



Algoritma PEMROGRAMAN



python

Sri Tria Siska, S.Kom., M.Kom., Hariyadi, S.Kom., M.Kom., Alfry Aristo Jansen Sinlae, S.Kom., M.Cs., Nova Tri Romadloni, S.Kom., M.Kom., Dini NurmalaSari, S.T., M.T., Heni Rachmawati, S.T., M.T., Wenda Novayani, S.ST., M.Eng., Trizaerah Armiani, S.Kom., M.Sc., I Wayan Rangga Pinastawa, M.Kom., Teuku Radillah, S.Kom., M.Kom., Fithri Selva Jumeilah, S.Kom., M.T.I., Mohammad Robihul Mufid, S.ST., M.Tr.Kom., Saniyatul Mawaddah, S.ST., M.Kom.

Editor: Yulya Muhammi, S.Kom., M.Kom.

ALGORITMA PEMROGRAMAN PYTHON

**Sri Tria Siska, S.Kom., M. Kom., Hariyadi, S.Kom., M.Kom,
Alfrey Aristo Jansen Sinlae, S.Kom., M.Cs., Nova Tri
Romadloni, M.Kom., Dini NurmalaSari, Heni Rachmawati,
Wenda Novayani, Trizaurah Armiani, S.Kom., M.Sc., I Wayan
Rangga Pinastawa, Teuku Radillah, Fithri Selva Jumeilah,
S.Kom., M.T.I., Mohammad Robihul Mufid, Saniyatul
Mawaddah**



Algoritma Pemrograman Python

Copyright© PT Penamudamedia, 2023

Penulis:

Sri Tria Siska, S.Kom., M. Kom., Hariyadi, S.Kom., M.Kom, Alfry Aristo Jansen Sinlae, S.Kom., M.Cs., Nova Tri Romadloni, M.Kom., Dini NurmalaSari, Heni Rachmawati, Wenda Novayani, Trizaurah Armiani, S.Kom., M.Sc., I Wayan Rangga Pinastawa, Teuku Radillah, Fithri Selva Jumeilah, S.Kom., M.T.I., Mohammad Robihul Mufid, Saniyatul Mawaddah

Editor:

Yulya Muhammi, S.Kom., M.Kom

ISBN:

978-623-09-6020-8

Desain Sampul:

Tim PT Penamuda Media

Tata Letak:

Enbookdesign

Diterbitkan Oleh

PT Penamuda Media

Casa Sidoarium RT 03 Ngentak, Sidoarium Dodeam Sleman Yogyakarta

HP/Whatsapp	: +6285700592256
Email	: penamudamedia@gmail.com
Web	: www.penamuda.com
Instagram	: @penamudamedia

Cetakan Pertama, Oktober 2023

x + 150, 15x23 cm

Hak cipta dilindungi oleh undang-undang

*Dilarang memperbanyak sebagian atau seluruh isi buku
tanpa izin Penerbit*

KATA PENGANTAR

Alhamdulillah dan Puji syukur dipanjangkan kehadiran Allah Subhanahu wa ta'ala atas rahmat-Nya dikarenakan Buku Algoritma Pemrograman Phyton ini dapat diselesaikan dengan baik. Buku ini disusun sedemikian rupa agar dapat digunakan dengan mudah sebagai panduan pembelajaran untuk mahasiswa atau siapa pun yang ingin memahami konsep dasar pemrograman dan algoritma menggunakan bahasa pemrograman Python. Python memiliki berbagai kegunaan, termasuk pengembangan web, pemrosesan data, kecerdasan buatan (AI), pemodelan matematika, pengembangan permainan, dan masih banyak lagi. Keberhasilan Python juga didukung oleh ekosistem yang kuat, termasuk berbagai pustaka dan kerangka kerja (framework) yang mendukung berbagai jenis pengembangan. Dalam buku ini diberikan teori konsep dan contoh kasus serta penyelesaian menggunakan bantuan bahasa pemrograman Python.

Terima kasih yang sebesar-besarnya kami ucapkan kepada pihak-pihak yang telah membantu dan mendukung pembuatan buku Algoritma Pemrograman Phyton ini. Harapan kami semoga buku ini dapat memberikan manfaat bagi para pembacanya. Buku ini masih jauh dari sempurna. Oleh sebab itu kami membuka kritik dan saran guna untuk menyempurnakan buku ini dalam edisi selanjutnya.

Pekanbaru, Oktober 2023

Yulya Muharmi, S.Kom., M.Kom

DAFTAR ISI

KATA PENGANTAR.....	v
DAFTAR ISI	vi
Bab 1 - Konsep Dasar Algoritma.....	1
A. Pengertian Algoritma	1
B. Fungsi Algoritma Pemograman.....	3
C. Jenis - Jenis Algoritma Pemograman	3
D. Syarat Algoritma	6
E. Ciri - Ciri Algoritma.....	6
F. Struktur Dasar Algoritma.....	7
Bab 2 - Notasi Algoritma	10
A. Pengertian Notasi Algoritma.....	10
B. Jenis - Jenis Flowchart	16
C. Karakteristik Algoritma	17
Bab 3 - Pemrograman Python.....	19
A. Latar Belakang Python	19
B. Instalasi Python.....	20
C. Menjalankan Kode Python.....	22
D. Mengapa Memilih Python?	24
E. Kontribusi dan Pengguna Python	24
F. Peran Python dalam Industri Teknologi Modern	25
G. Pengembangan Python dan Versi Utama	25

Bab 4 - Operator Operator Python	26
A. Operator Aritmatika	26
B. Operator Perbandingan	28
C. Operator Logika	30
D. Operator Penugasan / Assignment	32
E. Operator Identitas	34
F. Operator Keanggotaan	35
G. Operator Bitwise	36
Bab 5 - List, Dictionary, Tuple	40
A. List.....	40
B. Dictionary.....	45
C. Tupple.....	49
Bab 6- Tipe Data, Variable, Konstanta, Dan Nilai	53
A. Tipe Data Pada Python	53
B. Pemberian type data secara spesifik.....	60
C. Cara mengetahui tipe data pada python	61
D. Variable pada Python.....	61
E. Konstanta pada Python	63
F. Nilai pada Python	64
Bab 7 - Percabangan	66
A. Struktur percabangan <i>if</i>	66
B. Struktur percabangan <i>if/Else</i>	68
C. Struktur percabangan <i>if/Elif/Else</i>	70
D. Struktur percabangan bersarang	72

E. Latihan Soal	74
Bab 18- Library.....	75
A. Apa Itu Library?.....	76
B. Jenis-jenis Library	78
C. Menggunakan Library Python.....	81
Bab 9 - Algoritma Perulangan.....	85
A. Perulangan <i>while</i>	85
B. Perulangan <i>For</i>	93
Bab 10 - Fungsi.....	100
A. Sintak fungsi	100
B. Memanggil Fungsi	101
C. Jenis-jenis Fungsi.....	103
D. Parameter	106
E. Parameter Wajib.....	108
F. Parameter Opsional	109
G. Parameter Tidak Berurutan	111
H. Parameter *args.....	113
I. Parameter **kwargs	113
Bab 11 - Array	115
A. Array didalam Bahasa Python	116
B. Fungsi Array didalam Bahasa Python.....	118
C. Bagaimana cara menginisialisai Array dalam Python.....	119
Bab 12 - Algoritma Rekursi.....	126
A. Kelebihan Rekursi	127

B. Mendefinisikan Rekursi.....	128
C. Fase pada Rekursi	129
D. Membuat Fungsi Rekursi dengan Phyton	130
E. Studi Kasus 1 (Menampilkan Faktorial)	132
F. Studi Kasus 2 (Menghitung Fibonacci).....	134
G. Studi Kasus 3 (Menghitung Bilangan Berpangkat)	138
DAFTAR PUSTAKA	140
TENTANG PENULIS.....	144

x

Konsep Dasar Algoritma

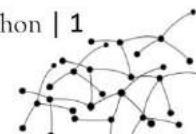
A. Pengertian Algoritma

Algoritma adalah metode atau langkah-langkah yang dirancang secara teratur dan berurutan untuk memecahkan atau menyelesaikan masalah melalui instruksi atau tindakan.

Algoritma dapat digunakan tidak hanya untuk menyelesaikan masalah komputer, tetapi juga untuk menyelesaikan masalah sehari-hari yang membutuhkan serangkaian proses atau langkah proses. Untuk lebih memahami apa itu algoritma, kita pelajari artinya dari beberapa sumber.

Pada beberapa sumber buku diperoleh pengertian dari algoritma di antaranya :

1. Algoritma menurut (Kani, 2020, 1.19) adalah suatu upaya dengan urutan operasi yang disusun secara logis dan sistematis untuk menyelesaikan suatu masalah untuk menghasilkan suatu output tertentu.
 2. Algoritma berasal dari kata algoris dan ritmis yang pertama kali diperkenalkan oleh Abu Ja'far Muhammad Ibn Musa Al Khwarizmi pada 825 M di



dalam buku Al-Jabr Wa-al Muqabla. Dalam bidang pemrograman, algoritma didefinisikan sebagai metode yang terdiri dari serangkaian langkah yang terstruktur dan sistematis untuk menyelesaikan masalah dengan bantuan komputer (Jando & Nani, 2018, 5).

3. Algoritma menurut Munir & Lidya, 2016, 5) adalah urutan langkah-langkah untuk menyelesaikan suatu persoalan.
4. Algoritma merupakan sekumpulan instruksi atau langkah-langkah yang dituliskan secara sistematis dan digunakan untuk menyelesaikan masalah / persoalan logika dan matematika dengan bantuan komputer (Sismoro, 2005, 29).

Algoritma Pemrograman adalah langkah atau instruksi sistematis yang dipergunakan dalam perhitungan atau pemecahan masalah. Dalam aktivitas pemrograman, algoritma digunakan untuk membuat dan mengembangkan program menggunakan alur logika tertentu.

Selain itu, algoritma pemrograman dalam praktiknya terdiri dari tiga tipe konstruksi, yakni *conditional*, *linear sequence* dan *looping*. Ketiganya merupakan algoritma atau logika umum yang sering digunakan dalam pembuatan program.

1. **Conditional** adalah logika yang menjalankan keputusan dari dua atau lebih percabangan berdasarkan kondisi tertentu.

2. **Linear sequence** adalah logika sekuensial yang hasil akhirnya diambil berdasarkan prosedur atau step by step.
3. **Looping** adalah logika yang menjalankan perintah secara berulang-ulang.

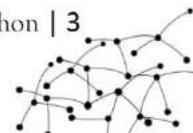
B. Fungsi Algoritma Pemograman

Secara garis besar algoritma berfungsi untuk memecahkan masalah. Seperti yang telah disebutkan bahwa logika merupakan aspek penting yang menjadi dasar pembuatan program. Adapun fungsi lain algoritma pemograman diantaranya :

1. Memecahkan masalah pada program yang melibatkan perhitungan - perhitungan rumit.
2. Alur prosesnya jelas sehingga kamu dapat menemukan titik kesalahan secara akurat ketika mendapatkan bug atau error.
3. Penggunaan yang lebih efektif dan efisien karena algoritma dapat menyederhanakan logika dan alur kerja program.
4. Meminimalisir terjadinya pengulangan penulisan kode yang sama.
5. Memudahkan programmer ketika menambahkan fitur baru dan mengupdate program.

C. Jenis - Jenis Algoritma Pemograman

Setelah memahami pengertian algoritma pemrograman dan cara kerjanya, kamu juga perlu mengetahui jenis-jenisnya. Jika kamu ingin menjadi seorang



programmer, setidaknya ada enam jenis algoritma pemrograman yang wajib dipahami, di antaranya:

1. Algoritma Recursive

Algoritma recursive adalah jenis algoritma yang memecahkan suatu permasalahan sedikit demi sedikit. Proses penyelesaian algoritma jenis ini dilakukan dengan cara membagi masalah menjadi beberapa kondisi yang serupa. Selain itu, algoritma recursive terbagi menjadi empat tipe, di antaranya:

- a. **Algoritma dinamis** adalah logika yang memakai teknik memorisasi, dimana hasil penyelesaian masalah akan disimpan dalam memori agar dapat digunakan kembali di masa yang akan datang.
- b. **Algoritma backtracking** adalah tipe logika recursive yang proses penyelesaian masalahnya dilakukan secara bertahap dengan mengeliminasi solusi yang tidak diperlukan.
- c. **Algoritma greedy** merupakan kebalikan dari algoritma dinamis, dimana logika tipe ini tidak akan mempertimbangkan hasil penyelesaian sebelumnya dalam menentukan keputusan.
- d. **Algoritma divide and conquer** adalah tipe yang membagi masalah menjadi dua bagian, yakni masalah itu sendiri dan metode penyelesaiannya.

2. Algoritma Sorting

Sesuai namanya, algoritma sorting adalah jenis logika pemrograman yang digunakan untuk menyusun data berdasarkan urutan tertentu. Misalnya, mengurutkan data dari angka kecil ke besar atau

berdasarkan abjad dari A sampai Z. Contoh algoritma jenis ini yaitu selection sort, insertion sort, merge sort, dan bubble sort.

3. Algoritma Hashing

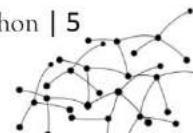
Algoritma hashing adalah jenis algoritma yang berfungsi untuk mencari data berdasarkan query pencarian sekaligus mencocokkannya dengan ID key yang sudah ditentukan. Contohnya penerapan algoritma hashing yaitu penggunaan verifikasi kata sandi ketika login ke akun tertentu.

4. Algoritma Searching

Algoritma searching adalah jenis logika yang berfungsi untuk mencari data tertentu berdasarkan query yang spesifik. Dalam hal ini, proses pencarian mencakup keseluruhan data termasuk yang sudah disortir ataupun belum. Contoh algoritma searching yaitu pencarian angka linier atau biner.

5. Algoritma Randomized

Algoritma randomized adalah logika pemrograman yang memanfaatkan nomor random guna menentukan langkah yang akan diambil selanjutnya. Metode pemecahan seperti ini bertujuan untuk meminimalisir kerumitan pemrograman. Contohnya, pemanfaatan nomor random yang muncul untuk menentukan pivot.



D. Syarat Algoritma

1. Agar mendapatkan hasil yang benar dan berakurasi tinggi maka harus memiliki tingkat kepercayaan yang tinggi (reability).
2. Melakukan suatu proses secara efisien (cost rendah).
3. Frekuensi kalkulasi dan pemrosesan disiapkan dengan cepat dan dalam waktu yang sesingkat-singkatnya.
4. Memiliki sifat yang umum.
5. Permasalahan yang di selesaikan tidak hanya satu masalah saja tetapi permasalahan yang di selesaikan lebih banyak.
6. Dapat di perluas dan dapat dikembangkan.
7. Suatu yang sudah di selesaikan harus di perluas agar lebih jauh lagi dan dikembangkan berdasarkan persyaratan yang sudah ada.
8. Dapat dengan mudah di pahami.

E. Ciri - Ciri Algoritma

1. Dalam menyelesaikan suatu permasalahan algoritma harus memiliki cara atau logika yang tepat.
2. Output yang dihasilkan harus sesuai dan benar untuk jangka yang sangat pendek.
3. Dalam menuliskan algoritma harus dengan bahasa yang standart dan berurut serta rapi agar tidak menimbulkan arti ganda.

4. Algoritma dapat diimplementasikan kedalam bahasa pemrograman jika ditulis dengan format yang mudah dipahami.
5. Dibutuhkan operasi yang harus terdefinisi dengan jelas.
6. Semua permasalahan Algoritma akan selesai jika beberapa proses cara telah dilakukan.

Ada 5 ciri-ciri yang sangat penting menurut Donald E. Knuth, yaitu:

1. Keterbatasan “*Finiteness*” Keterbatasan yaitu suatu algoritma yang akan selesai jika sudah membuat beberapa proses.
2. Kepastian “*Definiteness*” Kepastian yaitu semua cara algoritma harus di jelaskan dengan tepat dan bukan berarti ganda.
3. Masukan “*Input*” Algoriitma mempunyai nilai nol atau lebih beberapa data masukkan (input).
4. Keluaran “*Output*” Algoritma mempunyai nilai nol atau lebih beberapa data hasil keluaran (output).
5. Efektifitas (Effectiveness), Langkah-langkah algoritma harus efektif dan dikerjakan dalam waktu yang wajar.

F. Struktur Dasar Algoritma

1. Algoritma Sekuensial

Algoritma Sekuensial (*Sequence Algorithm*) adalah algoritma yang langkah-langkahnya dikerjakan atau dieksekusi secara urut dari awal hingga akhir sesuai dengan urutannya.



2. Algoritma Perulangan (*Looping Algorithm*)

Algoritma Perulangan atau *Looping Algorithm* adalah sebuah struktur dasar algoritma yang menjalankan beberapa langkah tertentu secara berulang-ulang sampai terpenuhinya suatu kondisi.

a. Struktur FOR

Struktur dasar algoritma perulangan menggunakan instruksi FOR digunakan untuk mengulang satu baris instruksi atau beberapa baris instruksi sampai jumlah perulangan yang disyaratkan terpenuhi. Ciri-ciri utama struktur perulangan menggunakan FOR adalah terdapat nilai awal dan nilai akhir yang menunjukkan syarat yang harus terpenuhi.

b. Struktur WHILE

Struktur *looping* dengan menggunakan WHILE berfungsi hampir mirip dengan FOR yaitu mengulang satu baris instruksi atau beberapa baris instruksi selama syarat yang ditentukan masih terpenuhi. Ciri-ciri utama dari struktur WHILE adalah syarat yang ditentukan akan diuji lebih dahulu sebelum instruksi-instruksi dieksekusi dalam perulangan.

c. Struktur DO...WHILE

Struktur *looping* dengan DO...WHILE digunakan untuk mengulangi satu baris instruksi atau beberapa baris instruksi sampai syarat yang ditetapkan tidak terpenuhi. Ciri-ciri utama dari struktur DO...WHILE ialah syarat akan diuji setelah instruksi dikerjakan seluruhnya atau bisa kita katakan pengujian pada syarat dilakukan dibelakang.

3. Algoritma Percabangan (*Conditional Algorithm*)

Algoritma percabangan atau Algoritma bersyarat adalah algoritma yang menjalankan instruksi selanjutnya apabila syarat yang ditetapkan sudah terpenuhi. Pada struktur ini tidak setiap instruksi akan dikerjakan, instruksi yang dikerjakan hanya yang memenuhi syarat saja. Pada bahasa pemrograman struktur ini sering digunakan menggunakan instruksi IF-THEN atau lebih dikenal instruksi jika-maka.

a. Struktur IF Sederhana

Bentuk dari struktur IF sederhana adalah **IF** (Syarat) **THEN** (Instruksi).

b. Struktur IF...THEN...ELSE...

