

COMPUTATIONAL DATA PREPROCESSING

COMPUTATIONAL DATA PREPROCESSING

with Python3

Tri Angga Dio S., Rolly M. Awangga
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, 2021

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

1	Python	1
2	Git & Github	71
3	Algoritma	95
4	Hash Map	101
5	Hash Tables	109
6	Kompleksitas	113
7	Big O Notation	123

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>	
1 Python	1
1.1 Mengenal Python	1
1.2 Kelebihan Python	2
1.3 Kekurangan Python	3
1.4 Instalasi Python	4

1.4.1	Linux	4
1.4.2	Windows	10
1.5	Menjalankan Python	19
1.5.1	Linux	19
1.5.2	Windows	24
1.6	Hello World	25
1.6.1	Syntax Dasar	26
1.6.2	Python Case Sensitivity	26
1.7	Komentar Python	26
1.8	Tipe Data Python	27
1.9	Variabel Python	28
1.10	Operator Python	30
1.10.1	Operator Aritmatika	30
1.10.2	Operator Perbandingan	32
1.10.3	Operator Penugasan	32
1.10.4	Prioritas Eksekusi Operator di Python	33
1.11	Kondisi Python	34
1.11.1	Kondisi If	34
1.11.2	Kondisi If Else	35
1.11.3	Kondisi Elif	35
1.12	Loop Python	36
1.12.1	While Loop	36
1.12.2	For Loop	37
1.12.3	Nested Loop	37
1.13	Number Python	37
1.13.1	Konversi Tipe Data Number Python	38
1.13.2	Fungsi Matematika Python	38
1.13.3	Fungsi Nomor Acak Python	39
1.13.4	Fungsi Matematika Python	40
1.13.5	Fungsi Trigonometri Python	40
1.13.6	Konstanta Matematika Python	41
1.14	String Python	41
1.14.1	Mengakses Nilai dalam String	41
1.14.2	Mengupdate String	42
1.14.3	Karakter Escape Python	42
1.14.4	Operator Spesial String Python	43
1.14.5	Operator Format String Python	44
1.14.6	Triple Quote Python	44

1.14.7	String Unicode Python	45
1.15	List Python	48
1.15.1	Membuat List Python	48
1.15.2	Akses Nilai Dalam List Python	49
1.15.3	Update Nilai Dalam List Python	49
1.15.4	Hapus Nilai Dalam List Python	49
1.15.5	Operasi Dasar Pada List Python	49
1.15.6	Indexing, Slicing dan Matrix Pada List Python	50
1.15.7	Method dan Fungsi Build-in Pada List Python	50
1.16	Tuple Python	51
1.16.1	Akses Nilai Dalam Tuple Python	52
1.16.2	Update Nilai Dalam Tuple Python	52
1.16.3	Hapus Nilai Dalam Tuple Python	52
1.16.4	Operasi Dasar Pada Tuple Python	53
1.16.5	Indexing, Slicing dan Matrix Pada Tuple Python	53
1.16.6	Fungsi Build-in Pada Tuple Python	53
1.17	Dictionary Python	54
1.17.1	Akses Nilai Dalam Dictionary Python	54
1.17.2	Update Nilai Dalam Dictionary Python	54
1.17.3	Hapus Elemen Dictionary Python	55
1.17.4	Fungsi Build-in Pada Dictionary Python	55
1.17.5	Method Build-in Pada Dictionary Python	55
1.18	Tanggal & Waktu Python	56
1.18.1	Apa itu Tick?	56
1.18.2	Apa itu TimeTuple Python?	57
1.18.3	Mendapatkan Waktu Saat Ini	58
1.18.4	Mendapatkan Waktu yang berformat	58
1.18.5	Mendapatkan kalender dalam sebulan	58
1.18.6	Modul time pada Python	58
1.18.7	Modul calendar pada Python	60
1.19	Fungsi Python	62
1.19.1	Mendefinisikan Fungsi Python	63
1.20	Modul Python	63
1.20.1	Import Statement	63
1.21	File I/O Python	64
1.21.1	Print	64
1.21.2	Membaca Input Keyboard	64
1.21.3	Fungsi Input Python	64

1.22	Exception	65
1.23	Object & Class Python	67
1.23.1	Istilah Dalam OOP	68
1.23.2	Membuat Class Python	68
1.23.3	Membuat Instance Objects	69
1.23.4	Mengakses Atribut	69
2	Git & Github	71
2.1	Version Control System (VCS)	73
2.2	Git	74
2.3	Github	75
2.4	Perbedaan Git & Github	76
2.5	Tutorial Membuat Akun Github	76
2.6	Tutorial Install Git	83
2.6.1	Windows 10	83
2.6.2	Linux	86
2.7	Generate SSH Key	87
2.8	Memasang SSH Key di Github	90
3	Algoritma	95
3.1	Algoritma	96
3.2	Cases	98
3.2.1	Worst Case	98
3.2.2	Average Case	98
3.2.3	Best Case	98
4	Hash Map	101
4.1	Hash Map	101
4.2	Tabel Hash vs Hashmap: Perbedaan antara Tabel Hash dan Hashmap dengan Python	104
4.3	Membuat Dictionary	105
4.4	Membuat Nested Dictionary	105
4.5	Melakukan Operasi pada Hash Table menggunakan Dictionary	106
4.6	Mengubah Dictionary menjadi Kerangka Data	107
5	Hash Tables	109
5.1	Hash Table	110
5.1.1	Mengakses Nilai dari Dictionary	111

5.1.2	Update Dictionary	111
5.1.3	Delete Dictionary	112
6	Kompleksitas	113
6.1	Kompleksitas	113
6.2	Asymptotic Notation	114
6.3	Big O Notation	114
6.3.1	Cara Membaca dan Memahami Big O	116
6.3.2	Kegunaan Pemahaman Big O	117
6.3.3	Contoh Big O Notasi dengan Python3	117
6.3.4	$O(n^2)$	118
6.3.5	Ingin merasakan perbedaan dari $O(n)$ dan $O(n^2)$????	118
7	Big O Notation	123

DAFTAR GAMBAR

1.1	Instalasi Anaconda Linux: Step 1	4
1.2	Instalasi Anaconda Linux: Step 2	5
1.3	Instalasi Anaconda Linux: Step 3	5
1.4	Instalasi Anaconda Linux: Step 4	6
1.5	Instalasi Anaconda Linux: Step 5	6
1.6	Instalasi Anaconda Linux: Step 6	7
1.7	Instalasi Anaconda Linux: Step 7	7
1.8	Instalasi Anaconda Linux: Step 8	8
1.9	Instalasi Anaconda Linux: Step 9	8
1.10	Instalasi Anaconda Linux: Step 10	9
1.11	Instalasi Anaconda Linux: Step 11	10
1.12	Instalasi Python Windows: Step 1	11
1.13	Instalasi Python Windows: Step 2	12

1.14	Instalasi Python Windows: Step 3	13
1.15	Instalasi Anaconda Windows: Step 7	17
1.16	Instalasi Anaconda Windows: Step 8	18
1.17	Instalasi Anaconda Windows: Step 9	18
1.18	Menjalankan Python Linux: Step 1	19
1.19	Menjalankan Python Linux: Step 2	20
1.20	Menjalankan Python Linux: Step 3	20
1.21	Menjalankan Python Linux: Step 4	21
1.22	Menjalankan Python Linux dengan Teks Editor: Step 1	21
1.23	Menjalankan Python Linux dengan Teks Editor: Step 2	22
1.24	Menjalankan Python Linux dengan Teks Editor: Step 3	22
1.25	Menjalankan Python Linux dengan Teks Editor: Step 4	23
1.26	Menjalankan Python Linux dengan Teks Editor: Step 5	23
1.27	Menjalankan Python Linux dengan Teks Editor: Step 6	24
1.28	Menjalankan Python Linux dengan Teks Editor: Step 7	24
1.29	Menjalankan Python Windows	25
2.1	Git dan Github	72
2.2	Version Control System	73
2.3	Sebelum VCS	73
2.4	Sesudah VCS	74
2.5	Git	74
2.6	Github	75
2.7	Tutorial Akun Github: Step 1	77
2.8	Tutorial Akun Github: Step 2	77
2.9	Tutorial Akun Github: Step 3	78
2.10	Tutorial Akun Github: Step 4	78
2.11	Tutorial Akun Github: Step 5	79
2.12	Tutorial Akun Github: Step 6	80

2.13	Tutorial Akun Github: Step 7	81
2.14	Tutorial Akun Github: Step 8	81
2.15	Tutorial Akun Github: Step 9	82
2.16	Tutorial Akun Github: Step 10	82
2.17	Tutorial Install Git-scm: Step 1	83
2.18	Tutorial Install Git-scm: Step 2	83
2.19	Tutorial Install Git-scm: Step 3	84
2.20	Tutorial Install Git-scm: Step 4	84
2.21	Tutorial Install Git-scm: Step 5	85
2.22	Tutorial Install Git-scm: Step 6	85
2.23	Tutorial Install Git-scm: Step 7	86
2.24	Tutorial Install Git Linux: Step 1	86
2.25	Tutorial Install Git Linux: Step 2	87
2.26	Tutorial Generate SSH Key: Step 1	88
2.27	Tutorial Generate SSH Key: Step 2	88
2.28	Tutorial Generate SSH Key: Step 3	89
2.29	Tutorial Generate SSH Key: Step 4	89
2.30	Tutorial Generate SSH Key: Step 5	90
2.31	Tutorial Memasang SSH Key: Step 1	91
2.32	Tutorial Memasang SSH Key: Step 2	91
2.33	Tutorial Memasang SSH Key: Step 3	92
2.34	Tutorial Memasang SSH Key: Step 4	92
2.35	Tutorial Memasang SSH Key: Step 5	93
2.36	Tutorial Memasang SSH Key: Step 6	93
2.37	Tutorial Memasang SSH Key: Step 7	94
3.1	Algoritma	96
4.1	Hash Map	101
4.2	<i>Output Dictionary</i>	108

5.1	Hash Tables	110
6.1	Kompleksitas	113
6.2	Big O Notation	114
6.3	Cheat Sheet Big O Notation	115
6.4	Cheat Sheet Big O Notation Sorting	116
6.5	Mencoba Big O Notation: Step 1	119
6.6	Mencoba Big O Notation: Step 2	119
6.7	Mencoba Big O Notation: Step 3	120
6.8	Mencoba Big O Notation: Step 4	120
6.9	Mencoba Big O Notation: Step 5	120
6.10	Mencoba Big O Notation: Step 6	121
6.11	Mencoba Big O Notation: Step 7	121
6.12	Mencoba Big O Notation: Step 8	122
6.13	Mencoba Big O Notation: Step 9	122
7.1	<i>Constant Time</i>	125
7.2	<i>Logarithmic Time</i>	126
7.3	<i>Linear Time</i>	126
7.4	<i>Quadratic Time</i>	127
7.5	<i>Exponential Time</i>	128

DAFTAR TABEL

Listings

1.1	Python Comment	26
1.2	Python Tipe Data	28
1.3	Python Variabel	29
1.4	Python Operator Aritmatika	30
1.5	Python Kondisi If	34
1.6	Python Kondisi If Else	35
1.7	Python Kondisi Elif	35
1.8	Python While Loop	36
1.9	Python For Loop	37
1.10	Python Nested Loop	37
4.1	Implementasi Hash Map	102
4.2	Dict dengan Kurung Kurawal	105
4.3	Dict dengan Kurung Kurawal	105
4.4	Dict dengan Kurung Kurawal	105
4.5	Nested Dict	105
4.6	Dict dengan nilai key	106
4.7	Dict with function	106
4.8	Dict For Loop	106

4.9	Dict For Loop	107
4.10	Dict For Loop	107
4.11	Mengubah Dictionary menjadi Kerangka Data	107
5.1	Implementasi Hash Table	111
5.2	Hasil Implementasi Hash Table	111
5.3	Implementasi Update Hash Table	111
5.4	Hasil Implementasi Update Hash Table	112
5.5	Implementasi Delete Hash Table	112
5.6	Hasil Implementasi Delete Hash Table	112

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

PYTHON

1.1 Mengenal Python

Python adalah bahasa pemrograman interpretatif multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal ini membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain. Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distrionya sudah menyertakan Python di dalamnya. Dengan kode yang simpel dan mudah diimplementasikan, seorang programmer dapat lebih mengutamakan pengembangan aplikasi yang dibuat, bukan malah sibuk mencari syntax error.

Contoh syntax python sederhana:

```
1 print("aku suka python")
```

Hanya dengan menuliskan kode print seperti yang diatas, anda sudah bisa mencetak apapun yang anda inginkan di dalam tanda kurung (). Dibagian akhir kode pun, anda tidak harus mengakhirnya dengan tanda semicolon ;.

1.2 Kelebihan Python

Berdasarkan survey yang dilakukan oleh situs stackoverflow, Bahasa pemrograman python merupakan bahasa yang menduduki urutan pertama sebagai bahasa yang paling banyak dibahas di dunia. Karena memiliki kelebihan atau keunggulan yang membuat bahasa satu ini digemari banyak orang diseluruh dunia. Keunggulan tersebut antara lain :

1. Memiliki library yang luas dan banyak; Bahasa pemrograman python memiliki banyak library yang siap anda gunakan yang berisi berbagai modul. Didalamnya terdapat berbagai macam kode untuk digunakan seperti regulas expressions, documentation-generation, unit testing, database, CGI, email, dan masih banyak lagi. Sehingga dengan adanya library ini, anda tidak perlu menulis lagi secara manual
2. Open Source atau gratis; Python merupakan bahasa pemrograman open source atau dapat anda unduh secara gratis. Bahasa satu ini dikembangkan dibawah lisensi open source yang disetujui oleh OSI dimana bahasa pemrograman ini bebas digunakan, dikembangkan, dan di distribusikan, bahkan termasuk tujuan komersial.
3. Mampu meningkatkan produktifitas developer; Sebagai bahasa yang mudah digunakan dan dipelajari membuat developer menjadi lebih produktif, apa lagi disertai dengan library yang luas. Selain itu dengan bahasa python, anda hanya perlu menulis kode lebih sedikit sehingga anda mempunyai banyak waktu untuk bisa mengerjakan yang lain.
4. Mendukung Internet Of Things (IoT) yang baik; Manfaat yang bisa anda dapatkan ketika menggunakan bahasa pemrograman python salah satunya yaitu mampu mendukung ekosistem Internet Of Things (IoT) dengan sangat baik, dimana python mampu menghubungkan benda-benda disekitar lingkungan kita kedalam sebuah internet yang menghubungkan satu sama lain.
5. Bahasa yang mudah dipelajari dan mudah digunakan; Python merupakan bahasa yang mudah dipelajari bahkan untuk pengembang pemula. Kode bahasa python ini mudah dibaca dan bisa menjalankan banyak fungsi kompleks dengan mudah, karena banyaknya library. Selain itu, proses pengembangan bahasa python bisa dilakukan dengan cepat menggunakan kode yang lebih sedikit. Bahkan tim kecil pun bisa menangani bahasa python secara efektif. Python merupakan bahasa yang sangat dinamis dimana dibangun berdasarkan tingkat keterbacaan kode yang tinggi. Sehingga kode tersebut mudah dipahami dan

dicerna. Berbagai macam jenis library memuat banyak perlengkapan dan fung-sionalitas yang sangat luar biasa, sehingga kemudahan dalam membangun pro-gram menjadi salah satu yang ditawarkan oleh bahasa pemrograman satu ini.

6. Fleksibel; Kode program yang ditulis menggunakan bahasa python dapat di-jalankan di hampir semua sistem operasi seperti Windows, Mac, maupun Linux, termasuk beberapa perangkat-perangkat seluler. Kode python dapat di inte-grasikan dengan aplikasi yang ditulis dalam bahasa pemrograman lain dengan mekanisme tertentu. Misal, kode python dapat dipanggil dari kode C/C++, dan sama halnya dengan perkembangan .NET Framework.

1.3 Kekurangan Python

Selain kelebihan diatas, bahasa pemrograman python juga memiliki kekurangan. Seperti halnya bahasa yang lain, tidak ada bahasa pemrograman yang sempurna. Berikut ulasan lengkap mengenai apa saja kelemahan bahasa pemrograman python.

1. Tidak cocok untuk aplikasi mobile; Python merupakan bahasa pemrograman yang sangat baik digunakan untuk platform dekstop dan server namun tidak dalam urusan komputasi aplikasi mobile. Pengembangan aplikasi dan game ku-rang cocok jika menggunakan python. Bahkan banyak yang mengatakan bahwa mustahil membuat game dalam bentuk tiga dimensi dengan grafis tinggi meng-gunakan python.
2. Eksekusi relatif lambat; Python merupakan bahasa interpreter yang bekerja den-gan menggunakan kompiler. Dimana ketika dijalankan, python akan bekerja lebih lambat jika dibandingkan dengan bahasa lain. Tetapi hal ini juga bergant-tung dari besar dan kecilnya program yang akan kita buat. Python tidak baik jika diperuntukkan dalam pekerjaan multi-processor atau multi-core. Python dinilai memiliki performa yang relatif lambat jika dibandingkan dengan ba-hasa pemrograman lainnya. Bahkan beberapa penugasan terdapat diluar dari jangkauan python, serupa bahasa pemrograman dinamis lainnya, python tidak secepat atau efisien sebagai statis, tidak seperti bahasa pemrograman compilasi serupa bahasa C.
3. Kesulitan jika beralih ke bahasa pemrograman lain; Python memiliki banyak library yang luas sehingga pengguna python terbiasa dengan fitur yang ada pada library tersebut. Hal ini tentu saja dapat menimbulkan masalah seperti pengguna mengalami kesulitan ketika belajar maupun beralih ke bahasa pemrograman yang lain. Karena kebiasaan menggunakan library tersebut membuat pengguna tidak mengetahui bagaimana asal pembuatan source code library tersebut.
4. Kesalahan saat runtime; Python merupakan bahasa pemrograman yang dinamis, dimana anda tidak perlu mendeklarasikan tipe variabel saat menulis kode. Meskipun ini memudahkan developer selama pengkodean, namun pastinya dapat meningkatkan terjadinya kesalahan saat run-time.

5. Efisiensi dan fleksibilitas tradeoff by; Python memberikan fleksibilitas tradeoff by dengan tidak diberikan secara meluas. Python menyiapkan bahasa pemrograman optimasi untuk kegunaan bersama dengan perangkat bantu yang dibutuhkan untuk di integrasikan dengan bahasa pemrograman lain.

1.4 Instalasi Python

Sebelum Anda menggunakan Python, Anda harus menginstalnya terlebih dahulu di sistem operasi komputer Anda. Saat ini Python memiliki 2 versi yang berbeda, yaitu Python versi 3 dan Python versi 2. Disini kita akan belajar bahasa pemrograman Python menggunakan versi terbaru 3. Cara menginstal python sangat mudah, ikuti panduan dibawah ini. Dibawah adalah panduan cara instal python di platform Linux, dan Windows

1.4.1 Linux

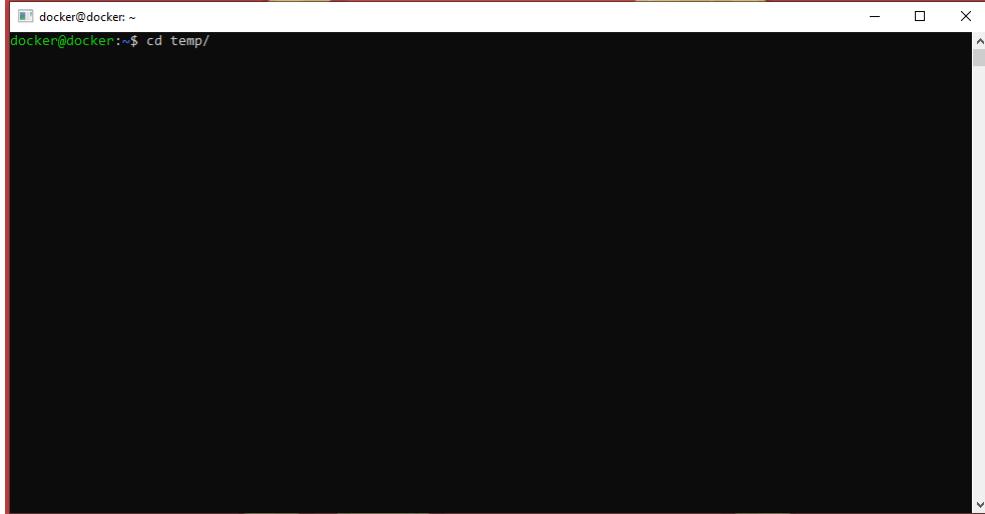
1.4.1.1 Anaconda

1. Buka browser, kunjungi <https://www.anaconda.com/products/individual>
2. lalu scroll sampai muncul menu seperti gambar dibawah, klik kanan pada Installer dan klik **Copy link address**



Gambar 1.1 Instalasi Anaconda Linux: Step 1

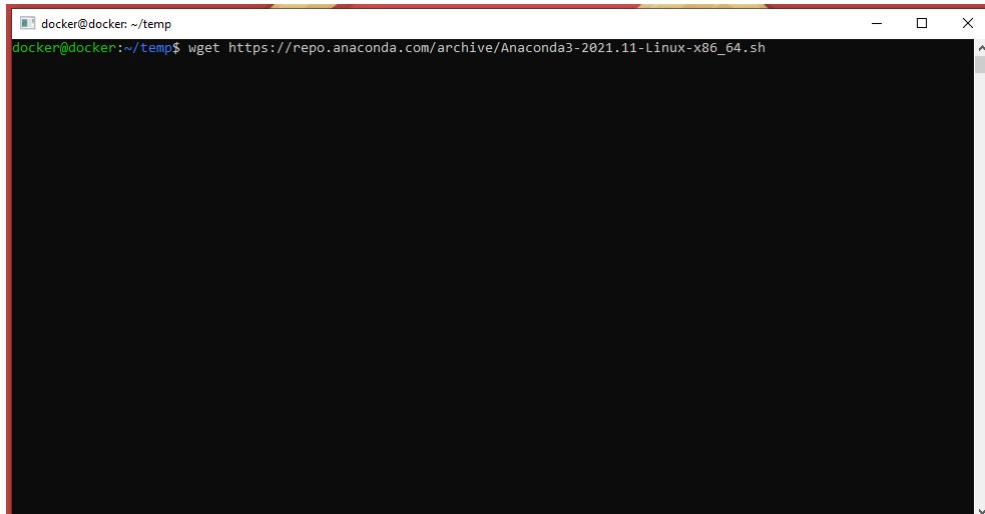
3. buka terminal lalu ketikkan **cd temp/**, tekan enter



```
docker@docker:~$ cd temp/
```

Gambar 1.2 Instalasi Anaconda Linux: Step 2

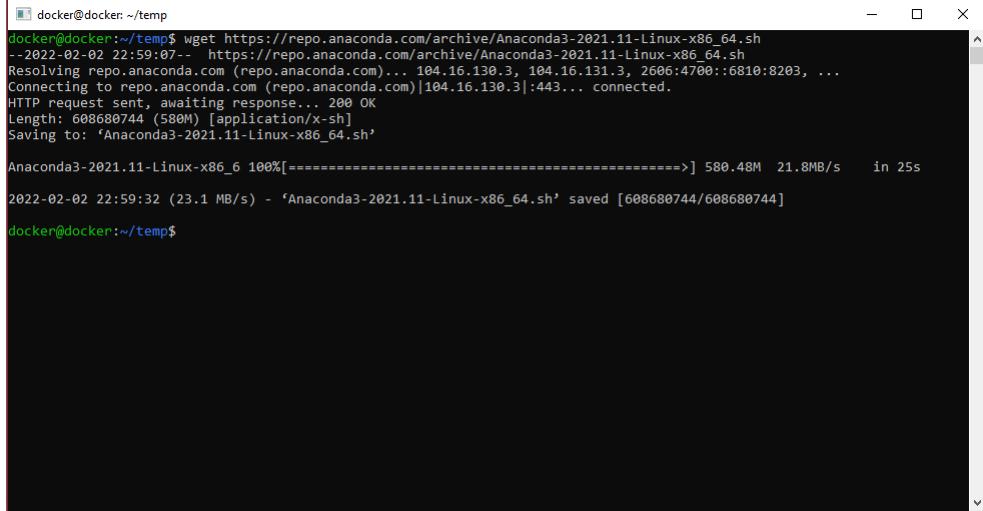
4. setelah itu ketikkan **wget link yang sudah dicopy**



```
docker@docker:~/temp$ wget https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-x86_64.sh
```

Gambar 1.3 Instalasi Anaconda Linux: Step 3

5. tekan enter, dan tunggu hingga download selesai



```

docker@docker:~/temp$ wget https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-x86_64.sh
--2022-02-02 22:59:07-- https://repo.anaconda.com/archive/Anaconda3-2021.11-Linux-x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3, 104.16.131.3, 2606:4700::6810:8203, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 608680744 (580M) [application/x-sh]
Saving to: 'Anaconda3-2021.11-Linux-x86_64.sh'

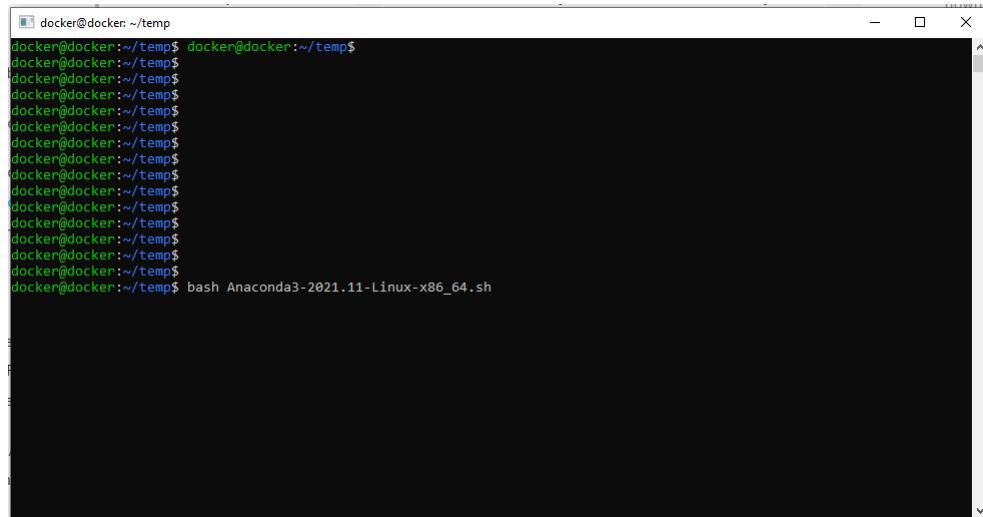
Anaconda3-2021.11-Linux-x86_6 100%[=====] 580.48M 21.8MB/s   in 25s

2022-02-02 22:59:32 (23.1 MB/s) - ‘Anaconda3-2021.11-Linux-x86_64.sh’ saved [608680744/608680744]

docker@docker:~/temp$
```

Gambar 1.4 Instalasi Anaconda Linux: Step 4

6. jika sudah ketikkan **bash nama file anaconda, sedikit tips, untuk lebih cepat ketikkan Anac lalu tekan Tab maka secara otomatis akan mengisi nama filenya**



```

docker@docker:~/temp$ bash Anaconda3-2021.11-Linux-x86_64.sh
```

Gambar 1.5 Instalasi Anaconda Linux: Step 5

7. lalu tekan Enter

```
docker@docker:~/temp$ docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$  
docker@docker:~/temp$ bash Anaconda3-2021.11-Linux-x86_64.sh  
Welcome to Anaconda3 2021.11  
In order to continue the installation process, please review the license  
agreement.  
Please, press ENTER to continue  
>>>
```

Gambar 1.6 Instalasi Anaconda Linux: Step 6

8. jika muncul gambar dibawah ini

```
docker@docker:~/temp$  
=====  
End User License Agreement - Anaconda Individual Edition  
=====  
Copyright 2015-2021, Anaconda, Inc.  
All rights reserved under the 3-clause BSD License:  
This End User License Agreement (the "Agreement") is a legal agreement between you and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Individual Edition (which was formerly known as Anaconda Distribution).  
Subject to the terms of this Agreement, Anaconda hereby grants you a non-exclusive, non-transferable license to:  
 * Install and use the Anaconda Individual Edition (which was formerly known as Anaconda Distribution),  
 * Modify and create derivative works of sample source code delivered in Anaconda Individual Edition from Anaconda's repository; and  
 * Redistribute code files in source (if provided to you by Anaconda as source) and binary forms, with or without modification subject to the requirements set forth below.  
Anaconda may, at its option, make available patches, workarounds or other updates to Anaconda Individual Edition. Unless the updates are provided with their separate governing terms, they are deemed part of Anaconda Individual Edition licensed to you as provided in this Agreement. This Agreement does not entitle you to any support for Anaconda Individual Edition.  
Anaconda reserves all rights not expressly granted to you in this Agreement.  
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:  
--More--
```

Gambar 1.7 Instalasi Anaconda Linux: Step 7

9. tekan dan tahan Enter sampai muncul gambar dibawah ini, lalu ketik yes, tekan enter

```
docker@docker: ~/temp
The following packages listed on https://www.anaconda.com/cryptography are included in the repository accessible through
Anaconda Individual Edition that relate to cryptography.

Last updated April 5, 2021

Do you accept the license terms? [yes/no]
[no] >>>
Please answer 'yes' or 'no':'
>>>
```

