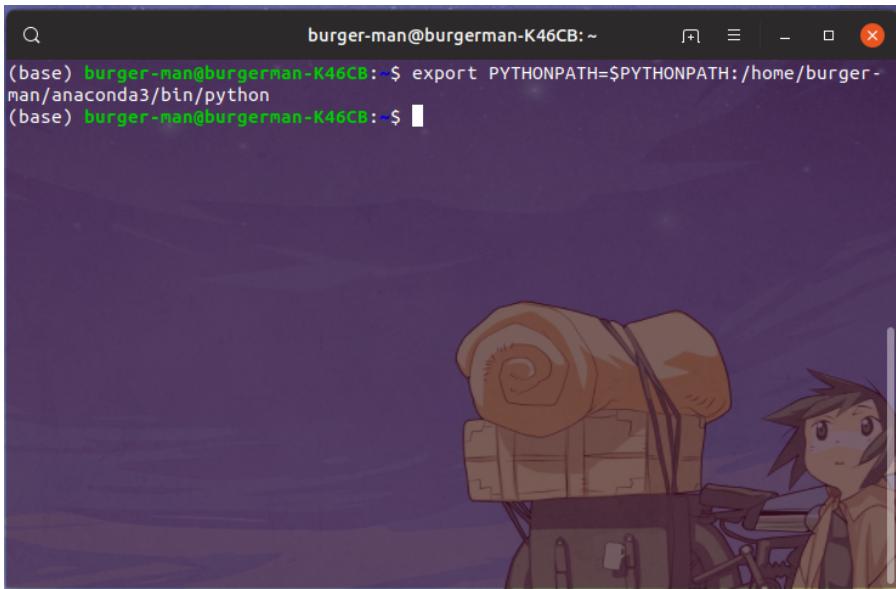
**Gambar 2.31 Path**

6. lalu pilih environment variable yang ingin ditambahkan, klik OK

**Gambar 2.32 Edit Environment Variable**

### 2.5.2 Linux (Ubuntu 19.04)

1. pertama kita buka terminal kita lalu ketikkan perintah export PYTHONPATH=\$PYTHONPATH: contoh seperti gambar 2.33, lalu enter

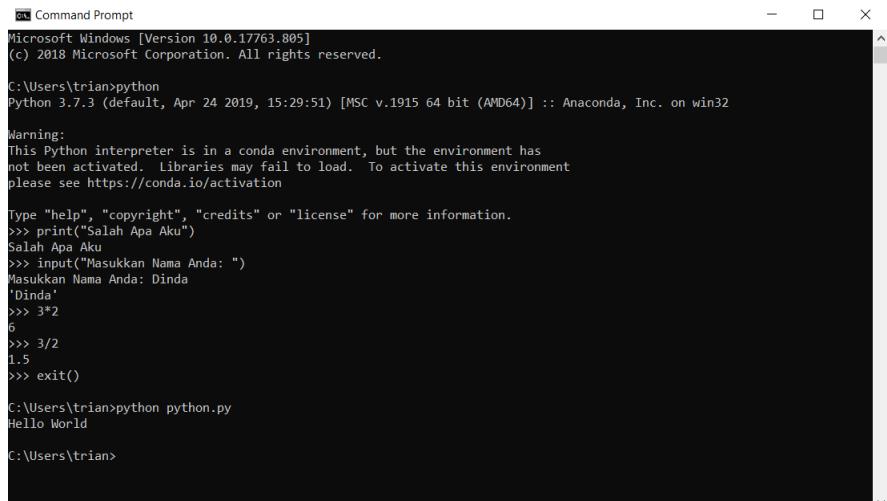


**Gambar 2.33** Gambar setpath

## 2.6 Command Line Interface/Interpreter

### 2.6.1 Windows (Windows 10)

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



```
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
6
>>> 3/2
1.5
>>> exit()

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

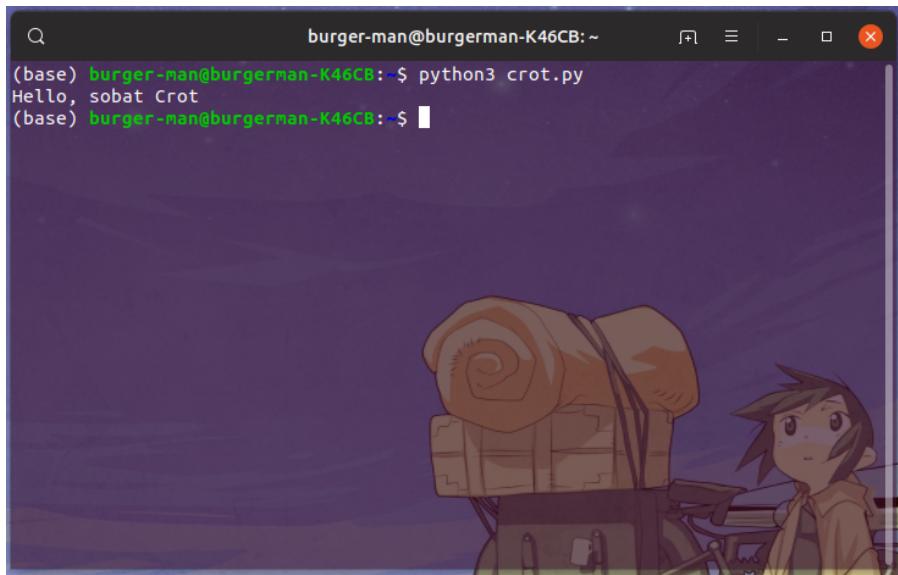
C:\Users\trian>
```