Pada struktur ini, terdapat dua kemungkinan instruksi yang akan dikerjakan berdasarkan hasil dari pengujian. Contoh jika syarat yang diujikan memperoleh hasil benar maka instruksi_1 dikerjakan, namun jika bernilai salah maka instruksi_2 yang dikerjakan.

c. IF Bersarang

Untuk struktur yang satu ini kita perlu belajar logika dan ketelitian, satu alasan yang pasti adalah struktur ini sering dipakai untuk tes kerja dalam bidang IT terutama pekerjaan yang berkaitan dengan perancangan sistem. Pada struktur ini juga kemungkinan akan banyak instruksi yang dikerjakan berdasarkan hasil pengujian,



0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 1
1 0 0 0 0 1 0 1 1 1 0 0 1 0 1 0 1 1 0 0 0
0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0
1 1 1 0 1 0 1 0 1 1 0 0 1 0 1 0 1 1 1 1
0 0 1 0 0 1 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0
0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1
0 0 1 1 1 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 0
1 1 1 0 1 0 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0
0 0 1 1 1 0 0 1 0 0 1 1 1 0 1 0 1 0 0 1 1
0 1 1 1 1 0 0 1 0 0 1 1 1 0 1 0 1 1 0 0 0
1 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 0 0 0
1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 0 0
1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 0 0
0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 0
1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 0
0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0
1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0
0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1
0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1
0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1

2

Notasi Algoritma

A. Pengertian Notasi Algoritma

Notasi algoritma adalah cara atau bentuk dalam menuliskan sebuah algoritma. Keberadaannya tidak dituliskan ke dalam bahasa pemrograman, tetapi bisa diterjemahkan ke dalam bahasa pemrograman tersebut. Menurut Rinaldi Munir dalam buku Algoritma dan Pemrograman (1997) Notasi Algoritma adalah desain berisi urutan langkah-langkah pencapaian solusi yang tidak tergolong bahasa pemrograman apapun.

Dengan begitu, notasi algoritma itulah menjadi dasar dari dibuatnya sebuah program komputer dan dapat diterjemahkan ke dalam berbagai bahasa pemrograman. Secara umum, keberadaannya juga terdiri dari tiga jenis yakni kalimat deskriptif, pseudocode dan flowchart akan dijabarkan pada penjelasan berikutnya.

Secara umum terdapat tiga cara dalam menuliskan algoritma yaitu sebagai berikut:

1. *Deskriptif*
2. *Pseudocode*
3. *Flowchart*

1. Kalimat Deskriptif

Notasi algoritma dengan menggunakan kalimat deskriptif disebut juga notasi alami. Notasi algoritma deskriptif dilakukan dengan cara menuliskan intruksi-intruksi yang musti dilaksanakan dalam bentuk untaian kalimat deskriptif dengan menggunakan bahasa yang jelas. Notasi deskriptif ini disarankan untuk algoritma yang pendek karena apabila untuk algoritma yang panjang notasi deskriptif kurang efektif. Secara garis besar notasi deskriptif tersusun atas tiga bagian utama, yaitu:

a. Bagian judul (*header*)

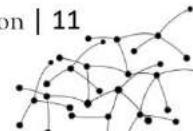
Merupakan bagian yang terdiri atas nama algoritma dan penjelasan atau spesifikasi algoritma tersebut.

b. Bagian deklarasi (*kamus*)

Merupakan bagian untuk mendefinisikan semua nama yang digunakan pada algoritma dapat berupa variabel, konstanta, tipe ataupun fungsi

c. Bagian deskripsi

Merupakan bagian inti pada struktur algoritma yang berisi uraian langkah-langkah penyelesaian masalah. Setiap bagian diatas diberikan dengan komentar untuk memperjelas maksud teks yang dituliskan. Berikut ini adalah beberapa contoh algoritma yang penulisannya ditulis secara deskripsi.



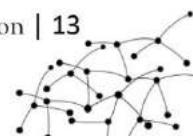
- 1) ***Algoritma Mengirim Surat:***
 - a) Ketik atau tulis surat
 - b) Siapkan sampul surat atau amplop
 - c) Masukkan surat ke dalam amplop yang tersedia
 - d) Lem amplop surat dengan baik
 - e) Tuliskan alamat pengiriman surat, jika tidak ingat, lebih dahulu ambil buku alamat & cari alamat yg dituju, lalu tulis alamat amplop surat.
 - f) Beli dan tempelkan perangko pada amplop
 - g) Pergi ke kantor pos dan membawa surat untuk diserahkan kepada pegawai pos setempat.
- 2) ***Algoritma Mengambil Uang di ATM:***
 - a) Mulai
 - b) Memasukan kartu dimesin ATM
 - c) Pilih bahasa
 - d) Masukan *password*
 - e) Pilih menu Tarik tunai
 - f) Pilih jenis rekening
 - g) Masukan jumlah uang yang ingin diambil
 - h) Jika saldo mencukupi, mesin ATM akan mengeluarkan uang sesuai dengan jumlah-nya, Jika tidak maka kembali ke Masukkan jumlah uang yang ingin diambil.
 - i) Ambil uangnya

- j) Ambil kartu ATM
 - k) Selesai
- 3) ***Algoritma Mengirim SMS***
- a) Buka Menu SMS
 - b) Pilih kontak / nomor telepon
 - c) Ketikan pesan
 - d) Tekan tombol kirim
 - e) SMS akan terkirim

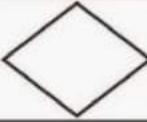
2. Flowchart

Flowchart bisa diartikan sebagai diagram alir atau bagan yang menggambarkan langkah-langkah dan struktur suatu algoritma atau program. Ilustrasi ini dinyatakan dalam simbol, setiap simbol mempunyai makna tertentu untuk proses tertentu. Simbol-simbol *flowchart* yang umumnya digunakan adalah simbol-simbol *flowchart* standart yang dikeluarkan oleh ANSI dan ISO. Aturan umum dalam *flowchart*:

- a. Semua simbol dari *flowchart* dihubungkan oleh garis aliran (*arrows*) yang mengindikasikan arah aliran bukan garis biasa.
- b. Garis aliran memasuki bagian atas simbol dan keluar dari bagian bawah, kecuali untuk simbol keputusan (decision), yang memiliki garis aliran yang keluar dari bawah atau samping.
- c. Aliran proses bergerak dari atas ke bawah.
- d. Awal dan akhir pada *flowchart* disimbolkan dengan terminal.



Tabel. Simbol - simbol dalam Flowchart

	Flow Direction symbol Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga connecting line.
	Terminator Symbol Yaitu simbol untuk permulaan (start) atau akhir (stop) dari suatu kegiatan
	Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses dalam lembar / halaman yang sama.
	Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses pada lembar / halaman yang berbeda.
	Processing Symbol Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer
	Simbol Manual Operation Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh computer
	Simbol Decision Simbol pemilihan proses berdasarkan kondisi yang ada.
	Simbol Input-Output Simbol yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya

Berikut ini adalah keuntungan dari metode penulisan algoritma dengan *flowchart*:

- Memberikan kemudahan kepada setiap orang yang memnbaca untuk membaca dan memahami algoritma.

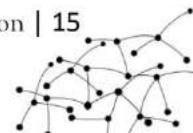
- b. Sangat cocok untuk digunakan pada studi kasus algoritma yang kerumitannya pada level tingkat rendah.
- c. Aliran proses programnya sangat rinci dengan pemodelan secara visual dengan symbol.
- d. Namun, tidak hanya kemudahannya saja, *flowchart* juga mempunyai kekurangan diantaranya:
- e. Tidak cocok untuk perepresentasikan alur program yang komplek.
- f. Membutuhkan tools khusus untuk merancang algoritmanya.
- g. Penjelasan dalam alur proses tidak detail karena keterbatasan ruang.

3. Pseudo Code

Pseudecode merupakan cara penulisan algoritma yang menyerupai bahasa pemrograman tingkat tinggi. Pada umumnya notasi *pseudecode* menggunakan bahasa yang mudah dimengerti secara umum dan juga lebih ringkas dari pada algoritma. *Pseudecode* berisi deskripsi dari algoritma pemrograman komputer yang menggunakan struktur sederhana dari beberapa bahasa pemrograman tetapi bahasa tersebut hanya ditujukan agar bisa terbaca dan dimengerti manusia.

Sehingga *pseudecode* tidak dipahami oleh komputer.

Agar notasi *pseudecode* dapat dimengerti oleh komputer maka musti diterjemahkan ke dalam sintaks bahasa pemrograman tertentu. Pada notasi



pseudecode tidak aturan tertentu yang resmi. Disarankan untuk menggunakan kata kunci yang umum digunakan seperti *if, then, else, while, do, for, repeat* dan lainnya.

Penulisan Pseudo code memiliki beberapa pedoman sebagai berikut:

- a. Memiliki bagian Header / Kepala yang menunjukkan judul dari algoritma, Komentar dan Deklarasi.
- b. Memiliki badan algoritma
- c. Memiliki bagian akhir algoritma yang menandakan bagian akhir dari suatu algoritma.
- d. Untuk menuliskan komentar diawali dengan karakter “{“ dan diakhiri dengan karakter “}”.

B. Jenis - Jenis Flowchart

Flowchart itu sendiri terdiri dari lima jenis, masing-masing jenis memiliki karakteristik dalam penggunaanya. Berikut adalah jenis-jenisnya:

1. Flowchart dokumen

Flowchart dokumen (*document flowchart*) atau bisa juga disebut dengan *paperwork flowchart*. Flowchart dokumen berfungsi untuk menelusuri alur form dari satu bagian ke bagian yang lain, termasuk bagaimana laporan diproses, dicatat, dan disimpan.

2. Flowchart program

Flowchart program Flowchart ini menggambarkan secara rinci prosedur dari proses program. Flowchart program terdiri dari dua macam, antara

lain: flowchart logika program (*program logic flowchart*) dan flowchart program komputer terinci (*detailed computer program flowchart*).

3. Flowchart proses

Flowchart proses adalah cara penggambaran rekayasa industrial dengan cara merinci dan menganalisis langkah-langkah selanjutnya dalam suatu prosedur atau sistem.

4. Flowchart sistem

Flowchart sistem adalah flowchart yang menampilkan tahapan atau proses kerja yang sedang berlangsung di dalam sistem secara menyeluruh. Selain itu flowchart sistem juga menguraikan urutan dari setiap prosedur yang ada di dalam sistem.

5. Flowchart skematik

Flowchart skematik ini menampilkan alur prosedur suatu sistem, hampir sama dengan flowchart sistem. Namun, ada perbedaan dalam penggunaan simbol-simbol dalam menggambarkan alur. Selain simbol-simbol, flowchart skematik juga menggunakan gambar-gambar komputer serta peralatan lainnya untuk mempermudah dalam pembacaan flowchart untuk orang awam.

C. Karakteristik Algoritma

Agar memahami lebih jauh mengenai algoritma tersebut, maka penting untuk mengetahui bagaimana karakteristik yang dimiliki terlebih dahulu. Seperti sudah kami singgung sebelumnya bahwa algoritma merupakan



sebuah perintah terstruktur dan logis, maka mempunyai sejumlah karakter diantaranya adalah.

1. **Teratur** Algoritma mempunyai ciri-ciri teratur atau *well-ordered*. Maka hal tersebut dapat membuktikan bahwa setiap langkah atau proses sudah tertata dalam urutan jelas sekaligus teratur.
2. **Tidak Ambigu** Karakteristik berikutnya adalah *unambiguous* alias tidak ambigu. Disinilah oprasi yang telah dijelaskan tersebut bisa dipahami oleh komputer tanpa adanya penyederhanaan lebih lanjut.
3. **Komputasi Efektif** Karakteristik terakhir adalah *effectively computable* alias komputasi efektif. Dimana komputer nantinya harus benar-benar bisa melakukan operasi sesuai dengan algoritma.

3

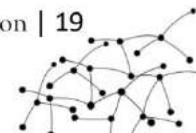
Pemrograman Python

Bab ini bertujuan untuk memberikan gambaran umum tentang bahasa pemrograman Python, menggali latar belakang, dan memberikan ulasan singkat tentang instalasi dan cara menjalankan kode Python.

A. Latar Belakang Python

Python adalah bahasa pemrograman tingkat tinggi yang pertama kali dikembangkan pada awal 1990-an oleh Guido van Rossum. Nama Python diambil dari penggemar acara komedi televisi "Monty Python's Flying Circus." Python dikenal dengan sintaksis yang jelas, mudah dibaca, dan mudah dipahami, membuatnya sangat populer di kalangan pengembang perangkat lunak. Bahasa ini memiliki filosofi desain yang mengedepankan kejelasan, kesederhanaan, dan keterbacaan kode.

Python memiliki berbagai kegunaan, termasuk pengembangan web, pemrosesan data, kecerdasan buatan (AI), pemodelan matematika, pengembangan permainan, dan masih banyak lagi. Keberhasilan Python juga didukung oleh ekosistem yang kuat, termasuk berbagai pustaka dan kerangka kerja (framework) yang mendukung berbagai jenis pengembangan.



B. Instalasi Python

Untuk memulai dengan Python, perlu menginstal interpreter Python di sistem komputer. Interpreter ini akan memungkinkan untuk menjalankan kode Python pada komputer. Python tersedia secara gratis dan dapat diunduh dari situs web resmi Python:

(<https://www.python.org/downloads/>).



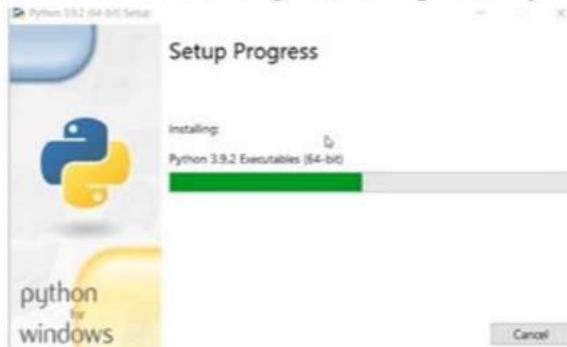
Gambar 1. Mengunduh Aplikasi Python



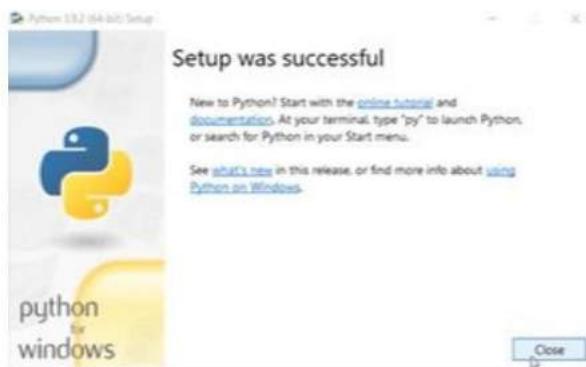
Gambar 2. Proses Menyimpan Hasil Unduhan Aplikasi Python



Gambar 3. Proses Penginstalan Aplikasi Python



Gambar 4. Proses Penginstalan Aplikasi Python Sementara Berjalan



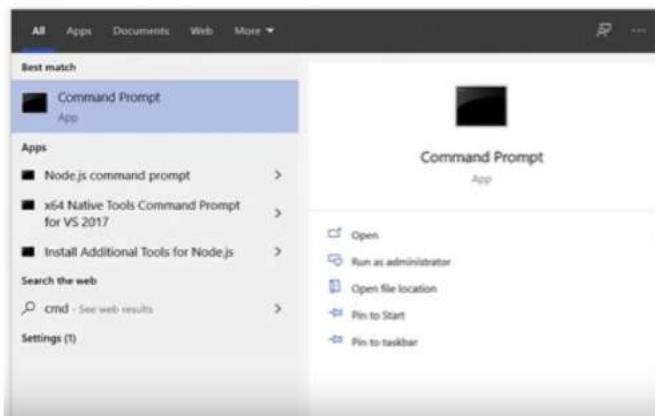
Gambar 5. Proses Penginstalan Berhasil Dilakukan

Proses instalasi Python relatif mudah dan mengikuti petunjuk yang disediakan pada situs web resmi. Selain itu, terdapat berbagai distribusi Python yang sudah mencakup banyak pustaka yang berguna, seperti Anaconda dan Miniconda, yang dapat mempermudah pengelolaan pustaka tambahan.

C. Menjalankan Kode Python

Setelah menginstal Python, dapat mulai menjalankan kode Python melalui terminal atau lingkungan pengembangan yang sesuai seperti IDLE, Jupyter Notebook, atau lingkungan pengembangan terintegrasi (IDE) seperti PyCharm.

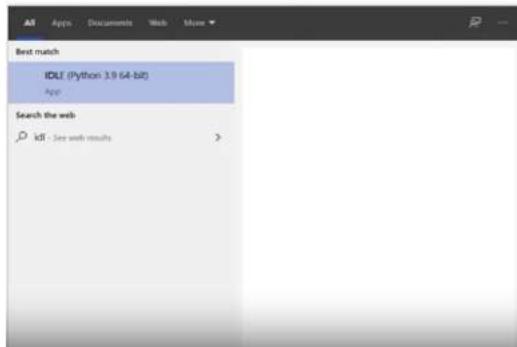
Untuk menjalankan kode Python, buat file dengan ekstensi .py (misalnya, program.py), tulis kode Python di dalamnya, dan jalankan dengan perintah python nama_file.py melalui terminal. Alternatifnya, jika menggunakan lingkungan pengembangan, cukup tekan tombol 'Run' atau ikuti petunjuk khusus lingkungan tersebut.



Gambar 6. Menjalankan Command Prompt

```
C:\Users\          >python --version
Python 3.9.2
C:\Users\          >python
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>> 5 * 4
20
>>> x = 10
>>> x
10
>>> quit()
C:\Users\
```

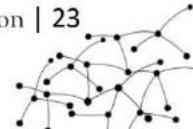
Gambar 7. Menjalankan Kode Python Melalui Command Prompt



Gambar 8. Menjalankan IDLE Python

```
File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>>
```

Gambar 9. Menjalankan Kode Python Pada IDLE



D. Mengapa Memilih Python?

Python menjadi bahasa pemrograman pilihan bagi banyak pengembang karena sejumlah alasan. Beberapa di antaranya meliputi:

1. Keterbacaan Kode: Sintaksis Python yang sederhana dan mudah dibaca memungkinkan pengembang untuk lebih fokus pada logika program daripada memikirkan tata bahasa yang rumit.
2. Mudah Dipelajari dan Dipahami: Python merupakan salah satu bahasa yang paling mudah dipelajari, membuatnya ideal bagi pemula untuk memahami konsep pemrograman dasar.
3. Kaya Akan Pustaka: Python memiliki banyak pustaka dan modul yang memperluas fungsionalitasnya untuk berbagai keperluan, dari pengembangan web hingga pemrosesan data.
4. Dukungan Komunitas: Python memiliki komunitas yang besar dan aktif yang membantu dalam menyelesaikan masalah, memberikan tutorial, dan memperbarui bahasa ini dengan peningkatan berkala.
5. Fleksibel dan Kuat: Python dapat digunakan untuk berbagai jenis proyek, dari aplikasi web hingga analisis data, dan memiliki kemampuan untuk mengintegrasikan dengan bahasa lain.