Gambar 1.8 Instalasi Anaconda Linux: Step 8

10. jika sudah seperti gambar dibawah, tekan enter

```
>>>
Please answer 'yes' or 'no':'
>>> yes

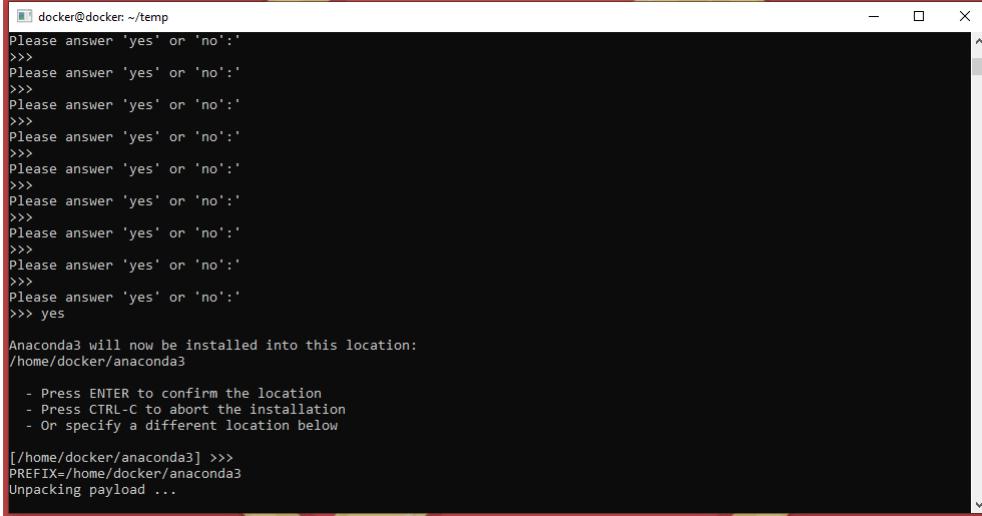
Anaconda3 will now be installed into this location:
/home/docker/anaconda3

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

[/home/docker/anaconda3] >>>
```

Gambar 1.9 Instalasi Anaconda Linux: Step 9

11. tunggu hingga proses selesai



The screenshot shows a terminal window with a black background and white text. At the top, it says "docker@docker: ~/temp". The window contains several lines of text from a script or log file. It asks for user input ("Please answer 'yes' or 'no':") multiple times, with the last response being "yes". It then states that Anaconda3 will be installed into the location "/home/docker/anaconda3". It provides instructions for confirmation: "Press ENTER to confirm the location", "Press CTRL-C to abort the installation", or "Or specify a different location below". Finally, it shows the command line with "[~/home/docker/anaconda3] >>>" followed by "PREFIX=/home/docker/anaconda3" and "Unpacking payload ...".

```
[docker@docker:~/temp]
Please answer 'yes' or 'no':'
>>>
Please answer 'yes' or 'no':'
>>> yes

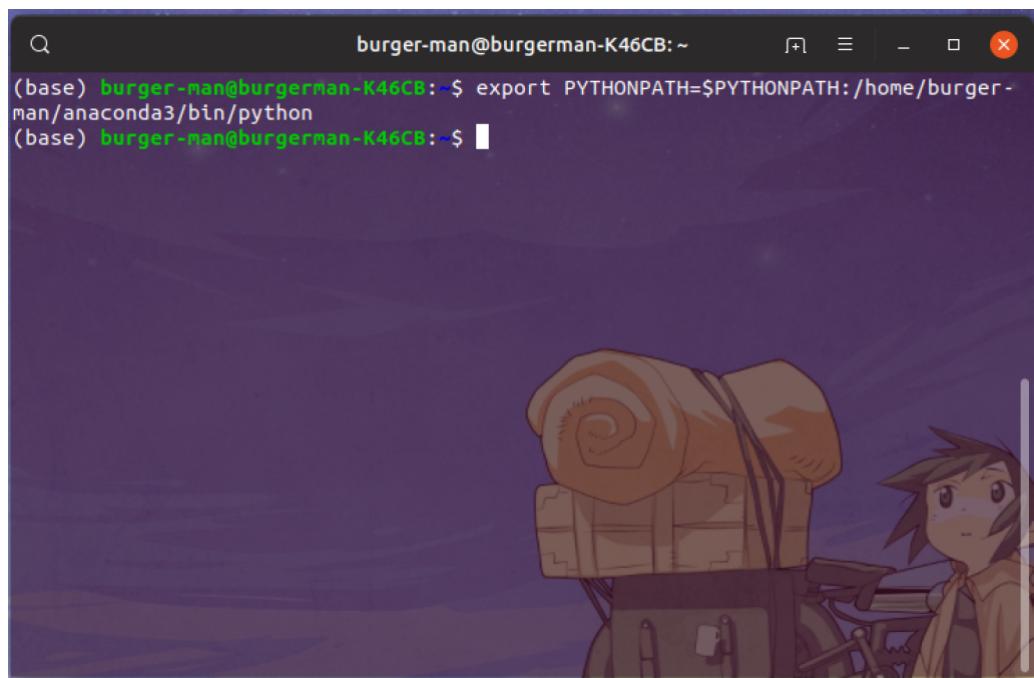
Anaconda3 will now be installed into this location:
/home/docker/anaconda3

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

[/home/docker/anaconda3] >>>
PREFIX=/home/docker/anaconda3
Unpacking payload ...
```

Gambar 1.10 Instalasi Anaconda Linux: Step 10

12. jika sudah selesai instalasi, ketikkan perintah berikut **export PYTHONPATH=\$PYTHONPATH path anaconda** seperti dibawah ini, lalu tekan enter



Gambar 1.11 Instalasi Anaconda Linux: Step 11

1.4.2 Windows

Untuk instalasi python terdapat 2 cara yaitu menggunakan Python langsung atau menggunakan Anaconda

1.4.2.1 *Python*

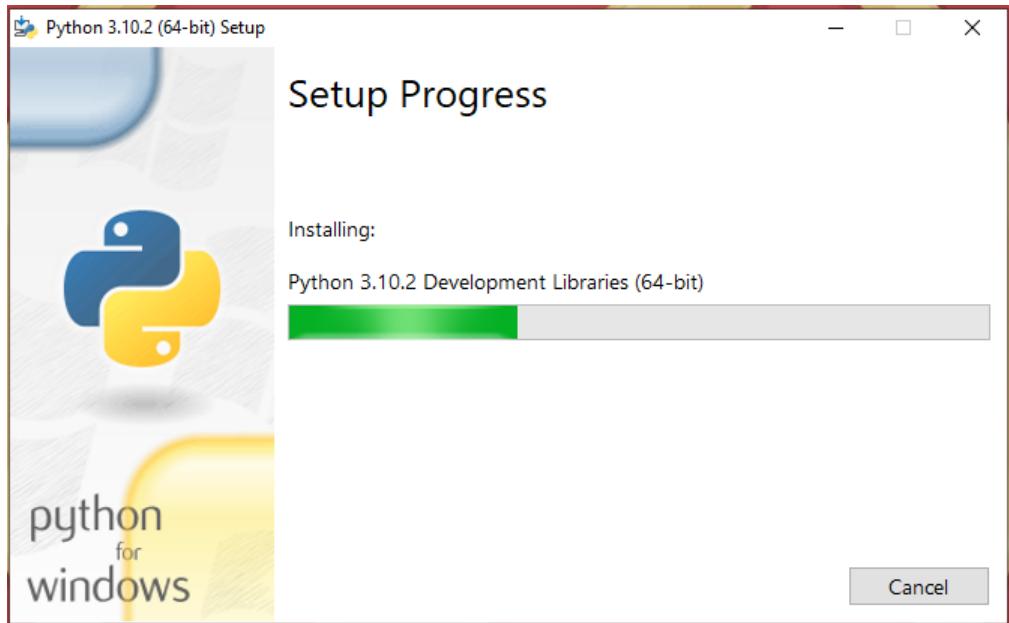
1. Buka browser, kunjungi <http://www.python.org/downloads/windows/>
2. Buka (klik 2x) file installer python yang baru saja di download, lalu akan muncul seperti gambar dibawah ini.



Gambar 1.12 Instalasi Python Windows: Step 1

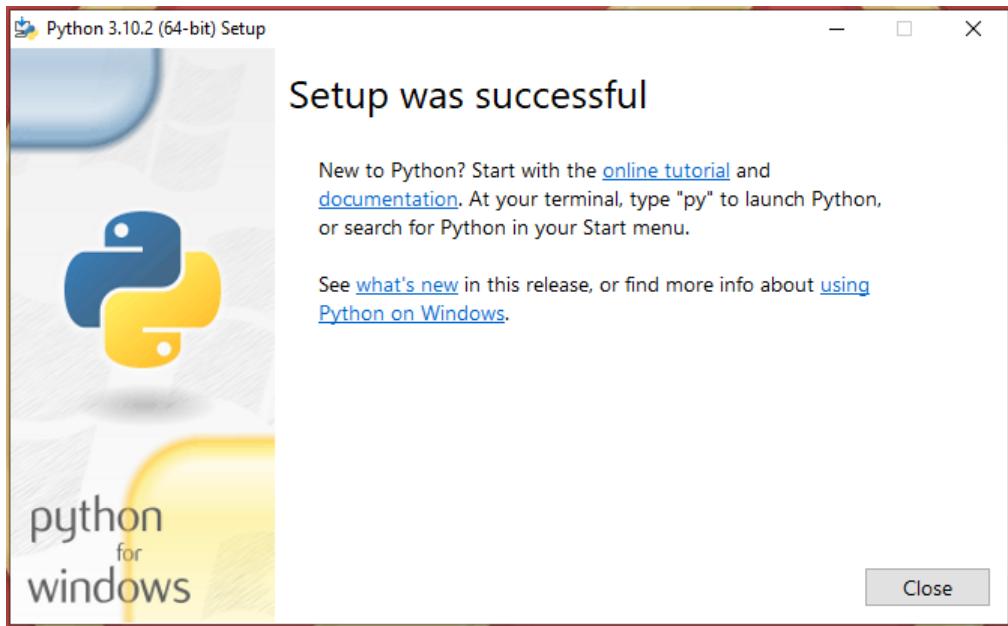
3. Pastikan sudah seperti gambar diatas, lalu klik tombol **Install Now**

4. Lalu akan muncul proses instalasi, tunggu sampai selesai.



Gambar 1.13 Instalasi Python Windows: Step 2

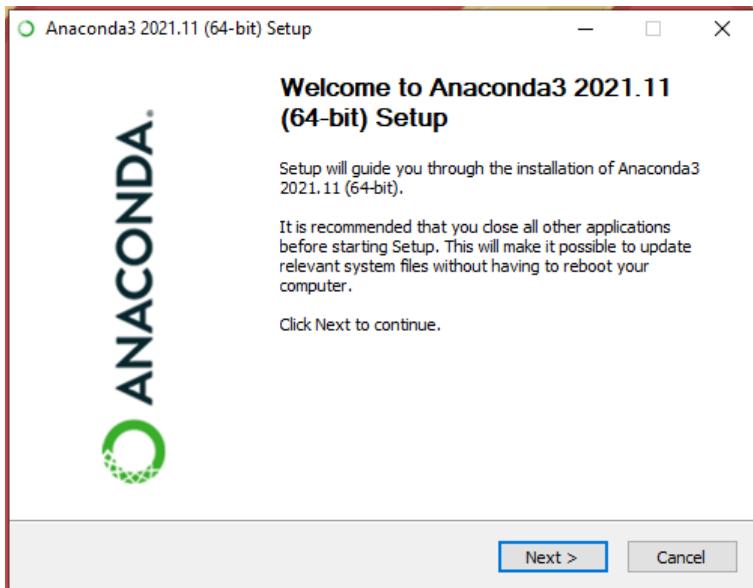
5. Jika sudah selesai akan muncul seperti gambar dibawah ini.



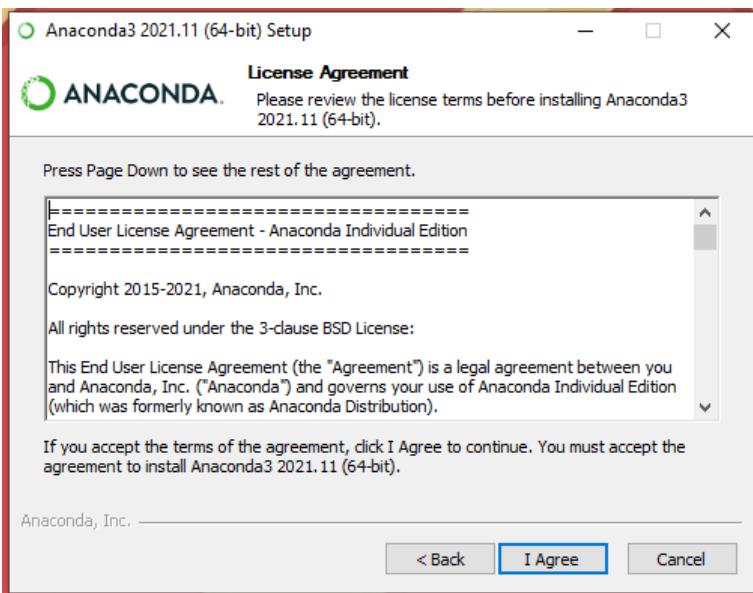
Gambar 1.14 Instalasi Python Windows: Step 3

1.4.2.2 *Anaconda*

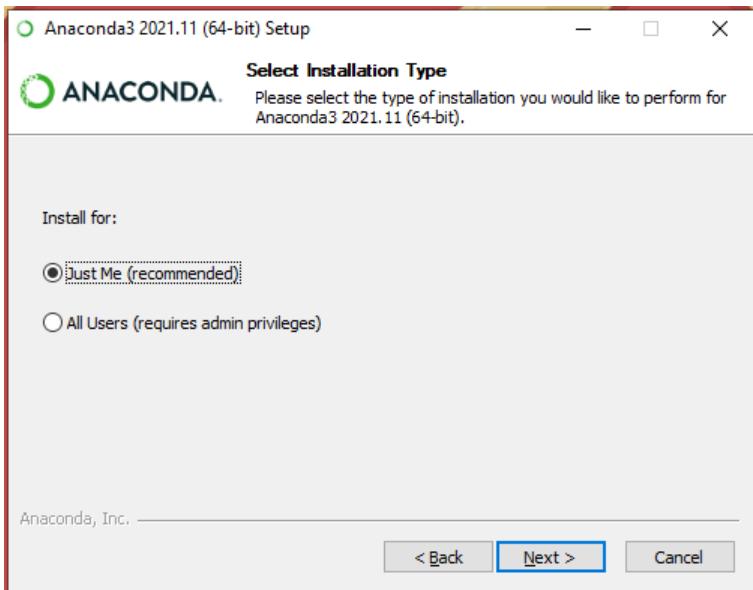
1. Buka browser, kunjungi <https://www.anaconda.com/products/individual> lalu download file Installer Anaconda sesuai Bit komputer/laptop anda.
2. Jika sudah download, klik (2x) filenya dan akan muncul seperti gambar dibawah ini.



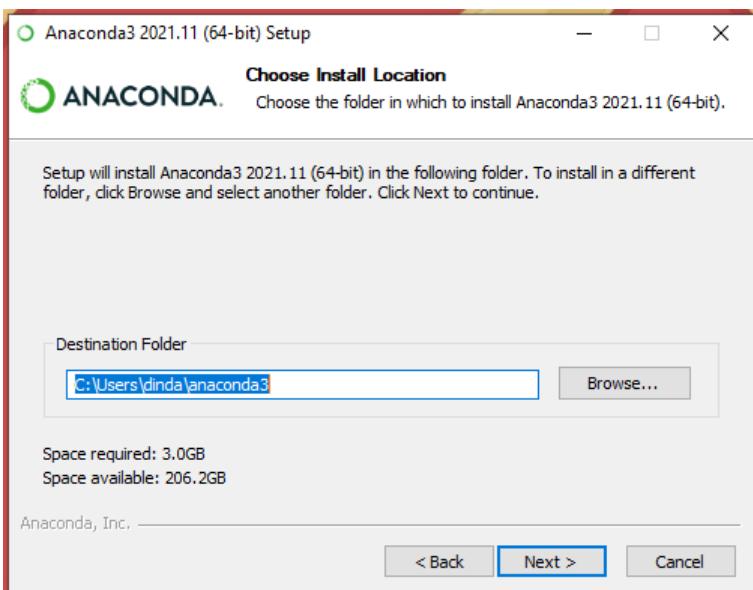
3. Klik next setelah itu akan muncul seperti gambar dibawah ini.



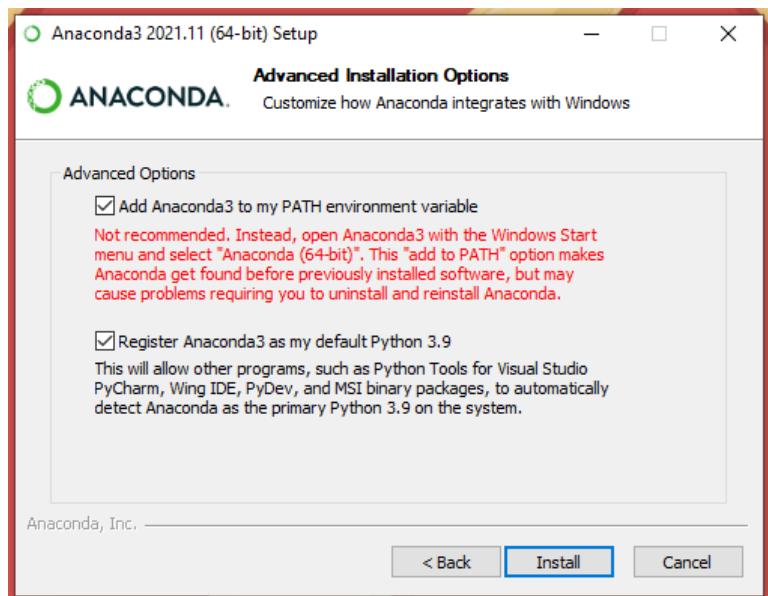
4. klik I Agree, lalu akan muncul gambar dibawah ini.



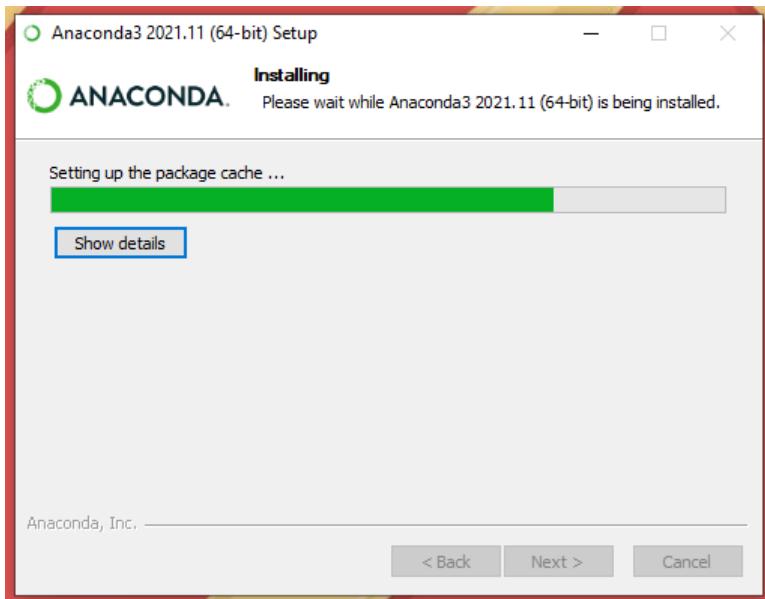
5. lalu ada pilihan Just Me atau All Users, disini boleh bebas pilih yang mana saja, disini saya memilih Just Me karena direkomendasikan oleh Anacondanya, jika sudah memilih langsung klik next



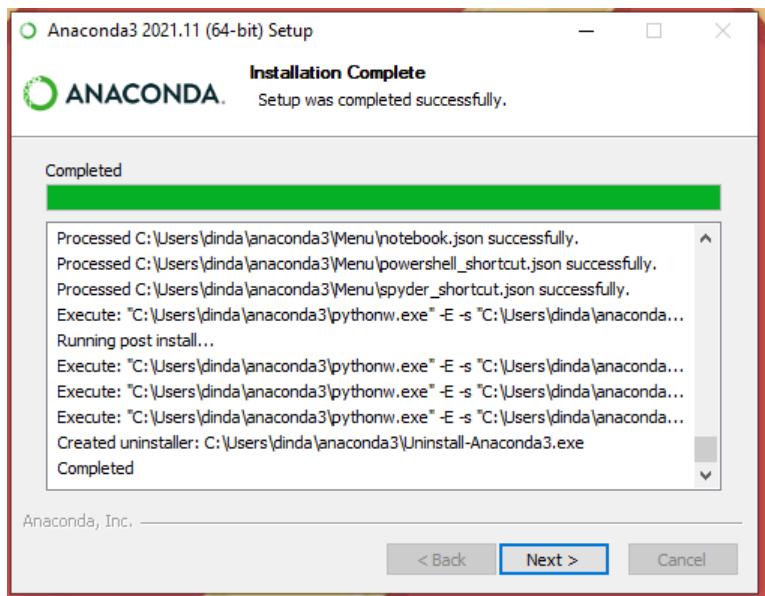
- setelah itu muncul destination folder adalah lokasi dimana ekstraksi file Anaconda akan diletakkan dimana, disini saya membiarkan seperti yang sudah diset default oleh Anaconda, klik next



- setelah itu muncuk Menu Advanced Options, disini saya mencentang option Add Anaconda3 to my PATH environtment variabel, setelah itu klik install

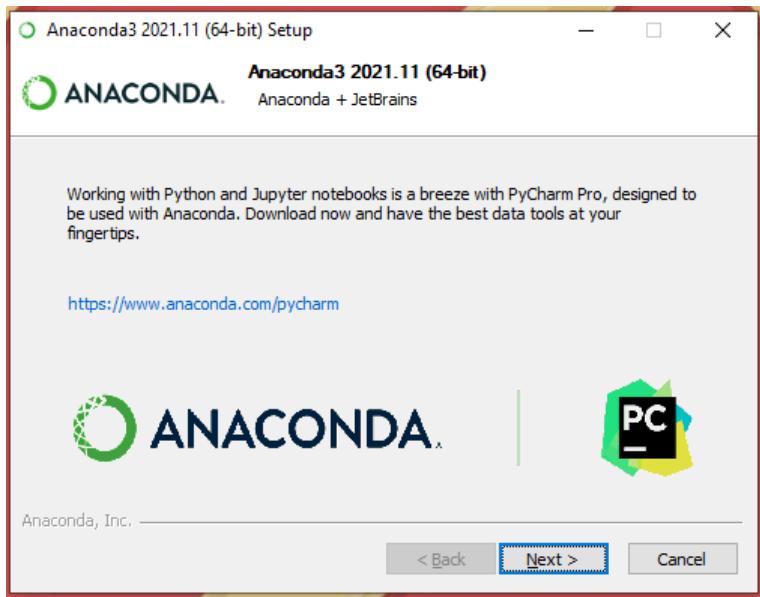


8. tunggu proses instalasi hingga selesai



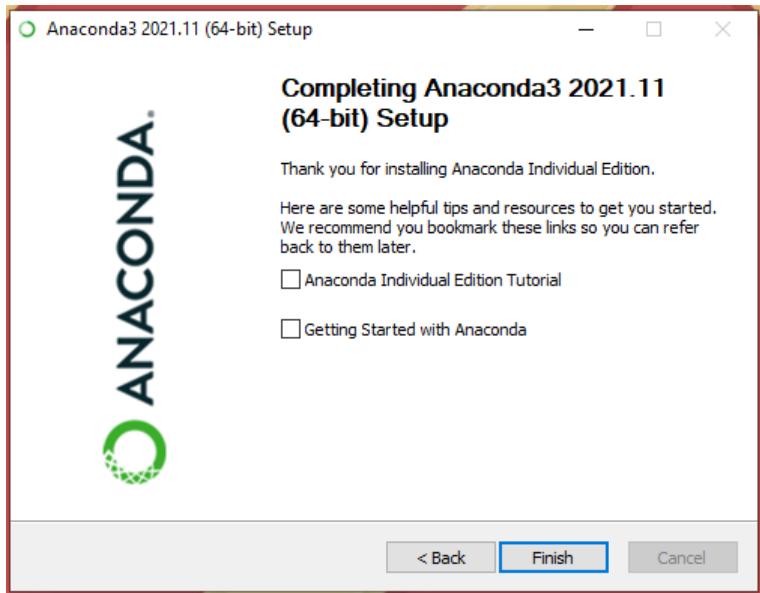
Gambar 1.15 Instalasi Anaconda Windows: Step 7

9. jika sudah muncul tulisan **Completed**, klik Next



Gambar 1.16 Instalasi Anaconda Windows: Step 8

10. instalasi anaconda di windows 10 sudah selesai, klik next



Gambar 1.17 Instalasi Anaconda Windows: Step 9

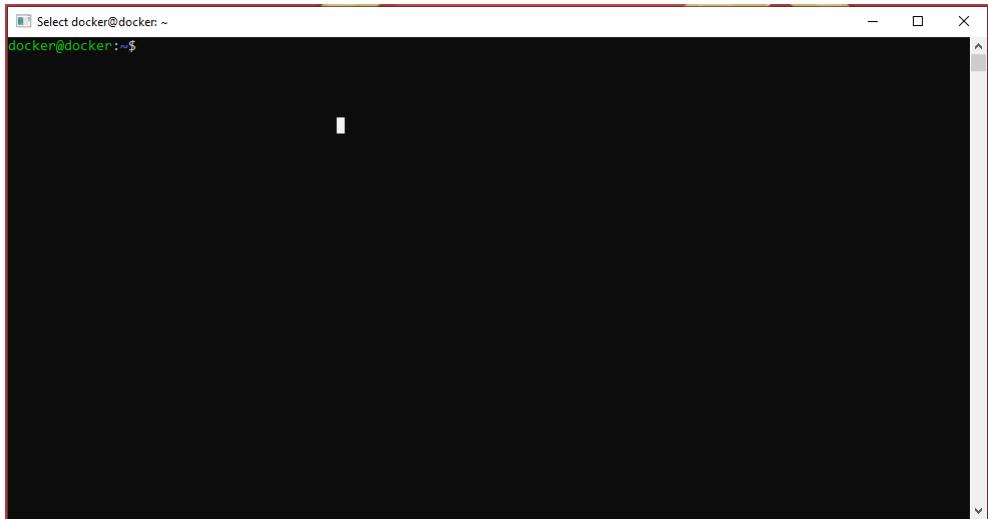
11. jika ingin mengunjungi website anaconda dan membaca manual secara lengkap boleh di centang kedua opsinya dan klik Finish, disini saya tidak mencentang kedua opsi.

1.5 Menjalankan Python

Untuk menjalankan Python ada banyak cara yang bisa dilakukan. Anda bisa menggunakan *shell*, terminal atau menggunakan IDE (Integrated Development Environment). Di bawah ini adalah langkah-langkah menjalankan Python dengan cara yang paling mudah.

1.5.1 Linux

1. Buka terminal CTRL + ALT + T



The screenshot shows a terminal window with a dark background. At the top left, it displays the title 'Select docker@docker: ~'. Below the title, there is a green prompt line starting with 'docker@docker:~\$'. The rest of the window is a solid black color, indicating that no text has been entered or displayed yet.