Gambar 2.34 CLI in Command Prompt

## 2.6.2 Linux (Ubuntu 19.04)

Untuk menjalankan perintah CLI cukup mudah yaitu sebagai berikut

1. Buka terminal lalu ketikkan **python namafile.py** seperti gambar 2.35, lalu enter



Gambar 2.35 Gambar running script dengan CLI

## BAB 3

---

# JUDUL BAGIAN KEDUA

---

### 3.1 Variabel

Variabel adalah sebuah tempat untuk menampung value dimemori, dapat dimisalkan seperti sebuah ruangan atau wadah, variabel dibagi dua berdasarkan ruang lingkup yaitu variable lokal dan global, untuk menentukan variabel global atau lokal itu tergantung dari tempat dideklarasikannya variabel pada program yang sedang dibuat. Variabel global yaitu variabel yang dapat diakses di semua lingkup dalam program yang sedang dibuat, dalam kata lain variabel global ini dapat dikenali oleh semua fungsi dan prosedur, sementara variabel lokal yaitu variabel yang dapat diakses hanya di lingkup khusus, dalam kata lain variabel lokal ini hanya bisa diakses pada fungsi/prosedur dimana variabel itu dideklarasikan.

Berikut merupakan standar-standar 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. Nama variabel tidak boleh diawali dengan angka
4. Karakter bersifat case-sensitive (huruf besar dan huruf kecil dibedakan), contoh: Nama dan NAMA keduanya memiliki arti yang berbeda dan merupakan variabel yang berbeda.
5. Nama variabel tidak boleh menggunakan kata kunci yang ada pada bahasa pemrograman python, contoh: if, else, while

## 3.2 Input dan Output

Input & output bertujuan agar pengguna dan program dapat berinteraksi. 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.

Contoh dari penggunaan input dan output adalah sebagai berikut:

```

1 #Input yang ditujukan untuk user
2 nama= informatics research center
3
4 #output yang didapatkan user
5 print( Hallo , nama , selamat datang )

```

## 3.3 Operasi Aritmatika

Python memiliki operasi aritmatika, antara lainnya seperti :

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

Penggunaan dari simbol simbol ini sama hal nya dengan fungsi aritmatika pada umumnya.

## 3.4 Perulangan

Dalam membuat sebuah program, terkadang kita memerlukan satu baris atau satu blok kode yang sama secara berulang, disini fungsi perulangan dipakai sehingga kita tidak perlu menulis baris atau blok kode yang sama secara terus menerus, dalam python perulangan dibagi menjadi 2, yaitu for dan while.

### 3.4.1 For

For merupakan perulangan yang akan mengulang kondisi true sampai batas yang telah ditentukan, biasanya digunakan untuk perulangan yang mana parameter pengulangannya menggunakan list atau range. Berikut ini merupakan contoh penggunaan sintaks perulangan for.

```
1 for i in range (0 ,10):  
2     print( i )
```

### 3.4.2 While

While merupakan perulangan yang akan terjadi apabila kondisinya True, perulangan akan terus berjalan hingga diperoleh kondisi False. Berikut ini merupakan contoh penggunaan sintaks perulangan 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!")
```

## 3.5 Kondisi

Pengambilan keputusan kadang diperlukan dalam sebuah program untuk menentukan tindakan apa yang akan dilakukan sesuai dengan kondisi yang terjadi, contoh kasus misalkan ada seorang anak bernama idam, seorang manusia yang membutuhkan makan, jika idam lapar maka idam akan makan. Maka dapat dijabarkan seperti dibawah ini :

Kondisi, jika :

Idam lapar

Maka :

Idam akan makan

Namun terkadang kondisi juga diberikan tambahan opsi sebuah kondisi tambahan, misalkan jika idam makan maka idam kenyang, namun jika tidak maka idam akan kelaparan. Penjabarannya dapat dilihat sebagai berikut :

Kondisi, jika :

Idam makan

Maka :

Idam akan kenyang

Jika tidak :

Idam akan kelaparan

Contoh diatas dapat ditulis dalam sintax python dengan menggunakan kondisi, pengkondisian dalam python dibagi menjadi 4, yaitu : IF, IF ELSE, ELIF, nested IF. Berikut merupakan pembahasannya.

**3.5.0.1 IF** IF adalah suatu struktur yang memiliki suatu perlakuan jika terjadi suatu kondisi. Akan tetapi, tidak terjadi sesuatu yang lain atau terjadi apa-apa ketika berada di dalam luar kondisi tersebut. 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")

```

**3.5.0.2 IF ELSE** IF 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 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 else:
7     print("a is greater than b")

```

**3.5.0.3 ELIF** Kondisi ELIF merupakan suatu struktur logika majemuk yang memiliki banyak pilihan aksi terhadap berbagai kemungkinan kejadian yang terjadi. 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")

```

**3.5.0.4 Nested IF** 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.")
```

## 3.6 Error

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

Contoh:

```
1 nama = "Dinda Majesty"
2 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 error SyntaxError: invalid syntax. error ini dapat diatasi dengan memperhatikan tata cara penulisan kode pada bahasa pemrograman python.