E. Kontribusi dan Pengguna Python

Python memiliki ekosistem yang kaya dan beragam, dengan kontribusi dari individu, komunitas, dan organisasi besar. Beberapa perusahaan terkemuka yang menggunakan Python dalam pengembangan perangkat lunak mereka meliputi Google, Facebook, Instagram,

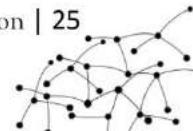
Netflix, dan Dropbox. Kontribusi terus-menerus dari pengguna dan komunitas Python berperan penting dalam pertumbuhan dan peningkatan bahasa ini.

F. Peran Python dalam Industri Teknologi Modern

Python memiliki peran yang signifikan dalam dunia teknologi modern. Digunakan di berbagai industri, termasuk teknologi informasi, kecerdasan buatan, ilmu data, dan pengembangan perangkat lunak, Python menjadi bahasa serbaguna yang memungkinkan solusi untuk berbagai tantangan. Pengembangan web dengan framework seperti Django dan Flask, analisis data dengan Pandas dan NumPy, atau kecerdasan buatan dengan TensorFlow dan PyTorch adalah beberapa contoh penerapan Python di industri.

G. Pengembangan Python dan Versi Utama

Python terus berkembang dan memperbarui diri dengan merilis versi baru secara berkala. Versi utama saat ini adalah Python 3.x, yang memiliki perbedaan signifikan dengan Python 2.x. Perbedaan ini meliputi perbaikan kesalahan, peningkatan fitur, dan peningkatan keamanan. Oleh karena itu, sangat disarankan untuk menggunakan Python versi terbaru untuk memanfaatkan semua peningkatan tersebut.



4

Operator Operator Python

Operator yang digunakan dalam Bahasa Pemrograman Python adalah simbol atau tanda yang digunakan untuk melakukan suatu operasi pada suatu nilai atau variable atau ekspresi, yang berfungsi untuk membuat kode program Python semakin mudah (Jubilee Enterprise, 2019).

A. Operator Aritmatika

Operator Aritmatika adalah simbol yang digunakan untuk melakukan operasi matematika atau aritmatika pada bilangan atau angka. Operator yang tersedia pada python sama dengan operator aritmatika pada Bahasa pemrograman lainnya (Stevens and Boucher, 2015). Tabel 1 berikut adalah operator aritmatika dalam python (Malhotra, 2020) :

Table 1 Operator Aritmatika

Operator	Nama	Contoh Penggunaan
+	Penambahan / Addition	$x + y$
-	Pengurangan Subtraction	$x - y$

*	Perkalian / Multiplication	$x * y$
/	Pembagian / Division	x / y
%	Modulus	$x \% y$
**	Pangkat / Exponentiation	$x ** y$
//	Pembagian Bulat / Floor Division	$x // y$

Contoh penggunaan operator aritmatika dalam Bahasa Python, sebagai berikut :

Contoh Penggunaan Operator Aritmatika

```

angkaA = 10
angkaB = 5

#Operator Penambahan ( + )
hasilTambah = angkaA + angkaB
print("Hasil Penambahan:", hasilTambah)

#Operator Pengurangan ( - )
hasilKurang = angkaA - angkaB
print("Hasil Pengurangan:", hasilKurang)

#Operator Perkalian ( * )
hasilKali = angkaA * angkaB
print("Hasil Perkalian:", hasilKali)

#Operator Pembagian ( / )
hasilBagi = angkaA / angkaB
print("Hasil Pembagian:", hasilBagi)

#Operator Modulus ( % )
hasilModulus = angkaA % angkaB
print("Hasil Modulus:", hasilModulus)

#Operator Pangkat ( ** )
hasilPangkat = angkaA ** angkaB
print("Hasil Pangkat:", hasilPangkat)

#Operator Pembagian Bulat ( // )
hasilBagiB = angkaA // angkaB
print("Hasil Pembagian Bulat:", hasilBagiB)

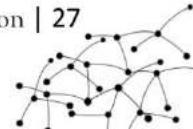
```

```

Hasil Penambahan: 15
Hasil Pengurangan: 5
Hasil Perkalian: 50
Hasil Pembagian: 2.0
Hasil Modulus: 0
Hasil Pangkat: 100000
Hasil Pembagian Bulat: 2

```

Gambar 1. Contoh Penggunaan Operator Aritmatika dalam Python



B. Operator Perbandingan

Operator perbandingan atau Comparison operator adalah operator yang dapat digunakan untuk membandingkan dua nilai atau dua ekspresi dan akan menghasilkan nilai True atau False (*Operators et al.*, no date). Operator perbandingan ini biasanya digunakan untuk menentukan hubungan dari suatu nilai. Berikut adalah operator perbandingan yang ada pada Python:

Table 2 Operator Perbandingan

Operator	Nama	Contoh Penggunaan
<code>==</code>	Equal / Sama Dengan	<code>x == y</code>
<code>!=</code>	Not Equal / Tidak Sama Dengan	<code>x != y</code>
<code>></code>	Greater Than / Lebih Besar	<code>x > y</code>
<code><</code>	Less Than / Lebih Kecil	<code>x < y</code>
<code>>=</code>	Greater Than or Equal / Lebih Besar Sama Dengan	<code>x >= y</code>
<code><=</code>	Less Than or Equal / Lebih Kecil Sama Dengan	<code>x <= y</code>

Berikut merupakan contoh penggunaan operator perbandingan dengan menggunakan kode program Python (*Addition and Floating*, no date) :

Contoh Penggunaan Operator Perbandingan

```
angkaX = 5
angkaY = 7

#Operator Sama Dengan ( == )
if angkaX == angkaY:
    print("angka X sama dengan angka Y")
else:
    print("angka X tidak sama dengan angka Y")

#Operator Tidak Sama Dengan ( != )
if angkaX != angkaY:
    print("angka X tidak sama dengan angka Y")
else:
    print("angka X sama dengan angka Y")

#Operator Kurang Dari ( < )
if angkaX < angkaY:
    print("angka X kurang dari angka Y")
else:
    print("angka X tidak kurang dari angka Y")

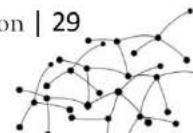
#Operator Lebih Dari ( > )
if angkaX > angkaY:
    print("angka X lebih dari angka Y")
else:
    print("angka X tidak lebih dari angka Y")

#Operator Kurang Dari Sama Dengan ( <= )
if angkaX <= angkaY:
    print("angka X kurang dari sama dengan angka Y")
else:
    print("angka X tidak kurang dari angka Y")

#Operator Lebih Dari Sama Dengan ( >= )
if angkaX >= angkaY:
    print("angka X lebih dari sama dengan angka Y")
else:
    print("angka X tidak lebih dari angka Y")
```

angka X tidak sama dengan angka Y
angka X tidak sama dengan angka Y
angka X kurang dari angka Y
angka X tidak lebih dari angka Y
angka X kurang dari sama dengan angka Y
angka X tidak lebih dari angka Y

Gambar 2. Contoh Penggunaan Operator Perbandingan dalam Python



C. Operator Logika

Dalam Bahasa pemrograman Python, operator logika digunakan untuk membandingkan dua nilai atau variable atau ekspresi, dan kemudian akan menghasilkan atau mengembalikan logika Boolean **True** atau **False** berdasarkan kondisi tersebut (Operators *et al.*, no date) (Stevens and Boucher, 2015). Berikut penjelasan mengenai operator logika dalam Python :

Table 3 Operator Logika

Operator	Keterangan	Contoh Penggunaan
and	Mengembalikan nilai True , jika kedua nilai atau pernyataan True	$x > 10 \text{ and } x < 20$
or	Mengembalikan nilai True , jika salah satu nilai atau pernyataan True	$x > 10 \text{ or } x < 20$
not	Kebalikan dari and , yaitu mengembalikan nilai False , jika kedua nilai atau pernyataan True	<code>Not(x > 10 and x < 20)</code>

Contoh penggunaan operator logika dalam Bahasa Python, sebagai berikut :

Contoh Penggunaan Operator Logika

```
usia = 27
IPK = 3.25

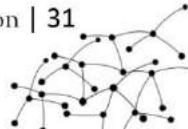
#Operator and
if usia >= 24 and IPK >= 3.00:
    print("Selamat, Anda diterima untuk pekerjaan ini")
else:
    print("Maaf, Anda tidak memenuhi syarat untuk pekerjaan ini")

#Operator or
if usia >= 24 or IPK >= 3.00:
    print("Selamat, Anda diterima untuk pekerjaan ini")
else:
    print("Maaf, Anda tidak memenuhi syarat untuk pekerjaan ini")

#Operator not
lulus = False
if not lulus:
    print("Maaf, Anda tidak memenuhi syarat untuk pekerjaan ini.")
else:
    print("Selamat, Anda diterima untuk pekerjaan ini")
```

```
Selamat, Anda diterima untuk pekerjaan ini
Selamat, Anda diterima untuk pekerjaan ini
Maaf, Anda tidak memenuhi syarat untuk pekerjaan ini.
```

Gambar 3. Contoh Penggunaan Operator Logika dalam Python



D. Operator Penugasan / Assignment

Operator assignment atau operator penugasan adalah operator yang digunakan untuk memasukkan suatu nilai ke dalam suatu variable. Simbol yang digunakan untuk operator penugasan dalam python adalah tanda sama dengan (=) (Paul Barry, 2016). Berikut adalah operator perbandingan yang dapat digunakan dalam python :

Table 4 Operator Penugasan

Operator	Contoh Penggunaan	Persamaan dengan
=	x = 5	$x = x * 3$
+=	x += 7	$x = x + 7$
-+	x -= 7	$x = x - 7$
*=	x *= 7	$x = x * 7$
/=	x /= 7	$x = x / 7$
%=	x %= 7	$x = x \% 7$
//=	x //= 7	$x = x // 7$
**=	x **= 7	$x = x ** 7$
&=	x &= 7	$x = x \& 7$
=	x = 7	$x = x 7$
^=	x ^= 7	$x = x ^ 7$
>>=	x >>= 7	$x = x >> 7$
<<=	x <<= 7	$x = x << 7$

Untuk membaca operator assignment dilakukan dari kanan ke kiri, misalnya $y = 10$, maka berarti “masukkan nilai 10 ke dalam variable y.

Contoh penggunaan operator penugasan lainnya, sebagai berikut :

Contoh Penggunaan Operator Penugasan

```
#Operator Penugasan ( = )
x = 25
print("Nilai x adalah",x)

#Operator Penugasan dan Penambahan ( += )
x += 5
print("Nilai x setelah penambahan 5 adalah",x)

#Operator Penugasan dan Pengurangan ( -= )
x -= 5
print("Nilai x setelah pengurangan 5 adalah",x)

#Operator Penugasan dan Perkalian ( *= )
x *= 5
print("Nilai x setelah perkalian 5 adalah",x)

#Operator Penugasan dan Pembagian ( /= )
x /= 5
print("Nilai x setelah pembagian 5 adalah",x)

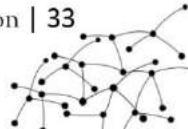
#Operator Penugasan dan Pembagian Bulat( //=
)
x //=" 5
print("Nilai x setelah pembagian bulat 5 adalah",x)

#Operator Penugasan dan Modulus ( %= )
x %= 5
print("Nilai x setelah modulus 5 adalah",x)

#Operator Penugasan dan Pangkat ( **= )
x **=" 5
print("Nilai x setelah pangkat 5 adalah",x)

Nilai x adalah 25
Nilai x setelah penambahan 5 adalah 30
Nilai x setelah pengurangan 5 adalah 25
Nilai x setelah perkalian 5 adalah 125
Nilai x setelah pembagian 5 adalah 25.0
Nilai x setelah pembagian bulat 5 adalah 5.0
Nilai x setelah modulus 5 adalah 0.0
Nilai x setelah pangkat 5 adalah 0.0
```

Gambar 4. Contoh Penggunaan Operator Penugasan dalam Python



E. Operator Identitas

Operator identitas dalam Python digunakan untuk membandingkan dua objek, apakah memiliki tipe yang sama atau tidak (Paul Barry, 2016). Berbeda dengan operator logika yang membandingkan nilai Boolean, operator identitas biasanya digunakan untuk memeriksa apakah dua buah objek atau variable memiliki identitas yang sama dan mengacu pada lokasi memori yang sama atau berbeda. Berikut merupakan penjelasan dari operator identitas beserta contoh penggunaanya :

Table 5 Operator Identitas

Operator	Keterangan	Contoh Penggunaan
is	Mengembalikan nilai True , jika kedua variable atau objek memiliki identitas yang sama	x is y
is not	Mengembalikan nilai True , jika kedua variable atau objek tidak memiliki identitas yang sama	y is not y

Contoh Penggunaan Operator Identitas

```
#Operator is
x = [2, 4, 5, 6, 8, 11]
y = x #y perujuk pada objek yang sama dengan x

if x is y:
    print("x dan y memiliki identitas yang sama.")
else:
    print("x dan y memiliki identitas yang berbeda.")

#Operator is not
x = [2, 4, 5, 6, 8, 11]
y = x #y perujuk pada objek yang sama dengan x

if x is not y:
    print("x dan y memiliki identitas yang berbeda.")
else:
    print("x dan y memiliki identitas yang sama.")

x dan y memiliki identitas yang sama.
x dan y memiliki identitas yang sama.
```

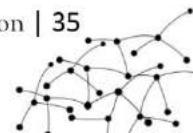
Gambar 5. Contoh Penggunaan Operator Identitas dalam Python

F. Operator Keanggotaan

Pada Python terdapat operator keanggotaan yang digunakan untuk melakukan pemeriksaan apakah suatu nilai merupakan anggota dari suatu *Iterble* (seperti List, Tuple, Set, String, atau Dictionary) atau bukan (Lambert, 2017). Operator ini juga dapat digunakan untuk melakukan pengujian apakah suatu urutan disajikan dalam suatu objek. Berikut penjelasan untuk operator keanggotaan :

Table 6 Operator Keanggotaan

Operator	Keterangan	Contoh Penggunaan
in	Mengembalikan nilai True , jika <i>iterable</i> atau urutan data dengan nilai yang ditentukan terdapat dalam objek yang	x in y



	diperiksa	
not in	Mengembalikan nilai True , jika <i>iterable</i> atau urutan data dengan nilai yang ditentukan tidak terdapat dalam objek yang diperiksa	y not in y

Contoh Penggunaan Operator Keanggotaan

```
#Operator in
daftarFilm = ["kartun", "action", "drama", "horor", "sinetron"]

film = "kartun"

if film in daftarFilm:
    print(f"{film} ada dalam daftar film favorit.")
else:
    print(f"{film} tidak ada dalam daftar film favorit.")

#Operator not in
daftarMinuman = ["teh es", "juice", "susu", "kopi", "teh tarik"]

minuman = "teh es"

if minuman not in daftarMinuman:
    print(f"{minuman} tidak ada dalam daftar menu minuman.")
else:
    print(f"{minuman} ada dalam daftar menu minuman.")

#Operator Keanggotaan dalam Loop
for minuman in daftarMinuman:
    if minuman == "juice":
        print(f"{minuman} ditemukan dalam daftar minuman.")
        break
```

kartun ada dalam daftar film favorit.
 teh es ada dalam daftar menu minuman.
 juice ditemukan dalam daftar minuman.

Gambar 6. Contoh Penggunaan Operator Keanggotaan dalam Python

G. Operator Bitwise

Operator selanjutnya yang terdapat pada Bahasa pemrograman Python adalah operator Bitwise. Operator ini biasanya digunakan untuk melakukan operasi *bit-level* terhadap bilangan bulat integer dalam bentuk format Biner

(Operators *et al.*, no date). Biasanya operator Bitwise berguna pada saat pengembangan perangkat keras atau Hardware atau IoT serta untuk melakukan manipulasi pada level bit. Penggunaan operator Bitwise dapat dilakukan terhadap operator AND, OR, XOR, shift kiri dan shift kanan. Berikut penjelasan mengenai operator Bitwise.

Table 7 Operator Bitwise

Operator	Nama	Keterangan
&	AND bitwise	Menghasilkan 1 jika kedua bit yang sesuai adalah 1, dan 0 jika tidak sesuai
	OR bitwise	Menghasilkan 1 jika salah satu bit yang sesuai adalah 1
^	XOR bitwise	Menghasilkan 1 jika hanya salah satu bit yang sesuai adalah 1. Atau Exclusive dari OR
~	NOT bitwise	Untuk mengubah setiap bit dalam angka dari 0 menjadi 1, dan dari 1 menjadi 0. Disebut juga sebagai negasi bitwise, atau membalikkan semua bit
<<	Zero fill left shift / Shift kiri	Menggeser bit ke kiri, dengan cara memberikan nilai 0 pada bagian kanan



>>	Signed right shift / Shift kanan	Menggeser bit ke kanan, dengan cara memberikan nilai 0 pada bagian kiri
-----------------	----------------------------------	---

Contoh Penggunaan Operator Bitwise

```
#Operator AND ( & )
a = 5 # Representasi biner: 0101
b = 3 # Representasi biner: 0011

hasil = a & b # Hasil dari operasi AND: 0001 (1 dalam desimal)
print("Hasil Operator AND",hasil)

#Operator OR ( | )
x = 12 # Representasi biner: 1100
y = 5 # Representasi biner: 0101

hasil = x | y # Hasil dari operasi OR: 1101 (13 dalam desimal)
print("Hasil Operator OR",hasil)

#Operator XOR ( ^ )
p = 18 # Representasi biner: 10010
q = 9 # Representasi biner: 01001

hasil = p ^ q # Hasil dari operasi XOR: 11011 (27 dalam desimal)
print("Hasil Operator XOR",hasil)

#Operator NOT ( ~ )
angka = 5 # Representasi biner: 00000101

hasil = ~angka # Hasil dari operasi NOT: 11111010 (-6 dalam desimal)
print("Hasil Operator NOT",hasil)

#Operator Shift Kiri ( << )
nilai = 8 # Representasi biner awal: 00001000

hasil = nilai << 2 # Hasil dari shift kiri: 00100000 (32 dalam desimal)
print("Hasil Operator Shift Kiri",hasil)

#Operator Shift Kanan ( >> )
angka = 16 # Representasi biner awal: 00010000

hasil = angka >> 3 # Hasil dari shift kanan: 00000010 (2 dalam desimal)
print("Hasil Operator Shift Kanan",hasil)
```

Hasil Operator AND 1
 Hasil Operator OR 13
 Hasil Operator XOR 27
 Hasil Operator NOT -6
 Hasil Operator Shift Kiri 32
 Hasil Operator Shift Kanan 2

Gambar 7. Contoh Penggunaan Operator Bitwise dalam Python

Pada contoh pertama terdapat parameter $a = 5$ (biner : 0101), parameter $b = 3$ (biner : 0011). Hasil dari $a \& b$ adalah 1 (biner : 0001). Operator bitwise AND menghasilkan bit yang sama diposisi yang sesuai dari dua bilangan.