Gambar 1.18 Menjalankan Python Linux: Step 1

2. ketik **python3** lalu tekan enter

```
docker@docker:~$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

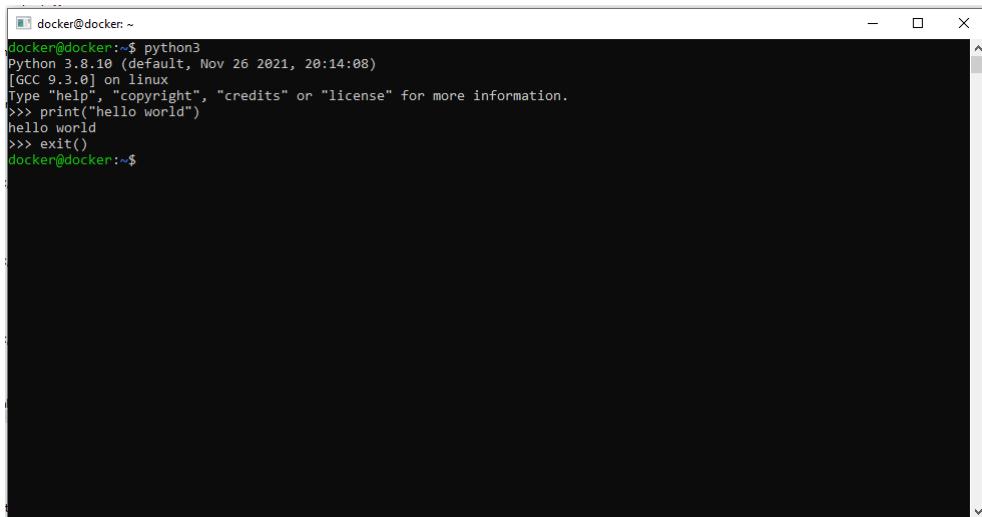
Gambar 1.19 Menjalankan Python Linux: Step 2

3. ketik **print("hello world")** tekan enter

```
docker@docker:~$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>>
```

Gambar 1.20 Menjalankan Python Linux: Step 3

4. selamat anda sudah berhasil menjalankan program python3
5. jika ingin keluar dari shell python3 bisa dengan mengetik **exit()** lalu enter



```
docker@docker: ~$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world")
Hello world
>>> exit()
docker@docker: ~$
```

Gambar 1.21 Menjalankan Python Linux: Step 4

atau

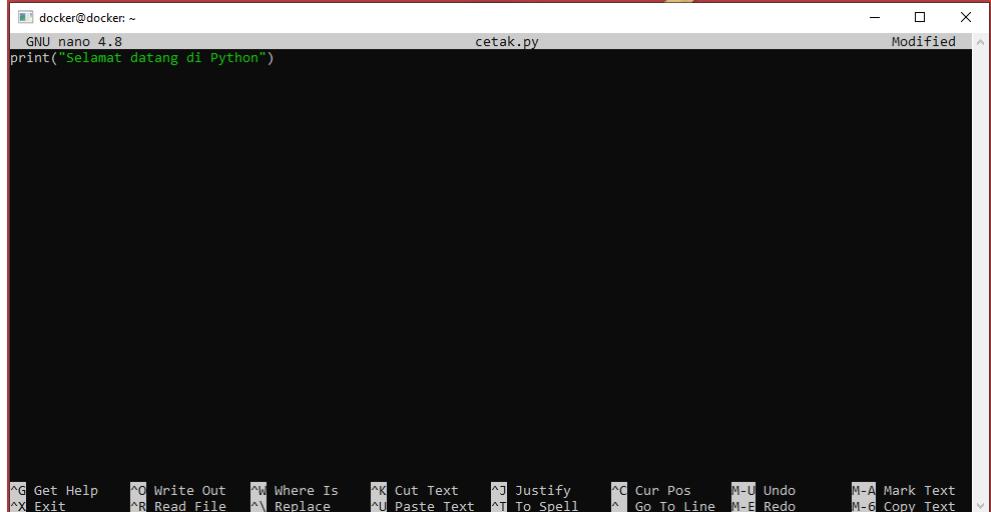
1. Gunakan teks editor, misalnya **nano cetak.py**



```
docker@docker: ~$ nano cetak.py
```

Gambar 1.22 Menjalankan Python Linux dengan Teks Editor: Step 1

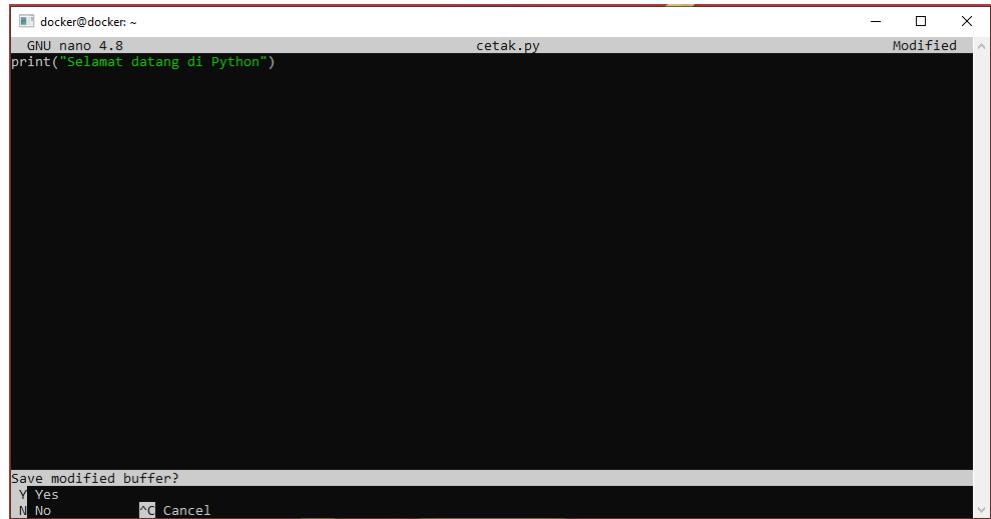
2. ketik **print("Selamat datang di Python")**



A screenshot of a terminal window titled "cetak.py". The window shows the command "docker@docker: ~" followed by the Python code "print("Selamat datang di Python")". The terminal has a dark background and light-colored text. A menu bar at the top includes "Modified". The bottom of the window shows a series of keyboard shortcuts for various functions like Get Help, Write Out, Cut Text, Paste Text, Undo, Redo, etc.

Gambar 1.23 Menjalankan Python Linux dengan Teks Editor: Step 2

3. lalu tekan tombol CTRL + X



A screenshot of a terminal window titled "cetak.py". The window shows the command "docker@docker: ~" followed by the Python code "print("Selamat datang di Python")". At the bottom of the screen, a modal dialog box appears with the message "Save modified buffer?". It contains three options: "Y Yes", "N No", and "C Cancel".

Gambar 1.24 Menjalankan Python Linux dengan Teks Editor: Step 3

4. setelah itu tekan tombol y

A screenshot of a terminal window titled "cetak.py". The window shows the command "GNU nano 4.8" at the top left. The main area contains the following Python code:

```
print("Selamat datang di Python")
```

The status bar at the bottom displays file-related options: "File Name to Write: cetak.py", "Get Help", "Cancel", "DOS Format", "Mac Format", "Append", "Prepend", "Backup File", and "To Files".

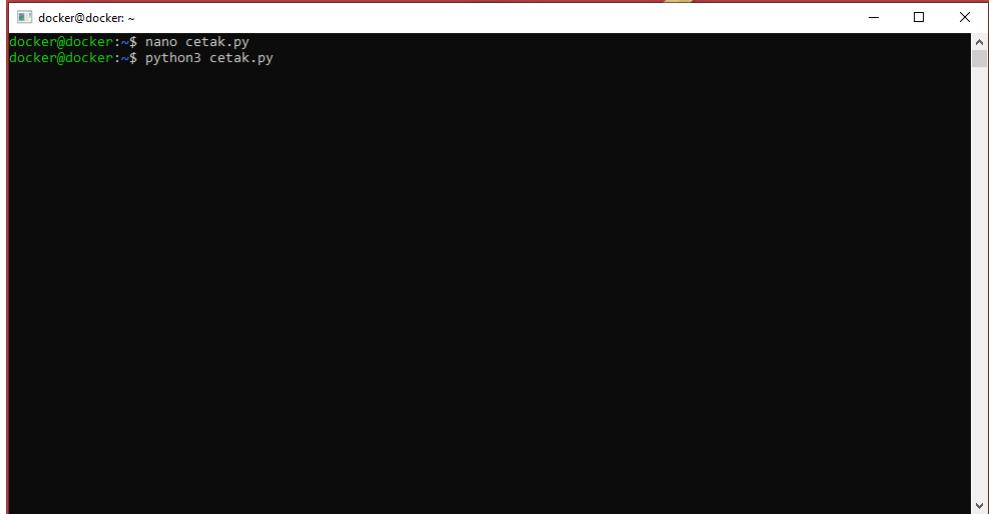
Gambar 1.25 Menjalankan Python Linux dengan Teks Editor: Step 4

5. lalu tekan Enter

A screenshot of a terminal window showing the command "nano cetak.py" being typed into the input field. The terminal prompt is "docker@docker:~\$".

Gambar 1.26 Menjalankan Python Linux dengan Teks Editor: Step 5

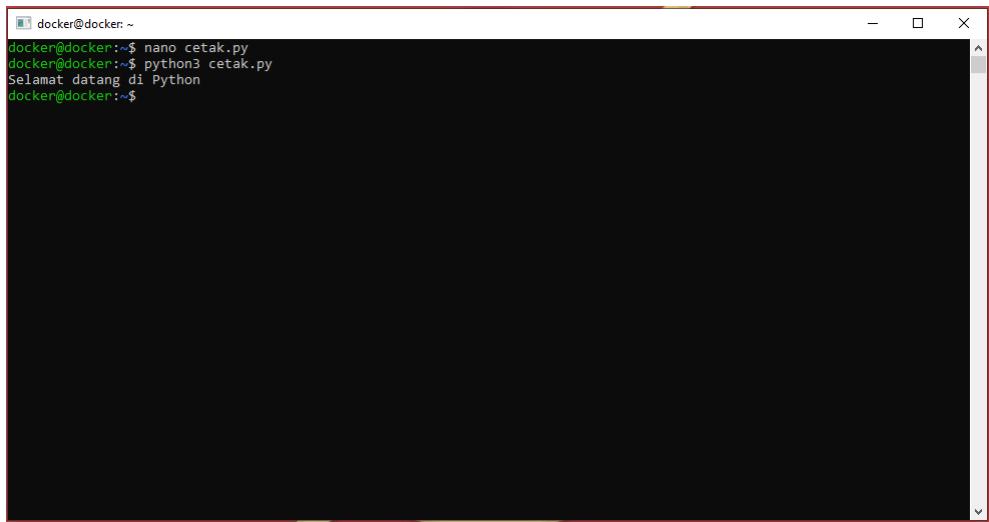
6. lalu ketikkan **python3 cetak.py**



```
docker@docker:~$ nano cetak.py
docker@docker:~$ python3 cetak.py
```

Gambar 1.27 Menjalankan Python Linux dengan Teks Editor: Step 6

7. lalu tekan Enter



```
docker@docker:~$ nano cetak.py
docker@docker:~$ python3 cetak.py
Selamat datang di Python
docker@docker:~$
```

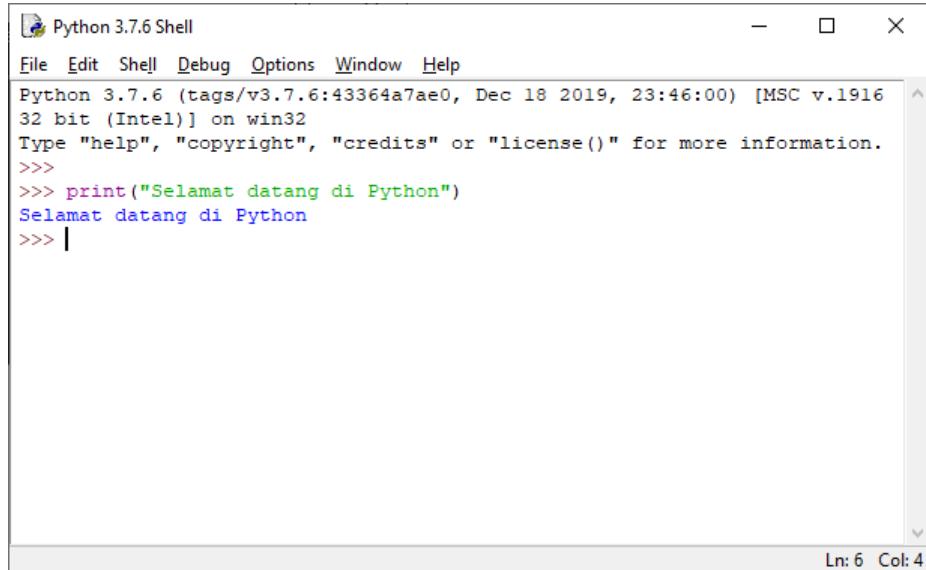
Gambar 1.28 Menjalankan Python Linux dengan Teks Editor: Step 7

1.5.2 Windows

1.5.2.1 Menggunakan Shell

1. Buka IDLE (python shell di windows), Anda bisa mencarinya di tombol START.

2. Tuliskan script Python Anda, contoh: print("Selamat datang di Python"). jika sudah tekan tombol ENTER, dan script Python akan dijalankan/eksekusi.



The screenshot shows the Python 3.7.6 Shell window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main area displays the Python version information and a help message. A code input area shows the command `>>> print("Selamat datang di Python")`, which is then executed, resulting in the output `Selamat datang di Python`. The status bar at the bottom right indicates "Ln: 6 Col: 4".

Gambar 1.29 Menjalankan Python Windows

3. Untuk keluar dari Python shell ketik exit()

1.5.2.2 Menggunakan Script Editor

1. Untuk menjalankan script yang disimpan dalam file, buka IDLE (python shell di windows), Anda bisa mencarinya di tombol START.
2. Klik menu File - New File
3. Tulis script Python pada window yang muncul, contoh:

```
: print("Belajar Python")
```
4. Simpan script lewat menu File - Save
5. Jalankan program dengan klik menu Run - Run Module

1.6 Hello World

Syntax bahasa Python hampir sama dengan bahasa pemrograman pada umumnya seperti Java atau PHP.

1.6.1 Syntax Dasar

Dibawah ini adalah contoh fungsi Python yang digunakan untuk mencetak. Di Python untuk mencetak cukup gunakan fungsi print() , dimana sesuatu yang akan dicetak harus diletakkan diantara kurung buka dan kurung tutup, bahkan di Python versi 2.x Anda tidak harus menggunakan tanda kurung kurawal, cukup pisahkan dengan spasi. Jika ingin mencetak tipe data String langsung, Anda harus memasukkannya ke dalam tanda kutip terlebih dahulu.

```
1 print("Hello World")
```

Saat anda menjalankan script diatas, Anda akan melihat output berupa text Hello World.

1.6.2 Python Case Sensitivity

Python bersifat case sensitif, ini artinya huruf besar dan huruf kecil memiliki perbedaan. Sebagai contoh jika Anda menggunakan fungsi print dengan huruf kecil print() akan berhasil. Lain hal jika anda menggunakan huruf kapital Print() atau PRINT() , akan muncul pesan error. Aturan ini berlaku untuk nama variabel ataupun fungsi-fungsi lainnya.

1.7 Komentar Python

Komentar (comment) adalah kode di dalam script Python yang tidak dieksekusi atau tidak dijalankan mesin. Komentar hanya digunakan untuk menandai atau memberikan keterangan tertulis pada script. Komentar biasa digunakan untuk membiarkan orang lain memahami apa yang dilakukan script. atau untuk mengingatkan kepada programmer sendiri jika suatu saat kembali mengedit script tersebut. Untuk menggunakan komentar anda cukup menulis tanda pagar #, diikuti dengan komentar Anda. Dibawah ini adalah contoh penggunaan komentar pada Python.

```
1 # Ini adalah komentar
2 # Tulisan ini tidak akan dieksekusi
3
4 #komentar dengan tanda pagar hanya bisa digunakan
5 #untuk
6 #satu
7 #baris
8 """
9 """
10 Penulisan Komentar lebih dari satu baris yaitu
11 dengan menggunakan kutip dua 3 kali dan
12 ditutup dengan kutip dua 3 kali juga
13 """
14
15 print("Hello World") #ini juga komentar
16
17 #print("Welcome")
```

```

19 # komentar bisa berisi spesial karakter !@#$%^&*() ,./; '[]\
20
21 #mencetak nama
22 print("Budi")
23
24 #mencetak angka/integer
25 print(123)

```

Listing 1.1 Python Comment

Saat anda menjalankan script diatas, Anda akan melihat output berupa Hello World, Budi dan 123, karena tulisan/komentar yang ditulis tidak dieksekusi.

1.8 Tipe Data Python

Tipe data adalah suatu media atau memori pada komputer yang digunakan untuk menampung informasi. Python sendiri mempunyai tipe data yang cukup unik bila kita bandingkan dengan bahasa pemrograman yang lain. Berikut adalah tipe data dari bahasa pemrograman Python :

Tipe Data	Contoh	Penjelasan
Boolean	True atau False	Menyatakan benar True yang bernilai 1 , atau salah False yang bernilai 0
String	"Ayo belajar Python"	Menyatakan karakter/kalimat bisa berupa huruf angka, dll (diapit tanda " atau ')
Integer	10 atau 100	Menyatakan bilangan bulat
Float	3.14 atau 1.22	Menyatakan bilangan yang mempunyai koma
Hexadecimal	9a atau 1d3	Menyatakan bilangan dalam format heksa (bilangan berbasis 16)
Complex	1 + 5j	Menyatakan pasangan angka real dan imajiner
List	[‘tri’, ‘angga’, ‘dio’, ‘simamora’] atau [1, 2, 3, 4]	Data uataian yang menyimpan berbagai tipe data dan isinya bisa diubah atau modifikasi
Tuple	(‘tri’, ‘angga’, ‘dio’, ‘simamora’) atau (1, 2, 3, 4)	Data uataian yang menyimpan berbagai tipe data dan isinya tidak bisa diubah atau modifikasi
Dictionary	{‘nama’: ‘tri angga dio simamora’}	Data uataian yang menyimpan berbagai tipe data berupa pasangan key dan value

Untuk mencoba berbagai macam tipe data, silahkan coba script Python dibawah ini.

```
1 #tipe data Boolean
2 print(True)
3
4 #tipe data String
5 print("Ayo belajar Python")
6 print('Belajar Python Sangat Mudah')
7
8 #tipe data Integer
9 print(20)
10
11 #tipe data Float
12 print(3.14)
13
14 #tipe data Hexadecimal
15 print(9a)
16
17 #tipe data Complex
18 print(5j)
19
20 #tipe data List
21 print([1,2,3,4,5])
22 print(["satu", "dua", "tiga"])
23
24 #tipe data Tuple
25 print((1,2,3,4,5))
26 print(("satu", "dua", "tiga"))
27
28 #tipe data Dictionary
29 print({"nama": "Budi", 'umur': 20})
30 #tipe data Dictionary dimasukan ke dalam variabel biodata
31 biodata = {"nama": "Andi", 'umur': 21} #proses inisialisasi variabel
     biodata
32 print(biodata) #proses pencetakan variabel biodata yang berisi tipe
     data Dictionary
33 print(type(biodata)) #fungsi untuk mengecek jenis tipe data. akan
     tampil <class 'dict'> yang berarti dict adalah tipe data
     dictionary
```

Listing 1.2 Python Tipe Data

1.9 Variabel Python

Variabel adalah lokasi memori yang dicadangkan untuk menyimpan nilai-nilai. Ini berarti bahwa ketika Anda membuat sebuah variabel Anda memesan beberapa ruang di memori. Variabel menyimpan data yang dilakukan selama program dieksekusi, yang nantinya isi dari variabel tersebut dapat diubah oleh operasi - operasi tertentu pada program yang menggunakan variabel.

Variabel dapat menyimpan berbagai macam tipe data. Di dalam pemrograman Python, variabel mempunyai sifat yang dinamis, artinya variabel Python tidak perlu

dideklarasikan tipe data tertentu dan variabel Python dapat diubah saat program dijalankan.

Penulisan variabel Python sendiri juga memiliki aturan tertentu, yaitu :

1. Karakter pertama harus berupa huruf atau garis bawah/underscore _
2. Karakter selanjutnya dapat berupa huruf, garis bawah/underscore _ atau angka
3. Karakter pada nama variabel bersifat sensitif (case-sensitif). Artinya huruf kecil dan huruf besar dibedakan. Sebagai contoh, **namaDepan** dan **namadepan** adalah variabel yang berbeda.

Untuk mulai membuat variabel di Python caranya sangat mudah, Anda cukup menuliskan variabel lalu mengisinya dengan suatu nilai dengan cara menambahkan tanda sama dengan = diikuti dengan nilai yang ingin dimasukan.

Dibawah ini adalah contoh penggunaan variabel dalam bahasa pemrograman Python

```
1 #proses memasukan data ke dalam variabel
2 nama = "John Doe"
3 #proses mencetak variabel
4 print(nama)
5
6 #nilai dan tipe data dalam variabel dapat diubah
7 umur = 20                      #nilai awal
8 print(umur)                    #mencetak nilai umur
9 type(umur)                     #mengecek tipe data umur
10 umur = "dua puluh satu"       #nilai setelah diubah
11 print(umur)                   #mencetak nilai umur
12 type(umur)                    #mengecek tipe data umur
13
14 namaDepan = "Budi"
15 namaBelakang = "Susanto"
16 nama = namaDepan + " " + namaBelakang
17 umur = 22
18 hobi = "Berenang"
19 print("Biodata\n", nama, "\n", umur, "\n", hobi)
20
21 #contoh variabel lainya
22 inivariabel = "Halo"
23 ini_juga_variabel = "Hai"
24 _inivariabeljuga = "Hi"
25 inivariabel222 = "Bye"
26
27 panjang = 10
28 lebar = 5
29 luas = panjang * lebar
30 print(luas)
```

Listing 1.3 Python Variabel

1.10 Operator Python

Operator adalah konstruksi yang dapat memanipulasi nilai dari operan. Sebagai contoh operasi $3 + 2 = 5$. Disini 3 dan 2 adalah operan dan + adalah operator. Bahasa pemrograman Python mendukung berbagai macam operator, diantaranya :

1.10.1 Operator Aritmatika

Operator	Contoh	Penjelasan
Penjumlahan	$1 + 3 = 4$	Menjumlahkan nilai dari masing-masing operan atau bilangan
Pengurangan	$4 - 1 = 3$	Mengurangi nilai operan di sebelah kiri menggunakan operan di sebelah kanan
Perkalian	$2 * 4 = 8$	Mengalikan operan/bilangan
Pembagian	$10 / 5 = 2$	Untuk membagi operan di sebelah kiri menggunakan operan di sebelah kanan
Sisa Bagi (Modulo)	$11 \% 2 = 1$	Mendapatkan sisa pembagian dari operan di sebelah kiri operator ketika dibagi oleh operan di sebelah kanan
Pangkat	$8 ** 2 = 64$	Memangkatkan operan disebelah kiri operator dengan operan di sebelah kanan operator
Pembagian Bulat	$10 // 3 = 3$	Sama seperti pembagian. Hanya saja angka di belakang koma dihilangkan

Dibawah ini adalah contoh penggunaan Operator Aritmatika dalam bahasa pemrograman Python

```

1 #OPERATOR ARITMATIKA
2
3 #Penjumlahan
4 print(13 + 2)
5 apel = 7
6 jeruk = 9
7 buah = apel + jeruk #
8 print(buah)
9
10 #Pengurangan
11 hutang = 10000
12 bayar = 5000
13 sisaHutang = hutang - bayar
14 print("Sisa hutang Anda adalah ", sisaHutang)
15
16 #Perkalian
17 panjang = 15
18 lebar = 8
19 luas = panjang * lebar
20 print(luas)

```

```
21  
22 #Pembagian  
23 kue = 16  
24 anak = 4  
25 kuePerAnak = kue / anak  
26 print("Setiap anak akan mendapatkan bagian kue sebanyak ", kuePerAnak  
      )  
27  
28 #Sisa Bagi / Modulus  
29 bilangan1 = 14  
30 bilangan2 = 5  
31 hasil = bilangan1 % bilangan2  
32 print("Sisa bagi dari bilangan ", bilangan1, " dan ", bilangan2, "  
      adalah ", hasil)  
33  
34 #Pangkat  
35 bilangan3 = 8  
36 bilangan4 = 2  
37 hasilPangkat = bilangan3 ** bilangan4  
38 print(hasilPangkat)  
39  
40 #Pembagian Bulat  
41 print(10//3)  
42 #10 dibagi 3 adalah 3.3333. Karena dibulatkan maka akan menghasilkan  
    nilai 3
```

Listing 1.4 Python Operator Aritmatika

1.10.2 Operator Perbandingan

Operator	Contoh	Penjelasan
Sama dengan	$1 == 1$	bernilai True Jika masing-masing operan memiliki nilai yang sama, maka kondisi bernilai benar atau True.
Tidak sama dengan	$2 != 2$	bernilai False Akan menghasilkan nilai kebalikan dari kondisi sebenarnya.
Tidak sama dengan	$2 <> 2$	bernilai False Akan menghasilkan nilai kebalikan dari kondisi sebenarnya.
Lebih besar dari	$5 > 3$	bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, maka kondisi menjadi benar.
Lebih kecil dari	$5 < 3$	bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, maka kondisi menjadi benar.
Lebih besar atau sama dengan	$5 >= 3$	bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, atau sama, maka kondisi menjadi benar.
Lebih kecil atau sama dengan	$5 <= 3$	bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, atau sama, maka kondisi menjadi benar.

1.10.3 Operator Penugasan

Operator penugasan digunakan untuk memberikan atau memodifikasi nilai ke dalam sebuah variabel.

Operator	Contoh	Penjelasan
Sama dengan	$a = 1$	Memberikan nilai di kanan ke dalam variabel yang berada di sebelah kiri.
Tambah sama dengan	$a += 2$	Memberikan nilai variabel dengan nilai variabel itu sendiri ditambah dengan nilai di sebelah kanan.
Kurang sama dengan	$a -= 2$	Memberikan nilai variabel dengan nilai variabel itu sendiri dikurangi dengan nilai di sebelah kanan.
Kali sama dengan	$a *= 2$	Memberikan nilai variabel dengan nilai variabel itu sendiri dikalikan dengan nilai di sebelah kanan.
Bagi sama dengan	$a /= 4$	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan.
Sisa bagi sama dengan	$a %= 3$	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan. Yang diambil nantinya adalah sisa baginya.
Pangkat sama dengan	$a **= 3$	Memberikan nilai variabel dengan nilai variabel itu sendiri dipangkatkan dengan nilai di sebelah kanan.
Pembagian bulat sama dengan	$a //= 3$	Membagi bulat operan sebelah kiri operator dengan operan sebelah kanan operator kemudian hasilnya diisikan ke operan sebelah kiri.

1.10.4 Prioritas Eksekusi Operator di Python

Dari semua operator diatas, masing-masing mempunyai urutan prioritas yang nantinya prioritas pertama akan dilakukan paling pertama, begitu seterusnya sampai dengan prioritas terakhir.

Operator	Keterangan
**	Aritmatika
, +, -	Bitwise
*, /, %, //	Aritmatika
+, -	Aritmatika
>>, <<	Bitwise
&	Bitwise
~	Bitwise
<=, <, >, >=	Perbandingan
<>, ==, !=	Perbandingan
=, %=, /=, //=, -=, +=, *=, **=	Penugasan
is, is not	Identitas
in, not in	Membership (Keanggotaan)
not, or, and	Logika

1.11 Kondisi Python

1.11.1 Kondisi If

Pengambilan keputusan (kondisi if) digunakan untuk mengantisipasi kondisi yang terjadi saat jalanya program dan menentukan tindakan apa yang akan diambil sesuai dengan kondisi. Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar True. Jika kondisi bernilai salah False maka statement/kondisi if tidak akan di-eksekusi. Dibawah ini adalah contoh penggunaan kondisi if pada Python

```

1 #Kondisi if adalah kondisi yang akan dieksekusi oleh program jika
   bernilai benar atau TRUE
2
3 nilai = 9
4
5 #jika kondisi benar/TRUE maka program akan mengeksekusi perintah
   dibawahnya
6 if(nilai > 7):
7     print("Sembilan Lebih Besar Dari Angka Tujuh") # Kondisi Benar ,
   Dieksekusi
8
9 #jika kondisi salah/FALSE maka program tidak akan mengeksekusi
   perintah dibawahnya
10 if(nilai > 10):
11     print("Sembilan Lebih Besar Dari Angka Sepuluh") # Kondisi Salah ,
   Maka tidak tereksekusi

```

Listing 1.5 Python Kondisi If

Dari contoh diatas, jika program dijalankan maka akan mencetak string "Sembilan Lebih Besar Dari Angka Tujuh" sebanyak 1 kali yaitu pada if pertama. Di if kedua statement bernilai salah, jadi perintah print("Sembilan Lebih Besar Dari Angka Sepuluh") tidak akan dieksekusi.

1.11.2 Kondisi If Else

Pengambilan keputusan (kondisi if else) tidak hanya digunakan untuk menentukan tindakan apa yang akan diambil sesuai dengan kondisi, tetapi juga digunakan untuk menentukan tindakan apa yang akan diambil/dijalankan jika kondisi tidak sesuai. Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif. Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar. Kondisi if else adalah kondisi dimana jika pernyataan benar True maka kode dalam if akan dieksekusi, tetapi jika bernilai salah False maka akan mengeksekusi kode di dalam else. Dibawah ini adalah contoh penggunaan kondisi if else pada Python

```
1 #Kondisi if else adalah jika kondisi bernilai TRUE maka akan
   dieksekusi pada if , tetapi jika bernilai FALSE maka akan
   dieksekusi kode pada else
2
3 nilai = 3
4 #Jika pernyataan pada if bernilai TRUE maka if akan dieksekusi ,
   tetapi jika FALSE kode pada else yang akan dieksekusi .
5 if(nilai > 7):
6     print("Selamat Anda Lulus")
7 else:
8     print("Maaf Anda Tidak Lulus")
```

Listing 1.6 Python Kondisi If Else

Pada contoh diatas, jika program dijalankan maka akan mencetak string "Maaf Anda Tidak Lulus" karena pernyataan pada if bernilai False

1.11.3 Kondisi Elif

Pengambilan keputusan (kondisi if elif) merupakan lanjutan/percabangan logika dari "kondisi if". Dengan elif kita bisa membuat kode program yang akan menyeleksi beberapa kemungkinan yang bisa terjadi. Hampir sama dengan kondisi "else", bedanya kondisi "elif" bisa banyak dan tidak hanya satu.

Dibawah ini adalah contoh penggunaan kondisi elif pada Python

```
1 #Contoh penggunaan kondisi elif
2
3 hari_ini = "Minggu"
4
5 if(hari_ini == "Senin"):
6     print("Saya akan kuliah")
7 elif(hari_ini == "Selasa"):
8     print("Saya akan kuliah")
9 elif(hari_ini == "Rabu"):
10    print("Saya akan kuliah")
```

```

11 elif(hari_ini == "Kamis"):
12     print("Saya akan kuliah")
13 elif(hari_ini == "Jumat"):
14     print("Saya akan kuliah")
15 elif(hari_ini == "Sabtu"):
16     print("Saya akan kuliah")
17 elif(hari_ini == "Minggu"):
18     print("Saya akan libur")

```

Listing 1.7 Python Kondisi Elif

Pada contoh diatas, jika program dijalankan maka akan mencetak string "Saya akan libur".

1.12 Loop Python

Secara umum, pernyataan pada bahasa pemrograman akan dieksekusi secara berurutan. Pernyataan pertama dalam sebuah fungsi dijalankan pertama, diikuti oleh yang kedua, dan seterusnya. Tetapi akan ada situasi dimana Anda harus menulis banyak kode, dimana kode tersebut sangat banyak. Jika dilakukan secara manual maka Anda hanya akan membuang-buang tenaga dengan menulis beratus-ratus bahkan berribu-ribu kode. Untuk itu Anda perlu menggunakan pengulangan di dalam bahasa pemrograman Python.

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

- While Loop
- For Loop
- Nested Loop

1.12.1 While Loop

Pengulangan While Loop di dalam bahasa pemrograman Python dieksesusi statement berkali-kali selama kondisi bernilai benar atau True.

Dibawah ini adalah contoh penggunaan pengulangan While Loop.

```

1 #Contoh penggunaan While Loop
2 #Catatan: Penentuan ruang lingkup di Python bisa menggunakan tab alih
   - alih menggunakan tanda kurung
3
4 count = 0
5 while (count < 9):
6     print ("The count is: ", count)
7     count = count + 1
8
9 print ("Good bye!")

```

Listing 1.8 Python While Loop

1.12.2 For Loop

Pengulangan for pada Python memiliki kemampuan untuk mengulangi item dari urutan apapun, seperti list atau string.

Dibawah ini adalah contoh penggunaan pengulangan For Loop.