3. Logic error merupakan kesalahan yang terjadi karena kesalahan pembacaan data pada command perintah seperti data tidak terbaca atau tidak ada, dan tidak sesuai dengan aturannya. Contoh kesalahan tipe data yaitu

```
1 a= 4
2 b=6
3
4 print(a+b)
```

4. TypeError, terjadi apabila kode melakukan operasi atau fungsi terhadap tipe data yang tidak sesuai. Contoh: melakukan penjumlahan terhadap tipe data string dan integer. error ini dapat diatasi dengan mengubah tipe data string menjadi integer.

```
1 a = "10"
2 b = 5
3
4 print(a + b)
```

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

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

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

Maka akan menghasilkan output eror IndentationError: expected an indented block

### 3.7 Try Except

Try Except merupakan salah satu bentuk penangan error di dalam bahasa pemrograman python, perintah try except ini memiliki fungsi untuk menangkap sebuah error dan tetap menjalankan program kita, sehingga program yang sedang dijalankan akan mengeksekusi program hingga akhir. Contohnya terdapat pada listing berikut

```
1 a="1"
2 b=2
3
4 try :
5     a+b
6 except:
7     print("Error , kedua tipe data berbeda")
```

## BAB 4

---

# FUNGSI DAN KELAS

---

### 4.1 Teori

#### 4.1.1 Fungsi

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

```
1 def nambahinAngka(angka1 , angka2):  
2     hasil = angka1 + angka2  
3     return hasil
```

Apabila kita dapat memberikan nilai ke angka1 dan angka2, dan apa bila sudah diberi nilai dan program sudah dijalankan, maka program pun akan mengembalikan nilai berupa hasil dari penjumlahan angka 1 dan angka 2.

## 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. Contohnya adalah sebagai berikut

```

1 def my biodata(nama, umur):
2     bio = "nama saya " + nama + " umur saya " + umur
3     return bio
4
5 def my study(kampus, prodi):
6     study = "saya berkuliah di " + kampus + " program studi " + prodi
7     return study

```

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

```

1 import fungsi
2 nama = "Dinda Majesty"
3 umur = "19 Tahun"
4 biodata = my biodata(nama, umur)
5 print(biodata)
6
7 kampus = "Politeknik Pos Indonesia"
8 prodi = "D4-Teknik Informatika"
9 kuliah = my study(kampus, prodi)
10 print(kuliah)

```

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

## 4.3 Class, Object, Atribute, and Method

Class atau Kelas merupakan sebuah blueprint/kerangka dari objek yang berisi fungsi dan dibuat untuk mendefenisikan objek dengan atribut yang sesuai dengan kelas yang telah dibuat yang nantinya akan diinisiasi. Objek adalah sebuah wujud yang dapat kita lakukan perintah sesuai dengan methodnya, 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, atribut ada 2 yaitu kelas atribut dan instansi atribut, perbedaannya hanya di letak, kalau kelas atribut ada di bawah kelas, dan instansi atribut ada didalam fungsi, atribut itu sebuah variabel yang dimiliki oleh parentnya seperti fungsi atau class. .Method merupakan kode program yang berisi tindakan atau perintah untuk menjalankan objek.

```
1 class Fungsi(object):  
2  
3     def Nama(self , namakamu):  
4         self.kamu = namakamu
```

## 4.4 Pemanggilan Class

Pemanggilan library kelas dapat dilakukan dengan cara import dan membuat objek dari kelas tersebut. Contohnya, kita memiliki le 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.

```
1 import Fungsi
```

## 4.5 Pemakaian Package Fungsi Apabila File Didalam Folder

Pemakaian Package fungsi apabila le terdapat didalam sebuah folder maka kita bisa menggunakan from folder import le dan from le import fungsi. Contohnya, kita memiliki folder src yang didalamnya terdapat le fungsi.py dan didalam fungsi.py terdapat fungsi Berhitung, untuk mengimportkan fungsi maka kita dapat mengetikkan kode seperti berikut.

```
1 from src import fungsi  
2 from fungsi import Berhitung
```

## 4.6 Pemakaian Package Kelas Apabila File didalam Folder

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

```
1 from src import fungsi  
2 Kelas = fungsi.Nama(namakamu)
```



## DAFTAR PUSTAKA

---

1. R. Awangga, “Sampeu: Servicing web map tile service over web map service to increase computation performance,” in *IOP Conference Series: Earth and Environmental Science*, vol. 145, no. 1. IOP Publishing, 2018, p. 012057.