0101

0011

-----&

0001, maka hasilnya adalah 1 atau 0001.

Pada contoh berikutnya, $x = 12$ (biner : 1100) dan $y = 5$ (biner : 0101), maka hasil $x | y = 13$ (biner : 1101)

1100

0101

-----|

1101, maka hasilnya adalah 13 atau 1101.

Pada contoh berikutnya, $p = 18$ (biner : 10010) dan $y = 9$ (biner : 01001), maka hasil $x ^ y = 27$ (biner : 11011)

10010

01001

-----^

11011, maka hasilnya adalah 27 atau 11011.

Pada contoh selanjutnya, angka = 5 (biner : 00000101), maka hasil operator NOT terhadap x atau -angka = -6 (biner : 11111010)

$00000101 \sim x = 11111010$, atau sama dengan -6.

Untuk contoh penggunaan Pada contoh berikutnya, $p = 18$ (biner : 10010) dan $y = 9$ (biner : 01001), maka hasil $x ^ y = 27$ (biner : 11011)

10010

01001

-----^

11011, maka hasilnya adalah 27 atau 11011.

5

List, Dictionary, Tuple

A. List

Subbab ini menyajikan salah satu tipe bawaan Python yang paling berguna: list.

List Adalah Urutan

Seperti string, list adalah urutan nilai. Dalam sebuah string, nilainya adalah karakter; dalam list, tipenya bisa apa saja. Nilai dalam list disebut **elemen** atau terkadang **item**.

Ada beberapa cara untuk membuat list baru; yang paling sederhana adalah dengan mengapit elemen dalam tanda kurung siku ([dan]):

[10, 20, 30, 40]

'crunchy frog', 'ram bladder', 'lark vomit')

Contoh pertama adalah list empat bilangan bulat. Yang kedua adalah list tiga string. Elemen list tidak harus bertipe sama. List berikut berisi string, float, integer, dan list lainnya:

```
['spam', 2.0, 5, [10, 20]]
```

List di dalam list lain disebut **nested**. List yang tidak berisi elemen disebut list kosong; Anda dapat membuatnya dengan tanda kurung kosong, [].

Anda dapat menetapkan nilai list ke variabel:

In [1]:

```
cheeses = ['Cheddar', 'Edam', 'Gouda']
numbers = [42, 123]
empty = []
print(cheeses, numbers, empty)
```

Out [1]:

```
'Cheddar', 'Edam', 'Gouda'] [42, 123] []
```

List Dapat Berubah

Sintaks untuk mengakses elemen list sama dengan mengakses karakter string yakni operator kurung siku. Ekspresi di dalam tanda kurung menentukan indeks. Ingatlah bahwa indeks dimulai dari 0:

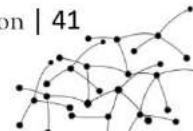
In [2]:

```
cheeses[0]
```

Out[2]:

```
'Cheddar'
```

Tidak seperti string, list bisa berubah. Ketika operator tanda kurung muncul di sisi kiri tugas,



operator ini mengidentifikasi elemen list yang akan ditugaskan:

In [3]:

numbers = [42, 123]

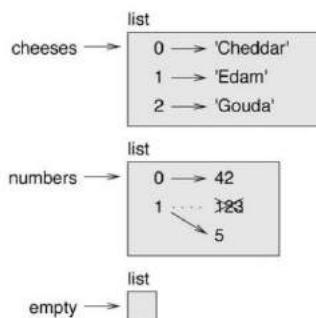
numbers[1] = 5

numbers

Out[3]:

[42, 5]

Unsur bilangan kesatu yang tadinya 123 kini menjadi 5. Gambar 6-1 menunjukkan diagram status cheeses, numbers, dan empty.



Gambar 1. Diagram keadaan.

List diwakili oleh kotak dengan kata “list” di luar dan elemen list di dalamnya. cheeses mengacu pada list dengan tiga elemen yang diindeks 0, 1 dan 2. numbers berisi dua elemen; diagram menunjukkan bahwa nilai elemen kedua telah ditetapkan ulang dari 123 menjadi 5. empty mengacu pada list tanpa elemen.

Indeks list bekerja dengan cara yang sama seperti indeks string:

- a. Ekspresi bilangan bulat apa pun dapat digunakan sebagai indeks.
- b. Jika Anda mencoba membaca atau menulis elemen yang tidak ada, Anda mendapatkan IndexError.
- c. Jika suatu indeks mempunyai nilai negatif, indeks tersebut dihitung mundur dari akhir list.

Operator in juga berfungsi pada list:

In [4]:

```
cheeses = ['Cheddar', 'Edam', 'Gouda']
```

```
'Edam' in cheeses
```

Out[4]:

```
True
```

In [5]:

```
'Brie' in cheeses
```

Out[5]:

```
False
```

Menelusuri List

Cara paling umum untuk menelusuri elemen list adalah dengan perulangan for. Sintaksnya sama dengan string:

In [6]:



```
for cheese in cheeses:
```

```
    print(cheese)
```

```
Out [6]:
```

```
Cheddar
```

```
Edam
```

```
Gouda
```

Penggunaan fungsi `for` seperti diatas berfungsi dengan baik jika Anda hanya perlu membaca elemen list. Namun jika Anda ingin menulis atau memperbarui elemen, Anda memerlukan indeks. Cara umum untuk melakukannya adalah dengan menggabungkan fungsi bawaan `range` dan `len`. Contoh:

```
In [7]:
```

```
for i in range(len(numbers)):
```

```
    numbers[i] = numbers[i] * 2
```

Perulangan ini melintasi list dan memperbarui setiap elemen. `len` mengembalikan jumlah elemen dalam list. `range` mengembalikan list indeks dari 0 hingga $n-1$, di mana n adalah panjang list. Setiap kali melalui loop, `i` mendapatkan indeks elemen berikutnya. Pernyataan penugasan menggunakan `i` untuk membaca nilai lama elemen dan menetapkan nilai baru.

Perulangan `for` pada list kosong tidak pernah menjalankan isi:

```
In [8]:
```

```
for x in []:
```

```
    print('This never happens.')
```

Meskipun sebuah list dapat berisi list lain, nested list tetap dihitung sebagai satu elemen. Panjang list ini ada empat:

```
['spam', 1, ['Brie', 'Roquefort', 'Poul le Veq'], [1, 2, 3]]
```

B. Dictionary

Subbab ini menyajikan tipe bawaan lain yang disebut dictionary. Dictionary adalah salah satu fitur terbaik Python; mereka adalah dasar dari banyak algoritma yang efisien dan elegan

Dictionary Adalah Pemetaan

Dictionary itu seperti list, tetapi lebih umum. Dalam sebuah list, indeks harus berupa bilangan bulat; dalam dictionary mereka bisa (hampir) jenis apa pun. Dictionary berisi kumpulan indeks, yang disebut **key**, dan kumpulan value. Setiap key dikaitkan dengan satu value. Asosiasi key dan value disebut **pasangan key-value** atau terkadang **item**.

Dalam bahasa matematika, dictionary mewakili **pemetaan** dari key ke value, jadi Anda juga bisa mengatakan bahwa setiap key “memetakan ke” suatu value. Sebagai contoh, kita akan membuat dictionary yang memetakan kata-kata dari bahasa Inggris ke bahasa Spanyol, sehingga key dan value semuanya berupa string. Fungsi dict membuat dictionary baru tanpa item. Karena



dict adalah nama fungsi bawaan, sebaiknya hindari menggunakannya sebagai nama variabel.

In [9]:

```
eng2sp = dict()
```

```
eng2sp
```

Out[9]:

```
{}
```

Tanda kurung berlekuk-lekuk, {}, melambangkan dictionary kosong. Untuk menambahkan item ke dictionary, Anda dapat menggunakan tanda kurung siku:

In [10]:

```
eng2sp['one'] = 'uno'
```

Baris ini membuat item yang memetakan dari key 'one' ke value 'uno'. Jika kita mencetak dictionary lagi, kita akan melihat pasangan key-value dengan titik dua di antara key dan value:

In [11]:

```
eng2sp
```

Out[11]:

```
{'one': 'uno'}
```

Format keluaran ini juga merupakan format masukan. Misalnya, Anda bisa membuat dictionary baru dengan tiga item:

In [12]:

```
eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}
```

Namun jika Anda mencetak eng2sp, Anda mungkin akan terkejut:

In [13]:

```
eng2sp
```

Out[13]:

```
{'one': 'uno', 'two': 'dos', 'three': 'tres'}
```

Urutan pasangan key-value mungkin tidak sama. Jika Anda mengetikkan contoh yang sama di komputer Anda, Anda mungkin mendapatkan hasil yang berbeda. Secara umum, urutan item dalam dictionary tidak dapat diprediksi. Namun hal itu tidak menjadi masalah karena elemen dictionary tidak pernah diindeks dengan indeks bilangan bulat. Sebagai gantinya, Anda menggunakan key untuk mencari value yang sesuai:

In [14]:

```
eng2sp['two']
```

Out[14]:

```
'dos'
```

Key 'two' selalu dipetakan ke value 'dos' sehingga urutan item tidak menjadi masalah. Jika key tidak ada dalam dictionary, Anda mendapatkan pengecualian:

In [15]:

```
eng2sp['four']
```



```
KeyError  
Traceback (most recent call last)  
Cell In[53], line 1  
----> 1 eng2sp['four']
```

KeyError: 'four'

Fungsi len berfungsi pada dictionary; ia mengembalikan jumlah pasangan key-value:

In [16]:

```
len(eng2sp)
```

Out[16]:

3

Operator in juga berfungsi pada dictionary; ini memberi tahu Anda apakah sesuatu muncul sebagai *key* dalam dictionary (muncul sebagai nilai saja tidak cukup).

In [17]:

```
'one' in eng2sp
```

Out[17]:

True

In [18]:

```
'uno' in eng2sp
```

Out[18]:

False

Untuk melihat apakah sesuatu muncul sebagai value dalam dictionary, Anda bisa menggunakan metode values, yang mengembalikan kumpulan value, lalu menggunakan operator in:

In [19]:

```
vals = eng2sp.values()
```

```
'uno' in vals
```

Out[19]:

```
True
```

Operator in menggunakan algoritma berbeda untuk list dan dictionary. Untuk list, ia mencari elemen list secara berurutan, seperti dalam “**Searching**”. Semakin panjang list, semakin lama pula waktu pencarinya.

Untuk dictionary, Python menggunakan algoritma yang disebut **hashtable** yang memiliki properti luar biasa yakni: operator in membutuhkan waktu yang sama, tidak peduli berapa banyak item yang ada dalam dictionary

C. Ttuple

Subbab ini menyajikan satu lagi tipe bawaan, tuple, dan kemudian memperlihatkan bagaimana list, dictionary, dan tuple bekerja bersama

Tuple Tidak Dapat Diubah

Tuple adalah urutan value. Value bisa bertipe apa saja, dan diindeks berdasarkan bilangan bulat, jadi dalam hal ini tuple sangat mirip dengan list. Perbedaan pentingnya adalah tuple tidak dapat diubah. Secara sintaksis, tuple adalah list value yang dipisahkan koma:



In [20]:

```
t = 'a', 'b', 'c', 'd', 'e'
```

Meskipun tidak diperlukan, tuple biasanya diapit dalam tanda kurung:

In [21]:

```
t = ('a', 'b', 'c', 'd', 'e')
```

Untuk membuat tuple dengan satu elemen, Anda harus menyertakan koma terakhir:

In [22]:

```
t1 = 'a',
```

```
type(t1)
```

Out[22]:

```
tuple
```

Value dalam tanda kurung bukanlah tuple:

In [23]:

```
t2 = ('a')
```

```
type(t2)
```

Out[23]:

```
str
```

Cara lain untuk membuat tuple adalah fungsi bawaan tuple. Tanpa argumen, ini menciptakan tuple kosong:

In [24]:

```
t = tuple()
```

```
t
```

```
Out[24]:
```

```
()
```

Jika argumennya berupa sequence (string, list, atau tuple), hasilnya adalah tuple dengan elemen sequence:

```
In [25]:
```

```
t = tuple('lupins')
```

```
t
```

```
Out[25]:
```

```
('l', 'u', 'p', 'i', 'n', 's')
```

Karena tuple adalah nama fungsi bawaan, sebaiknya hindari menggunakannya sebagai nama variabel. Kebanyakan operator list juga bekerja pada tuple. Operator kurung siku mengindeks elemen:

```
In [26]:
```

```
t = ('a', 'b', 'c', 'd', 'e')
```

```
t[0]
```

```
Out[26]:
```

```
'a'
```

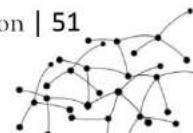
Dan operator irisan memilih serangkaian elemen:

```
In [27]:
```

```
t[1:3]
```

```
Out[27]:
```

```
('b', 'c')
```



Namun jika Anda mencoba mengubah salah satu elemen tuple, Anda mendapatkan kesalahan:

In [30]:

```
t[0] = 'A'
```

```
-----  
TypeError                           Traceback (most recent  
call last)
```

```
Cell In[90], line 1
```

```
----> 1 t[0] = 'A'
```

```
TypeError: 'tuple' object does not support item  
assignment
```

Karena tuple tidak dapat diubah, Anda tidak dapat mengubah elemennya. Namun Anda dapat mengganti satu tuple dengan tuple lainnya:

In [31]:

```
t = ('A',) + t[1:]
```

```
t
```

Out[31]:

```
('A', 'b', 'c', 'd', 'e')
```

Pernyataan ini membuat tuple baru dan kemudian membuat t merujuk padanya.

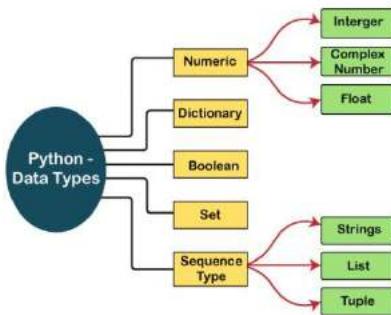
6

Tipe Data, Variabel, Konstanta, Dan Nilai

A. Tipe Data Pada Python

Tipe data (data type) adalah konsep yang mengacu pada jenis nilai atau kumpulan nilai yang dapat disimpan dan dioperasikan dalam program komputer. Jenis tipe data yang digunakan dalam sebuah program akan mempengaruhi bagaimana program tersebut berperilaku.

Tipe data pada pemrograman dengan python diatur pada saat menetapkan nilai ke variable(Lambert, 2017), hal ini berbeda dengan beberapa bahasa pemrograman lain yang mengharuskan mendefinisikan tipe data terlebih dahulu. Berikut adalah gambaran tipe data yang ada di python:



Gambar 1. Tipe Data pada Python

(Javatpoint.com, 2023)

1. Tipe Numerik

a. Interger(int)

Tipe data int digunakan untuk menyimpan nilai bilangan bulat. Seperti: 23, -17, 0, dll

Contoh program:

```
x = 5
print(x)
```

b. Float(float)

Tipe data float digunakan untuk menyimpan nilai bilangan decimal. Seperti angka 3.14, -0.34, 1.0, dll

Contoh program:

```
x = 5.3
print(x)
```

c. Complex Number

Tipe data complex digunakan untuk menyimpan bilangan komplek. Seperti bilangan: 5 + 4j, 2 – 3y.

Contoh program:

```
x=5 + 4j
print(x)
```

Output: (5+4j)

2. Sequence type

a. String(str)

Tipe data string digunakan untuk menyimpan teks. Suatu teks dapat didefinisikan dengan Python menggunakan tanda kutip tunggal(' '), ganda(" "), atau tiga kali (""" """) atau "" ""(Stevens and Boucher, 2015). Dimana 'hello' dan "hello" adalah dua hal yang sama. Penulisan teks dalam banyak baris atau multiline dapat menggunakan kutip tiga.

Contoh program

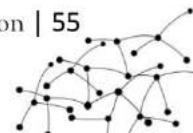
```
nama='Wenda novayani'  
sub ="Python Code"  
pwd='123456'  
  
teks = """\nSelamat kamu adalah  
orang pilihan,  
ingat belajar python  
itu mudah, mudah banget  
semoga kesuksesan mengiringi Anda  
"""  
  
print("Hii...",nama)  
print("selamat Datang di ",sub)  
print("Password kamu:",pwd)  
print(teks)
```

Output:

```
Hii... Wenda novayani  
selamat Datang di Python Code  
Password kamu: 123456
```

```
Selamat kamu adalah  
orang pilihan,  
ingat belajar python  
itu mudah, mudah banget  
semoga kesuksesan mengiringi Anda
```

Ketika ada operator tambah(+) pada string seperti "python"+" programming" , maka akan menjadi "python programming," dimana operator + digunakan



untuk menggabungkan dua string. Sedangkan operator kali (*) pada string seperti "Python code" *2 menghasilkan "Python codePython code", dimana operator * disebut sebagai operator pengulangan.

Contoh program:

```
str1 = 'hello, saya senang' #string str1
str2 = 'belajar python' #string str2
print (str1*2) #mencetak str1 dua kali
print (str1 + str2) #mencetak gabungan str1 dan str2
```

Output:

hello, saya senanghello, saya senang

hello, saya senangbelajar python

b. List

Tipe data list digunakan untuk menyimpan daftar nilai yang dapat diubah (*mutable*) dan di-indeks kedalam satu variabel, dimana nilai dalam list dapat berisi data dengan tipe berbeda. Data yang disimpan dalam list dipisahkan dengan koma (,) dan dimasukkan ke dalam bagian persegi []. List dapat diurutkan, dapat diubah, dan memungkinkan nilai duplikat.

Untuk mendapatkan akses ke data daftar list, dapat menggunakan indeks, dimulai dari indeks [0], jika menggunakan indeks negatif maka akan mulai dari data akhir.

Contoh program:

```
list = [34, True, 40, "male"]
list1=["durian", "jeruk", "kedondong"]
```

```
list2=["anggur", "mangga", "anggur"]
print(list[1])
print(list[-1])
```

Output

```
[34, True, 40, 'male']
```

```
jeruk
```

```
anggur
```

c. Tuple

Tipe data ini mirip dengan list, tetapi bersifat tidak dapat diubah (immutable), data dalam tuple dipisahkan dengan koma dan dimasukkan dalam tanda kurung (). Seperti: (1, 2, 3), ('a', 'b', 'c').