```

1 #Contoh pengulangan for sederhana
2 angka = [1,2,3,4,5]
3 for x in angka:
4     print(x)
5
6 #Contoh pengulangan for
7 buah = ["nanas", "apel", "jeruk"]
8 for makanan in buah:
9     print ("Saya suka makan", makanan)
```

Listing 1.9 Python For Loop

1.12.3 Nested Loop

Bahasa pemrograman Python memungkinkan penggunaan satu lingkaran di dalam loop lain. Bagian berikut menunjukkan beberapa contoh untuk menggambarkan konsep tersebut.

Dibawah ini adalah contoh penggunaan Nested Loop.

```

1 #Contoh penggunaan Nested Loop
2 #Catatan: Penggunaan modulo pada kondisional mengasumsikan nilai
3             selain nol sebagai True(benar) dan nol sebagai False(salah)
4
5 i = 2
6 while(i < 100):
7     j = 2
8     while(j <= (i/j)):
9         if not(i%j): break
10        j = j + 1
11        if (j > i/j) : print(i, " is prime")
12        i = i + 1
13
14 print("Good bye!")
```

Listing 1.10 Python Nested Loop

1.13 Number Python

Number adalah tipe data Python yang menyimpan nilai numerik. Number adalah tipe data yang tidak berubah. Ini berarti, mengubah nilai dari sejumlah tipe data akan menghasilkan objek yang baru dialokasikan.

Objek Number dibuat saat Anda memberikan nilai pada-nya. Sebagai contoh :

angkaPertama = 1 angkaKedua = 33

Python mendukung beberapa tipe data Number diantaranya :

- Int
- Float
- Complex

Berikut ini adalah beberapa contoh dari Tipe data Number pada Python :

Int	Float	Complex
20	0.1	3.14j
300	1.20	35.j
-13	-41.2	3.12e-12j
020	32.23+e123	.873j
-0103	-92.	-.123+0J
-0x212	-32.52e10	3e+123J
0x56	60.2-E13	4.31e-4j

1.13.1 Konversi Tipe Data Number Python

Pada Python Anda bisa mengkonversi tipe data dengan menggunakan fungsi. Dibawah ini adalah beberapa fungsi untuk mengkonversi tipe data number Python.

- `int(x)` untuk meng-konversi x menjadi plain integer.
- `long(x)` untuk meng-konversi x menjadi long integer.
- `float(x)` untuk meng-konversi x menjadi floating point number.
- `complex(x)` untuk meng-konversi x menjadi complex number dengan real part x dan imaginary part zero.
- `complex(x, y)` untuk meng-konversi x dan y menjadi complex number dengan real part x dan imaginary part y. x dan numeric expressions y.

1.13.2 Fungsi Matematika Python

Pada bahasa pemrograman Python terdapat fungsi untuk melakukan perhitungan matematis, berikut adalah daftarnya:

Penjelasan	Penggunaan	Penjelasan
Absolute	abs(x)	Nilai absolut dari x:(positive) jarak antara x and 0.
Ceiling	ceil(x)	Ceiling dari x: integer terkecil yang kurang dari x.
Cmp	cmp(x, y)	-1 if $x < y$, 0 if $x == y$, or 1 if $x > y$. Tidak berlaku lagi dengan Python 3. Sebaliknya gunakan return $(x>y)-(x < y)$.
Eksponen	exp(x)	Nilai eksponen dari x: e^x
Fabs	fabs(x)	Nilai absolut dari x.
Floor	floor(x)	Nilai dasar dari x: internet terbesar tidak lebih besar dari x.
Log	log(x)	Logaritma dari x, untuk $x > 0$.
Log 10	log10(x)	Basis 10 logaritma dari x, untuk $x > 0$.
Max	max(x1, x2,...)	Argumen terbesar: Nilai terdekat dengan tak terhingga positif
Min	min(x1, x2,...)	Argumen terkecil: nilai yang paling mendekati tak berhingga negatif.
Modf	modf(x)	Bagian pecahan dan bilangan bulat dari x dalam tupel dua item. Kedua bagian memiliki tanda yang sama dengan x. Bagian integer dikembalikan sebagai float.
Pow	pow(x, y)	Nilai $x^{**}y$.
Round	round(x [,n])	X dibulatkan menjadi n digit dari titik desimal. Putaran Python jauh dari nol sebagai tie-breaker: round (0.5) adalah 1.0 dan round (-0.5) adalah -1.0.
Akar Kuadrat	sqrt(x)	Akar kuadrat x untuk $x > 0$.

1.13.3 Fungsi Nomor Acak Python

Nomor acak digunakan untuk aplikasi permainan, simulasi, pengujian, keamanan, dan privasi. Python mencakup fungsi berikut yang umum digunakan. Berikut adalah daftarnya :

1.13.4 Fungsi Matematika Python

Pada bahasa pemrograman Python terdapat fungsi untuk melakukan perhitungan matematis, berikut adalah daftarnya:

Penjelasan	Penggunaan	Penjelasan
Choice	choice(seq)	Item acak dari list, tuple, atau string.
RandRange	randrange ([start,] stop [,step])	Elemen yang dipilih secara acak dari jangkauan (start, stop, step).
Random	random()	A random float r, sehingga 0 kurang dari atau sama dengan r dan r kurang dari 1
Seed	seed([x])	Menetapkan nilai awal integer yang digunakan dalam menghasilkan bilangan acak. Panggil fungsi ini sebelum memanggil fungsi modul acak lainnya. Tidak ada pengembalian
Shuffle	shuffle(lst)	Mengacak daftar dari daftar di tempat. Tidak ada pengembalian
Floor	floor(x)	The floor of x: the largest integer not greater than x.
Uniform	uniform(x, y)	Sebuah float acak r, sedemikian rupa sehingga x kurang dari atau sama dengan r dan r kurang dari y.

1.13.5 Fungsi Trigonometri Python

Python mencakup fungsi berikut yang melakukan perhitungan trigonometri. Berikut adalah daftarnya :

Penjelasan	Penggunaan	Penjelasan
Acos	acos(x)	Kembalikan kosinus x, di radian.
Asin	asin(x)	Kembalikan busur sinus x, dalam radian.
Atan	atan(x)	Kembalikan busur singgung x, di radian.
Atan 2	atan2(y, x)	Kembali atan (y / x), di radian.
Kosinus	cos(x)	Kembalikan kosinus x radian.
Hypot	hypot(x, y)	Kembalikan norma Euclidean, $\sqrt{x^2 + y^2}$.
Sin	sin(x)	Kembalikan sinus dari x radian.
Tan	tan(x)	Kembalikan tangen x radian.
Derajat	degrees(x)	Mengonversi sudut x dari radian ke derajat.
Radian	radians(x)	Mengonversi sudut x dari derajat ke radian.

1.13.6 Konstanta Matematika Python

Modul ini juga mendefinisikan dua konstanta matematika. Berikut adalah daftarnya :

Penjelasan	Penggunaan	Penjelasan
Pi	pi	Konstanta Pi matematika
e	e	Konstanta e matematika

1.14 String Python

String adalah jenis yang paling populer di bahasa pemrograman. Kita bisa membuatnya hanya dengan melampirkan karakter dalam tanda kutip. Python memperlakukan tanda kutip tunggal sama dengan tanda kutip ganda. Membuat string semudah memberi nilai pada sebuah variabel.

Dibawah ini adalah contoh sederhana dari sebuah string pada bahasa pemrograman Python.

```
1 hello = "Hello World" # hello adalah variabel yang diisi oleh string
2 print(hello)
```

1.14.1 Mengakses Nilai dalam String

Python tidak menggunakan tipe karakter titik koma ; Ini diperlukan sebagai string dengan panjang satu, sehingga juga dianggap sebagai substring.

Untuk mengakses substring, gunakan tanda kurung siku untuk mengiris beserta indeks atau indeks untuk mendapatkan substring Anda. Sebagai contoh :

```
1 hello = "Hello World"
2 print(hello[0])
```

1.14.2 Mengupdate String

Anda dapat “memperbarui” string yang ada dengan (kembali) menugaskan variabel ke string lain. Nilai baru dapat dikaitkan dengan nilai sebelumnya atau ke string yang sama sekali berbeda sama sekali. Sebagai contoh

```
1 hello = "Hello World"
2 print("Updated String :- ", hello[:6] + 'Python')
```

1.14.3 Karakter Escape Python

Dibawah ini adalah tabel dari daftar karakter escape atau karakter non-printable yang dapat diwakili/ditulis dengan awalan notasi backslash.

Notasi Backslash	Karakter Hexa	Penjelasan
\a	0x07	Bell atau alert
\b	0x08	Backspace
\cx	-	Control-x
\C-x	-	Control-x
\e	0x1b	Escape
\f	0x0c	Formfeed
\M-\C-x	-	Meta-Control-x
\n	0x0a	Newline
\nnn	-	Octal notation, dimana n berada di range 0..7
\r	0x0d	Carriage return
\s	0x20	Space
\t	0x09	Tab
\v	0x0b	Vertical tab
\x	-	Character x
\xnn	-	Notasi Hexadecimal, dimana n berada di range 0..9, a..f, atau A..F

1.14.4 Operator Spesial String Python

Asumsikan variabel string adalah ‘Belajar’ dan variabel b adalah ‘Python’, lalu dibawah ini adalah operator yang bisa dipakai pada kedua string di variabel tersebut. a = ”Belajar” b = ”Python”

Berikut adalah daftar operator spesial string pada Python :

Operator	Contoh	Penjelasan
+	a + b	akan menghasilkan BelajarPython Concatenation - Menambahkan nilai pada kedua sisi operator
*	a*2	akan menghasilkan BelajarBelajar Pengulangan - Membuat string baru, menggabungkan beberapa salinan dari string yang sama
[]	a[1]	akan menghasilkan e Slice - Memberikan karakter dari indeks yang diberikan
[:]	a[1:4]	akan menghasilkan ela Range Slice - Memberikan karakter dari kisaran yang diberikan
in	B in a	akan menghasilkan 1 Keanggotaan - Mengembalikan nilai true jika ada karakter dalam string yang diberikan
not in	Z not in a	akan menghasilkan 1 Keanggotaan - Mengembalikan nilai true jika karakter tidak ada dalam string yang diberikan
r/R	print r'\n' prints\n dan print R' \n'prints\n Raw String -	Menekan arti aktual karakter Escape. Sintaks untuk string mentah sama persis dengan string biasa kecuali operator string mentah, huruf “r”, yang mendahului tanda petik. “R” bisa berupa huruf kecil (r) atau huruf besar (R) dan harus ditempatkan tepat sebelum tanda kutip pertama.
%		Format - Melakukan format String

1.14.5 Operator Format String Python

Salah satu fitur Python yang paling keren adalah format string operator %. Operator ini unik untuk string dan membuat paket memiliki fungsi dari keluarga printf C () C. berikut adalah contoh sederhananya :

```
1 print ("My name is %s and weight is %d kg!" % ('Andi', 21))
```

Berikut adalah daftar lengkap simbol yang bisa digunakan bersamaan dengan % :

Operator	Penjelasan
%c	character
%s	Konversi string melalui str () sebelum memformat
%i	Dianggap sebagai bilangan bulat desimal
%d	Dianggap sebagai bilangan bulat desimal
%u	Unsigned decimal integer
%o	Bilangan bulat oktal
%x	Bilangan bulat heksadesimal (huruf kecil)
%X	Bilangan bulat heksadesimal (huruf besar)
%e	Notasi eksponensial (dengan huruf kecil 'e')
%E	Notasi eksponensial (dengan huruf besar 'E')
%f	Bilangan real floating point
%g	Yang lebih pendek dari% f dan% e
%G	Lebih pendek dari% f dan% E

1.14.6 Triple Quote Python

Python triple quotes digunakan dengan membiarkan string untuk ditulis dalam beberapa baris, termasuk kata kerja NEWLINEs, TABs, dan karakter khusus lainnya. Sintaks untuk triple quotes terdiri dari tiga tanda kutip tunggal atau ganda ditulis berturut-turut : Berikut adalah contohnya :

```
1 kutipantiga = """this is a long string that is made up of
2 several lines and non-printable characters such as
3 TAB (\t) and they will show up that way when displayed.
4 NEWLINEs within the string, whether explicitly given like
5 this within the brackets [ \n ], or just a NEWLINE within
6 the variable assignment will also show up."""
7
```

```
7 print (kutipantiga)
```

1.14.7 String Unicode Python

Pada Python 3, semua string diwakili dalam Unicode. Sedangkan pada Python 2 disimpan secara internal sebagai 8-bit ASCII, maka diperlukan lampiran ‘u’ untuk membuatnya menjadi Unicode. Tetapi hal ini tidak lagi diperlukan sekarang. :

Metode String Built-in

Python menyertakan metode built-in berikut untuk memanipulasi string

Operator	Penjelasan
capitalize()	Meng-kapitalkan huruf pertama string
center(width, fillchar)	Mengembalikan string yang dilapisi dengan fillchar dengan string asli yang dipusatkan pada total width kolom.
count(str, beg = 0,end = len(string))	Menghitung berapa kali str yang terjadi dalam string atau dalam substring string jika memulai indeks beg dan end index end diberikan.
decode(encoding = 'UTF-8',errors = 'strict')	Dekode string menggunakan codec yang terdaftar untuk pengkodean. Encoding default ke pengkodean string default.
encode(encoding = 'UTF-8',errors = 'strict')	Mengembalikan versi string yang dikodekan string; Pada kesalahan, default adalah menaikkan ValueError kecuali jika kesalahan diberikan dengan 'ignore' atau 'replace'.
endswith(suffix, beg = 0, end = len(string))	Menentukan apakah string atau substring string (jika memulai indeks memohon dan mengakhiri akhir indeks diberikan) berakhir dengan akhiran; Mengembalikan nilai true jika benar dan salah.
expandtabs(tabsize = 8)	Memperluas tab dalam string ke banyak ruang; Default ke 8 spasi per tab jika tabsize tidak tersedia.
find(str, beg = 0 end = len(string))	Tentukan jika str terjadi dalam string atau dalam substring string jika memulai indeks beg dan end index end diberikan return index jika ditemukan dan -1 sebaliknya.
index(str, beg = 0, end = len(string))	Sama seperti find (), namun menimbulkan pengecualian jika str tidak ditemukan.
isalnum()	Mengembalikan true jika string memiliki minimal 1 karakter dan semua karakternya alfanumerik dan false sebaliknya.
isalpha()	Mengembalikan true jika string memiliki minimal 1 karakter dan semua karakter adalah abjad dan false sebaliknya.
isdigit()	Mengembalikan true jika string hanya berisi digit dan false sebaliknya.
islower()	Mengembalikan true jika string memiliki setidaknya 1 karakter casing dan semua karakter casing dalam huruf kecil dan false sebaliknya.
isnumeric()	Mengembalikan true jika string unicode hanya berisi karakter numerik dan false sebaliknya.
isspace()	Mengembalikan true jika string hanya berisi karakter spasi dan false sebaliknya.

Operator	Penjelasan
istitle()	Mengembalikan true jika string benar “titlecased” dan false sebaliknya.
isupper()	Mengembalikan true jika string memiliki setidaknya satu karakter casing dan semua karakter casing ada dalam huruf besar dan false sebaliknya.
join(seq)	Merges (concatenates) representasi string elemen dalam urutan seq menjadi string, dengan string pemisah.
len(string)	Mengembalikan panjang string
ljust(width[, fillchar])	Mengembalikan string berlapis ruang dengan string asli dibiarkan dibenarkan ke kolom lebar total.
lower()	Mengonversi semua huruf besar dalam bentuk string menjadi huruf kecil.
lstrip()	Menghapus semua spasi utama dalam string.
maketrans()	Mengembalikan tabel terjemahan untuk digunakan dalam fungsi terjemahan.
max(str)	Mengembalikan karakter alfabetik dari string str.
min(str)	Mengembalikan min karakter abjad dari string str.
replace(old, new [, max])	Menggantikan semua kemunculan lama dalam string dengan kejadian baru atau paling maksimal jika max diberikan.
rfind(str, beg = 0,end = len(string))	Sama seperti find (), tapi cari mundur dalam string.
rindex(str, beg = 0, end = len(string))	Sama seperti index (), tapi cari mundur dalam string.
rjust(width,[, fillchar])	Mengembalikan string berlapis ruang dengan senar asli benar-dibenarkan untuk total kolom lebar.
rstrip()	Menghapus semua spasi spasi string.
split(str="'", num=string.count(str))	Membagi string sesuai dengan pemisah str (ruang jika tidak disediakan) dan mengembalikan daftar substring; Terpecah menjadi paling banyak substring jika diberikan.
splitlines(num=string.count('\n'))	Membagi string sama sekali (atau num) NEWLINEs dan mengembalikan daftar setiap baris dengan NEWLINEs dihapus.
startswith(str, beg=0,end=len(string))	Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise.

Operator	Penjelasan
strip([chars])	Lakukan kedua lstrip () dan rstrip () pada string
swapcase()	Kasus invers untuk semua huruf dalam string.
title()	Mengembalikan versi string “titlecased”, yaitu, semua kata diawali dengan huruf besar dan sisanya huruf kecil.
translate(table, deletechars="")	Menerjemahkan string sesuai dengan tabel terjemahan str (256 karakter), menghapus string del.
upper()	Mengonversi huruf kecil dalam bentuk string ke huruf besar.
zfill (width)	Mengembalikan string asli yang tertinggal dengan angka nol ke total karakter lebar; Dimaksudkan untuk angka, zfill () mempertahankan tanda apapun yang diberikan (kurang satu nol).
isdecimal()	Mengembalikan nilai true jika string unicode hanya berisi karakter desimal dan false sebaliknya.

1.15 List Python

Dalam bahasa pemrograman Python, struktur data yang paling dasar adalah urutan atau lists. Setiap elemen-elemen berurutan akan diberi nomor posisi atau indeksnya. Indeks pertama dalam list adalah nol, indeks kedua adalah satu dan seterusnya.

Python memiliki enam jenis urutan built-in, namun yang paling umum adalah list dan tuple. Ada beberapa hal yang dapat Anda lakukan dengan semua jenis list. Operasi ini meliputi pengindeksan, pengiris, penambahan, perbanyak, dan pengecekan keanggotaan. Selain itu, Python memiliki fungsi built-in untuk menemukan panjang list dan untuk menemukan elemen terbesar dan terkecilnya.

1.15.1 Membuat List Python

List adalah tipe data yang paling serbaguna yang tersedia dalam bahasa Python, yang dapat ditulis sebagai daftar nilai yang dipisahkan koma (item) antara tanda kurung siku. Hal penting tentang daftar adalah item dalam list tidak boleh sama jenisnya.

Membuat list sangat sederhana, tinggal masukkan berbagai nilai yang dipisahkan koma di antara tanda kurung siku. Dibawah ini adalah contoh sederhana pembuatan list dalam bahasa Python.

```
1 #Contoh sederhana pembuatan list pada bahasa pemrograman python
2 list1 = ['kimia', 'fisika', 1993, 2017]
3 list2 = [1, 2, 3, 4, 5]
4 list3 = ["a", "b", "c", "d"]
```

1.15.2 Akses Nilai Dalam List Python

Untuk mengakses nilai dalam list python, gunakan tanda kurung siku untuk mengeiris beserta indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut.

Berikut adalah contoh cara mengakses nilai di dalam list python :

```
1 #Cara mengakses nilai di dalam list Python
2 list1 = ['fisika', 'kimia', 1993, 2017]
3 list2 = [1, 2, 3, 4, 5, 6, 7]
4
5 print ("list1 [0]: ", list1[0])
6 print ("list2 [1:5]: ", list2 [1:5])
```

1.15.3 Update Nilai Dalam List Python

Anda dapat memperbarui satu atau beberapa nilai di dalam list dengan memberikan potongan di sisi kiri operator penugasan, dan Anda dapat menambahkan nilai ke dalam list dengan metode append(). Sebagai contoh :

```
1 list = ['fisika', 'kimia', 1993, 2017]
2 print ("Nilai ada pada index 2 : ", list[2])
3
4 list[2] = 2001
5 print ("Nilai baru ada pada index 2 : ", list[2])
```

1.15.4 Hapus Nilai Dalam List Python

Untuk menghapus nilai di dalam list python, Anda dapat menggunakan salah satu pernyataan del jika Anda tahu persis elemen yang Anda hapus. Anda dapat menggunakan metode remove() jika Anda tidak tahu persis item mana yang akan dihapus. Sebagai contoh :

```
1 #Contoh cara menghapus nilai pada list python
2
3 list = ['fisika', 'kimia', 1993, 2017]
4
5 print (list)
6 del list[2]
7 print ("Setelah dihapus nilai pada index 2 : ", list)
```

1.15.5 Operasi Dasar Pada List Python

List Python merespons operator + dan * seperti string; Itu artinya penggabungan dan pengulangan di sini juga berlaku, kecuali hasilnya adalah list baru, bukan sebuah String.

Sebenarnya, list merespons semua operasi urutan umum yang kami gunakan pada String di bab sebelumnya. Dibawah ini adalah tabel daftar operasi dasar pada list python.

Python Expression	Hasil	Penjelasan
len([1, 2, 3, 4])	4	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Halo!'] * 4	['Halo!', 'Halo!', 'Halo!', 'Halo!']	Repetition
2 in [1, 2, 3]	True	Membership
for x in [1,2,3] : print (x,end = ' ')	1 2 3	Iteration

1.15.6 Indexing, Slicing dan Matrix Pada List Python

Karena list adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk list seperti yang mereka lakukan untuk String.

Dengan asumsi input berikut :

```
L = ['C++', 'Java', 'Python']
```

Python Expression	Hasil	Penjelasan
L[2]	'Python'	Offset mulai dari nol
L[-2]	'Java'	Negatif: hitung dari kanan
[1:]	['Java', 'Python']	Slicing mengambil bagian

1.15.7 Method dan Fungsi Build-in Pada List Python

Python menyertakan fungsi built-in sebagai berikut :

Python Function	Penjelasan
cmp(list1, list2) #	Tidak lagi tersedia dengan Python 3
len(list)	Memberikan total panjang list.
max(list)	Mengembalikan item dari list dengan nilai maks.
min(list)	Mengembalikan item dari list dengan nilai min.
list(seq)	Mengubah tuple menjadi list.

Python menyertakan methods built-in sebagai berikut

Python Function	Penjelasan
list.append(obj)	Menambahkan objek obj ke list
list.count(obj)	Jumlah pengembalian berapa kali obj terjadi dalam list
list.extend(seq)	Tambahkan isi seq ke list
list.index(obj)	Mengembalikan indeks terendah dalam list yang muncul obj
list.insert(index, obj)	Sisipkan objek obj ke dalam list di indeks offset
list.pop(obj = list[-1])	Menghapus dan mengembalikan objek atau obj terakhir dari list
list.remove(obj)	Removes object obj from list
list.reverse()	Membalik list objek di tempat
list.sort([func])	Urutkan objek list, gunakan compare func jika diberikan

1.16 Tuple Python

Sebuah tupel adalah urutan objek Python yang tidak berubah. Tupel adalah urutan, seperti daftar. Perbedaan utama antara tupel dan daftarnya adalah bahwa tupel tidak dapat diubah tidak seperti List Python. Tupel menggunakan tanda kurung, sedangkan List Python menggunakan tanda kurung siku.

Membuat tuple semudah memasukkan nilai-nilai yang dipisahkan koma. Secara opsional, Anda dapat memasukkan nilai-nilai yang dipisahkan koma ini di antara tanda kurung juga. Sebagai contoh :

```

1 #Contoh sederhana pembuatan tuple pada bahasa pemrograman python
2
3 tup1 = ('fisika', 'kimia', 1993, 2017)
4 tup2 = (1, 2, 3, 4, 5)
5 tup3 = "a", "b", "c", "d"

```

Tupel kosong ditulis sebagai dua tanda kurung yang tidak berisi apa-apa, contohnya : tup1 = (); Untuk menulis tupel yang berisi satu nilai, Anda harus memasukkan koma, meskipun hanya ada satu nilai, contohnya : tup1 = (50,) Seperti indeks String, indeks tuple mulai dari 0, dan mereka dapat diiris, digabungkan, dan seterusnya

1.16.1 Akses Nilai Dalam Tuple Python

Untuk mengakses nilai dalam tupel, gunakan tanda kurung siku untuk mengiris bersama indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut. Sebagai contoh :

```
1 #Cara mengakses nilai tuple
2
3 tup1 = ('fisika', 'kimia', 1993, 2017)
4 tup2 = (1, 2, 3, 4, 5, 6, 7)
5
6 print ("tup1[0]: ", tup1[0])
7 print ("tup2[1:5]: ", tup2[1:5])
```

1.16.2 Update Nilai Dalam Tuple Python

Tuple tidak berubah, yang berarti Anda tidak dapat memperbarui atau mengubah nilai elemen tupel. Anda dapat mengambil bagian dari tupel yang ada untuk membuat tupel baru seperti ditunjukkan oleh contoh berikut.

```
1 tup1 = (12, 34.56)
2 tup2 = ('abc', 'xyz')
3
4 # Aksi seperti dibawah ini tidak bisa dilakukan pada tuple python
5 # Karena memang nilai pada tuple python tidak bisa diubah
6 # tup1[0] = 100;
7
8 # Jadi, buatlah tuple baru sebagai berikut
9 tup3 = tup1 + tup2
10 print (tup3)
```

1.16.3 Hapus Nilai Dalam Tuple Python

Menghapus elemen tuple individual tidak mungkin dilakukan. Tentu saja, tidak ada yang salah dengan menggabungkan tupel lain dengan unsur-unsur yang tidak diinginkan dibuang.

Untuk secara eksplisit menghapus keseluruhan tuple, cukup gunakan del statement. Sebagai contoh

```
1 tup = ('fisika', 'kimia', 1993, 2017)
2 print(tup)
3
4 # hapus tuple dengan statement del
5 del tup
6
7 # lalu buat kembali tuple yang baru dengan elemen yang diinginkan
8 tup = ('Bahasa', 'Literasi', 2020)
9 print("Setelah menghapus tuple : ", tup)
```

1.16.4 Operasi Dasar Pada Tuple Python

Tupel merespons operator + dan * sama seperti String; Mereka berarti penggabungan dan pengulangan di sini juga berlaku, kecuali hasilnya adalah tupel baru, bukan string.

Sebenarnya, Tuple merespons semua operasi urutan umum yang kami gunakan pada String di bab sebelumnya. Dibawah ini adalah tabel daftar operasi dasar pada Tuple python

Python Expression	Hasil	Penjelasan
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Halo!',) * 4	('Halo!', 'Halo!', 'Halo!', 'Halo!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1,2,3) : print(x, end = ' ')	1 2 3	Iteration

1.16.5 Indexing, Slicing dan Matrix Pada Tuple Python

Karena tupel adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk tupel seperti pada String, dengan asumsi masukan berikut

Dengan asumsi input berikut : T = ('C++', 'Java', 'Python')

Python Expression	Hasil	Penjelasan
T[2]	'Python'	Offset mulai dari nol
T[-2]	'Java'	Negatif: hitung dari kanan
T[1:]	('Java', 'Python')	Slicing mengambil bagian

1.16.6 Fungsi Build-in Pada Tuple Python

Python menyertakan fungsi built-in sebagai berikut

Python Function	Penjelasan
cmp(tuple1, tuple2) #	Tidak lagi tersedia dengan Python 3
len(tuple)	Memberikan total panjang tuple.
max(tuple)	Mengembalikan item dari tuple dengan nilai maks.
min(tuple)	Mengembalikan item dari tuple dengan nilai min.
tuple(seq)	Mengubah seq menjadi tuple.

1.17 Dictionary Python

Dictionary Python berbeda dengan List ataupun Tuple. Karena setiap urutannya berisi key dan value. Setiap key dipisahkan dari value-nya oleh titik dua (:), item dipisahkan oleh koma, dan semuanya tertutup dalam kurung kurawal. Dictionary kosong tanpa barang ditulis hanya dengan dua kurung kurawal, seperti ini: .

Nilai kamus bisa berupa tipe apa pun, namun key harus berupa tipe data yang tidak berubah seperti string, angka, atau tupel.

1.17.1 Akses Nilai Dalam Dictionary Python

Untuk mengakses elemen Dictionary, Anda dapat menggunakan tanda kurung siku yang sudah dikenal bersama dengan key untuk mendapatkan nilainya. Berikut adalah contoh sederhananya :

```

1 #Contoh cara membuat Dictionary pada Python
2
3 dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
4 print ("dict['Name']: ", dict['Name'])
5 print ("dict['Age']: ", dict['Age'])
```

1.17.2 Update Nilai Dalam Dictionary Python

Anda dapat memperbarui Dictionary dengan menambahkan entri baru atau pasangan nilai kunci, memodifikasi entri yang ada, atau menghapus entri yang ada seperti ditunjukkan pada contoh sederhana yang diberikan di bawah ini.

```

1 #Update dictionary python
2
3 dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
4 dict['Age'] = 8; # Mengubah entri yang sudah ada
5 dict['School'] = "DPS School" # Menambah entri baru
6
```

```

7 print ("dict['Age']: ", dict['Age'])
8 print ("dict['School']: ", dict['School'])

```

1.17.3 Hapus Elemen Dictionary Python

Anda dapat menghapus elemen Dictionary individual atau menghapus keseluruhan isi Dictionary. Anda juga dapat menghapus seluruh Dictionary dalam satu operasi.

Untuk menghapus seluruh Dictionary secara eksplisit, cukup gunakan del statement. Berikut adalah contoh sederhana :

```

1 #Contoh cara menghapus pada Dictionary Python
2
3 dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
4
5 del dict['Name'] # hapus entri dengan key 'Name'
6 dict.clear()     # hapus semua entri di dict
7 del dict        # hapus dictionary yang sudah ada
8
9 print ("dict['Age']: ", dict['Age'])
10 print ("dict['School']: ", dict['School'])

```

1.17.4 Fungsi Build-in Pada Dictionary Python

Python menyertakan fungsi built-in sebagai berikut :

Python Function	Penjelasan
cmp(dict1, dict2)	Membandingkan unsur keduaanya.
len(dict)	Memberikan panjang total Dictionary. Ini sama dengan jumlah item dalam Dictionary.
str(dict)	Menghasilkan representasi string yang dapat dicetak dari Dictionary
type(variable)	Mengembalikan tipe variabel yang lulus. Jika variabel yang dilewatkan adalah Dictionary, maka akan mengembalikan tipe Dictionary.

1.17.5 Method Build-in Pada Dictionary Python

Python menyertakan method built-in sebagai berikut :

Python Method	Penjelasan
dict.clear()	Menghapus semua elemen Dictionary
dict.copy()	Mengembalikan salinan Dictionary
dict.fromkeys()	Buat Dictionary baru dengan kunci dari seq dan nilai yang disetel ke nilai.
dict.get(key, default=None)	For key, nilai pengembalian atau default jika tombol tidak ada dalam Dictionary
dict.has_key(key)	Mengembalikan true jika key dalam Dictionary, false sebaliknya
dict.items()	Mengembalikan daftar dari pasangan tuple dictionary (key, value)
dict.keys()	Mengembalikan daftar key dictionary
dict.setdefault(key, default=None)	Mirip dengan get (), tapi akan mengatur dict [key] = default jika kunci belum ada di dict
dict.update(dict2)	Menambahkan pasangan kunci kata kunci dict2 ke dict
dict.values()	Mengembalikan daftar nilai dictionary

1.18 Tanggal & Waktu Python

Program Python dapat menangani tanggal dan waktu dengan beberapa cara. Konversi antara format tanggal adalah tugas umum untuk komputer. Modul waktu dan kalender Python melacak tanggal dan waktu.

1.18.1 Apa itu Tick?

Interval waktu adalah bilangan floating-point dalam satuan detik. Instansi tertentu dalam waktu dinyatakan dalam hitungan detik sejak pukul 12:00 1 Januari 1970.

Dibawah ini adalah contoh penggunaanya.

```
1 import time; # Digunakan untuk meng-import modul time
```

```
2 ticks = time.time()
3 print("Berjalan sejak 12:00am, January 1, 1970:", ticks)
```

1.18.2 Apa itu TimeTuple Python?

Banyak fungsi waktu Python menangani waktu sebagai tuple dari 9 nomor, seperti yang terdapat pada tabel di bawah ini.

Index	Field	Value
0	4-digit year	2008
1	Bulan	1 sampai 12
2	Hari	1 sampai 31
3	Jam	0 sampai 23
4	Menit	0 sampai 59
5	Detik	0 sampai 61
6	Hari dalam Minggu	0 sampai 6 (0 adalah Senin)
7	Hari dalam Bulan	1 sampai 366
8	Daylight savings	-1, 0, 1, -1 means library determines DST

Tuple di atas setara dengan struktur struct_time. Struktur ini memiliki atribut berikut

Index	Attribute	Value
0	tm_year	2008
1	tm_mon	1 sampai 12
2	tm_mday	1 sampai 31
3	tm_hour	0 sampai 23
4	tm_min	0 sampai 59
5	tm_sec	0 sampai 61
6	tm_wday	0 sampai 6 (0 adalah Senin)
7	tm_yday	1 sampai 366
8	tm_isdst	-1, 0, 1, -1 means library determines DST 0, 1, -1 means library determines DST -1, 0, 1, -1 means library determines DST

1.18.3 Mendapatkan Waktu Saat Ini

Untuk menerjemahkan waktu instan dari satu detik sejak nilai floating-point ke waktu menjadi tupel waktu, lewati nilai floating-point ke fungsi (mis., `Localtime`) yang mengembalikan waktu tupel dengan semua sembilan item valid.

```
1 import time;
2
3 localtime = time.localtime(time.time())
4 print("Waktu lokal saat ini :", localtime)
```

1.18.4 Mendapatkan Waktu yang berformat

Anda dapat memformat kapan saja sesuai kebutuhan Anda, namun metode sederhana untuk mendapatkan waktu dalam format yang mudah dibaca adalah `asctime()`

```
1 import time;
2
3 localtime = time.asctime( time.localtime(time.time()) )
4 print("Waktu lokal saat ini :", localtime)
```

1.18.5 Mendapatkan kalender dalam sebulan

Modul kalender memberikan berbagai macam metode untuk dimainkan dengan kalender tahunan dan bulanan. Di sini, kami mencetak kalender untuk bulan tertentu (Jan 2008)

```
1 import calendar
2
3 cal = calendar.month(2008, 1)
4 print("Dibawah ini adalah kalender:")
5 print(cal)
```

1.18.6 Modul time pada Python

Ada modul waktu populer yang tersedia dengan Python yang menyediakan fungsi untuk bekerja dengan waktu dan untuk mengkonversi antara representasi. Dibawah ini adalah tabel dari modul time pada python yang ada.

Python Function	Penjelasan
time.altzone	Diimbangi zona waktu DST lokal, dalam detik di sebelah barat UTC, jika seseorang didefinisikan. Ini negatif jika zona waktu DST lokal berada di sebelah timur UTC (seperti di Eropa Barat, termasuk Inggris). Gunakan saja ini jika siang hari tidak nol.
time.asctime([tupletime])	Menerima time-tupel dan mengembalikan string 24-karakter yang dapat dibaca seperti ‘Tue Dec 11 18:07:14 2008’.
time.clock()	Mengembalikan waktu CPU saat ini sebagai jumlah floating-point detik. Untuk mengukur biaya komputasi dari berbagai pendekatan, nilai time.clock lebih bermanfaat daripada time.time () .
time.ctime([secs])	Seperti asctime (localtime (detik)) dan tanpa argumen seperti asctime ()
time.gmtime([secs])	Menerima instan yang diungkapkan dalam hitungan detik sejak zaman dan mengembalikan waktu tuple t dengan waktu UTC. Catatan: t.tm_isdst selalu 0
time.localtime([secs])	Menerima instan yang dinyatakan dalam hitungan detik sejak zaman dan mengembalikan waktu tuple t dengan waktu setempat (t.tm_isdst adalah 0 atau 1, tergantung pada apakah DST berlaku seketika oleh peraturan lokal).
time.mktime(tupletime)	Menerima instan dinyatakan sebagai time-tuple di waktu setempat dan mengembalikan nilai floating point

Ada dua atribut penting yang tersedia dengan modul waktu:

Python Method	Penjelasan
time.timezone	Atribut time.timezone adalah offset dalam detik zona waktu lokal (tanpa DST) dari UTC (> 0 di Amerika; <= 0 di sebagian besar Eropa, Asia, Afrika).
time.tzname	Atribut time.tzname adalah sepasang string yang bergantung pada lokal, yang merupakan nama zona waktu lokal tanpa dan dengan DST.

1.18.7 Modul calendar pada Python

Modul kalender menyimpan fungsi yang berhubungan dengan kalender, termasuk fungsi untuk mencetak kalender teks untuk bulan atau tahun tertentu.

Secara default, kalender mengambil hari Senin sebagai hari pertama dalam minggu dan minggu sebagai yang terakhir. Untuk mengubah ini, fungsi call calendar.setfirstweekday () .

Berikut adalah daftar fungsi yang tersedia dengan modul kalender:

Python Function	Penjelasan
calendar.calendar(year,w=2,l=1,c=6)	Mengembalikan string multiline dengan kalender untuk tahun tahun yang diformat menjadi tiga kolom yang dipisahkan oleh ruang c. W adalah lebar karakter setiap tanggal; Setiap baris memiliki panjang $21 * w + 18 + 2 * c$. L adalah jumlah baris untuk setiap minggu.
calendar.firstweekday()	Mengembalikan pengaturan saat ini untuk hari kerja yang dimulai setiap minggu. Secara default, saat kalender pertama kali diimpor, ini adalah 0, yang berarti Senin.
calendar.isleap(year)	Pengembalian True jika tahun adalah tahun kabisat; Jika tidak, False
calendar.leapdays(y1,y2)	Mengembalikan jumlah lompatan hari dalam tahun-tahun dalam rentang (y1, y2).
calendar.month(year,month,w=2,l=1)	Mengembalikan string multiline dengan kalender untuk bulan-bulan tahun, satu baris per minggu ditambah dua baris header. W adalah lebar karakter setiap tanggal; Setiap baris memiliki panjang $7 * w + 6$. L adalah jumlah baris untuk setiap minggu.
calendar.monthcalendar(year,month)	Mengembalikan daftar dafatar int. Setiap sublist menunjukkan seminggu. Hari di luar bulan-bulan tahun diatur ke 0; Hari dalam bulan ditetapkan ke hari ke bulan, 1 dan ke atas.

Python Function	Penjelasan
calendar.monthrange(year,month)	Mengembalikan dua bilangan bulat. Yang pertama adalah kode hari kerja untuk hari pertama bulan bulan di tahun; Yang kedua adalah jumlah hari dalam sebulan. Kode hari kerja adalah 0 (Senin) sampai 6 (Minggu); Angka bulan adalah 1 sampai 12.
calendar.prcal(year,w=2,l=1,c=6)	Seperti kalender cetak.calendar (tahun, w, l, c).
calendar.prmonth(year,month,w=2,l=1)	Seperti kalender cetak. Bulan (tahun, bulan, w, l).
calendar.setfirstweekday(weekday)	Mengatur hari pertama setiap minggu sampai hari kerja kode hari kerja. Kode hari kerja adalah 0 (Senin) sampai 6 (Minggu).
calendar.timegm(tupletime)	Kebalikan dari time.gmtime: menerima waktu instan dalam bentuk tupel waktu dan mengembalikan detik yang sama seperti jumlah floating-point dalam hitungan detik sejak zaman.
calendar.weekday(year,month,day)	Mengembalikan kode hari kerja untuk tanggal yang ditentukan. Kode hari kerja adalah 0 (Senin) sampai 6 (Minggu); Bulan adalah 1 (Januari) sampai 12 (Desember).

1.19 Fungsi Python

Fungsi adalah blok kode terorganisir dan dapat digunakan kembali yang digunakan untuk melakukan sebuah tindakan/action. Fungsi memberikan modularitas yang lebih baik untuk aplikasi Anda dan tingkat penggunaan kode yang tinggi.

1.19.1 Mendefinisikan Fungsi Python

Anda dapat menentukan fungsi untuk menyediakan fungsionalitas yang dibutuhkan. Berikut adalah aturan sederhana untuk mendefinisikan fungsi dengan Python.

- Fungsi blok dimulai dengan def kata kunci diikuti oleh nama fungsi dan tanda kurung (()).
- Setiap parameter masukan atau argumen harus ditempatkan di dalam tanda kurung ini. Anda juga dapat menentukan parameter di dalam tanda kurung ini.
- Pernyataan pertama dari sebuah fungsi dapat berupa pernyataan opsional - string dokumentasi fungsi atau docstring.
- Blok kode dalam setiap fungsi dimulai dengan titik dua (:) dan indentasi.
- Pernyataan kembali [ekspresi] keluar dari sebuah fungsi, secara opsional menyampaikan kembali ekspresi ke pemanggil. Pernyataan pengembalian tanpa argumen sama dengan return None.

Contoh fungsi

```
1 def printnama(nama):  
2     print(nama)  
3     return
```

1.20 Modul Python

Modul memungkinkan Anda mengatur kode Python secara logis. Mengelompokkan kode terkait ke dalam modul membuat kode lebih mudah dipahami dan digunakan. Modul adalah objek Python dengan atribut yang diberi nama yang bisa Anda bind dan dijadikan referensi.

Secara sederhana modul adalah file yang terdiri dari kode Python. Modul dapat mendefinisikan fungsi, kelas dan variabel. Modul juga bisa menyertakan kode yang bisa dijalankan “runable”.

Berikut adalah contoh modul sederhana pada Python :

```
1 def printnama(nama):  
2     print(f"Halo: {nama}")  
3     return
```

1.20.1 Import Statement

Anda dapat menggunakan file sumber Python apapun sebagai modul dengan mengeksekusi pernyataan impor di file sumber Python lainnya. Impornya memiliki sintaks berikut.

Ketika interpreter menemukan sebuah pernyataan import, ia mengimpor modul jika modul tersebut ada di jalur pencarian. Jalur pencarian adalah daftar direktori

yang ditafsirkan juru bahasa sebelum mengimpor modul. Misalnya, untuk mengimpor modul hello.py, Anda perlu meletakkan perintah berikut di bagian atas script.

```
1 # Import module support
2 import module_nama
3
4 # Anda bisa memanggil fungsi defined sebagai berikut
5 module_nama.printnama("Andy")
```

1.21 File I/O Python

Disini kita akan belajar semua fungsi dasar I/O yang tersedia pada Python 3. Jika Anda ingin mempelajari lebih detail, lihat dokumentasi standar Python.

1.21.1 Print

Cara termudah untuk menghasilkan output adalah dengan menggunakan pernyataan cetak di mana Anda bisa melewati nol atau lebih banyak ekspresi yang dipisahkan dengan koma. Fungsi ini mengubah ekspresi yang Anda berikan ke string dan menulis hasilnya ke output standar sebagai berikut :

```
1 print("Python adalah bahasa pemrograman yang simpel")
```

1.21.2 Membaca Input Keyboard

Python 2 memiliki dua fungsi built-in untuk membaca data dari input standar, yang secara default berasal dari keyboard. Fungsi ini adalah input() dan raw_input()

Dengan Python 3, fungsi raw_input() tidak digunakan lagi. Selain itu, input() berfungsi membaca data dari keyboard sebagai string, terlepas dari apakah itu tertutup dengan tanda kutip (" atau ") atau tidak.

1.21.3 Fungsi Input Python

Fungsi input([prompt]) setara dengan raw_input, kecuali mengasumsikan bahwa input adalah ekspresi Python yang valid dan mengembalikan hasil yang dievaluasi ke Anda.

```
1 >>> x = input("something:")
2 something:10
3
4 >>> x
5 '10'
6
7 >>> x = input("something:")
8 something:'10' #memasukkan kembali variabel x dengan data string
9
10 >>> x
11 "'10'"
```

1.22 Exception

Python menyediakan dua fitur yang sangat penting untuk menangani kesalahan tak terduga dalam program Python Anda dan menambahkan kemampuan debugging di dalamnya.

- Exception Handling

- Assertions Exception adalah sebuah peristiwa, yang terjadi selama pelaksanaan program yang mengganggu aliran normal instruksi program. Secara umum, ketika skrip Python menemukan situasi yang tidak dapat diatasi, hal itu menimbulkan pengecualian. Exception adalah objek Python yang mewakili kesalahan.

Ketika skrip Python menimbulkan Exception, ia harus menangani Exception begitu saja sehingga berhenti dan berhenti.

Nama	Penjelasan
Exception	Kelas dasar untuk semua pengecualian / exception
StopIteration	Dibesarkan ketika metode (iterator) berikutnya dari iterator tidak mengarah ke objek apa pun.
SystemExit	Dibesarkan oleh fungsi sys.exit () .
StandardError	Kelas dasar untuk semua pengecualian built-in kecuali StopIteration dan SystemExit.
ArithmetricError	Kelas dasar untuk semua kesalahan yang terjadi untuk perhitungan numerik.
OverflowError	Dibesarkan saat perhitungan melebihi batas maksimum untuk tipe numerik.
FloatingPointError	Dibesarkan saat perhitungan floating point gagal.
ZeroDivisonError	Dibesarkan saat pembagian atau modulo nol dilakukan untuk semua tipe numerik.
AssertionError	Dibesarkan jika terjadi kegagalan pernyataan Assert.
AttributeError	Dibesarkan jika terjadi kegagalan referensi atribut atau penugasan.
EOFError	Dibesarkan bila tidak ada input dari fungsi raw_input () atau input () dan akhir file tercapai.
ImportError	Dibesarkan saat sebuah pernyataan impor gagal.
KeyboardInterrupt	Dibesarkan saat pengguna menyela eksekusi program, biasanya dengan menekan Ctrl + c .
LookupError	Kelas dasar untuk semua kesalahan pencarian.
IndexError	Dibesarkan saat sebuah indeks tidak ditemukan secara berurutan.
KeyError	Dibesarkan saat kunci yang ditentukan tidak ditemukan dalam kamus.
NameError	Dibesarkan saat pengenal tidak ditemukan di namespace lokal atau global.
UnboundLocalError	Dibesarkan saat mencoba mengakses variabel lokal dalam suatu fungsi atau metode namun tidak ada nilai yang ditugaskan padanya.
EnvironmentError	Kelas dasar untuk semua pengecualian yang terjadi di luar lingkungan Python.
IOError	Dibesarkan saat operasi input / output gagal, seperti pernyataan cetak atau fungsi open () saat mencoba membuka file yang tidak ada.
OSError	Dibangkitkan untuk kesalahan terkait sistem operasi.

SyntaxError	Dibesarkan saat ada kesalahan dengan sintaks Python.
IndentationError	Dibesarkan saat indentasi tidak ditentukan dengan benar.
SystemError	Dibesarkan saat penafsir menemukan masalah internal, namun bila kesalahan ini ditemui juru bahasa Python tidak keluar.
SystemExit	Dibesarkan saat juru bahasa Python berhenti dengan menggunakan fungsi sys.exit (). Jika tidak ditangani dalam kode, menyebabkan penafsir untuk keluar.
TypeError	Dibesarkan saat operasi atau fungsi dicoba yang tidak valid untuk tipe data yang ditentukan.
ValueError	Dibesarkan ketika fungsi bawaan untuk tipe data memiliki jenis argumen yang valid, namun argumen tersebut memiliki nilai yang tidak valid yang ditentukan.
RuntimeError	Dibesarkan saat kesalahan yang dihasilkan tidak termasuk dalam kategori apa pun.
NotImplementedError	Dibesarkan ketika metode abstrak yang perlu diimplementasikan di kelas warisan sebenarnya tidak dilaksanakan.

1.23 Object & Class Python

Python telah menjadi bahasa berorientasi objek sejak bahasa Python sendiri dibuat. Untuk membuat dan menggunakan kelas dan objek pada Python benar-benar mudah. Pada tutorial ini Anda akan dibantu untuk menjadi ahli dalam penggunaan pemrograman berorientasi objek Python.

Jika Anda tidak memiliki pengalaman sebelumnya dengan pemrograman berorientasi objek (OOP), Anda mempelajarinya terlebih dahulu agar Anda dapat memahami konsep dasarnya.

Jika memang sudah mengerti konsep dasar OOP berikut ini adalah pengenalan dari Object-Oriented Programming (OOP) untuk membantu Anda.

1.23.1 Istilah Dalam OOP

Istilah	Penjelasan
Class	Prototipe yang ditentukan pengguna untuk objek yang mendefinisikan seperangkat atribut yang menjadi ciri objek kelas apa pun. Atribut adalah data anggota (variabel kelas dan variabel contoh) dan metode, diakses melalui notasi titik.
Class variable	Sebuah variabel yang dibagi oleh semua contoh kelas. Variabel kelas didefinisikan dalam kelas tapi di luar metode kelas manapun. Variabel kelas tidak digunakan sesering variabel contoh.
Data member	Variabel kelas atau variabel contoh yang menyimpan data yang terkait dengan kelas dan objeknya.
Function overloading	Penugasan lebih dari satu perilaku ke fungsi tertentu. Operasi yang dilakukan bervariasi menurut jenis objek atau argumen yang terlibat.
Instance variable	Variabel yang didefinisikan di dalam sebuah metode dan hanya dimiliki oleh instance kelas saat ini.
Inheritance	Pengalihan karakteristik kelas ke kelas lain yang berasal darinya.
Instance	Objek individu dari kelas tertentu. Objek obj yang termasuk dalam Lingkaran kelas, misalnya, adalah turunan dari Lingkaran kelas.
Instantiation	Penciptaan sebuah instance dari sebuah kelas.
Method	Jenis fungsi khusus yang didefinisikan dalam definisi kelas.
Object	Contoh unik dari struktur data yang didefinisikan oleh kelasnya. Objek terdiri dari kedua anggota data (variabel kelas dan variabel contoh) dan metode.
Operator overloading	Penugasan lebih dari satu fungsi ke operator tertentu.

1.23.2 Membuat Class Python

Statement class digunakan untuk membuat definisi kelas baru. Nama kelas segera mengikuti kata kunci diikuti oleh titik dua sebagai berikut.

```
1 class ClassName: 'Optional class documentation string' class_suite
```

Dibawah ini adalah contoh cara membuat class dan penggunaanya :

```

1 class Employee:
2     empCount = 0
3
4     def __init__(self, name, salary):
5         self.name = name
6         self.salary = salary
7         Employee.empCount += 1
8
9     def displayCount(self):
10        print("Total Employee %d" % Employee.empCount)
11
12    def displayEmployee(self):
13        print("Name : ", self.name, ", Salary: ", self.salary)

```

1.23.3 Membuat Instance Objects

Untuk membuat instances kelas, Anda memanggil class menggunakan nama class dan meneruskan argumen apa pun yang metode init terima.

```

1 # variabel emp1 akan diisi oleh object karyawan bernama Zara dan
2 # salary 2000
3 emp1 = Employee("Zara", 2000)
4
5 # variabel emp2 akan diisi oleh object karyawan bernama Manni dan
6 # salary 5000
7 emp2 = Employee("Manni", 5000)

```

1.23.4 Mengakses Atribut

Anda mengakses atribut objek menggunakan dot operator dengan objek. Variabel kelas akan diakses dengan menggunakan nama kelas sebagai berikut :

```

1 emp1.displayEmployee()
2 emp2.displayEmployee()
3 print ("Total Employee %d" % Employee.empCount)

```

Contoh lengkapnya, silahkan lihat kode dibawah ini.

```

1 class Employee:
2     'Common base class for all employees'
3     empCount = 0
4
5     def __init__(self, name, salary):
6         self.name = name
7         self.salary = salary
8         Employee.empCount += 1
9
10    def displayCount(self):
11        print ("Total Employee %d" % Employee.empCount)
12
13    def displayEmployee(self):
14        print ("Name : ", self.name, ", Salary: ", self.salary)
15
16

```