Contoh program:

```
data=("pisang", "cherry", "jeruk", "kiwi", "melon", 15000)

#mencetak data indek 1
print(data[1])

#mencetak data index terakhir
print(data [-1])

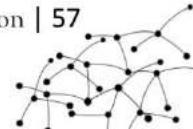
#mencetak data index 2 terakhir
print(data [-2])

#mencetak data selain indek 4
print(data [:4])

#mencetak data dari indek 2 dan berakhir di indek 4 (not included)
print(data[2:4])
```

Output

```
cherry
15000
melon
('pisang', 'cherry', 'jeruk', 'kiwi')
('jeruk', 'kiwi')
```



3. Boolean(bool)

Tipe data boolean digunakan untuk menyimpan nilai kebenaran, yaitu True atau False. Tipe data ini biasanya digunakan dalam pengkondisian. Contoh: True, False. Beberapa info penting tentang Boolean:

- Hampir semua nilai dievaluasi menjadi True jika memiliki value.
- String apa pun bernilai True, kecuali string kosong.
- Angka apa pun bernilai True, kecuali 0.
- List, tuple, set, dan dict apa pun adalah True, kecuali yang kosong.

Contoh program:

Kode Program	Output
print(bool("hi.."))	True
print(bool(123))	True
print(bool(["I", "Love", "python"]))	True
print(bool(0))	False
print(bool("")))	False

4. Set

Tipe data set digunakan untuk menyimpan data banyak item dalam satu variable, dimana datanya tidak memiliki urutan yang tetap(unordered), sehingga tidak dapat menggunakan indek dalam pemanggilan data(*unindexed*) serta tidak boleh berisi data yang sama (duplikat). Tipe data set juga tidak dapat dirubah (*unchangeable*), namun item datanya dapat dihapus dan ditambah.

Data dalam tipe data set berada dalam kurung kurawal {} dimana setiap data dipisahkan dengan tanda koma, jenis data dalam tipe set dapat dalam berbagai jenis.

Contoh program:

```
dataku = {'Wenda', 25, 165,'Python code'}  
  
#mencetak isi set  
print(dataku)  
  
# menambahkan data ke set  
dataku.add(30)  
print(dataku)  
  
#menghapus item data dari set  
dataku.remove(25)  
print(dataku)
```

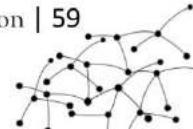
Output

```
{'Wenda', 'Python code', 165, 30}  
{165, 'Wenda', 25, 'Python code', 30}  
{'Wenda', 25, 'Python code', 30}
```

5. Dictionary (dict)

Dictionary adalah tipe data pada python yang berfungsi untuk menyimpan kumpulan data/nilai dengan pendekatan “key-value”. Karena setiap urutannya berisi key dan value, dimana key dan value-nya dipisahkan dengan titik dua (:), item dipisahkan oleh koma, dan semuanya tertutup dalam kurung kurawal. Dictionary kosong tanpa isi ditulis hanya dengan dua kurung kurawal {}. Data dalam dictionary dapat terdiri dari berbagai macam data termasuk tipe dictionary juga.

Dictionary memiliki sifat:



- a. *unordered* (tidak berurutan), sehingga key/atribut yang tidak dapat diakses menggunakan indek.
- b. *changeable* (dapat diubah), sehingga data yang telah dimasukkan dapat diubah.
- c. *unique*, alias tidak dapat menerima dua keys yang sama, sehingga jika ada dua key yang sama, key yang didefinisikan terakhir akan menimpa nilai dari key yang didefinisikan lebih awal.

Contoh program:

```
karyawan = {"nama": "Wenda", "gaji":45000,"perusahaan":"PT.Solutex"}  
print(karyawan)  
  
#menambahkan data hobi  
karyawan['hobi']="shopping"  
print(karyawan)  
  
#menghapus data perusahaan  
karyawan.pop('perusahaan')  
print(karyawan)  
  
#mengganti nama karyawan  
karyawan['nama']="Wendano"  
print(karyawan)  
  
#mencari nilai dari gaji  
x = karyawan.get("gaji")  
print(x)
```

Output:

```
{'nama': 'Wenda', 'gaji': 45000, 'perusahaan': 'PT.Solutex'}  
{'nama': 'Wenda', 'gaji': 45000, 'perusahaan': 'PT.Solutex', 'hobi': 'shopping'}  
{'nama': 'Wenda', 'gaji': 45000, 'hobi': 'shopping'}  
{'nama': 'Wendano', 'gaji': 45000, 'hobi': 'shopping'}  
45000
```

B. Pemberian type data secara spesifik

Dalam python, pemberian tipe data secara spesifik dapat dilakukan dengan cara menuliskan tipe datanya sebelum nilai, seperti contoh berikut:

```
x = float(20.5)  
y = str("Hello Python")
```

```
z = list(("Jeruk", "nenas", "cherry"))
```

C. Cara mengetahui tipe data pada python

Cara mengetahui tipe data pada python dapat menggunakan fungsi type(). Contoh:

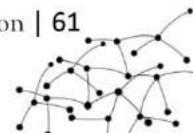
Kode Program	Output
a = 5 b = 7.0 c = "Salam dari kota Rumbai" d =1j+2 print(type(a)) print(type(b)) print(type(c)) print(type(d))	<class 'int'> <class 'float'> <class 'str'> <class 'complex'>

D. Variable pada Python

Variabel adalah suatu tempat untuk menyimpan nilai atau data di dalam memori, dan nanti dapat dipanggil kembali untuk dibaca atau diganti. Pembuatan variabel dalam Python tidak memerlukan deklarasi tipe data secara eksplisit seperti pada bahasa pemrograman lainnya. Variable di python dibuat dengan format : **nama_variabel=nilai**

Contoh sintak :

umur=20 } nilai
 |
 |
 nama_variabel



Beberapa informasi penting tentang variable pada python:

1. Pengisian nilai (assignment): pengisian nilai ke dalam variabel dengan menggunakan tanda sama dengan (=).
2. Python tidak memiliki perintah untuk mendeklarasikan variable, dimana variabel dibuat saat pertama kali memberikan nilai padanya.

Contoh:

```
umur = 24  
nama= "Wenda Novayani"
```

3. Variabel tidak perlu dideklarasikan dengan tipe data tertentu, dan bahkan dapat diubah tipenya setelah ditetapkan.

Contoh:

```
x = 24 #variabel x bertipe int  
x= "Wendano" #variabel x sekarang bertipe str
```

4. Pemberian nama variable pada dasarnya bebas, namun ada aturan dan pengecualian:

Table 8. Aturan nama Variabel

No	Aturan nama variabel	Contoh kode
A	Nama variable bersifat <i>case sensitive</i> , huruf kapital dan huruf kecil akan berbeda	<pre>x = 5 X = "Hello" print(x) print(X)</pre> X tidak sama dengan x
B	Nama variable hanya boleh diawali oleh huruf (a-z,A-Z) atau garis bawah (_), (tidak boleh angka atau karakter khusus lainnya)	<pre>_kota= "Pekanbaru" #benar 1kota= "Pekanbaru" #salah</pre>

C	Hanya boleh berisikan alfanumerik dan under score (A-z, 0-9, _)	X1=10.2 #benar X_\$a=3 #salah s3ze=3 #benar kot@="Pekanbaru" #salah
D	Tidak boleh ada spasi, jika lebih dari satu suku kata maka gunakan under score atau digabungkan	Nama_depan="Kayysah" #salah namaDepan="kayysah" #benar nama_depan="kay" #benar
E	Nama variabel tidak boleh menggunakan kata kunci yang sudah ada dalam python	if, while, for, dst #tidak boleh

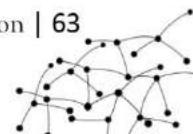
E. Konstanta pada Python

Konstanta pada Python, tidak ada konsep konstanta yang benar-benar tetap dan tidak dapat diubah seperti dalam beberapa bahasa pemrograman lainnya. Dalam Python, variabel yang dianggap sebagai konstanta hanyalah konvensi dan disarankan untuk menggunakan huruf besar dengan kata-kata terpisah oleh garis bawah (UPPER_CASE) untuk nama variabel tersebut, agar pembaca kode mengetahui bahwa nilai tersebut seharusnya tidak diubah.

Contoh penggunaan konvensi untuk konstanta dalam Python:

```
PI = 3.14159
NAMA_BULAN = "September"
```

Namun, Python tidak secara khusus melarang perubahan nilai variabel tersebut. Misalnya, masih dapat menulis:



```
PI = 3.14 # Ini mengubah nilai PI
```

Membuat nilai yang benar-benar tidak dapat diubah, dapat menggunakan tipe data tuple yang bersifat immutable (tidak dapat diubah). Meskipun demikian, praktik ini tidak umum dan tidak digunakan sebagai konstanta dalam Python.

F. Nilai pada Python

Dalam Python, nilai dapat berupa angka, string, boolean, dan tipe data lainnya. Nilai merupakan data dasar yang disimpan dalam sebuah variabel dan dapat diolah menggunakan operator matematika, operator logika, serta fungsi-fungsi bawaan Python (built-in functions). Pemberian nilai pada python disebut juga dengan assignment.

Misalnya untuk variabel integer, dapat mengisinya dengan nilai yang berupa angka bulat seperti 5, 10, -8, dan sebagainya. Sebagai contoh kode program sederhana untuk penggunaan nilai pada Python:

```
# mendefinisikan variabel x dan mengisinya dengan nilai 5
x = 10

# mencetak nilai variabel x
print(x)

# melakukan operasi matematika terhadap variabel x
y = x + 3.5

# mencetak nilai variabel y
print(y)
```

Penjelasan program pada contoh di atas:

Mendefinisikan variabel ‘x’ dengan mengisinya dengan nilai 5. Kemudian mencetak nilai variabel ‘x’

tersebut menggunakan fungsi `print()`. Selanjutnya melakukan operasi matematika penjumlahan antara variabel `x` dengan angka 3.5 dan hasilnya disimpan dalam variabel `y`. Lalu mencetak nilai variabel `y` menggunakan fungsi `print()`.

Pemberian nilai dapat diberikan pada variabel dalam 3 cara yaitu:

1. Assigning a Single Value to Multiple Variables:

Contoh program:

```
x = y = z = 10
print(x) # Output: 10
print(y) # Output: 10
print(z) # Output: 10
```

2. Assigning Multiple Values to Multiple Variables:

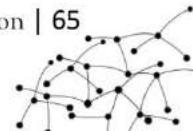
Contoh program:

```
x, y, z = 10, 20.3, 30
print(x) # Output: 10
print(y) # Output: 20.3
print(z) # Output: 30
```

3. Assigning Values to Variables in One Line:

Contoh Program:

```
x = "durian"; y = "anggur"; z = 5
print(x) # Output: durian
print(y) # Output: anggur
print(z) # Output: 5
```



7

Percabangan

Percabangan dalam *python* digunakan untuk mengatur alur eksekusi program berdasarkan kondisi tertentu. Anda dapat menggunakan pernyataan 'if', 'elif' (singkatan dari "else if"), dan 'else' untuk membuat percabangan. Selain percabangan, struktur ini juga disebut *control flow, decision, struktur kondisi, struktur if*.

Adapun tipe *control flow* dalam *python* sebagai berikut:

A. Struktur percabangan *if*

Percabangan *if* adalah pernyataan pengambilan keputusan yang paling simpel. Percabangan ini digunakan untuk menentukan apakah pernyataan yang diambil sesuai dengan eksekusi.

Adapun *syntax* dari percabangan *if* yaitu:

if kondisi:

pernyataan()

Kondisi merupakan sebuah variabel atau nilai yang bertipe data boolean (*true/false*).

1. Jika kondisi yang dievaluasi bernilai true, pernyataan dalam *if* dieksekusi
2. Jika kondisi yang dievaluasi menjadi false, pernyataan dalam *if* tidak dieksekusi

Kondisi True	Kondisi False
angka = 10 if angka > 0: → #pernyataan #kode lainnya	angka = -5 if angka > 0: → #pernyataan → #kode lainnya

Contoh 1 (percabangan *if*)

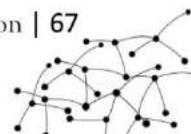
```
x = 10
if x > 0:
    print("x lebih besar dari 0")
print("pernyataan selesai")
```

Output

```
x lebih besar dari 0
pernyataan selesai
```

Pada contoh diatas, variabel x dideklarasikan dengan nilai = 10. Disini jika nilai x lebih besar dari 0 maka kondisi bernilai *true*, dan pernyataan dieksekusi dan kemudian dilanjutkan dengan kode lainnya diluar perkondisian.

Namun jika variabel x dideklarasikan dengan nilai = -5, maka kondisi tidak memenuhi karena x kurang dari 0 dan pernyataan diabaikan serta dilanjutkan dengan kode lainnya diluar perkondisian (lihat contoh program 2).



Contoh 2 (percabangan *if*)

```
x = -5  
if x > 0:  
    print("x lebih besar dari 0")  
print("pernyataan selesai")
```

Output

pernyataan selesai

B. Struktur percabangan *if/Else*

Percabangan *if .. else* adalah struktur percabangan yang memiliki nilai *if* untuk perkondisian dan juga *else* jika perkondisian bernilai *false*.

Adapun *syntax* dari percabangan *if..else* yaitu:
if kondisi:

pernyataan(), kondisi is True

else:

pernyataan(), kondisi is False

Kondisi pernyataan *if ... else* mengevaluasi kondisi yang diberikan:

Jika kondisi bernilai *True* atau benar, maka:

1. Pernyataan di dalam *if* dieksekusi
2. Pernyataan di dalam *else* diabaikan

Jika kondisi bernilai *False* atau salah, maka:

1. Penyataan di dalam *else* dieksekusi
2. Pernyataan di dalam *if* diabaikan

Kondisi True	Kondisi False
angka = 10 if angka > 0: → #pernyataan else: #Pernyataan #kode lainnya	angka = -5 if angka > 0: → #pernyataan else: #Pernyataan #kode lainnya

Contoh 3 (percabangan *if...else*)

```
x = 10
if x > 0:
    print("x lebih besar dari 0")
else:
    print("x lebih kecil dari 0")

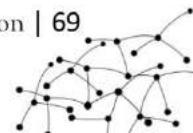
print("pernyataan selesai")
```

Output

```
x lebih besar dari 0
pernyataan selesai
```

Contoh 4 (percabangan *if...else*)

```
x = -5
if x > 0:
    print("x lebih besar dari 0")
```



```
else:
```

```
    print("x lebih kecil dari 0")
```

```
    print("pernyataan selesai")
```

Output

```
x lebih kecil dari 0  
pernyataan selesai
```

Pada contoh program 3 dan 4 untuk kasus percabangan *if...else* agak sedikit berbeda dengan kasus percabangan *if*. Contoh program 3 mendeklarasikan variabel x dengan nilai 10 dengan perkondisian bahwa nilai x harus lebih dari 0 dan kondisi terpenuhi atau bernilai benar maka pernyataan di dalam *if* dieksekusi dan pernyataan di dalam *else* diabaikan.

Akan tetapi, jika perkondisian tidak terpenuhi atau bernilai salah (contoh program 4) maka pernyataan didalam *if* diabaikan sedangkan pernyataan didalam *else* dieksekusi.

C. Struktur percabangan *if/Elif/Else*

Struktur percabangan *if ... else* digunakan untuk mengeksekusi pernyataan diantara dua alternatif. Akan tetapi, jika Anda memerlukan lebih dari dua alternatif maka Anda dapat menggunakan struktur percabangan *if... elif... else*.

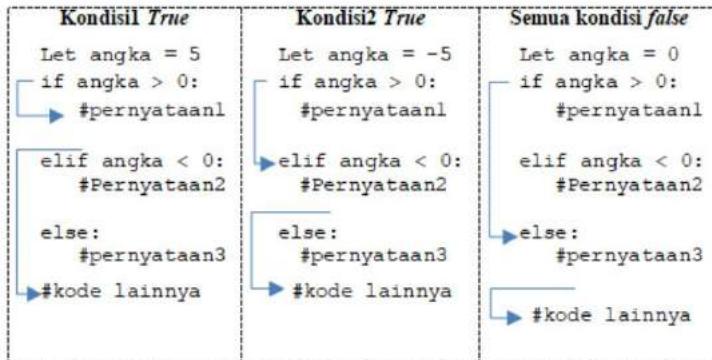
Adapun *syntax* dari percabangan *if...elif...else* yaitu:
if kondisi1:

```
    pernyataan1()
```

```
elif kondisi2:  
    pernyataan2()  
else:  
    pernyataan3()
```

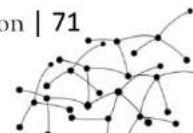
Penjelasan disini:

1. Jika **kondisi1** dievaluasi bernilai **true** atau **benar** maka pernyataan1 dieksekusi
2. Jika **kondisi1** dievaluasi bernilai **false** atau **salah** maka **kondisi2** dievaluasi dengan ketentuan:
 - a. Jika **kondisi2** dievaluasi bernilai **true** atau **benar**, maka pernyataan2 dieksekusi
 - b. Jika **kondisi2** dievaluasi bernilai **false** atau **salah**, maka pernyataan3 dieksekusi



Contoh 5 (percabangan *if... elif... else*)

```
nilai = 80  
  
if nilai >= 90:  
    print("Nilai Anda A")  
  
elif nilai >= 80:
```



```
print("Nilai Anda B")  
else:  
    print("Nilai Anda C")
```

Output

Nilai Anda B

Pada contoh 5 diatas, Anda membuat dan mendeklarasikan variabel nilai dengan nilai 80. Disini Anda melakukan pengecekan kondisi pada pernyataan *if* dan bernilai salah maka pernyataan diabaikan, kemudian melakukan pengecekan kondisi kembali pada pernyataan *elif* dan bernilai benar maka pernyataan dieksekusi dan pernyataan *else* diabaikan.

D. Struktur percabangan bersarang

Struktur percabangan bersarang digunakan jika terdapat perkondisian *if* didalam perkondisian *if*.

Adapun *syntax* dari percabangan bersarang yaitu:

#outer if statement

if kondisi1:

pernyataan()

#inner if statement

if kondisi2:

pernyataan()

Catatan:

1. Anda dapat menambahkan *else* dan *elif* didalam *inner if statement* untuk melakukan perkondisian
2. Anda juga dapat menambahkan *inner if statement* didalam *outer else* atau *elif*
3. Anda dapat menerapkan *if statement* bersarang

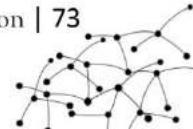
Contoh 6 (percabangan bersarang)

```
nilai = 85  
  
if nilai >= 70:  
    if nilai >= 80:  
        print("Baik")  
    else:  
        print("Cukup")  
else:  
    print("Kurang")
```

Output

Baik

Pada contoh percabangan bersarang diatas terdapat percabangan di dalam percabangan. Variabel nilai adalah 85 kemudian dilakukan pengecekan perkondisian pada *outer if statement*. Maka kondisi bernilai benar karena kondisi nilai lebih besar sama dengan 70 kemudian dilakukan pengecekan kembali pada *inner if statement* dan kondisi nilai lebih besar sama dengan 80 maka kondisi bernilai benar maka pernyataan dieksekusi.