```
17 # variabel emp1 akan diisi oleh object karyawan bernama Zara dan
    salary 2000
18 emp1 = Employee("Zara", 2000)
19
20 # variabel emp2 akan diisi oleh object karyawan bernama Manni dan
    salary 5000
21 emp2 = Employee("Manni", 5000)
22
23 emp1.displayEmployee()
24 emp2.displayEmployee()
25
26 print("Total Employee %d" % Employee.empCount)
```

BAB 2

GIT & GITHUB



Gambar 2.1 Git dan Github

Seiring berjalananya waktu, setiap pekerjaan manusia selalu mengalami pembaharuan untuk lebih efisien. Tujuannya tidak lain adalah menyederhanakan prosedur pekerjaan yang cenderung berbelit-belit. Begitupun dengan programmer dalam menyusun kode script yang rumit dan panjang membutuhkan kerja tim. Seperti halnya Git dan GitHub muncul untuk membantu pekerjaan tim programmer dalam menyusun kode script. Git dan GitHub merupakan dua platform yang didirikan oleh satu perusahaan dengan tujuan sama serta fitur yang berbeda. Namun kedua platform ini sangat membantu pekerjaan programmer dalam menyusun kode script secara tim. Seluruh pekerjaan juga dapat dipantau dan dievaluasi dengan mudah karena penggunaan kontrol sistem atau bisa disebut Version Control System (VCS). Saat ini, hampir menjadi kewajiban bagi programmer untuk menggunakan kedua platform ini dalam pekerjaannya. Lebih jelasnya, ketahui terlebih dahulu pengertian dari kedua platform berikut ini.

2.1 Version Control System (VCS)



Gambar 2.2 Version Control System

Version Control System (VCS) adalah sebuah sistem yang melakukan source code management (SCM) untuk mengelola perubahan di setiap dokumen, program komputer, website, dan kumpulan pemrograman lainnya.

- ▣ Skripsi (Baru Mulai)
- ▣ Skripsi (Revisi Ke-1)
- ▣ Skripsi (Revisi Ke-2)
- ▣ Skripsi (Revisi Ke-3)
- ▣ Skripsi (Revisi Ke-4)
- ▣ Skripsi (Revisi Ke-5)
- ▣ Skripsi (Revisi Ke-6)
- ▣ Skripsi (Revisi Ke-7)
- ▣ Skripsi (Revisi Ke-8)
- ▣ Skripsi (Revisi Ke-9)
- ▣ Skripsi (Revisi Ke-10)
- ▣ Skripsi (Revisi Ke-11)
- ▣ Skripsi (Udah jadi)
- ▣ Skripsi Alhamdulillah Lolos
- ▣ Skripsi Alhamdulillah Wis Udah

Gambar 2.3 Sebelum VCS

Lebih jelasnya kamu bisa merasakan contoh kasus yang biasa dilakukan oleh seorang mahasiswa dalam mengerjakan skripsinya. Setiap melakukan revisi, file yang telah lalu tidak akan dibuang dan akan disimpan dengan nama yang berbeda. Sedangkan yang terbaru akan disimpan dengan nama; misal “Skripsi (Revisi Ke-2)”. Kegiatan tersebut akan dilakukan secara terus menerus hingga dalam satu folder skripsi terdapat file Ms. Word dengan kuantitas yang banyak. Tujuan tersebut tidak lain adalah untuk menyimpan history pekerjaan mahasiswa. Hingga akhirnya file

terakhir selesai dengan nama “Skripsi Alhamdulillah Wis Udah”. Konsep pekerjaan tersebut dianggap tidak efisien oleh banyak developer karena kapasitas penyimpanan akan membengkak. VCS disini berfungsi untuk membantu penyimpanan berupa history tanpa menyimpan file baru, yang tersimpan hanya perubahan data. Sehingga kapasitas penyimpanan file menjadi ringan.



Gambar 2.4 Sesudah VCS

Seperti halnya pada gambar diatas, setiap perubahan data secara manual akan menghasilkan lebih banyak file. Sedangkan pada VCS mengusung konsep menyimpan rekaman perubahan dengan satu file saja.

2.2 Git



Gambar 2.5 Git

Git merupakan software berbasis Version Control System (VCS) yang bertugas untuk mencatat perubahan seluruh file atau repository suatu project. Developer software biasa menggunakan Git untuk distributed revision (VCS terdistribusi), hal ini bertujuan untuk menyimpan database tidak hanya ke satu tempat. Namun semua orang yang terlibat dalam penyusunan kode dapat menyimpan database ini. Prosedur yang diterapkan ini dapat membantu antar divisi project untuk memantau dan

menghubungkan (merge) antar ekstensi yang berbeda dengan mudah. Sehingga aplikasi yang dibuat oleh sebuah tim project dapat berfungsi tanpa menghubungkan secara manual. Terdapat istilah commit pada Git yang berfungsi untuk menyimpan riwayat perubahan data pada file. Melalui commit, developer dapat kembali ke source code sebelumnya dengan istilah checkout. Untuk mengoperasikan Git, kamu perlu menginstall software terlebih dahulu sehingga pekerjaan ini dapat dilakukan secara offline (tidak terkoneksi internet). Software ini juga tersedia secara gratis melalui web unduhannya di Git Downloading.

2.3 Github



Gambar 2.6 Github

GitHub merupakan layanan cloud yang berguna untuk menyimpan dan mengelola sebuah project yang dinamakan repository (repo git). Cara kerja pada GitHub harus terkoneksi pada internet sehingga tidak perlu meng-install sebuah software ke dalam perangkat keras. Hal ini memberikan keringanan penyimpanan komputer yang kita gunakan karena file project tersimpan oleh cloud GitHub. Konsep kerja GitHub pada dasarnya sama dengan Git yaitu dapat menulis source code secara individu atau tim. User interface yang tersedia pada GitHub lebih menarik dan mudah dipahami oleh pengguna awal. Pekerjaan secara tim, pengguna juga bisa melihat siapa penulis kode dan tanggal berapa kode tersebut dibuat. Terdapat fitur lain pada GitHub yaitu kita dapat membaca berbagai blog dan feed yang dibuat oleh sesama pengguna. Hal ini dimanfaatkan oleh pengguna seluruh dunia untuk saling berbagi ide pemrograman dan berdiskusi dalam menyelesaikan masalah. Tentunya postingan yang ada pada GitHub berkaitan dengan pemrograman. Sehingga GitHub telah menjadi forum diskusi para programmer seperti halnya media sosial. Semenjak GitHub diakuisisi oleh Microsoft di tahun 2018, platform ini berkembang semakin baik dan unggul. Sehingga mayoritas programmer lebih mengenal GitHub dalam program VCS dari pada pesaingnya seperti GitLab dan Atlassian BitBucket.

2.4 Perbedaan Git & Github

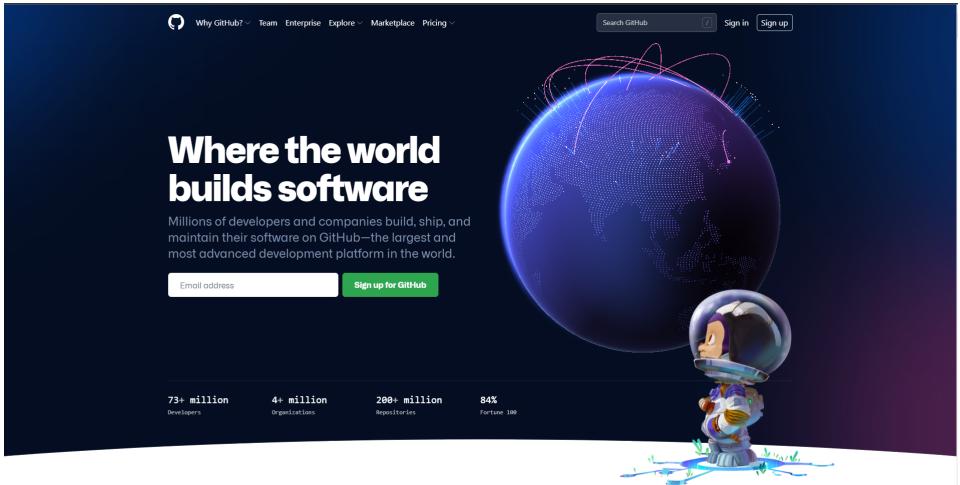
Perbedaan antara Git dan GitHub sangat unik dan memiliki keunggulan masing-masing. Berikut ini perbedaan dari kedua platform tersebut.

Git	Github
1. Meng-install software di penyimpanan lokal	1. Host melalui layanan cloud
2. Dikelola oleh The Linux Foundation	2. Diakuisisi oleh Microsoft pada 2018
3. Berfokus pada version control dan code sharing	3. Berfokus pada source code hosting terpusat
4. Akses secara offline	4. Akses secara online
5. Tidak menggunakan fitur user management	5. Menggunakan user management
6. Menyediakan desktop interface bernama “Git GUI”	6. Menggunakan nama desktop interface “GitHub Desktop”
7. Bersaing dengan Mercurial, Subversion, IBM, Rational Team, Concert, dan ClearCase	7. Bersaing dengan GitLab dan Atlassian BitBucket
8. Open sourced licensed	8. Pilihan bagi pengguna gratis dan pengguna berbayar

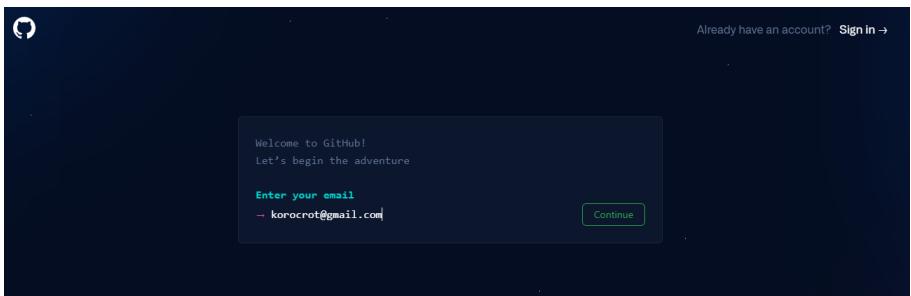
2.5 Tutorial Membuat Akun Github

Disini saya akan membahas bagaimana cara mudah membuat akun Github beserta gambar.

1. kunjungi website <https://github.com>, lalu klik sign up untuk mendaftar

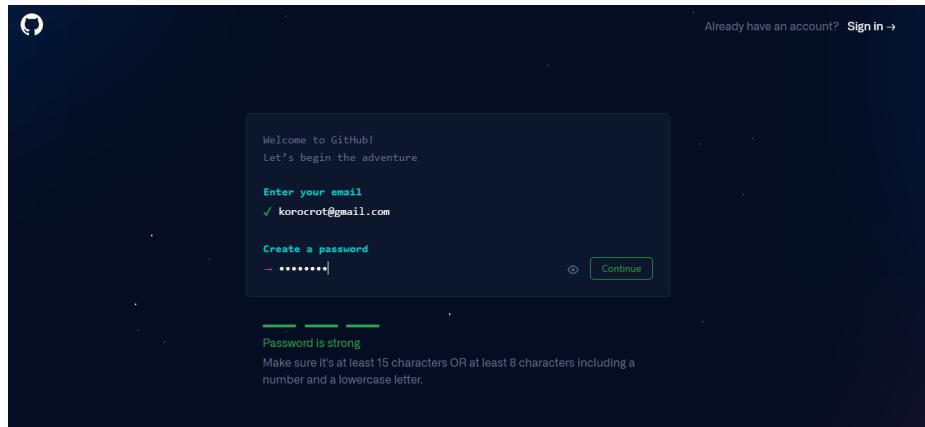


Gambar 2.7 Tutorial Akun Github: Step 1



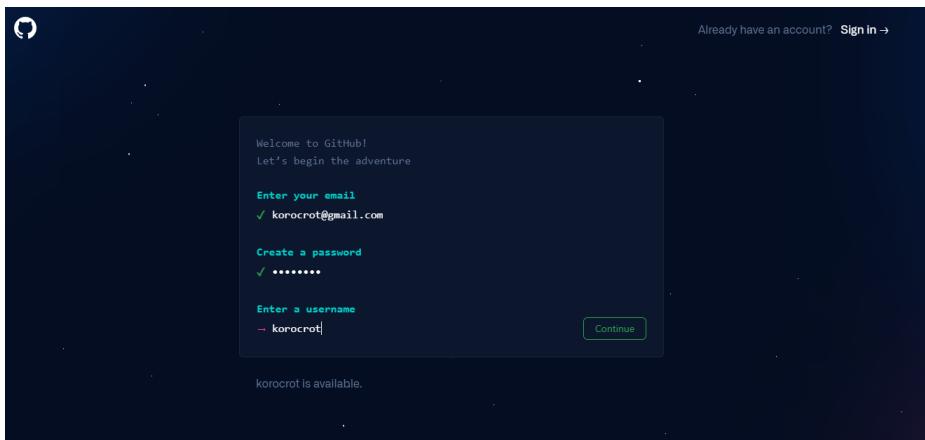
Gambar 2.8 Tutorial Akun Github: Step 2

3. lalu masukkan password yang dengan kombinasi yang direkomendasikan oleh Github



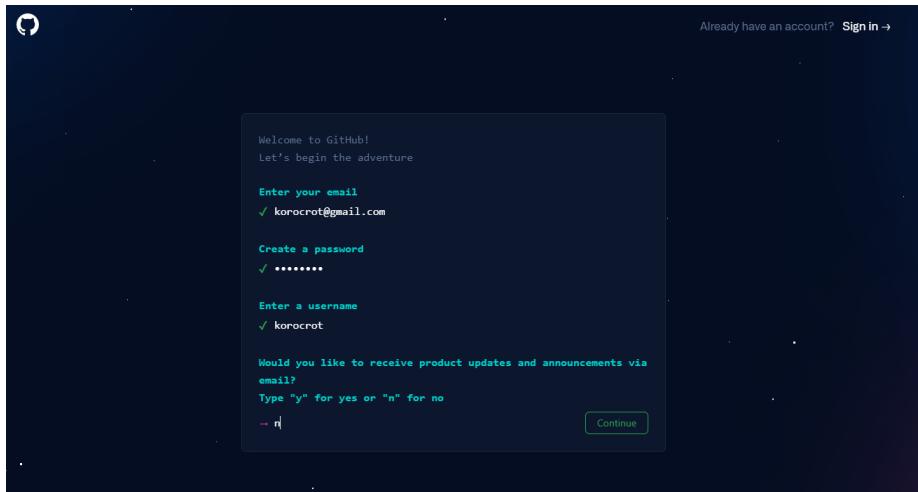
Gambar 2.9 Tutorial Akun Github: Step 3

- masukkan username yang unik, jika username tidak tersedia coba beberapa kombinasi seperti angka dan huruf



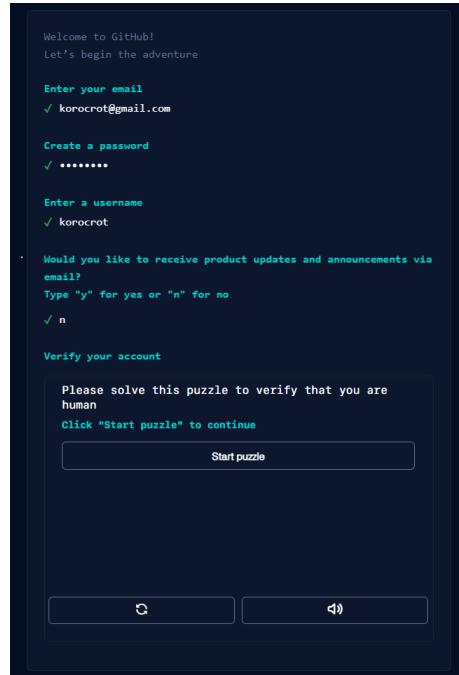
Gambar 2.10 Tutorial Akun Github: Step 4

- lalu setelah itu akan muncul penawaran dari Github untuk notifikasi product dan announcement, saya disini memilih "n"



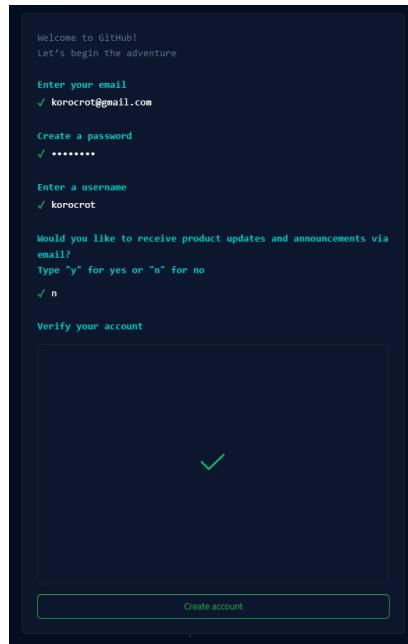
Gambar 2.11 Tutorial Akun Github: Step 5

6. setelah itu akan diminta untuk menyelesaikan puzzle untuk verifikasi bahwa yang mendaftar adalah manusia, selesaikan puzzle berdasarkan perintah yang diberikan



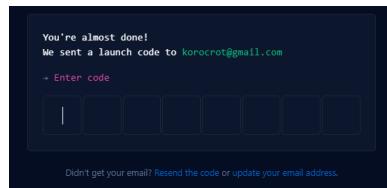
Gambar 2.12 Tutorial Akun Github: Step 6

7. jika berhasil bisa langsung klik **Create account**



Gambar 2.13 Tutorial Akun Github: Step 7

8. lalu Github akan mengirimkan kode verifikasi ke email yang didaftarkan



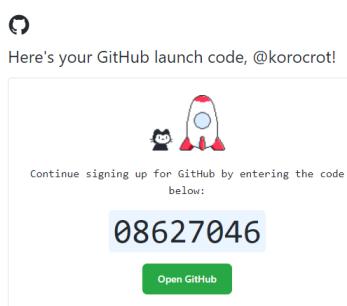
Gambar 2.14 Tutorial Akun Github: Step 8

9. copy dan paste code yang ada di email ke kotak **Enter code** git halaman github,
Note: jangan menutup tab Github

Your GitHub launch code [Kotak Masuk](#)

GitHub <noreply@github.com>
kepada saya ▾

01.36 (0 menit yang lalu)



Once completed, you can start using all of GitHub's features to explore, build, and share projects.

Not able to enter the code? Paste the following link into your browser:

https://github.com/users/korocrot/emails/190246493/confirm_verification/08627046?via_launch_code_email=true

Gambar 2.15 Tutorial Akun Github: Step 9

10. jika kode yang dimasukkan benar maka kamu akan dialihkan ke halaman dashboard github

Create your first project
Ready to start building? Create a repository for your first project or fork an existing repository to keep contributing to it.

[Create repository](#) [Import repository](#)

Recent activity
When you see actions across GitHub, we'll provide links to that activity here.

Learn Git and GitHub without any code!
Using the Hello World guide you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#) [Start a project](#)

All activity
Introduce yourself
The easiest way to introduce yourself on GitHub is by creating a README about you. You can start here:

Korocrot / README.md

- 1 - I'd like to contribute to ...
- 2 - I'm currently learning ...
- 3 - I'm looking to collaborate on ...
- 4 - How to reach me ...
- 5 -

[Dismiss this](#) [Continue](#)

Discover interesting projects and people to populate your personal news feed.
Your news feed helps you keep up with recent activity on repositories you watch or star and people you follow.

[Explore GitHub](#)

ProTip! The feed shows events from people you follow and repositories you watch or star.

Subscribe to your news feed

© 2022 GitHub, Inc.

Help About Shop Contact GitHub Pricing API Training Status Security Terms Privacy Docs

Gambar 2.16 Tutorial Akun Github: Step 10

2.6 Tutorial Install Git

Disini saya akan membahas bagaimana cara mudah menginstall Git pada Windows 10 & Linux beserta gambar.

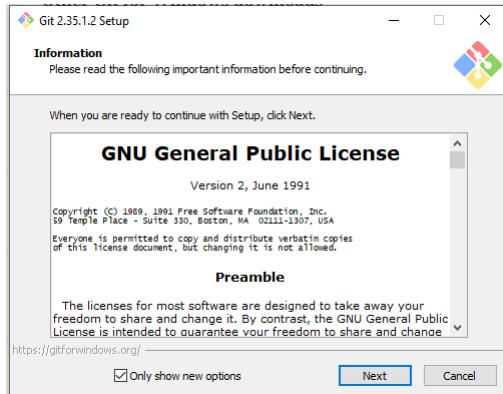
2.6.1 Windows 10

1. kunjungi website git-scm <https://git-scm.com/>
2. lalu klik menu **Download**
3. akan muncul seperti gambar dibawah, lalu klik **Download for Windows**



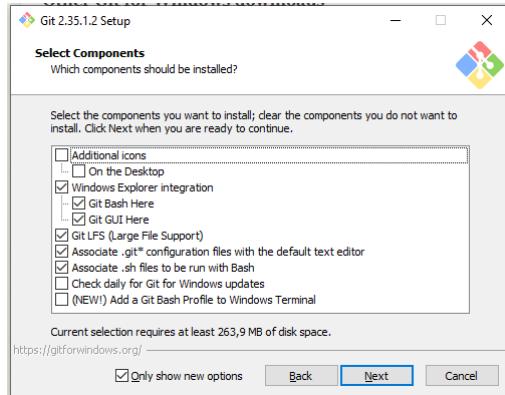
Gambar 2.17 Tutorial Install Git-scm: Step 1

4. setelah berhasil download, klik (2x) dan akan muncul menu seperti dibawah ini, klik next



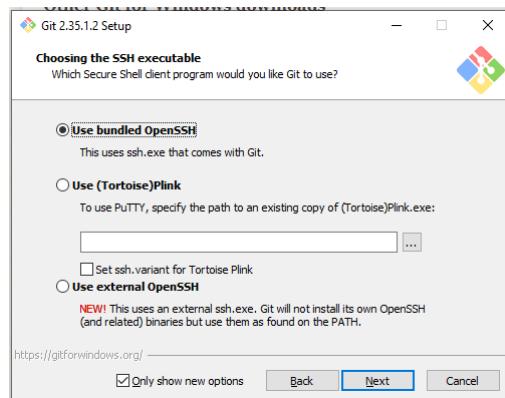
Gambar 2.18 Tutorial Install Git-scm: Step 2

5. klik next



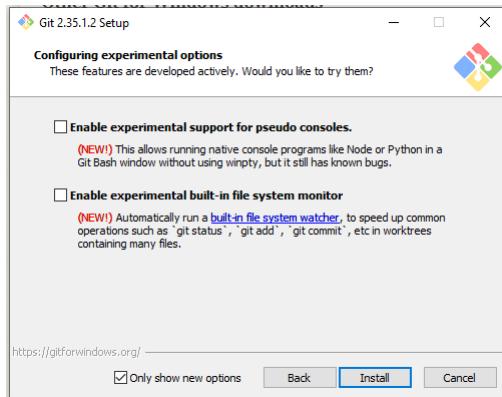
Gambar 2.19 Tutorial Install Git-scm: Step 3

6. klik next



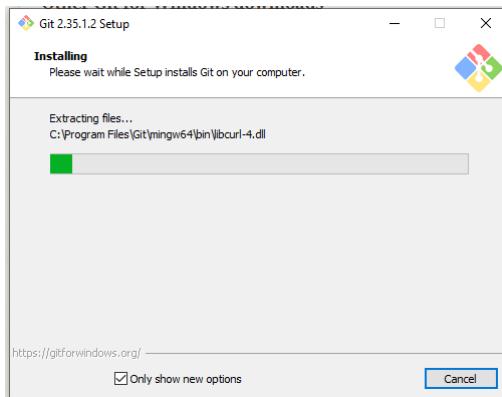
Gambar 2.20 Tutorial Install Git-scm: Step 4

7. klik install



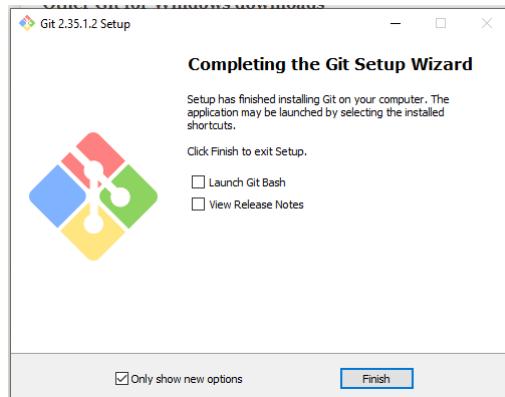
Gambar 2.21 Tutorial Install Git-scm: Step 5

8. tunggu instalasi hingga selesai



Gambar 2.22 Tutorial Install Git-scm: Step 6

9. klik finish



Gambar 2.23 Tutorial Install Git-scm: Step 7

2.6.2 Linux

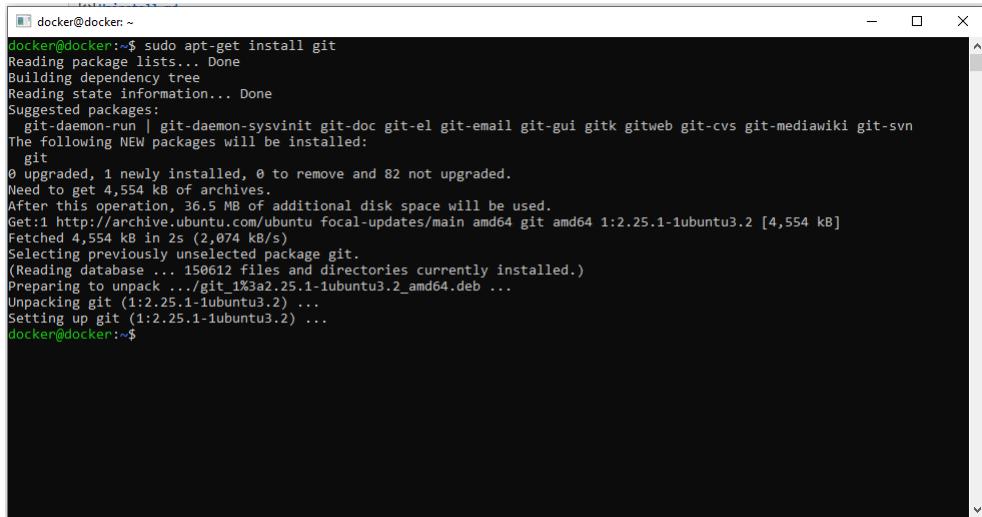
untuk instalasi linux bisa dibilang cukup mudah tidak serumit windows, berikut caranya.

1. buka terminal dan ketikkan **sudo apt-get install git -y** seperti gambar dibawah ini, lalu tekan Enter

A screenshot of a terminal window on a Linux system. The command "sudo apt-get install git" is typed into the terminal. The terminal window has a red border and a black background.

Gambar 2.24 Tutorial Install Git Linux: Step 1

2. tunggu hingga instalasi selesai seperti gambar dibawah ini



```
docker@docker:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git
0 upgraded, 1 newly installed, 0 to remove and 82 not upgraded.
Need to get 4,554 kB of archives.
After this operation, 36.5 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 git amd64 1:2.25.1-1ubuntu3.2 [4,554 kB]
Fetched 4,554 kB in 2s (2,074 kB/s)
Selecting previously unselected package git.
(Reading database ... 150612 files and directories currently installed.)
Preparing to unpack .../git_1%3a2.25.1-1ubuntu3.2_amd64.deb ...
Unpacking git (1:2.25.1-1ubuntu3.2) ...
Setting up git (1:2.25.1-1ubuntu3.2) ...
docker@docker:~$
```

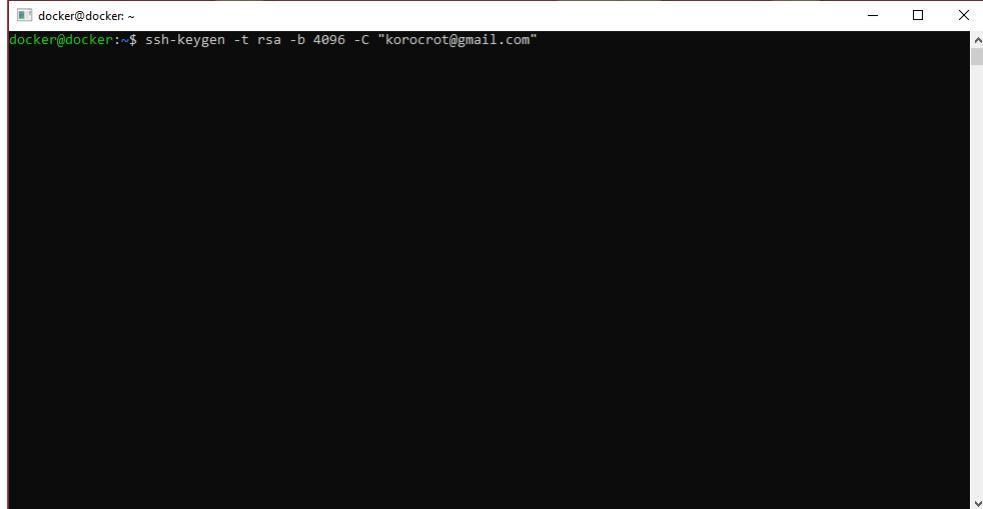
Gambar 2.25 Tutorial Install Git Linux: Step 2

2.7 Generate SSH Key

Pada section ini saya akan memberikan tutorial cara generate SSH key yang nantinya SSH key tersebut digunakan untuk mengakses website Github.

1. pertama buka git bash terminal, lalu ketikkan perintah dibawah, lalu tekan Enter

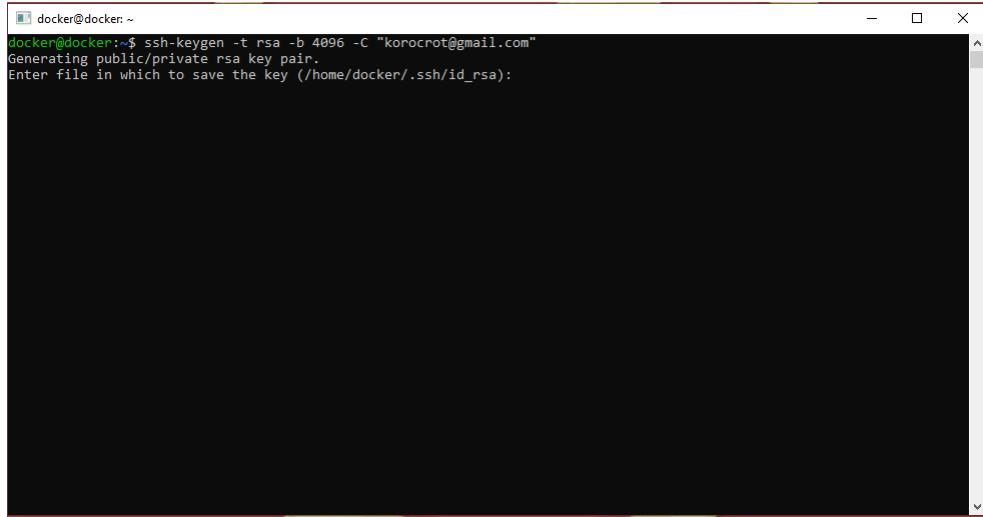
```
ssh-keygen -t rsa -b 4096 -C "email@anda.com"
```



```
docker@docker:~$ ssh-keygen -t rsa -b 4096 -C "korocrot@gmail.com"
```

Gambar 2.26 Tutorial Generate SSH Key: Step 1

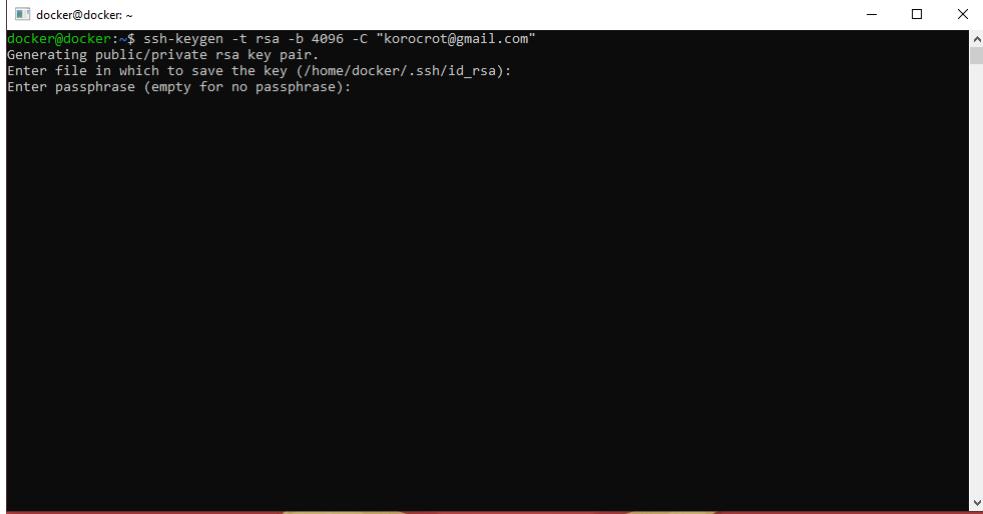
2. setelah itu akan muncul informasi isian seperti gambar dibawah ini, tekan Enter saja



```
docker@docker:~$ ssh-keygen -t rsa -b 4096 -C "korocrot@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/docker/.ssh/id_rsa):
```

Gambar 2.27 Tutorial Generate SSH Key: Step 2

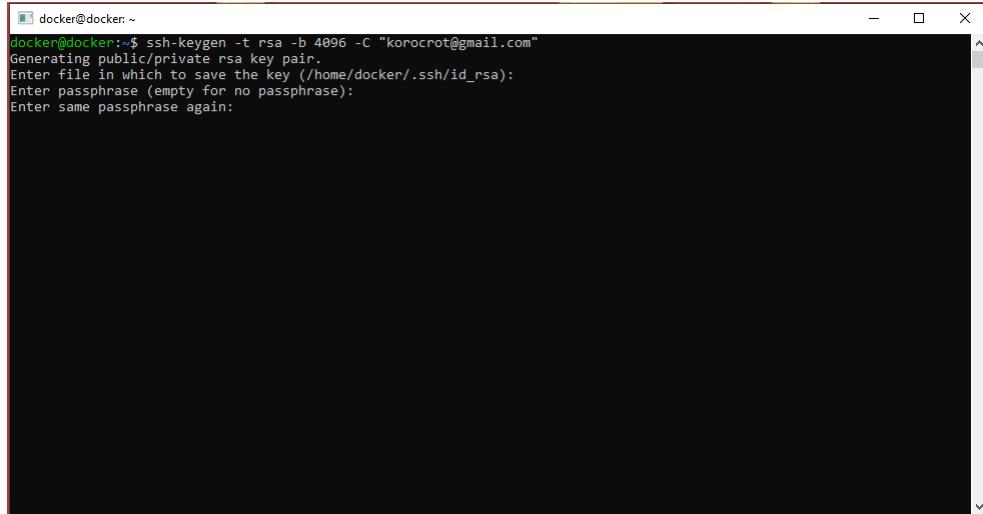
3. akan muncul lagi isian berikutnya, tekan Enter saja



```
docker@docker:~$ ssh-keygen -t rsa -b 4096 -C "korocrot@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/docker/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
```

Gambar 2.28 Tutorial Generate SSH Key: Step 3

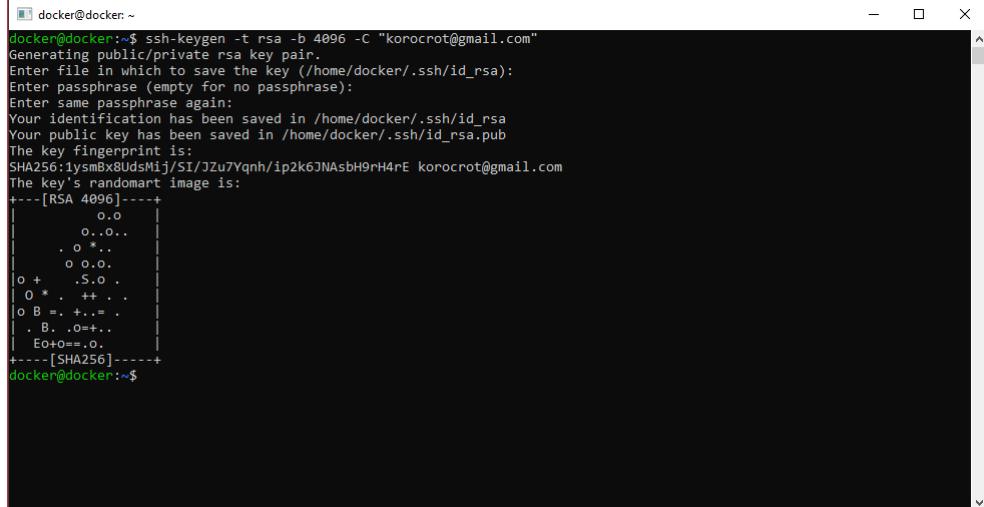
4. akan muncul lagi isian, tekan Enter saja



```
docker@docker:~$ ssh-keygen -t rsa -b 4096 -C "korocrot@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/docker/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Gambar 2.29 Tutorial Generate SSH Key: Step 4

5. jika sudah selesai akan muncul seperti gambar dibawah ini



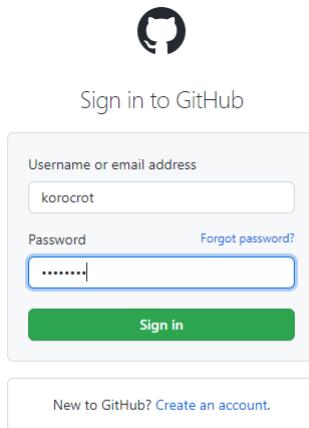
```
docker@docker:~$ ssh-keygen -t rsa -b 4096 -C "korocrot@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/docker/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/docker/.ssh/id_rsa
Your public key has been saved in /home/docker/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:1ysmBxBUdshMij/SI/JZu7Yqnh/ip2k6JNAsbH9rH4rE korocrot@gmail.com
The key's randomart image is:
+---[RSA 4096]---+
|          o.o   |
|         o.o..  |
|        .o *..  |
|       o o.o.  |
|      o +   .S.o . |
|     O * .  ++ .. |
|    o B =. +..= . |
|   . B. .-=+.  |
|  Eo+=...o.  |
+---[SHA256]---+
docker@docker:~$
```

Gambar 2.30 Tutorial Generate SSH Key: Step 5

2.8 Memasang SSH Key di Github

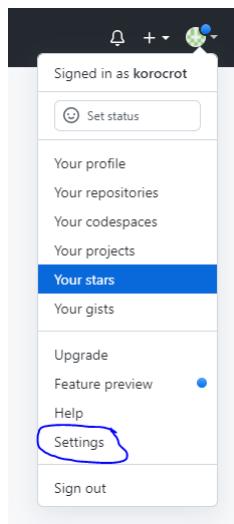
Section ini akan memberikan tutorial bagaimana memasang SSH Key di Github agar komputer bisa terkoneksi dengan akun Github

1. kunjungi <https://github.com> dan login menggunakan akun yang sudah didaftarkan sebelumnya



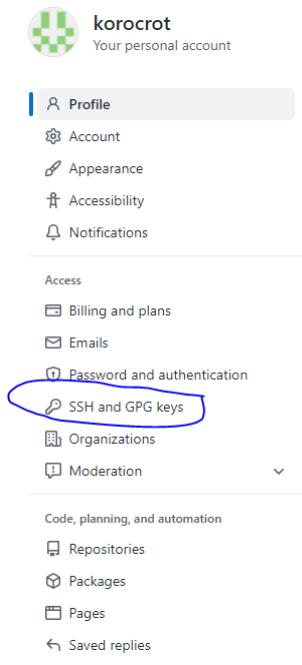
Gambar 2.31 Tutorial Memasang SSH Key: Step 1

2. jika sudah login klik profile dan klik settings



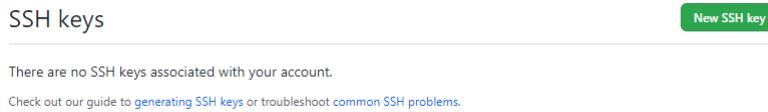
Gambar 2.32 Tutorial Memasang SSH Key: Step 2

3. setelah itu klik SSH and GPG Keys



Gambar 2.33 Tutorial Memasang SSH Key: Step 3

4. lalu klik new ssh keys



Gambar 2.34 Tutorial Memasang SSH Key: Step 4

5. lalu buka git bash anda dan ketikkan perintah dibawah, lalu enter, dan copy output yang dikeluarkan

```
1 cat .ssh/id_rsa.pub
```

```
[ docker@docker: ~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQACQDT0GewJjKQT6D/hjZFYTtW6/kxbUotOCjk2G/8bQK18Y8eCqW2YZHvdnNutbvqWdyBGedpCdy+G8fxvXVDI
yGi+Ct8rm3jdtpVp+9hYj0M7htCMN4tXPxnUaJghBezwxvKYE3c9gHMPtyoPyIMb5edhc6Tee+/SGK8k0E64JWhd1cP8vBaJhyw5r1FnVxTSMwx6L1xKx
i3pu9/xGdlyGJDof0scMp2RL2XCZY9hkjInZ9D892mpTBotWdvB+fjjiChTrUvs+Yvvu7REM1faHPaq84XYmMCeb72BwNx+p6Ty2H7LfY4VPMpuVZjvp
w/jxnNqi++8GhzEunhQ1Tqcr3tAUvYc1Y7kxmOpSTu6Ytovx8N8231ADF+zEqXYvxq8QtBcvNWQ/0opRy+cj6T+9VPNjNpYAoxsy16ozhED08xt6j9+LM
Vs1tpuDVnZz9o9EMEsL+yLTzrHtd/jYgORQkU9jKgdBxjK0KqXIAe6utUL7TrqtZPVZDwWxnhFo+Hdcf7DNi03udNEYe4MGivJSnADT2IQj03ZYH/OgaQw
AHlCaML1VrLvje=jKnDHCKM1paCfoN9LISWs66ppelNbdstZG/WY1M6KhwJZdlq163DrV2Lfup98c+bPYhvUVIL2ZarejH3nrFQxxiiL6I4j9Jr8fc/N8
nu== korocrot@gmail.com
docker@docker: ~$
```

Gambar 2.35 Tutorial Memasang SSH Key: Step 5

- setelah itu isi title dengan isian bebas, lalu paste yang sudah dicopy di bagian form bawah seperti gambar dibawah ini, lalu klik add ssh key

SSH keys / Add new

Title

docker

Key

```
ssh-rsa
AAAAAB3NzaC1yc2EAAAQABAAQACQDT0GewJjKQT6D/hjZFYTtW6/kxbUotOCjk2G/8bQK18Y8eCqW2YZHvdnNutbvq
WdyBGedpCdy+G8fxvXVDI
yGi+Ct8rm3jdtpVp+9hYj0M7htCMN4tXPxnUaJghBezwxvKYE3c9gHMPtyoPyIMb5edhc6Tee+/SGK8k0E64JWhd1cP8vBaJhyw5r1FnVxTSMwx6L1xKx
i3pu9/xGdlyGJDof0scMp2RL2XCZY9hkjInZ9D892mpTBotWdvB+fjjiChTrUvs+Yvvu7REM1faHPaq84XYmMCeb72BwNx+p6Ty2H7LfY4VPMpuVZjvp
w/jxnNqi++8GhzEunhQ1Tqcr3tAUvYc1Y7kxmOpSTu6Ytovx8N8231ADF+zEqXYvxq8QtBcvNWQ/0opRy+cj6T+9VPNjNpYAoxsy16ozhED08xt6j9+LM
Vs1tpuDVnZz9o9EMEsL+yLTzrHtd/jYgORQkU9jKgdBxjK0KqXIAe6utUL7TrqtZPVZDwWxnhFo+Hdcf7DNi03udNEYe4MGivJSnADT2IQj03ZYH/OgaQw
AHlCaML1VrLvje=jKnDHCKM1paCfoN9LISWs66ppelNbdstZG/WY1M6KhwJZdlq163DrV2Lfup98c+bPYhvUVIL2ZarejH3nrFQxxiiL6I4j9Jr8fc/N8
nu== korocrot@gmail.com
```

Add SSH key

Gambar 2.36 Tutorial Memasang SSH Key: Step 6

- jika tampilan seperti dibawah ini maka memasang ssh key sudah selesai

SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

 docker SHA256:1ysmBx8UdsMij/SI/JZu7Yqnh/ip2k6JNAsbH9rH4rE Added on Feb 6, 2022 Never used — Read/write	Delete
--	------------------------

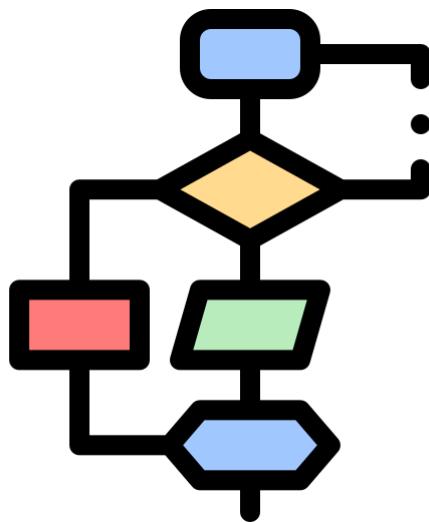
Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

Gambar 2.37 Tutorial Memasang SSH Key: Step 7

BAB 3

ALGORITMA

3.1 Algoritma



Gambar 3.1 Algoritma

sebelum jauh kita bahas tentang yang lainnya, mari kita bahas apa itu Algoritma:

Algoritma adalah sebuah proses atau seperangkat aturan yang harus diikuti dalam operasi pemecahan masalah Sederhananya, algoritma adalah serangkaian proses yang dilakukan secara berurutan untuk menyelesaikan sebuah permasalahan. Algoritma bisa bermacam-macam tergantung kepada siapa yang membuat algoritma tersebut. Namun permasalahannya adalah algoritma mana yang lebih efektif dan efisien? Algoritma sangat banyak digunakan dikehidupan sehari-hari, contohnya adalah bagaimana anda memasak air, bagaimana memasak mie instan, bagaimana menyalakan mesin motor, dan lain sebagainya yang dimana aktivitas tersebut terdapat proses alur yang jika kita ikuti maka akan mencapai suatu tujuan yang ingin kita capai, seperti ingin menyalakan motor contoh yang saya ketahui adalah sebagai berikut,

1. ambil kunci motor
2. masukkan kunci motor pada **main switch**
3. putar hingga ke arah on
4. lalu tekan tombol starter pada motor
5. motor menyala

Kita bisa sebut algoritma diatas adalah algoritma K untuk menyalakan sebuah motor, lalu beberapa tahun kemudian ada algoritma baru yaitu algoritma KL dimana algoritma ini bisa menyelesaikan masalah diatas dengan lebih cepat, kita tidak diperlukan lagi untuk memasukkan kunci motor pada **main switch**, ketika anda sudah dekat dengan motor cukup putar alat yang ada di main switch hingga on dan starter motor, kita contohnya dibawah ini

1. ambil kunci motor
2. putar tuas main switch ke arah on
3. tekan tombol starter pada motor
4. motor menyala

Jika kita bandingkan dua algoritma diatas K dan KL sama-sama menyelesaikan masalah yaitu bagaimana motor bisa menyala, akan tetapi ada 2 perbedaan yaitu di Time dan Space, mari kita lihat perbedaannya, dari segi langkah algoritma K memiliki 5 langkah agar motor bisa menyala, akan tetapi algoritma KL hanya memiliki 4 langkah, jika kita berikan 1 langkah = 1 detik maka algoritma KL bisa menghemat waktu hingga 1 detik. Lalu kita bandingkan dengan Space, algoritma KL ternyata untuk bisa mempercepat 1 detik maka dibutuhkan berat, material, teknologi, dan lainnya lebih tinggi 2x dari algoritma K. Tentunya 2 algoritma bisa menjadi solusi terbaik bagi beberapa masalah yang dihadapi.

3.2 Cases

Untuk memenuhi beberapa petunjuk, ada 3 case untuk merepresentasikan suatu algoritma untuk mencapai atau menyelesaikan masalahnya.

3.2.1 Worst Case

Worst case atau bisa kita terjemahkan ke Bahasa Indonesia adalah kasus terburuk merupakan definisi dari sebuah *class* yang menunjukkan runtime terburuk dari sebuah algoritma. Perancang algoritma biasanya akan memberikan nilai input untuk mencegah sebuah algoritma berjalan secara efisien untuk mengetahui kasus terburuk dari sebuah algoritma.

Contoh kasusnya adalah untuk setiap nilai n tertentu, runtime yang dilakukan oleh suatu algoritma atau program dapat bervariasi terhadap nilai n . Untuk program yang diberikan nilai n , maka waktu eksekusi terburuk adalah waktu eksekusi maksimum dimana maksimum diambil dari instance berukuran n . Worst case adalah kasus dimana algoritma sering dijumpai dan sangat mudah untuk dianalisa. Worst case juga bisa menjadi penjelasan mengapa sebuah program berjalan dengan sangat lambat dalam situasi apapun.

3.2.2 Average Case

Average case atau kasus rata-rata adalah definisi dari sebuah perilaku algoritma yang dimana runtime sebuah algoritma akan membutuhkan lebih banyak waktu untuk diselesaikan untuk beberapa kasus tertentu, sebagian besar tidak. Ukuran ini menggambarkan ekspektasi dari pengguna algoritma.

Kasus rata-rata sering dijumpai oleh pengguna algoritma dikarenakan algoritma ini akan menyelesaikan sebuah kasus dalam runtime yang rata-rata, tidak terlalu lama, tidak juga sangat cepat.

3.2.3 Best Case

Best case adalah definisi masalah algoritme yang menunjukkan runtime terbaiknya pada suatu kasus. Algoritma yang termasuk dalam best case adalah algoritma yang bekerja paling sedikit. Akan tetapi pada kenyataannya algoritma dengan best case sangat jarang ditemukan.

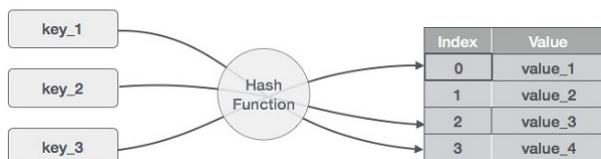
Mengetahui kasus terbaik untuk suatu algoritma sangat berguna meskipun situasinya jarang terjadi dalam praktik secara langsung. Dalam banyak kasus, hal ini memberikan informasi tentang keadaan optimal dari suatu algoritma. Misalnya, best case untuk Counting Search adalah ketika mencari nilai yang diinginkan contohnya adalah v . Counting Search akan menghitung berapa kali v muncul. Jika hitungan yang dihitung adalah nol, maka item tersebut tidak ditemukan, sehingga algoritma akan mengembalikan nilai *false* jika ditemukan maka algoritma akan mengembalikan nilai *true*. Ada beberapa perhatian yang harus dianalisis yaitu Counting Search

akan menelusuri seluruh daftar, oleh karena itu meskipun perilaku kasus terburuknya adalah $O(n)$ dilain sisi kasus terbaiknya atau best case tetap $O(n)$.

BAB 4

HASH MAP

4.1 Hash Map



Gambar 4.1 Hash Map

Hash Map merupakan struktur data yang memiliki indeks. Hash map menggunakan fungsi hash untuk menghitung indeks dengan key ke dalam buckets atau slot. Nilaunya dipetakan ke buckets dengan indeks yang sesuai dan key yang unik dan tidak berubah. Kita ibaratkan has map dengan sebuah lemari yang memiliki laci dengan label untuk barang-barang yang disimpan di dalamnya. Misalnya, menyimpan informasi pengguna dan email sebagai keynya, dan kita dapat memetakan nilai yang

terkait dengan pengguna tersebut seperti nama depan, nama belakang, dll ke dalam buckets.

Hashmap adalah tipe struktur data yang memetakan kunci ke pasangan nilainya (menerapkan tipe data array abstrak). Ini pada dasarnya menggunakan fungsi yang menghitung nilai indeks yang pada gilirannya menyimpan elemen yang akan dicari, dimasukkan, dihapus, dll. Ini membuatnya mudah dan cepat untuk mengakses data. Secara umum, tabel hash menyimpan pasangan key value dan key tersebut dihasilkan menggunakan fungsi hash. Jadi fungsi pencarian dan penyisipan elemen data menjadi lebih cepat karena nilai key itu sendiri menjadi indeks yang menyimpan data.

Fungsi hash adalah inti dari implementasi hash map. Fungsi hash mengambil kunci dan menerjemahkannya ke indeks buckets di daftar buckets. Hashing yang ideal harus menghasilkan indeks yang berbeda untuk setiap key. Namun, hal ini bisa menyebabkan collisions. Saat hashing memberikan indeks yang ada, kita cukup menggunakan buckets untuk beberapa nilai dengan appending list atau dengan rehashing.

Dalam python, dict merupakan contoh dari hash map. Kita akan melihat implementasi hash map dari awal untuk mempelajari cara membangun dan menyesuaikan struktur data tersebut agar dapat mengoptimalkan pencarian. Desain hash map akan mencakup fungsi-fungsi berikut:

1. `set_val(key, value)`: Menyisipkan pasangan key-value ke dalam peta hash. Jika nilai sudah ada di peta hash, perbarui nilainya.
2. `get_val(key)`: Mengembalikan nilai di mana key yang ditentukan dimapping, atau “Tidak ada catatan yang ditemukan” jika map ini tidak berisi pemetaan untuk key tersebut.
3. `delete_val(key)`: Menghapus pemetaan untuk key tertentu jika hash map berisi pemetaan untuk key tersebut.

Berikut contoh implementasi hash map:

```

1 class HashTable :
2
3     # Create empty bucket list of given size
4     def __init__(self, size):
5         self.size = size
6         self.hash_table = self.create_buckets()
7
8     def create_buckets(self):
9         return [[] for _ in range(self.size)]
10
11    # Insert values into hash map
12    def set_val(self, key, val):
13
14        # Get the index from the key
15        # using hash function
16        hashed_key = hash(key) % self.size
17
18        # Get the bucket corresponding to index
19        bucket = self.hash_table[hashed_key]
```

```
20
21     found_key = False
22     for index, record in enumerate(bucket):
23         record_key, record_val = record
24
25         # check if the bucket has same key as
26         # the key to be inserted
27         if record_key == key:
28             found_key = True
29             break
30
31     # If the bucket has same key as the key to be inserted ,
32     # Update the key value
33     # Otherwise append the new key-value pair to the bucket
34     if found_key:
35         bucket[index] = (key, val)
36     else:
37         bucket.append((key, val))
38
39 # Return searched value with specific key
40 def get_val(self, key):
41
42     # Get the index from the key using
43     # hash function
44     hashed_key = hash(key) % self.size
45
46     # Get the bucket corresponding to index
47     bucket = self.hash_table[hashed_key]
48
49     found_key = False
50     for index, record in enumerate(bucket):
51         record_key, record_val = record
52
53         # check if the bucket has same key as
54         # the key being searched
55         if record_key == key:
56             found_key = True
57             break
58
59     # If the bucket has same key as the key being searched ,
60     # Return the value found
61     # Otherwise indicate there was no record found
62     if found_key:
63         return record_val
64     else:
65         return "No record found"
66
67 # Remove a value with specific key
68 def delete_val(self, key):
69
70     # Get the index from the key using
71     # hash function
72     hashed_key = hash(key) % self.size
73
74     # Get the bucket corresponding to index
75     bucket = self.hash_table[hashed_key]
```

```

76     found_key = False
77     for index, record in enumerate(bucket):
78         record_key, record_val = record
79
80         # check if the bucket has same key as
81         # the key to be deleted
82         if record_key == key:
83             found_key = True
84             break
85
86     if found_key:
87         bucket.pop(index)
88     return
89
90 # To print the items of hash map
91 def __str__(self):
92     return "\n".join(str(item) for item in self.hash_table)
93
94
95 hash_table = HashTable(50)
96
97 # insert some values
98 hash_table.set_val('gfg@example.com', 'some value')
99 print(hash_table)
100 print()
101
102 hash_table.set_val('portal@example.com', 'some other value')
103 print(hash_table)
104 print()
105
106 # search/access a record with key
107 print(hash_table.get_val('portal@example.com'))
108 print()
109
110 # delete or remove a value
111 hash_table.delete_val('portal@example.com')
112 print(hash_table)

```

Listing 4.1 Implementasi Hash Map

4.2 Tabel Hash vs Hashmap: Perbedaan antara Tabel Hash dan Hashmap dengan Python

Berikut perbedaan antara Hash Table dengan Hash Map.

Hash Table	Hash Map
Disinkronkan	Tidak Disinkronkan
Cepat	Lambat
Mengizinkan satu kunci nol dan lebih dari satu nilai nol	Tidak mengizinkan kunci atau nilai

4.3 Membuat Dictionary

Dictionary dapat dibuat dengan dua cara:

1. Menggunakan kurung kurawal () Berikut contoh pembuatan dictionary menggunakan kurung kurawal.

```
1 my_dict={ 'Dave' : '001' , 'Ava': '002' , 'Joe': '003' }
2 print(my_dict)
3 type(my_dict)
```

Listing 4.2 Dict dengan Kurung Kurawal

2. menggunakan fungsi dict() Python memiliki fungsi bawaan dict() yang dapat digunakan untuk membuat dictionary, berikut contoh penggunaannya:

```
1 new_dict=dict()
2 print(new_dict)
3 type(new_dict)
```

Listing 4.3 Dict dengan Kurung Kurawal

Dalam contoh di atas, dictionary kosong dibuat karena tidak ada pasangan key-value yang diberikan sebagai parameter ke fungsi dict(). Jika teman-teman ingin menambahkan nilai, teman-teman dapat melakukan hal berikut:

```
1 new_dict=dict(Dave = '001' , Ava= '002' , Joe= '003')
2 print(new_dict)
3 type(new_dict)
```

Listing 4.4 Dict dengan Kurung Kurawal

4.4 Membuat Nested Dictionary

Nested Dictionary merupakan dictionary yang terletak di dalam dictionary. Berikut contoh nested dictionary:

```
1 emp_details = { 'Employee': { 'Dave': { 'ID': '001',
2                                         'Salary': 2000,
3                                         'Designation':'Python Developer'
4 },
5   'Ava': { 'ID': '002',
6           'Salary': 2300,
7           'Designation': 'Java Developer' },
8   'Joe': { 'ID': '003',
9           'Salary': 1843,
          'Designation': 'Hadoop Developer'
}}}
```

Listing 4.5 Nested Dict

4.5 Melakukan Operasi pada Hash Table menggunakan Dictionary

Berikut beberapa operasi yang dapat dilakukan pada hash table dalam python melalui dictionary:

1. Mengakses Nilai Nilai dictionary bisa diakses dengan berbagai cara seperti berikut:

- (a) Menggunakan nilai key Nilai dictionary dapat diakses menggunakan nilai key sebagai berikut:

```
1 my_dict={'Dave' : '001' , 'Ava': '002' , 'Joe': '003'}
2 my_dict['Dave']
```

Listing 4.6 Dict dengan nilai key

Outputnya yaitu 001

- (b) Menggunakan fungsi Ada beberapa fungsi bawaan yang dapat digunakan seperti get(), keys(), values(), dll.

```
1 my_dict={'Dave' : '001' , 'Ava': '002' , 'Joe': '003'}
2 print(my_dict.keys())
3 print(my_dict.values())
4 print(my_dict.get('Dave'))
```

Listing 4.7 Dict with function

Output sebagai berikut:

```
1 dict_keys(['Dave', 'Ava', 'Joe'])
2 dict_values(['001', '002', '003'])
3 001
```

- (c) Menerapkan for loop Perulangan for memungkinkan Anda mengakses pasangan nilai key dictionary dengan mudah dengan mengulanginya. Sebagai contoh:

```
1 my_dict={'Dave' : '001' , 'Ava': '002' , 'Joe': '003'}
2 print("All keys")
3 for x in my_dict:
4     print(x)          #prints the keys
5 print("All values")
6 for x in my_dict.values():
7     print(x)          #prints values
8 print("All keys and values")
9 for x,y in my_dict.items():
10    print(x, ":" , y)      #prints keys and values
```

Listing 4.8 Dict For Loop

Output:

```
1 Semua kunci
2 Dave
3 Ava
4 Joe
```

```
5 Semua nilai  
6 001  
7 002  
8 003  
9 Semua kunci dan nilai  
10 Dave : 001  
11 Ava : 002  
12 Joe : 003
```

2. Memperbarui Nilai Dictionary adalah tipe data yang bisa berubah dan oleh karena itu, teman-teman dapat memperbarunya jika diperlukan. Misalnya, jika saya ingin mengubah ID karyawan bernama Dave dari '001' menjadi '004' dan jika saya ingin menambahkan pasangan nilai kunci lain ke kamus saya, saya dapat melakukan hal berikut:

```
1 my_dict={'Dave' : '001' , 'Ava': '002' , 'Joe': '003'}
2 my_dict['Dave'] = '004' #Updating the value of Dave
3 my_dict['Chris'] = '005' #adding a key-value pair
4 print(my_dict)
```

Listing 4.9 Dict For Loop

Output:

```
1 { 'Dave': '004', 'Ava': '002', 'Joe': '003', 'Chris': '005'}
```

3. Menghapus Elemen Ada sejumlah fungsi yang memungkinkan Anda untuk menghapus item dari dictionary seperti del(), pop(), popitem(), clear(), dll. Misalnya:

```
1 my_dict={'Dave': '004', 'Ava': '002', 'Joe': '003', 'Chris': '005'
2         }
3 del my_dict['Dave'] #removes key-value pair of 'Dave'
4 my_dict.pop('Ava') #removes the value of 'Ava'
5 my_dict.popitem() #removes the last inserted item
6 print(my_dict)
```

Listing 4.10 Dict For Loop

Output: 'Joe': '003'. Output di atas menunjukkan bahwa semua elemen kecuali 'Joe: 003' telah dihapus dari kamus menggunakan berbagai fungsi.

4.6 Mengubah Dictionary menjadi Kerangka Data

Seperti yang telah teman-teman lihat sebelumnya, saya telah membuat nested dictionary yang berisi nama karyawan dan detailnya dipetakan ke sana. Sekarang untuk membuat tabel yang jelas dari itu, saya akan menggunakan library pandas untuk meletakkan semuanya sebagai kerangka data.

```

4         'Designation': 'Python Developer'
5     },
6     'Ava': {'ID': '002',
7             'Salary': 2300,
8             'Designation': 'Java Developer'},
9     'Joe': {'ID': '003',
10            'Salary': 1843,
11            'Designation': 'Hadoop Developer'}
12   }})
13 df=pd.DataFrame(emp_details[ 'Employee' ])
14 print(df)

```

Listing 4.11 Mengubah Dictionary menjadi Kerangka Data**Output:**

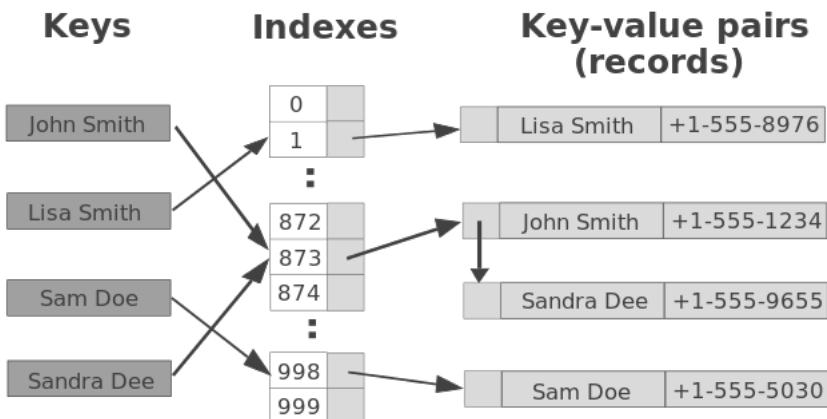
	Dave	Ava	Joe
Designation	Python Developer	Java Developer	Hadoop Developer
ID	001	002	003
Salary	2000	2300	1843

Gambar 4.2 *Output Dictionary*

BAB 5

HASH TABLES

5.1 Hash Table



Gambar 5.1 Hash Tables

Tabel hash adalah jenis struktur data di mana alamat atau nilai indeks elemen data dihasilkan dari fungsi hash. Itu membuat mengakses data lebih cepat karena nilai indeks berperilaku sebagai kunci untuk nilai data. Dengan kata lain tabel Hash menyimpan pasangan nilai kunci tetapi kuncinya dihasilkan melalui fungsi hashing. Jadi fungsi pencarian dan penyisipan elemen data menjadi lebih cepat karena nilai kunci itu sendiri menjadi indeks larik yang menyimpan data. Dalam Python, tipe data Kamus mewakili implementasi tabel hash. Kunci dalam kamus memenuhi persyaratan berikut.

1. Kunci kamus adalah hashable yaitu dihasilkan oleh fungsi hashing yang menghasilkan hasil unik untuk setiap nilai unik yang diberikan ke fungsi hash.
2. Urutan elemen data dalam kamus tidak tetap.

Jadi kita melihat implementasi tabel hash dengan menggunakan tipe data kamus seperti di bawah ini.

5.1.1 Mengakses Nilai dari Dictionary

Untuk mengakses elemen kamus, Anda dapat menggunakan tanda kurung siku bersama dengan kunci untuk mendapatkan nilainya.

```
1 # Declare a dictionary
2 dict = { 'Name': 'Angga', 'Age': 21, 'Class': 'Four' }
3
4 # Accessing the dictionary with its key
5 print("dict['Name']: ", dict['Name'])
6 print("dict['Age']: ", dict['Age'])
```

Listing 5.1 Implementasi Hash Table

Ketika kode di atas dijalankan, menghasilkan hasil sebagai berikut:

```
1 dict['Name']: Angga
2 dict['Age']: 21
```

Listing 5.2 Hasil Implementasi Hash Table

5.1.2 Update Dictionary

Anda dapat memperbarui kamus dengan menambahkan entri baru atau pasangan nilai kunci, memodifikasi entri yang ada, atau menghapus entri yang ada seperti yang ditunjukkan di bawah ini dalam contoh sederhana.

```
1 # Declare a dictionary
2 dict = { 'Name': 'Angga', 'Age': 21, 'Class': 'Four' }
3 dict['Age'] = 20 # update existing entry
4 dict['College'] = "Poltekpos" # Add new entry
5 print("dict['Age']: ", dict['Age'])
6 print("dict['College']: ", dict['College'])
```

Listing 5.3 Implementasi Update Hash Table

Ketika kode di atas dijalankan, menghasilkan hasil sebagai berikut:

```
1 dict['Age']: 20
2 dict['School']: Poltekpos
```

Listing 5.4 Hasil Implementasi Update Hash Table

5.1.3 Delete Dictionary

Anda dapat menghapus elemen kamus satu per satu atau menghapus seluruh konten kamus. Anda juga dapat menghapus seluruh kamus dalam satu operasi. Untuk menghapus seluruh kamus secara eksplisit, cukup gunakan pernyataan **del**.

```
1 # Declare a dictionary
2 dict = {'Name': 'Angga', 'Age': 21, 'Class': 'Four'}
3 del dict['Name'] # remove entry with key 'Name'
4 dict.clear()      # remove all entries in dict
5 del dict         # delete entire dictionary
6
7 print("dict['Age']: ", dict['Age'])
8 print("dict['School']: ", dict['School'])
```

Listing 5.5 Implementasi Delete Hash Table

Ini menghasilkan hasil berikut. Perhatikan bahwa pengecualian dimunculkan karena kamus setelah del dict tidak ada lagi.