E. Latihan Soal

Kasus Sederhana: Penentuan Diskon Belanja

Anda adalah seorang kasir di sebuah toko. Anda ingin membuat program sederhana untuk menghitung diskon yang diberikan kepada pelanggan berdasarkan total belanja mereka. Berikut adalah aturannya:

1. Jika total belanja pelanggan mencapai atau melebihi Rp 100.000, mereka akan mendapatkan diskon 10%.
2. Jika total belanja pelanggan kurang dari Rp 100.000, mereka tidak mendapatkan diskon.

Buat program Python yang meminta pengguna untuk memasukkan total belanja mereka dan kemudian mencetak jumlah diskon yang mereka dapatkan (jika ada) atau mencetak pesan bahwa mereka tidak mendapatkan diskon.

8

Library

Python telah muncul sebagai bintang terang yang memikat para pengembang dan pemula sekalipun. Python tidak hanya menjadi tren, tetapi juga menjadi bahasa pemrograman yang sangat populer. Mengapa begitu? Jawabannya sederhana: Python menawarkan kombinasi yang sangat menarik antara kesederhanaan dan kekuatan, yang menjadikannya pilihan utama untuk berbagai jenis proyek.

Salah satu alasan mengapa Python sangat digemari adalah Karena Pustaka Python Yang Luas (Ryane Puspa, 2021). Memang benar ekosistem pustaka atau library dari bahasa Python sangatlah luar biasa. Python dilengkapi dengan berbagai library yang sangat kuat dan kaya, yang dapat memperluas kemampuannya secara signifikan. Dengan library-library yang dapat digunakan untuk komputasi ilmiah, analisis data, pengembangan web, dan untuk kecerdasan buatan, Python menjadi pilihan utama untuk berbagai aplikasi, mulai dari analisis data hingga pengembangan web dan pembelajaran mesin.



A. Apa Itu Library?

Library adalah kumpulan kode yang menyediakan fungsionalitas tertentu untuk digunakan dalam aplikasi lain. Dalam konteks Python, menurut (RevoU, 2023) *library* biasanya terdiri dari beberapa *module* dan *package* yang bekerja bersama untuk menyediakan sekumpulan fungsionalitas. Library berguna untuk membantu menyelesaikan tugas-tugas dalam program Anda dengan lebih cepat dan efisien.

Sebagai Analoginya bayangkan jika Anda ingin memasak makanan di rumah. Untuk memasak, Anda tidak perlu membuat segala sesuatu dari nol, seperti membuat sendiri garam, gula, atau bumbu-bumbu. Sebaliknya, Anda pergi ke toko dan membeli bahan-bahan siap pakai, seperti garam, gula, atau bumbu masak. Ini membuat memasak menjadi lebih cepat dan mudah, bukan?

Contohnya, mari kita katakan Anda ingin membuat program untuk menggambar grafik. Tanpa library, Anda harus menulis kode yang sangat panjang dan rumit untuk menggambar bentuk-bentuk dan warna-warni di layar komputer. Tapi dengan menggunakan library grafik yang sudah ada, seperti "Matplotlib" dalam bahasa pemrograman Python, Anda dapat membuat grafik dengan mudah hanya dengan beberapa baris kode.

Jadi, dalam konteks ini, library adalah seperti "bahan masak" yang membantu Anda memasak makanan (program) Anda dengan lebih cepat dan mudah, tanpa harus membuat semuanya dari awal. Dengan library, Anda dapat membangun program yang lebih keren dan

kompleks tanpa harus jadi seorang "chef" pemrograman yang ulung!

Berikut beberapa poin penting tentang library:

1. **Reusabilitas:** Library memungkinkan pengembang untuk menggunakan kembali kode yang sudah ada. Ini menghemat waktu dan upaya dalam pengembangan perangkat lunak, karena pengembang tidak perlu membangun kembali fungsi-fungsi umum yang telah ada dalam library.
2. **Abstraksi:** Library menyediakan abstraksi tingkat tinggi untuk tugas-tugas tertentu. Sebagai contoh, dalam bahasa pemrograman Python, library seperti NumPy menyediakan abstraksi untuk komputasi ilmiah dan manipulasi array, sehingga pengembang tidak perlu khawatir tentang detail implementasi yang kompleks.
3. **Perluasan Fungsionalitas:** Library memungkinkan pengembang untuk menambahkan fungsionalitas baru ke aplikasi mereka dengan cepat. Misalnya, jika Anda memerlukan kemampuan grafis dalam aplikasi Python Anda, Anda dapat menggunakan library seperti Matplotlib atau Pygame.
4. **Stabilitas:** Library yang telah diuji secara ekstensif oleh komunitas pengembang cenderung lebih stabil dan memiliki lebih sedikit bug daripada kode yang baru ditulis. Ini dapat meningkatkan keandalan perangkat lunak Anda.
5. **Komunitas:** Pengembang perangkat lunak sering kali berkontribusi pada library open source. Hal ini menciptakan komunitas yang mendukung pertukaran



pengetahuan dan pemecahan masalah, serta memastikan bahwa library tersebut terus diperbarui dan ditingkatkan.

6. **Pengoptimalan:** Beberapa library dibangun dengan pengoptimalan khusus untuk tugas-tugas tertentu, seperti pemrosesan gambar, analisis data, atau kecerdasan buatan. Ini memungkinkan pengembang untuk mengakses performa tinggi tanpa harus menghabiskan waktu untuk mengoptimalkan kode mereka sendiri.

Anda dapat mengimpor library atau modul dengan perintah **import** dalam kode Python Anda dan kemudian mengakses fungsi dan objek yang ada dalam library tersebut untuk memenuhi kebutuhan Anda. Penggunaan library adalah salah satu aspek yang membuat Python menjadi bahasa pemrograman yang sangat kuat dan fleksibel.

B. Jenis-jenis Library

Setiap Library memiliki peran uniknya sendiri, terdapat library Standar dan Library yang dibuat oleh pengembang luar atau Komunitas. pemahaman tentang bagaimana mereka berfungsi akan membantu Anda dalam mengatasi berbagai tantangan pemrograman yang mungkin Anda hadapi.

Berikut adalah beberapa jenis library yang umum dalam dunia pemrograman:

1. NumPy

NumPy, atau Numerical Python, adalah pustaka fundamental yang membantu Python menjadi bahasa

yang sangat kuat untuk komputasi numerik. NumPy memperkenalkan konsep array multidimensi yang memungkinkan Anda untuk dengan mudah menyimpan dan memanipulasi data numerik. Dengan NumPy, Anda dapat melakukan operasi matematika, statistik, dan pengolahan data dengan lebih efisien.

2. Pandas

Pandas adalah teman terbaik ketika Anda berurusan dengan data tabular seperti spreadsheet. Pustaka ini menyediakan struktur data DataFrame yang kuat untuk menyimpan, mengelola, dan menganalisis data. Anda dapat menggabungkan, mengindeks, dan mengolah data dengan mudah menggunakan Pandas.

3. Matplotlib

Matplotlib adalah seniman dalam dunia visualisasi data. Pustaka ini memungkinkan Anda untuk membuat grafik dan plot yang menggambarkan data Anda dengan cara yang mudah dipahami. Dengan Matplotlib, Anda dapat menciptakan berbagai jenis grafik, termasuk grafik garis, histogram, dan scatter plot.

4. SciPy

SciPy adalah teman setia bagi ilmuwan dan peneliti. Pustaka ini memperluas NumPy dengan algoritma-algoritma ilmiah yang lebih tingkat, seperti optimisasi, integrasi, dan interpolasi. Jika Anda terlibat dalam penelitian ilmiah, SciPy adalah alat yang tak tergantikan.



5. TensorFlow

TensorFlow telah memimpin revolusi dalam bidang pembelajaran mesin. Pustaka ini memungkinkan Anda untuk membuat dan melatih model neural networks untuk berbagai tugas, seperti klasifikasi gambar, pengenalan wicara, dan banyak lagi. TensorFlow digunakan secara luas di industri dan penelitian.

6. Scikit-learn

Scikit-learn adalah pustaka yang sangat berguna dalam dunia pembelajaran mesin. Ia menyediakan algoritma-algoritma pembelajaran mesin yang siap pakai dan alat-alat untuk pemrosesan data yang hebat. Scikit-learn cocok untuk tugas-tugas seperti klasifikasi, regresi, dan pengelompokan.

7. Keras

Keras adalah antarmuka tingkat tinggi untuk pembuatan dan pelatihan model neural networks. Ia sangat mudah digunakan, cocok untuk pemula yang ingin memahami deep learning. Keras dapat berjalan di atas TensorFlow, Theano, atau Microsoft Cognitive Toolkit (CNTK).

8. Seaborn

Seaborn adalah pustaka yang mempercantik visualisasi statistik Anda. Ia berbasis pada Matplotlib dan dirancang khusus untuk menciptakan grafik statistik yang informatif dan estetis. Dengan Seaborn, Anda dapat dengan mudah membuat grafik yang menggambarkan distribusi data dan hubungan antar variabel.

9. Plotly

Plotly adalah alat yang luar biasa untuk membuat visualisasi interaktif. Ia memungkinkan Anda untuk membuat grafik interaktif yang dapat dinavigasi dan dijelajahi dalam aplikasi web atau notebook Jupyter. Plotly cocok untuk membuat peta interaktif, grafik interaktif, dan banyak lagi.

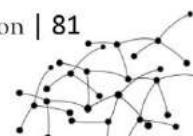
10. PyTorch

PyTorch adalah pustaka yang sangat populer di kalangan peneliti dan praktisi deep learning. Ia memberikan fleksibilitas tinggi dalam pembuatan dan pelatihan model neural networks. PyTorch sangat cocok untuk eksperimen dalam penelitian kecerdasan buatan.

Serta Masih Banyak lagi jenis lainnya. Setiap jenis library memiliki peran dan fungsionalitasnya sendiri, dan pemilihan library yang tepat sangat tergantung pada jenis proyek yang Anda kerjakan. Mempelajari dan memahami berbagai jenis library ini dapat sangat membantu dalam pengembangan perangkat lunak yang efisien, kuat dan berbagai masalah dalam pemrograman Python.

C. Menggunakan Library Python

Untuk menggunakan Library Python ada beberapa hal yang harus diperhatikan. Bagi anda yang ingin menggunakan Library standar Python, anda hanya perlu meng-import Modul library nya terlebih dahulu sedangkan untuk library external anda perlu melakukan instalasi library terlebih dahulu sebelum melakukan import. Untuk library standar dapat ditemukan di dokumentasi resmi Python



yang cukup lengkap. Beberapa modul yang paling umum digunakan dalam library standar Python adalah Math, Random dan DateTime.

1. Contoh Penggunaan Library Standar

Mari kita lihat beberapa contoh penggunaan library standar dalam Python Menggunakan Modul Math.

```
math.py > ...
1 import math
2 ^
3 # Menghitung akar kuadrat
4 akar_kuadrat = math.sqrt(16)
5 print(akar_kuadrat) # Output: 4.0
6 ^
7 # Menghitung sin dan cos
8 sin_30 = math.sin(math.radians(30))
9 cos_60 = math.cos(math.radians(60))
10 print(sin_30) # Output: 0.4999999999999994
11 print(cos_60) # Output: 0.5000000000000001
```

Pada gambar di atas terlihat penggunaan math, modul ini merupakan salah satu modul yang termasuk dalam library standar Python. Pertama dilakukan pemanggilan modul math menggunakan import ke dalam program python yang kemudian dapat melakukan penggunaan fungsi `sqrt()` pada Modul Math.

2. Contoh Penggunaan Library Eksternal

Untuk menggunakan library eksternal caranya hampir mirip dengan library standar namun ada beberapa langkah yang perlu anda lakukan terlebih dahulu yaitu instalasi. Berikut adalah contoh penggunaan library numpy :

a. instalasi NumPy

Sebelum Anda dapat menggunakannya, pastikan Anda sudah menginstal NumPy. Jika Anda belum menginstalnya, Anda dapat melakukannya dengan perintah berikut menggunakan pip, yang merupakan manajer paket Python:

pip install numpy

b. Impor NumPy ke dalam Proyek

Setelah Anda menginstal NumPy, Anda harus mengimpor modul NumPy ke dalam skrip atau proyek Python Anda. Ini dilakukan dengan menggunakan pernyataan **import**:

 **import numpy as np** # np adalah alias yang umum digunakan untuk NumPy

c. Membuat Array NumPy

NumPy terkenal karena kemampuannya untuk membuat dan mengelola array multidimensi. Anda dapat membuat array NumPy dengan berbagai cara. Berikut adalah beberapa contoh:



```
# Membuat array dari daftar Python
my_list = [1, 2, 3, 4, 5]
my_array = np.array(my_list)

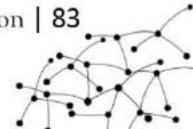
# Membuat array berisi angka nol
zeros_array = np.zeros((3, 3)) # Membuat matriks 3x3 dengan nol

# Membuat array berisi angka satu
ones_array = np.ones((2, 4)) # Membuat matriks 2x4 dengan satu

# Membuat array dengan urutan angka
range_array = np.arange(0, 10, 2) # Membuat array [0, 2, 4, 6, 8]
```

d. Melakukan Operasi Matematika

NumPy juga memiliki keunggulan untuk melakukan operasi matematika pada array dengan mudah. Contoh di bawah ini menunjukkan beberapa operasi dasar:



```
# Penjumlahan
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])
result = array1 + array2 # Hasilnya adalah [5, 7, 9]

# Perkalian
result = array1 * 2 # Hasilnya adalah [2, 4, 6]

# Perkalian antara dua array
result = array1 * array2 # Hasilnya adalah [4, 10, 18]
```

e. Mengakses dan Memanipulasi Data dalam Array

Anda dapat mengakses elemen-elemen dalam array NumPy dan memanipulasinya seperti berikut:

```
# Mengakses elemen berdasarkan indeks
my_array = np.array([1, 2, 3, 4, 5])
element = my_array[2] # Mengakses elemen ke-3 (indeks dimulai dari 0)

# Memotong (slicing) array
sub_array = my_array[1:4] # Mengambil elemen dari indeks 1 hingga 3 (indeks 4 tidak termasuk)

# Mengganti nilai elemen
my_array[0] = 10 # Mengganti nilai elemen pertama menjadi 10
```

f. Menggunakan Fungsi-Fungsi NumPy Lainnya

NumPy memiliki berbagai fungsi yang dapat membantu Anda dalam komputasi numerik, seperti **sum**, **mean**, **max**, **min**, dan banyak lagi. Anda dapat memeriksa dokumentasi NumPy untuk informasi lebih lanjut tentang fungsi-fungsi ini.

Sama halnya dengan library eksternal lainnya cara instalasi serta import dan penggunaan fungsi-fungsinya bisa dilihat pada dokumentasi yang tersedia pada tiap web library yang ada.

9

Algoritma Perulangan

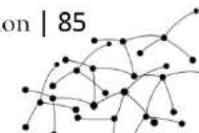
Algoritma perulangan (*looping*) adalah cara atau metode yang digunakan untuk melakukan perintah secara berulang-ulang pada kondisi tertentu sampai suatu kondisi tersebut terpenuhi. Perulangan merupakan aturan dalam pemrograman untuk menjalankan kode program yang sama secara berulang kali tanpa harus mengetik ulang kode yang sama. Pada algoritma perulangan memiliki struktur atau kondisi perulangan yang terdiri dari kondisi awal (perulangan), kondisi proses (saat terjadi perulangan) dan kondisi berhenti (saat perulangan terpenuhi)(Santoso, 2022)

Dalam bahasa pemrograman *Python* struktur perulangan (*looping*) terdiri dari 2 bagian, yaitu :

1. Perulangan *While*
 2. Perulangan *For*

A. Perulangan *while*

Perulangan *while* merupakan proses mengulang blok kode program pada suatu kondisi yang terpenuhi. Perulangan tersebut dapat terjadi secara terus menerus dan akan berhenti jika pada proses perulangan pada blok kode program tersebut ditentukan batas *increment*.



perulangannya (bernilai *true or false*)(Peter Wentworth and Meyers, 2012). Untuk penulisan struktur sintaks perulangan *while* sebagai berikut :

```
Start;  
while <kondisi>  
# blok kode perulangan  
increment
```

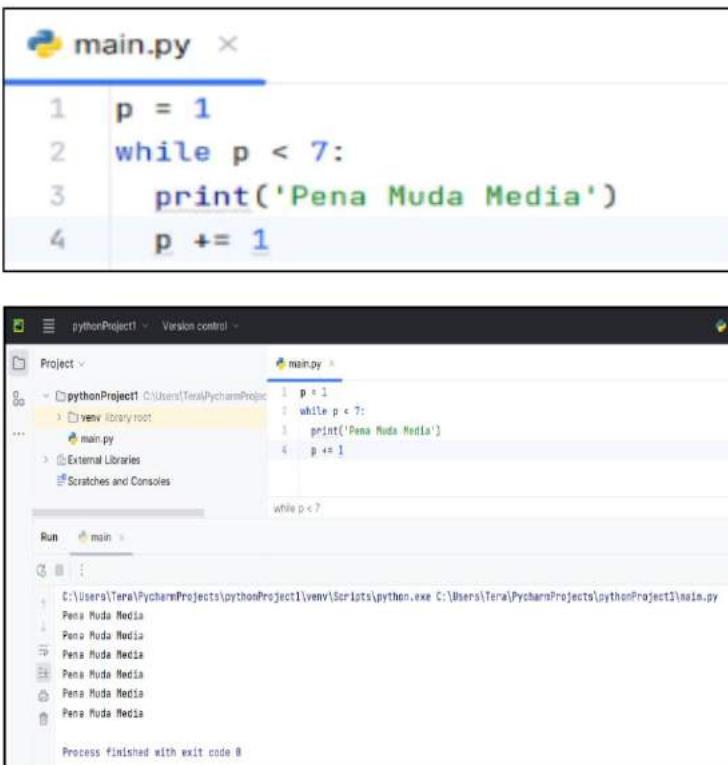
Adapun penjelasan sintaks diatas adalah *start* merupakan inisialisasi *variable counter*, *while* merupakan *keyword* yang wajib diisi, *<kondisi>*: merupakan ekspresi logika dan **blok kode program** merupakan baris kode yang akan dilakukan perulangan (jika kondisi perulangan terpenuhi) dan *increment* merupakan peningkatan jumlah variabel pada jumlah tertentu, misalkan variabel “p”, untuk mengubah variabel dapat digunakan dengan menambahkan angka 1, menjadi $p = p + 1$ atau $p += 1$

Berikut ini adalah contoh sintaks perulangan *while* pada *Python*:

```
main.py ×  
1 p = 1  
2 while p < 7:  
3     print('Pena Muda Media')
```

Hasil dari sintaks diatas, jika dijalankan akan menghasilkan tulisan berulang “Pena Muda Media” tanpa henti, karena pada kondisi *while p < 7* akan selalu bernilai *true*, sehingga terjadi *infinity loop* atau perulangan tanpa batas. Oleh karena itu sintaks tersebut perlu dicantumkan sintaks *increment* agar *infinity loop* tidak terjadi. Untuk menghentikan *infinity loop* tersebut dapat menekan *ctrl +*

c atau menutup paksa *Pycharm* atau sejenis IDE lainnya dengan menekan tombol stop. Berikut ini sintaks yang benar dengan mencantumkan sintaks *increment*



The screenshot shows the PyCharm interface. On the left is the Project tool window with a 'pythonProject1' folder containing 'main.py'. The main editor window shows the following Python code:

```
1 p = 1
2 while p < 7:
3     print('Pena Muda Media')
4     p += 1
```

The code is highlighted with syntax coloring. The run tool window at the bottom shows the output of the execution:

```
C:\Users\Tera\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\Tera\PycharmProjects\pythonProject1\main.py
Pena Muda Media
```

Below the output, it says "Process finished with exit code 0".