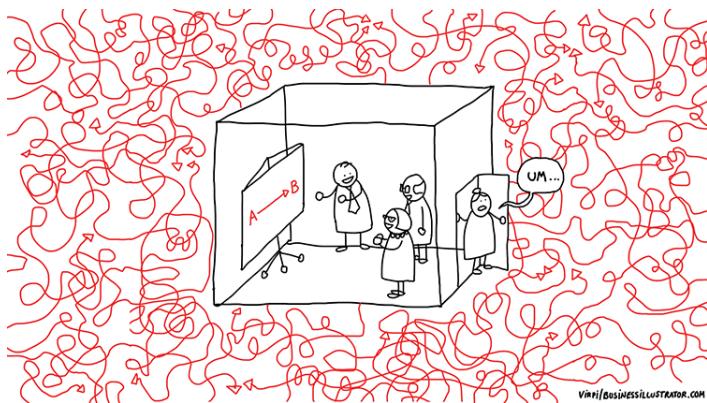
```
1 dict['Age']:
2 Traceback (most recent call last):
3   File "test.py", line 8, in <module>
4     print "dict['Age']: ", dict['Age'];
5 TypeError: 'type' object is unsubscriptable
```

Listing 5.6 Hasil Implementasi Delete Hash Table

BAB 6

KOMPLEKSITAS

6.1 Kompleksitas

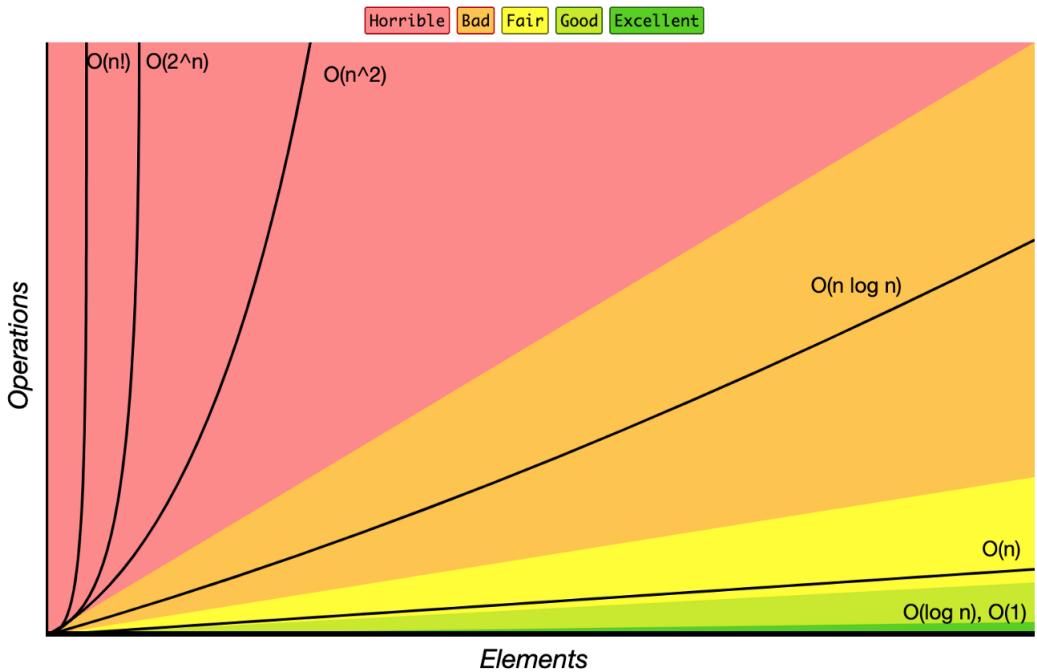


Gambar 6.1 Kompleksitas

Kompleksitas adalah suatu indikator antarhubungan di dalam suatu program yang memengaruhi cara bagaimana hubungan ini akan dikelola dan keahlian yang dibutuhkan untuk mengelolanya. Semua program dihasilkan dari banyak fungsi dan proses yang saling berhubungan dimana semuanya sangat kompleks. Kompleksitas dibagi menjadi 2 yaitu Time Complexity dan Space Complexity.

6.2 Asymptotic Notation

6.3 Big O Notation



Gambar 6.2 Big O Notation

Big o notation dapat kita fahami sebagai notasi atau lambang matematika yang menggambarkan tingkat dari kompleksitas atau kerumitan suatu sistem. Big o notation biasanya dilambangkan sebagai $O(n)$. Prinsip-prinsip big o notation ini dapat kita terapkan pada ilmu komputer untuk mengelompokkan atau mengklasifikasikan algoritma pemrograman berdasarkan kerumitannya. Pada penerapannya, big o notasi ini mengukur tingkat lamanya waktu proses (running time) dan space atau resource yang digunakan berbanding lurus dengan bertambahnya input data.

Big o notasi menjelaskan suatu fungsi yang diidentifikasi berdasarkan pertumbuhan datanya. Fungsi atau algoritma yang berbeda tetapi tingkat pertumbuhannya

sama dapat dilambangkan dengan big o notasi yang sama. Para developer atau programmer biasanya mendefinisikan tingkat kerumitan code atau algoritma yang kita buat menggunakan big o notasi ini. Dengan demikian, hal ini memudahkan cara berkomunikasi yang baik antar programmer dalam membahas kerumitan suatu algoritma. Berikut adalah cheat sheet dari beberapa operasi umum pada struktur data dalam bahasa pemograman.

Data Structure	Time Complexity								Space Complexity	
	Average				Worst					
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion		
Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	
Stack	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)	
Queue	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)	
Singly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)	
Doubly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)	
Skip List	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))	
Hash Table	N/A	O(1)	O(1)	O(1)	N/A	O(n)	O(n)	O(n)	O(n)	
Binary Search Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)	
Cartesian Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(n)	O(n)	O(n)	O(n)	
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	
Red-Black Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	
Splay Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)	
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	
KD Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)	

Gambar 6.3 Cheat Sheet Big O Notation

Sebagai cotoh, konsep array memiliki notasi $O(1)$ untuk cara mengkases data dan $O(n)$ untuk proses input data. Dari kedua grafik tersebut kita mendapatkan informasi bahwa konsep array ini memiliki performance terbaik untuk cara akses dan tidak terpengaruh terhadap banyaknya data. Adapun untuk proses insert data dia akan meningkat secara aritmetik dan prosesnya masih di bilang sangat bagus. Berikut data cheat sheet untuk teknik melakukan sorting dan nilai big o notasinya.

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	
Quicksort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(\log(n))$
Mergesort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Timsort	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Heapsort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(1)$
Bubble Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Selection Sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Tree Sort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(n)$
Shell Sort	$\Omega(n \log(n))$	$\Theta(n(\log(n))^2)$	$O(n(\log(n))^2)$	$O(1)$
Bucket Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n^2)$	$O(n)$
Radix Sort	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$
Counting Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$	$O(k)$
Cubesort	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$

Gambar 6.4 Cheat Sheet Big O Notation Sorting

6.3.1 Cara Membaca dan Memahami Big O

Kompleksitas dalam waktu artinya adalah seberapa lama kode program yang kita untuk menyelesaikan sebuah masalah yang kompleks. Kompleksitas ruang sama halnya dengan kompleksitas waktu, semakin rumit masalah yang ingin diselesaikan maka ruang sebagai penyimpanan atau bisa disebut memory akan semakin banyak digunakan. Dibawah ini ada contoh untuk membaca dan memahami Big O Notation dan penjelasannya.

1. $O(\log n)$ berarti tingkat kompleksitas akan berbanding lurus dengan \log dari banyaknya jumlah data. Algoritma yang masuk kategori $O(\log n)$ maka algoritma tersebut masuk kekategori yang sangat bagus.
2. $O(1)$ tingkat kompleksitas yang konstan dengan model grafik horizontal. Algoritma yang memiliki kategori $O(1)$ adalah algoritma yang terbaik dikarenakan time complexity dan space complexity yang paling cepat dan sedikit memakan ruang memory.
3. $O(n)$ tingkat kompleksitas yang linear atau berbanding lurus dengan banyaknya data sehingga akan membentuk garis diagonal. Dalam matematika bisa disebut linear.

4. $O(n+2)$ atau $O(2n+5)$ sama dengan tingkat kompleksitas $O(n)$. Jadi konstanta tidak dimasukan dalam penghitungan tingkat kompleksitas.

6.3.2 Kegunaan Pemahaman Big O

Akan muncul beberapa pertanyaan apakah teori Big O Notation akan berguna dalam kehidupan sehari-hari programmer atau hanya berguna ketika akan mengikuti test penerimaan kerja? Jawabannya adalah ya untuk kedua kondisi tersebut. Pada penggunaan sehari-hari pengetahuan big o notasi ini akan sangat berguna dalam meningkatkan kualitas serta performance dari algoritma-algoritma yang kita buat. Kita sebagai programmer atau penulis kode program akan mengetahui seberapa cepat atau bisa dibilang Time Complexity dari kode program yang kita tulis ketika data yang diproses semakin besar.

6.3.3 Contoh Big O Notasi dengan Python3

6.3.3.1 $O(1)$ $O(1)$ adalah Big O Notasi paling baik diantara semua kategori dikarenakan $O(1)$ konstan sehingga tidak memiliki tambahan operasi apapun untuk menambah waktu apapun. $O(1)$ terdiri dari operasi sedarhana yang tidak membutuhkan iterasi atau perulangan dalam kode program, contohnya adalah sebagai berikut.

```
1 ini_list = ['tri', 'angga', 'dio', 'simamora']
2 _ = ini_list[0]
```

Kode program Python diatas akan masuk ke kategori $O(1)$ dikarenakan kode program mengetahui index yang akan dituju dalam hal ini index 0 yang berisi 'tri'.

6.3.3.2 $O(\log n)$ $O(\log n)$ adalah kategori Big O Notatio yang berada dibawah $O(1)$ dikarenakan kategori Big O Notasi ini memiliki iterasi dan setiap iterasinya akan dilipat menjadi 2 (kurang, tambah, bagi, kali), berikut contohnya.

```
1 t = 100
2 while t > 10:
3     t -= 10
```

Input n sebesar 100 akan tetapi dalam iterasi dikurangi 10 sehingga perulangan bisa diakhiri dengan lebih cepat.

6.3.3.3 $O(n)$ $O(n)$ adalah Big O Notasi yang sering kita terapkan dalam kode program, $O(n)$ adalah kategori iterasi dalam kode program yang menggunakan seluruh element n yang diberikan, berikut contohnya

```
1 ini_list = ['tri', 'angga', 'dio']
2 for i in ini_list:
3     print(i)
```

Kode program diatas adalah salah satu contoh penggunaan $O(n)$, banyak sekali library built in python yang mengadopsi $O(n)$ contohnya adalah ketika kita ingin mengetahui sebuah index dari list berdasarkan nilainya maka kode program tersebut akan masuk kekategori $O(n)$ mengapa demikian? mari kita coba

```

1 # ada 1 built in pada python yaitu .index() untuk menemukan index ke
  berapa berdasarkan nilai , mari kita bedah mengapa bisa masuk
  kategori O(n)
2
3 ini_list = ['tri', 'angga', 'dio']
4
5 index_tri = ini_list.index('tri') # maka akan berisi 0
6
7 # berikut kode searchnya
8 def get_index(value_search):
9     index = 0
10    list_data = ['tri', 'angga', 'dio', 'simamora']
11    for value in list_data:
12        if value == value_search:
13            return index
14        index += 1
15    return None

```

Bisa terlihat untuk list search dengan pengembalian dimana index itu berada masuk dalam kategori O(n)

6.3.4 $O(n^2)$

$O(n^2)$ bisa dibilang kategori yang semestinya dihindarkan jika ingin mengolah data yang banyak, dikarenakan $O(n^2)$ memiliki perulangan didalam perulangan, atau bisa dibilang nested loops. Nested loop akan aman digunakan jika data yang digunakan sangat sedikit, akan tetapi jika data yang diolah mencapai jutaan maka kategori $O(n^2)$ harus dipertimbangkan dalam penggunaannya. Berikut contoh kode program $O(n^2)$.

```

1 ini_list_list = [[ 'tri' , 0], [ 'angga' , 1], [ 'dio' , 2], [ 'simamora' ,
  3]]
2 value = 'dio'
3 for i in ini_list_list:
4     for j in i:
5         if j == value:
6             print(i[1])
7             break
8     else:
9         continue
10    break

```

6.3.5 Ingin merasakan perbedaan dari $O(n)$ dan $O(n^2)$????

Jika anda penasaran apa yang saya maksud jika mengolah data yang besar, mari kita coba. Jika anda sudah memiliki akun Github, dan sudah mengikuti semua tutorial Github diatas maka anda sudah bisa mengikuti tutorial ini, jika belum maka anda harus menyelesaikan tutorial diatas. Markicob (mari kita coba)

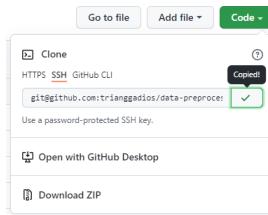
1. kunjungi Github Repository ¹, maka akan muncul seperti dibawah ini.

¹<https://github.com/trianggadios/data-preprocessing-computational>

The screenshot shows a GitHub repository page for 'data-preprocessing-computational'. The repository has 1 branch and 0 tags. The file tree includes 'requirements' (yesterday), 'algo' (initial commit yesterday), '.gitignore' (initial commit yesterday), 'LICENSE' (Initial commit yesterday), 'README.md' (Initial commit yesterday), 'daftar-slang-bahasa-indonesia.csv' (initial commit yesterday), 'qa.csv' (initial commit yesterday), 'requirements.txt' (requirements yesterday), and 'testing.py' (initial commit yesterday). The README.md file is open, displaying the text 'data-preprocessing-computational'. On the right side, there are sections for 'About' (No description, website, or topics provided), 'Releases' (No releases published, Create a new release), 'Packages' (No packages published, Publish your first package), 'Contributors' (dindamajesty13 Dinda Majesty, trianggadios Tri Angga Dio Simamora), and 'Languages' (Python 100.0%).

Gambar 6.5 Mencoba Big O Notation: Step 1

2. setelah itu klik bagian Code dan klik logo copy hingga ada tulisan **Copied!**



Gambar 6.6 Mencoba Big O Notation: Step 2

3. buka git bash anda, lalu ketikkan **git clone [url yang sudah dicopy bisa di-paste]**, tekan enter

```
MINGW64/c/Users/dinda
dinda@LAPTOP-43MM3U7K MINGW64 ~
$ git clone git@github.com:trianggadinos/data-preprocessing-computational.git
```

Gambar 6.7 Mencoba Big O Notation: Step 3

4. tunggu hingga selesai

```
MINGW64/c/Users/dinda
dinda@LAPTOP-43MM3U7K MINGW64 ~
$ git clone git@github.com:trianggadinos/data-preprocessing-computational.git
Cloning into 'data-preprocessing-computational'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 18 (delta 3), reused 14 (delta 3), pack-reused 0
Receiving objects: 100% (18/18), 102.05 KiB | 129.00 KiB/s, done.
Resolving deltas: 100% (3/3), done.

dinda@LAPTOP-43MM3U7K MINGW64 ~
$ |
```

Gambar 6.8 Mencoba Big O Notation: Step 4

5. lalu masuk ke directory dengan mengetikkan **cd data-preprocessing-computational/**, tekan enter

```
MINGW64/c/Users/dinda
dinda@LAPTOP-43MM3U7K MINGW64 ~
$ git clone git@github.com:trianggadinos/data-preprocessing-computational.git
Cloning into 'data-preprocessing-computational'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 18 (delta 3), reused 14 (delta 3), pack-reused 0
Receiving objects: 100% (18/18), 102.05 KiB | 129.00 KiB/s, done.
Resolving deltas: 100% (3/3), done.

dinda@LAPTOP-43MM3U7K MINGW64 ~
$ cd data-preprocessing-computational/|
```

Gambar 6.9 Mencoba Big O Notation: Step 5

6. lalu ketikkan perintah berikut ini **pip install -r requirements.txt**, tekan enter, tunggu hingga selesai

```
dinda@LAPTOP-43MM3U7K MINGW64 ~/data-preprocessing-computational (main)
$ pip install -r requirements.txt
Collecting pandas==1.4.0
  Using cached pandas-1.4.0-cp310-cp310-win_amd64.whl (10.6 MB)
Collecting psutil==5.9.0
  Using cached psutil-5.9.0-cp310-cp310-win_amd64.whl (245 kB)
Collecting numpy==1.21.0
  Using cached numpy-1.22.2-cp310-cp310-win_amd64.whl (14.7 MB)
Collecting pytz>=2020.1
  Using cached pytz-2021.3-py2.py3-none-any.whl (503 kB)
Collecting python-dateutil<2.8.1
  Using cached python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
Collecting six<1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, pytz, python-dateutil, numpy, psutil, pandas
Successfully installed numpy-1.22.2 pandas-1.4.0 psutil-5.9.0 python-dateutil-2.8.2 pytz-2021.3 six-1.16.0
WARNING: You are using pip version 21.2.4; however, version 22.0.3 is available.
You should consider upgrading via the 'C:\Users\dinda\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip' command.
dinda@LAPTOP-43MM3U7K MINGW64 ~/data-preprocessing-computational (main)
$ |
```

Gambar 6.10 Mencoba Big O Notation: Step 6

7. lalu jalankan script python dengan mengetikkan **python testing.py**, tekan enter, tunggu hingga kode program selesai

```
dinda@LAPTOP-43MM3U7K MINGW64 ~/data-preprocessing-computational (main)
$ python testing.py
dinda@LAPTOP-43MM3U7K MINGW64 ~/data-preprocessing-computational (main)
$ |
```

Gambar 6.11 Mencoba Big O Notation: Step 7

8. terdapat 2 algo yaitu algo1 dan algo2, algo1 menggunakan $O(n^2)$ sedangkan algo2 menggunakan $O(n)$
9. untuk melihat hasil kompleksitas waktu bisa mengetikkan perintah berikut, **cat algo1_time.txt**

```

MINGW64:/c/Users/dinda/data-preprocessing-computational
$ python testing.py
dinda@APTOP-43MM3U7K MINGW64 ~/data-preprocessing-computational (main)
$ cat algo1_time.txt
13.13219404220581
dinda@APTOP-43MM3U7K MINGW64 ~/data-preprocessing-computational (main)
$ |

```

Gambar 6.12 Mencoba Big O Notation: Step 8

10. terlihat gambar diatas yaitu hasil komputasi $O(n^2)$ yaitu 13 detik
11. jika ingin mengetahui berapa detik yang dihasilkan oleh $O(n)$, ketikkan perintah berikut **cat algo2_time.txt**, tekan enter

```

MINGW64:/c/Users/dinda/data-preprocessing-computational
$ python testing.py
dinda@APTOP-43MM3U7K MINGW64 ~/data-preprocessing-computational (main)
$ cat algo2_time.txt
13.13219404220581
dinda@APTOP-43MM3U7K MINGW64 ~/data-preprocessing-computational (main)
$ cat algo2_time.txt
0.063327789306664062
dinda@APTOP-43MM3U7K MINGW64 ~/data-preprocessing-computational (main)
$ |

```

Gambar 6.13 Mencoba Big O Notation: Step 9

12. sangat berbeda jauh bukan? mungkin akan tidak terasa jika data yang diolah sangat kecil akan tetapi jika menggunakan data yang besar maka akan terlihat sangat jauh perbedaannya.

BAB 7

BIG O NOTATION

Programmer profesional menggunakan cara yang efektif dan seefisien mungkin dalam menyelesaikan suatu permasalahan. Menemukan cara yang efektif dan efisien salah satu caranya yaitu dengan meminimalisir kompleksitas dari algoritma yang digunakan. Kompleksitas suatu algoritma terbagi menjadi 2, yaitu Time Complexity dan Space Complexity.

Time Complexity merupakan lama waktu yang diperlukan untuk menjalankan suatu algoritma. Sedangkan Space Complexity adalah ukuran ruang penyimpanan yang kita gunakan untuk menjalankan suatu algoritma. Dan disini saya hanya akan membahas tentang Time Complexity. Sebelum itu saya ingin teman-teman memahami terlebih dahulu apa itu algoritma. Sederhananya, algoritma adalah serangkaian proses yang dilakukan secara berurutan untuk menyelesaikan sebuah permasalahan. Algoritma bisa bermacam-macam tergantung kepada siapa yang membuat algoritma tersebut. Namun permasalahannya adalah menentukan algoritma mana yang lebih efektif dan efisien untuk menyelesaikan permasalahan yang kita hadapi.

Contoh simpel dalam kehidupan kita sehari-hari, ketika kita ingin pergi ke suatu tempat tetapi ada banyak jalan yang bisa kita lalui untuk bisa sampai di tempat tujuan, namun permasalahannya adalah rute mana yang paling cepat yang bisa kita ambil untuk sampai di tempat tujuan.

Regular Big O

2 $O(1)$ = it's just a constant number

$2n + 10$ $O(n)$ = n has the largest effect

$5n^2$ $O(n^2)$ = n^2 has the largest effect

Time Complexity Analysis adalah suatu cara sederhana untuk mengetahui berapa lama waktu yang dibutuhkan untuk menjalankan suatu algoritma dengan input tertentu (n). Biasanya lebih dikenal dengan sebutan Big-O Notation. Big O Notation digunakan untuk mengukur tingkat kompleksitas suatu algoritma. Big-O Notation adalah cara untuk mengkonversi keseluruhan langkah-langkah suatu algoritma kedalam bentuk Aljabar, yaitu dengan menghiraukan konstanta yang lebih kecil dan koefisien yang tidak berdampak besar terhadap keseluruhan kompleksitas permasalahan yang diselesaikan oleh algoritma tersebut. Mari kita liat contoh dibawah ini:

Sederhananya, semua contoh yang ada diatas mengatakan bahwa “kita hanya akan melihat faktor yang memiliki dampak paling besar terhadap nilai yang dihasilkan oleh algoritma tersebut”.

Terdapat beberapa macam time complexity, diantaranya:

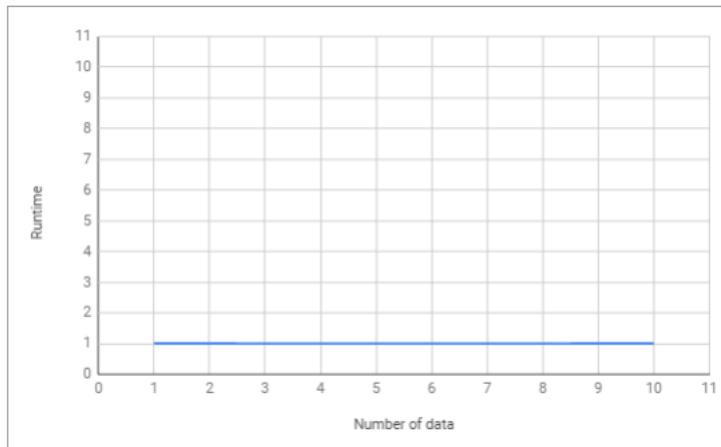
1. $O(1)$ — Constant Time Constant Time artinya banyaknya input yang diberikan kepada sebuah algoritma, tidak akan mempengaruhi waktu proses (runtime) dari algoritma tersebut. Suatu algoritma dengan $T(n) O(1)$ dikatakan memiliki kompleksitas waktu yang konstan.

```

1 let myArray = [1, 5, 0, 6, 1, 9, 9, 2];
2 function getFirst(input){
3     return input[0]; // selalu melakukan 1 langkah
4 }
5 let firstEl = getFirst(myArray);

```

Contoh diatas, terdapat sebuah fungsi untuk mengambil elemen pertama dari sebuah input array. Kita bisa melihat bahwa berapapun jumlah array yang diberikan kepada fungsi tersebut, dia akan selalu melakukan 1 hal, yaitu mengambil elemen pertama. Itu artinya jumlah input yang diberikan tidak mempengaruhi waktu proses (runtime) dari algoritma tersebut.



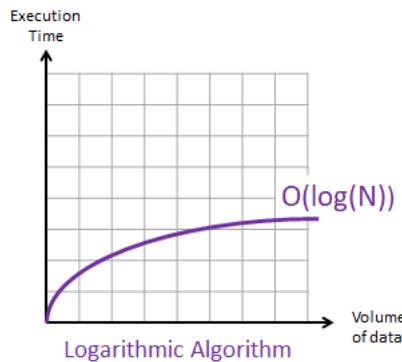
Gambar 7.1 *Constant Time*

2. $O(\log n)$ — Logarithmic Time Logarithmic Time artinya ketika kita memberikan input sebesar n terhadap sebuah fungsi, jumlah tahapan yang dilakukan oleh fungsi tersebut berkurang berdasarkan suatu faktor. Salah satu contohnya adalah algoritma Binary Search. Binary Search adalah algoritma yang kita gunakan dalam mencari posisi nilai dari suatu array dengan cara ‘mengeliminasi’ setengah dari array input untuk mempercepat proses pencarian.

```

1 let sortedArray = [11, 24, 30, 43, 51, 61, 73, 86];
2 function isExists(number, array){
3     var midPoint = Math.floor( array.length /2 );
4     if( array[midPoint] === num) return true;
5     let isFirstHalf = false;
6     if( array[midPoint] < num ) isFirstHalf = true;
7
8     else if( array[midPoint] > num ) isFirstHalf = false;
9     if (array.length == 1) return false;
10    else {
11        // memanggil fungsi yang sama dengan mengeleminiasi
12        // setengah dari input array
13        if (isFirstHalf)
14            return isExists(number, getFirstHalf(array));
15        else
16            return isExists(number, getSecondHalf(array));
17    }
18 isExists (24, sortedArray); // return true
19 isExists (27, sortedArray); // return false

```



Gambar 7.2 Logarithmic Time

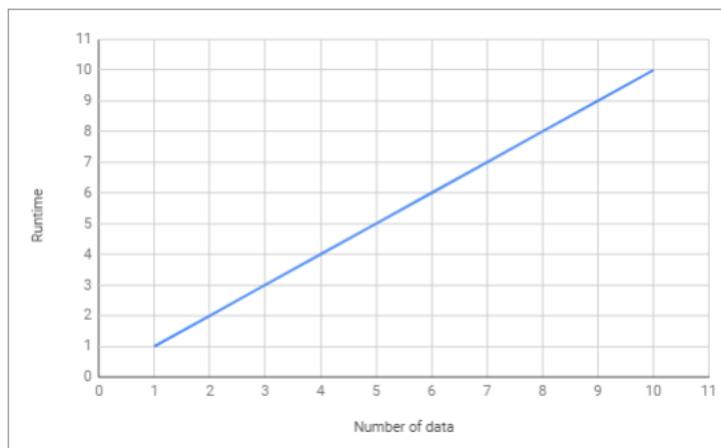
3. $O(n)$ — Linear Time Linear Time adalah ketika runtime dari fungsi kita berbanding lurus dengan jumlah input yang diberikan.

```

1 let myArray = [1, 5, 0, 6, 1, 9, 9, 2];
2 function getMax(input){
3     var max = 0;
4     for (var i=0; i<input.length; i++){
5         if (max < input[i])
6             max = input[i];
7     }
8     return max;
9 }
10 let maxNumber = getMax(myArray);

```

Kita bisa melihat bahwa semakin banyak jumlah input yang diberikan, maka waktu proses/runtime dari fungsi tersebut akan semakin besar.



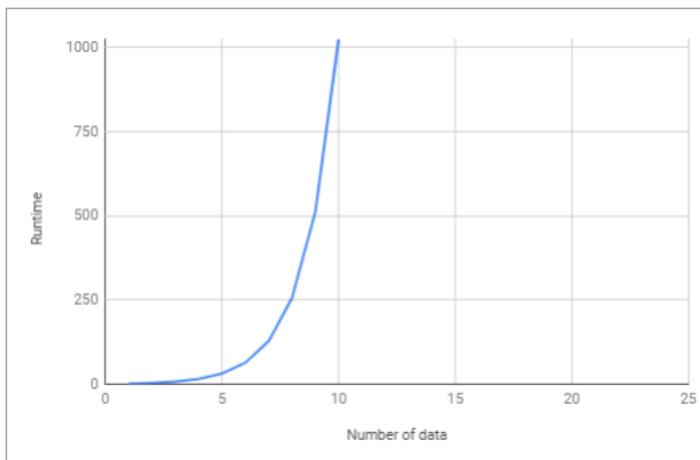
Gambar 7.3 Linear Time

4. $O(n^2)$ — Quadratic Time Quadratic Time adalah ketika runtime dari fungsi kita adalah sebesar n^2 , dimana n adalah jumlah input dari fungsi tersebut. Hal tersebut bisa terjadi karena kita menjalankan fungsi linear didalam fungsi linear ($n \cdot n$).

```

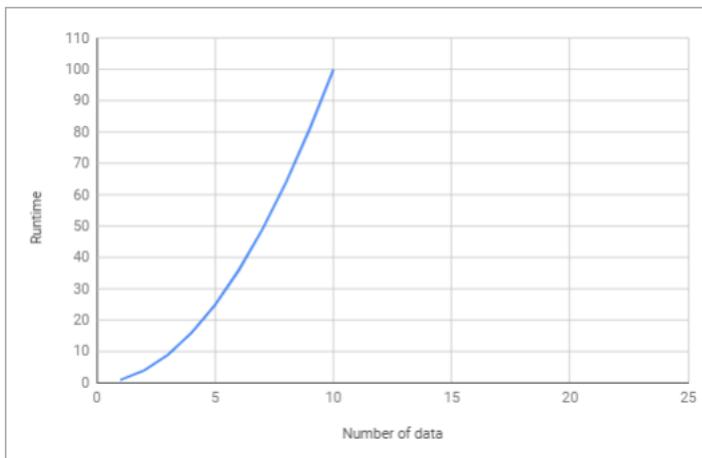
1 let myArray = [1, 5, 0, 6, 1, 9, 9, 2];
2 function sort(input){
3     var sortedArray = [];
4     for (var i=0; i<input.length; i++){ // O(n)
5         let min = input[i];
6         for (var j=i+1; i<input.length; i++){ // O(n)
7             if (input[i] < input[j])
8                 min = input[j];
9         }
10        sortedArray.push(min);
11    }
12    return sortedArray;
13 }
14 let sortedArray = sort(myArray);

```



Gambar 7.4 *Quadratic Time*

5. $O(2^n)$ — Exponential Time Exponential Time biasanya digunakan dalam situasi dimana kita tidak terlalu tahu terhadap permasalahan yang dihadapi, sehingga mengharuskan kita mencoba setiap kombinasi dan permutasi dari semua kemungkinan.



Gambar 7.5 *Exponential Time*