Gambar 1. Hasil eksekusi perulangan *while*

Adapun penjelasan dari sintaks diatas, yaitu variable **p=1** merupakan *variable counter* yang digunakan untuk menentukan jumlah perulangan, sedangkan **while p < 7** menjelaskan bahwa perulangan akan dilakukan sebanyak **p** atau perulangan dilakukan sebanyak kurang dari 7 (6 kali perulangan). Pada blok sintaks **print ('Pena Muda Media')** digunakan untuk menampilkan teks atau tulisan **Pena Muda Media** sebagai objek perulangan, dan **p+=1**

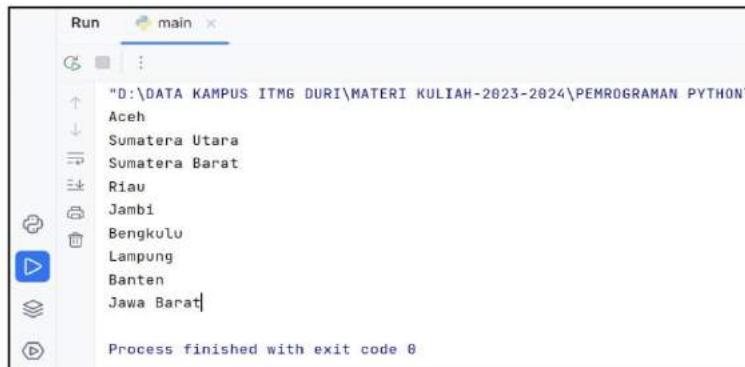
merupakan sintaks yang digunakan untuk menambahkan nilai varibel p sebanyak 1 angka pada setiap perulangan.

1. Perulangan *While* Menggunakan List

Perulangan *while* menggunakan list dilakukan untuk iterasi pada elemen didalam list pada kondisi perulangan yang diberikan tidak terpenuhi lagi. Adapun contoh sintaks perulangan *while* menggunakan *List* pada Bahasa pemrograman *Python* sebagai berikut :

```
listProvinsi = [
    'Aceh', 'Sumatera Utara', 'Sumatera Barat',
    'Riau', 'Jambi',
    'Bengkulu', 'Lampung', 'Banten', 'Jawa Barat'
]
# Contoh_while_menggunakan_list
p = 0
while p < len(listProvinsi):
    print(listProvinsi[p])
    p += 1
```

Adapun hasil dari eksekusi sintaks diatas sebagai berikut :



The screenshot shows a code editor window with a tab labeled "Run" and a file named "main". The code in the editor is identical to the one provided above. The output pane shows the results of the execution:

```
"D:\DATA KAMPUS ITMG DURI\MATERI KULIAH-2023-2024\PEMROGRAMAN PYTHON"
  Aceh
  Sumatera Utara
  Sumatera Barat
  Riau
  Jambi
  Bengkulu
  Lampung
  Banten
  Jawa Barat
Process finished with exit code 0
```

2. Perulangan While Menggunakan *Continue*

Perulangan *while* menggunakan *continue* dilakukan untuk melewati (skip) iterasi saat ini ke iterasi selanjutnya. Adapun contoh sintaks perulangan *while* dengan menggunakan *continue* sebagai berikut :

```
listProvinsi = [
    'Aceh', 'Sumatera Utara', 'Sumatera Barat',
    'Riau', 'Jambi',
    'Bengkulu', 'Lampung', 'Banten'
]
# lewati jika p adalah bilangan genap
# dan p lebih dari 0
p = -1
while p < len(listProvinsi):
    p += 1
    if p % 2 == 0 and p > 0:
        print('lewati')
        continue
    # tidak dieksekusi ketika continue dipanggil
```

Hasil dari eksekusi sintaks diatas dapat dilihat pada Gambar 3

```
Run  While_Continue <
D:\DATA KAMPUS ITMG DURI\MATERI KULIAH-2023-2024\PEMROGRAMAN PYTHON\
+ Aceh
+ Sumatera Utara
lewati
+ Riau
lewati
lewati
Bengkulu
lewati
Banten
lewati
Process finished with exit code 0
PythonProject4 > PythonProject4 > While_Continue.py
```

Gambar 3. Eksekusi *while* menggunakan *Continue*

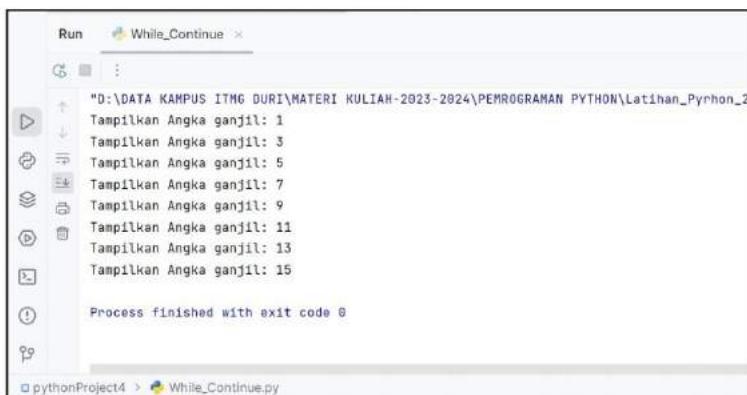
Pada hasil eksekusi sintaks diatas, saat *variable p* merupakan bilangan genap bernilai lebih dari satu, maka perintah `print(listProvinsi[p])` tidak akan

dieksekusi, akan tetapi dilewati (skip). Contoh sintaks lainnya sebagai berikut :

```
# Contoh ke 2 perulangan while dengan continue
p = 1

while p <= 16:
    if p % 2 == 0:
        p += 1
    continue # Lanjutkan ke iterasi
berikutnya jika angka genap
    print("Tampilkan Angka ganjil:", p)
    p += 1
```

Adapun hasil dari eksekusi sintaks diatas dapat dilihat pada Gambar 4.



```
Run While_Continue ×
D:\DATA KAMPUS ITNG DURI\MATERI KULIAH-2023-2024\PEMROGRAMAN PYTHON\Latihan_Python_2
Tampilkan Angka ganjil: 1
Tampilkan Angka ganjil: 3
Tampilkan Angka ganjil: 5
Tampilkan Angka ganjil: 7
Tampilkan Angka ganjil: 9
Tampilkan Angka ganjil: 11
Tampilkan Angka ganjil: 13
Tampilkan Angka ganjil: 15
Process finished with exit code 0
pythonProject4 > While_Continue.py
```

Gambar 4. Contoh *while* menggunakan *Continue*

3. Perulangan While Menggunakan *Break*

Perulangan *while* menggunakan *break* dilakukan untuk menghentikan eksekusi perulangan pada proses kondisi terpenuhi. Penggunaan *break* ini sebanarnya mirip dengan penggunaan *continue*, hanya saja pada saat pemanggilan *break* ini

perulangan akan diberhentikan secara paksa. Adapun contoh sintaks perulangan *while* dengan menggunakan *break* sebagai berikut :

```
# Contoh perulangan while dengan break
NamaBunga = [
    'Mawar', 'Tulip', 'Anggrek', 'Melur',
    'Melati', 'Sakura', 'Kamboja', 'Teratai'
]
Bunga_Yang_Dicari = input('Input Nama Bunga Yang
Dicari: ')
p = 0
while p < len(NamaBunga):
    if NamaBunga[p].lower() ==
Bunga_Yang_Dicari.lower():
        print('Ketemu di index', p)
        break
    print('Bukan Bunga', NamaBunga[p])
    p += 1
```

Sintaks diatas akan menghentikan proses perulangan berdasarkan bunga yang dicari, pada contoh ini bunga dicari adalah sakura yang berada pada index ke - 5 (perhitungan index dimulai dari angka 0). Adapun hasil eksekusi sintaks tersebut dapat dilihat pada Gambar 5.



Gambar 5. Eksekusi *while* menggunakan *break*

Pada perulangan break ini kita dapat menambahkan blok kode *else* yang bertujuan untuk menghentikan perulangan tanpa secara paksa. Adapun contoh sintaks perulangan tersebut sebagai berikut :

```
NamaBunga = [
    'Mawar', 'Tulip', 'Anggrek', 'Melur',
    'Malati', 'Sakura', 'Kamboja', 'Teratai'
]
Bunga_Yang_Dicari = input('Input Nama Bunga Yang
Dicari: ')
p = 0
while p < len(NamaBunga):
    if NamaBunga[p].lower() ==
Bunga_Yang_Dicari.lower():
        print('Ketemu di index', p)
        break
    print('Bukan Bunga', NamaBunga[p])
    p += 1
else :
    print('Mohon Maaf, Bunga yang anda cari tidak
ada didalam List.')
```

Adapun hasil dari eksekusi sintaks diatas dapat dilihat pada Gambar 6.

```
Run White_Break
D:\DATA KAMPUS ITNG DIBI\MATERT KULIAH 2023-2024\PEMROGRAMAN PYTHON\1.a
Input Nama Bunga Yang Dicari: Sakura
Bukan Bunga Mawar
Bukan Bunga Tulip
Bukan Bunga Anggrek
Bukan Bunga Melur
Bukan Bunga Malati
Bukan Bunga Sakura
Bukan Bunga Kamboja
Bukan Bunga Teratai
Mohon Maaf, Bunga yang anda cari tidak ada didalam List.

Process finished with exit code 0
```

Gambar 6. *while* menggunakan *break* dengan *else*

B. Perulangan *For*

Perulangan *for* merupakan perintah iterasi pada suatu nilai *sequence* seperti *list*, *tuple*, *string* dan lain-lain. Perulangan *for* pada bahasa pemrograman *Python* sedikit berbeda dengan perulangan mayoritas bahasa pemrograman lainnya, dikarenakan perulangan *for* pada *Python* lebih dikenal sebagai *foreach* pada bahasa pemrograman lainnya seperti *PHP*. Penggunaan perulangan *for* pada pemrograman *Python* biasanya untuk memproses *array* atau himpunan. Berikut ini adalah struktur perulangan *for* bahasa pemrograman pada *Python*(Kuhlman, 2013).

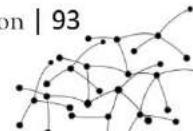
```
TR = [a, b, ...]  
for p in TR:
```

Adapun penjelasan struktur diatas sebagai berikut :

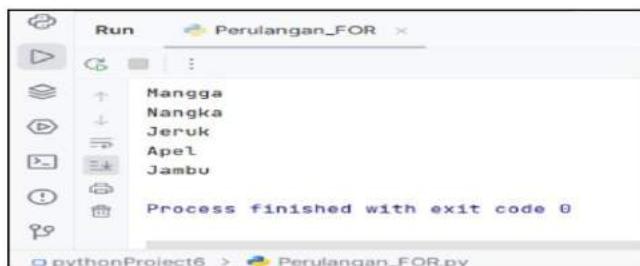
1. TR merupakan variabel suatu himpunan / Array (bias berupa tipe data seperti data *list*, *tuple*, *string*, *dictionary* atau *set*).
2. for p in TR: merupakan perulangan *for* yang dilakukan berdasarkan banyaknya jumlah elemen yang ada pada variabel TR tersebut.

Untuk lebih jelasnya dapat dilihat pada contoh sebagai berikut :

```
Buah =  
['Mangga', 'Nangka', 'Jeruk', 'Apel', 'Jambu']  
for p in Buah:  
    print(p)
```



Adapun hasil *output* dari sintaks tersebut, dapat dilihat pada Gambar 7.



The screenshot shows a Python code editor window titled "Perulangan_FOR". The code inside the editor is:

```
for buah in ["Mangga", "Nangka", "Jeruk", "Apel", "Jambu"]:  
    print(buah)
```

The output window below the code shows the printed values:

```
Mangga  
Nangka  
Jeruk  
Apel  
Jambu  
Process finished with exit code 0
```

Gambar 7. Perulangan menggunakan *For*

1. Perulangan *For* Menggunakan *List*

Perulangan *for* menggunakan list hampir sama dengan penggunaan *while*, yaitu perulangan dilakukan untuk iterasi pada elemen didalam list. Adapun contoh sintaks perulangan *for* menggunakan *list* pada Bahasa pemrograman *Python* sebagai berikut:

```
listProvinsi = [  
    'Aceh', 'Sumatera Utara', 'Sumatera Barat',  
    'Riau', 'Jambi',  
    'Bengkulu', 'Lampung', 'Banten', 'Jawa Barat'  
]  
for Provinsi in listProvinsi:  
    print(Provinsi)
```

Adapun hasil dari eksekusi sintaks diatas dapat dilihat pada Gambar 8.

```
Run For_Pada_List <...>
D:\DATA KAMPUS ITMG DURI\MATERI KULIAH-2023-2024\PEMROGRAMAN PYTHON>
Aceh
Sumatera Utara
Sumatera Barat
Riau
Jambi
Bengkulu
Lampung
Banten
Jawa Barat
Process finished with exit code 0
```

Gambar 8. Perulangan *For* menggunakan *list*

2. Perulangan *For* Menggunakan *Range*

Selain perulangan *for* menggunakan *list*, *for* juga dapat digunakan pada fungsi *range*. Adapun contoh sintaks perulangan *for* menggunakan *range* sebagai berikut :

```
# Perulangan For pada Range untuk Angka 0 sampai
# 7
for p in range(7):
    print("Perulangan Angka ke -", p)
```

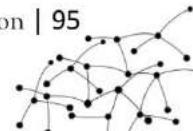
Adapun hasil dari eksekusi sintaks diatas dapat dilihat pada Gambar 9.

```
Run For_Pada_Range <...>
D:\DATA KAMPUS ITMG DURI\MATERI KULIAH-2023-2024\PEMROGRAMAN PYTHON>
Perulangan Angka ke - 0
Perulangan Angka ke - 1
Perulangan Angka ke - 2
Perulangan Angka ke - 3
Perulangan Angka ke - 4
Perulangan Angka ke - 5
Perulangan Angka ke - 6
Process finished with exit code 0
```

Gambar 9. Perulangan *For* menggunakan *range*

Selain sintaks diatas, contoh lainnya sebagai berikut :

```
# Bilangan Genap Kelipatan 4
for bilangan_genap in range(2, 24, 4):
    print(bilangan_genap)
```



```
# Bilangan Ganjil Kelipatan 2
for bilangan_ganjil in range(1, 24, 2):
    print(bilangan_ganjil)
```

3. Perulangan *For*Menggunakan *Tuple*

Perulangan *for* menggunakan *tuple*, untuk mengulang elemen didalam suatu *tuple*. Adapun contoh sintaks perulangan *for* menggunakan *tuple* sebagai berikut :

```
TupleMakanan = ('Rendang', 'Bakso', 'Soto',
                  'Batagor')
for Makanan in TupleMakanan:
    print("Saya suka Makan " + Makanan)
```

Adapun hasil dari eksekusi sintaks diatas dapat dilihat pada Gambar 10

The screenshot shows a terminal window titled 'for Makanan in TupleMakanan'. The window has a toolbar with 'Run' and a file icon. The main area displays the output of the code execution:

```
"D:\DATA KAMPUS ITMG DURI\MATERI KULIAH-2023-2024\PEMROGRAMAN PYTHON\I
Saya suka Makan Rendang
Saya suka Makan Bakso
Saya suka Makan Soto
Saya suka Makan Batagor
Process finished with exit code 0
```

Gambar 10. Perulangan *For*menggunakan *tuple*

Selain sintaks diatas, contoh lainnya sebagai berikut :

```
Deret_tuple = (15, 25, 35, 45, 55)

for p in range(len(Deret_tuple)):
    print(f"Elemen Tuple ke-{p}:
(Deret_tuple[p])")
```

4. Perulangan *For* Menggunakan *String*

Perulangan *for* menggunakan *string*, digunakan untuk mengulang karakter didalam suatu *string* (Herho, 2017). Adapun contoh sintaks perulangan *for* menggunakan *string* sebagai berikut :

```
for karakter_string in "PenaMudaMedia":  
    print(karakter_string)
```

Adapun hasil dari eksekusi sintaks diatas dapat dilihat pada Gambar 11



The screenshot shows a terminal window titled 'Run' with the title bar 'For_String'. The terminal displays the following code and its execution results:

```
for karakter_string in "PenaMudaMedia":  
    print(karakter_string)
```

The output shows each character of the string 'PenaMudaMedia' printed sequentially:

```
P  
e  
n  
a  
M  
u  
d  
a  
M  
e  
d  
l  
a
```

At the bottom of the terminal, the message 'Process finished with exit code 0' is visible.

Gambar 11. Perulangan *For* menggunakan *string*

Selain sintaks diatas, contoh lainnya sebagai berikut :

```
# Menghitung berapa kali karakter tertentu  
# tampil dalam suatu string  
Data_string = "Pena Muda Media!"  
count = 0  
for char in Data_string:  
    if char == 'M':  
        count += 1  
print("Jumlah karakter 'M' dalam string:",  
     count)
```

5. Perulangan *For* Menggunakan *Break*

Perulangan *for* menggunakan *break*, digunakan untuk menghentikan perulangan secara paksa

(interupsi)(Sugiana, 2002). Adapun contoh sintaks perulangan *for* menggunakan *break* sebagai berikut :

```
for p in range(5, 25):
    # hentikan jika p == 20
    if (p == 20):
        break

    print(p)
```

Adapun hasil dari eksekusi sintaks diatas dapat dilihat pada Gambar 12

```
Run For_Break
D:\DATA KAMPUS ITMG DURI\HATERI KULIAH-2023-2
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
Process finished with exit code 0
```

Gambar 12. Perulangan *For*menggunakan *break*

6. Perulangan *For*Menggunakan *Continue*

Perulangan *for* menggunakan *continue*, digunakan untuk melewati iterasi dalam proses perulangan, dan melanjutkan ke iterasi selanjutnya. Adapun contoh sintaks perulangan *for* menggunakan *continue* sebagai berikut :

```
for p in range(5, 15):
    # lewati (Skip) jika p == 9
    if (p == 9):
        continue
    print(p)
```

Adapun hasil dari eksekusi sintaks diatas dapat dilihat pada Gambar 13

```
Run For_Continue <-->
      + - = <--> 
      "D:\DATA KAMPUS ITMG DURI\MATERI KULIAH-2023-2"
      5
      6
      7
      8
      10
      11
      12
      13
      14
      Process finished with exit code 0
```

Gambar 13. Perulangan *For* menggunakan *continue*

Dari hasil *output* pada Gambar 13 tersebut, dapat dilihat bahwa deretan angka dimulai dari *range* 5, dan pada deret angka ke - 8 untuk angka ke - 9 dilewati dan dilanjutkan langsung ke angka ke - 10, dan proses perulangan berhenti sebelum samapi angka ke deretan angka ke -15.

0	0	1	1	1	0	1	0	0	0	0	1	1	0	1	0	1	0	0	0	1
1	0	0	0	0	0	1	0	1	1	1	0	0	1	0	1	0	1	1	0	0
0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0
1	1	1	1	0	1	0	1	0	1	1	0	0	1	0	1	1	1	1	1	1
0	0	1	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0
0	1	0	0	1	0	1	0	0	1	1	0	0	0	1	1	0	0	1	0	1
0	0	1	1	1	0	1	0	0	1	0	1	0	1	0	1	1	0	0	0	0
1	1	1	0	0	1	0	1	0	1	1	0	0	1	0	1	1	1	1	1	1
0	1	0	1	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1	0	1
1	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1
0	1	0	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1

10

Fungsi

Fungsi adalah sekumpulan perintah kode program yang dijadikan satu dan dapat digunakan kembali atau reusable, cukup sekali mendefinisikannya dan dapat menggunakananya berulang-ulang (Ramalho 2022). Fungsi di Python ditandai dengan kata def.

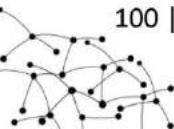
A. Sintak fungsi

Ada beberapa aturan dasar sintak penulisan fungsi selalu diawali dengan kata kunci def spasi nama fungsi dalam kurung parameter jika ada dan diakhiri titik dua. Untuk format penulisan dapat dilihat di bawah ini:

```
def <namaFungsi> (parameter):
    statements
```

Sama dengan penggunaan perulangan dan kondisi, blok kode fungsi diawali setelah penulisan titik dua. Semua perintah dalam blok kode fungsi akan menjorok ke dalam dari pendefinisan fungsi. Contoh dari penulisan sebuah fungsi sederhana dengan nama helloworld():

```
def helloworld():
```



```
print('Hello world!!!')
```

Dari contoh di atas, diketahui nama fungsinya adalah helloworld dan dimana parameter di dalam kurung tidak ada. Jika kode fungsi ini dijalankan maka akan mencetak

```
Hello world!!!
```

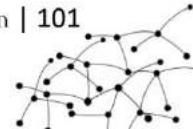
B. Memanggil Fungsi

Kode fungsi tidak akan dieksekusi jika nama fungsi tidak pernah dipanggil. Maka dari itu, untuk menjalankan fungsi cukup dengan menuliskan namafungsi atau dengan format

```
namaFungsi()
```

Dengan memanggil nama fungsi maka kemudian perintah di dalam fungsi akan dieksekusi. Cara memanggil fungsi cukup dengan mengetik nama fungsi dan diikuti tanda kurung () seperti di bawah ini:

```
#contoh fungsi
def helloworld():
    print('Hello world!!!')
#panggil fungsi
helloworld()
```



Output yang akan dihasilkan

Hello world!!!

Jika nama fungsi dipanggil berkali-kali maka perintah dalam fungsi tersebut akan di eksekusi berkali kali.

```
helloworld()
```

```
helloworld()
```

```
helloworld()
```

Output

Hello world!!!

Hello world!!!

Hello world!!!

Untuk lebih jelasnya, perhatikan contoh berikut:

```
def luasPersegi():
```

```
luas=sisi*sisi
```

```
print('Luas:',luas)
```

```
#pemanggilan ke-1
```

```
sisi=4
```

```
luasPersegi()
```

```
#pemanggilan ke-2
```

```
sisi=7
```

```
luasPersegi()
```

Dengan adanya fungsi luasPersegi() perhitungan luas persegi dengan variasi nilai sisi yang beragam dapat dilakukan berkali-kali. Cukup dengan sekali mendefinisikan fungsi, fungsi dapat dipanggil berkali-kali. Hal inilah yang menjadi keunggulan dari penggunaan fungsi.

C. Jenis-jenis Fungsi

Fungsi dapat dibedakan menjadi 2 jenis jika dikategorikan berdasarkan nilai balik(Mattes 2015), yaitu:

1. Fungsi tanpa nilai balik
2. Fungsi dengan nilai balik

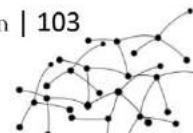
Pada beberapa contoh sebelumnya sudah banyak contoh fungsi yang tidak mengembalikan nilai. Karena dari contoh sebelumnya belum ada fungsi yang menggunakan return yang berfungsi untuk mengembalikan nilai. Nilai yang dikembalikan fungsi dapat digunakan untuk dimanipulasi lagi.

Adapun format penulisan fungsi dengan nilai balik adalah sebagai berikut:

```
def <namaFungsi> (parameter):  
    statements  
    return nilai
```

Berikut contoh fungsi dengan nilai balik:

```
def tarifparkir():  
    tarif=5000  
    if(jam>=1):
```



```
tarif=tarif+(jam*3000)  
return tarif
```

Struktur penulisan fungsi dengan nilai balik hanya memiliki perbedaan pada statement return di akhir fungsi. Dimana return bermakna mengembalikan sebuah nilai ketika fungsi tersebut dipanggil. Jika statement return dieksekusi maka kode program setelah return dalam fungsi tersebut tidak akan dieksekusi. Maka dari itu, return biasanya diurutan terakhir dalam fungsi. Karena ada nilai yang akan dikembalikan, maka cara pemanggilan fungsi dengan nilai balik agak berbeda, yaitu dengan menyiapkan variabel untuk menampung nilai balik dari fungsi.

```
variabel=namaFungsi()
```

Contoh

```
#cara 1 menyimpan pada variabel
```

```
jam=5
```

```
biaya=tarifparkir()
```

```
print('Biaya parkir: ',biaya)
```

Atau langsung digunakan tanpa variabel

```
jam=5
```

```
#cara 2
```

```
print('Biaya parkir: ',tarifparkir())
```

Dari contoh pemanggilan fungsi dengan menggunakan variabel di atas, variabel biaya akan menyimpan nilai tarif yang diperoleh dari fungsi tarifparkir(). Nilai yang

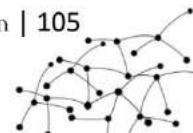
dikembalikan oleh fungsi dengan nilai balik dapat diolah kembali seperti melanjutkan operasi selanjutnya atau hanya sekedar menyimpannya dalam sebuah variabel.

Sebuah fungsi dapat memiliki return lebih dari 1. Dengan ketentuan, jika satu return sudah dieksekusi maka kode program di bawah return tersebut tidak akan dieksekusi. Melihat cara kerja return maka ini akan mengingatkan dengan perintah break pada perulangan. Perhatikan contoh fungsi di bawah ini yang memiliki return lebih dari satu:

```
def cariMax():
    if(bil1>bil2):
        return bil1 #return pertama
    return bil2 #return kedua

bil1=30
bil2=15
maks=cariMax()
print('Bilangan terbesar dari ', bil1,' dan ', bil2, ':', maks)
```

Fungsi cariMax() berfungsi untuk membandingkan bil1 dan bil2, jika bil1 lebih besar dari bil2 maka return pertama yang akan dieksekusi dan return kedua akan dilewati atau diabaikan. Namun, jika bil1 tidak lebih besar dari bil2, maka return kedua yang akan dieksekusi. Sehingga keluaran yang dihasilkan oleh kode program di atas adalah:



Bilangan terbesar dari 30 dan 15 : 30

Apabila nilai bil1=20 dan bil2=45 maka akan menghasilkan keluaran

Bilangan terbesar dari 20 dan 45 : 45

D. Parameter

Sebuah nilai/variabel dapat dilemparkan atau dikirimkan ke dalam sebuah fungsi melalui parameter. Sehingga, akan menghasilkan fungsi yang dinamis yang tergantung dengan nilai parameter yang diberikan. Misal ketika ingin membuat sebuah fungsi tambahLima() dimana setiap bilangan yang dikirimkan ke fungsi akan menambahkannya dengan 5 sehingga kode programnya adalah sebagai berikut:

```
def tambahLima(bilangan):  
    print(bilangan+5)
```

Agar fungsi ini dieksekusi, fungsi ini perlu dipanggil terlebih dahulu. Cara memanggil fungsinya cukup dengan nama fungsi diikuti dengan nilai/variabel yang akan dikirimkan ke parameter.

```
#passing by value  
tambahLima(10)  
#passing by variabel  
nilai=10  
tambahLima(nilai)
```

Mengirimkan nilai langsung ke parameter dikenal dengan istilah passing by value dan mengirimkan nilai ke

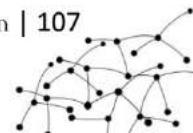
parameter melalui sebuah variabel dikenal dengan istilah passing by variable. Kedua metode pengiriman nilai ini akan menghasilkan keluaran yang sama yaitu sama-sama mengembalikan nilai 15.

Contoh lain dari fungsi yang memiliki parameter adalah kode penentuan predikat kelulusan seperti di bawah ini:

```
def predikat(ipk):
    if ipk>3.5:
        return 'dengan pujian'
    elif ipk>3:
        return 'sangat memuaskan'
    else:
        return 'memuaskan'
    #passing by value
    print(predikat(3.7))

#passing by variable
nilai=3.2
print(predikat(nilai))
```

Terdapat satu parameter pada fungsi predikat(), yaitu: ipk. Nilai ipk dapat dikirimkan dengan menggunakan value atau variable.



E. Parameter Wajib

Fungsi dengan parameter seperti contoh sebelumnya fungsi tambahLima() sudah menggunakan parameter wajib. Dimana setiap pemanggilan tambahLima() harus memberikan nilai untuk parameter bilangan. Karena jika tidak, maka akan terdapat pesan error seperti ini:

```
def tambahLima(bilangan):  
    print(bilangan+5)  
    #passing by value  
tambahLima()
```

TypeError

Traceback (most recent call last)

Cell In[8], line 4

```
2   print(bilangan+5)  
3 #passing by value  
----> 4 tambahLima()
```

TypeError: tambahLima() missing 1 required positional argument: 'bilangan'

Hal ini disebabkan parameter bilangan adalah contoh dari parameter wajib, dimana setiap pemanggilan fungsi wajib mengirimkan nilai ke parameter tersebut. Sebagai contoh:

```
def hitung(nilai1,nilai2):
```

```
return nilai1+nilai2
```

```
print('hasil:',hitung(15,20))
```

Jika dijalankan akan menghasilkan keluaran seperti ini:

```
hasil: 35
```

Pada kode pemanggilan fungsi hitung nilai yang dikirimkan adalah 15 dan 20. Bilangan 15 akan disimpan ke dalam parameter nilai1 dan bilangan 20 akan disimpan ke nilai2. Hal ini karenakan, pengiriman nilai akan dilakukan secara terurut. Misal, kode pemanggilan fungsi hitung diubah menjadi:

```
print('hasil:',hitung(20,15))
```

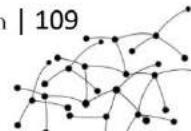
akan tetap menghasilkan output yang sama yaitu 35 hanya saja berbeda dinilai parameter nilai1 menjadi 20 dan nilai2 menjadi 15.

F. Parameter Opsional

Pada Python juga terdapat parameter opsional atau dikenal juga dengan default. Parameter ini tidak wajib diisi dikarenakan dia sudah memiliki nilai default jika tidak diisi. Perhatikan contoh fungsi yang memiliki parameter wajib:

```
def datadiri(nama, tinggi, satuan):
```

```
    print('Nama:',nama,' tinggi badan:',tinggi,satuan)
```



```
datadiri('Sari',168,'cm')
```

Dari kode program di atas didapatkan keluaran:

Nama: Sari tinggi badan: 168 cm

Dimana dari ketiga parameter semuanya adalah parameter wajib, sehingga nilai semua parameter harus dikirimkan setiap fungsi dipanggil. Sekarang parameter dari fungsi datadiri() akan dirubah menjadi seperti di bawah ini:

```
def datadiri(nama, tinggi, satuan='cm'):  
    print('Nama:',nama,', tinggi badan:',tinggi,satuan)
```

Pada fungsi di atas, sudah didefinisikan tiga parameter yaitu:

1. nama: parameter wajib
2. tinggi: parameter wajib
3. satuan: parameter opsional

Dari ketiga parameter tersebut, hanya parameter satuan yang boleh tidak diisi atau tidak diberikan nilai pada saat fungsi dipanggil. Jika parameter satuan tidak diisi maka secara default isi dari parameter satuan adalah 'cm'. Dengan demikian cara memanggil fungsi datadiri() terdapat dua cara, yaitu dengan memberikan nilai satuan atau tidak seperti yang dapat dilihat pada kode di bawah ini:

```
#mengirimkan nilai satuan  
datadiri('Sari',1.68,'m')  
#menggunakan nilai default  
datadiri('Sari',168)  
Jika di run, keluaran yang akan diperoleh adalah:  
Nama: Sari, tinggi badan: 1.68 m  
Nama: Sari, tinggi badan: 168 cm
```

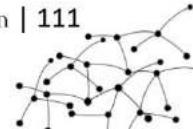
G. Parameter Tidak Berurutan

Sebelum membahas materi mari perhatikan kode program di bawah ini:

```
def hitunggaji(gapok, tunjangan=0, lembur=0):  
    print('Gaji Pokok:',gapok)  
    print('Tunjangan:', tunjangan)  
    print('Lembur:', lembur)  
    return gapok+tunjangan+lembur
```

Pada hitunggaji(), terdapat 2 parameter opsional yaitu tunjangan dan lembur dengan nilai default sama dengan 0. Mari coba panggil fungsi hitunggaji().

```
total=hitunggaji(1000000, 200000)  
print('Total gaji:',total)
```



Keluaran yang dihasilkan dari kode pemanggilan di atas adalah sebagai berikut:

Gaji Pokok: 1000000

Tunjangan: 200000

Lembur: 0

Total gaji: 1200000

Dari kode pemanggilan fungsi hanya ada dua nilai yang dikirimkan yaitu 1.000.000 ke parameter gapok dan 200.000 ke parameter tunjangan. Hal ini disebabkan, pengiriman nilai secara default dilakukan sesuai dengan urutan penulisan parameter pada definisi fungsi.

Lalu bagaimana jika ingin mengirimkan nilai hanya ke parameter gapok dan lembur saja? Apakah bisa? Jawabannya bisa, dengan cara pada saat pemanggilan nama fungsi harus menuliskan nama parameter dan diikuti nilai pada parameter tersebut. Sebagai contoh:

```
total=hitunggaji(gapok=1000000, lembur=200000)  
print("Total gaji:",total)
```

Dengan cara memanggil fungsi yang diikuti nama parameter maka urutan pengiriman nilai tidak sesuai dengan urutan nama parameter pada saat pendefinisian fungsi, tetapi tergantung dengan urutan pemanggilan nama parameter pada saat pemanggilan fungsi. Menggunakan cara ini passing parameter dapat dilakukan secara acak atau tidak harus terurut lagi.

Sehingga, output yang akan dihasilkan dari kode di atas adalah sebagai berikut:

Gaji Pokok: 1000000

Tunjangan: 0

Lembur: 200000

Total gaji: 1200000

Dapat dilihat bahwa, parameter tunjangan tidak diberikan nilai pada saat pemanggilan sehingga nilai yang digunakannya adalah nilai default yaitu 0. Sedangkan untuk parameter menggunakan nilai yang dikirimkan pada saat pemanggilan fungsi.

H. Parameter *args

Terkadang parameter yang dibutuhkan pada fungsi adalah parameter yang memiliki jumlah nilai yang tidak menentu layaknya array. Hal ini dapat dilakukan dengan cara menggunakan tanda asterik atau bintang pada nama parameter.

Sebagai contoh:

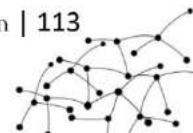
```
def total(*bilangan):
    print(bilangan)
total(1, 2, 3, 4, 5, 6, 'kadang')
```

Dari contoh di atas memiliki output:

```
(1, 2, 3, 4, 5, 6, 'kadang')
```

I. Parameter **kwargs

Fungsi parameter keyword arguments atau sering disingkat dengan kwargs adalah parameter yang memiliki jumlah parameter tidak terbatas. Cukup dengan memanggil



fungsi dengan diikuti dengan pendefenisian nama parameter. Seperti contoh di bawah ini, hanya ada satu parameter dengan nama keywords pada fungsi terbesar(). Namun pada pemangilan, terdapat 3 parameter yang dikirimkan.

```
def terbesar(**keywords):
    max = 0;
    for key in keywords:
        max = keywords[key]
    return max

print (terbesar(bil1=3, bil2=4, bil5=10))
```

0	0	1	1	1	0	1	0	0	0	1	1	0	1	0	1	0	0	0	1
1	0	0	0	0	0	1	0	1	1	1	0	0	1	0	1	0	1	1	0
0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0
1	1	1	1	0	1	0	1	0	1	1	0	0	1	0	1	1	1	1	1
0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	0	0	0	0	0
0	1	0	0	1	0	0	1	1	0	1	0	0	0	1	0	0	1	0	1
0	0	1	1	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0	0
1	1	1	0	1	0	0	1	0	1	1	0	0	1	0	1	1	1	1	1
0	1	1	0	0	1	0	0	1	1	1	0	1	1	0	1	0	0	0	0
0	0	1	1	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0	0
1	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

11

Array

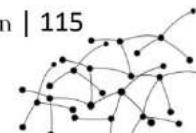
Array adalah sebuah struktur data yang digunakan dalam pemrograman untuk menyimpan sekumpulan nilai atau elemen yang serupa atau sejenis. Elemen-elemen ini disimpan dalam urutan tertentu dan dapat diakses menggunakan indeks. Indeks adalah bilangan bulat yang digunakan untuk mengidentifikasi lokasi atau posisi elemen dalam array. Analogi array dalam kehidupan sehari-hari dapat membantu Anda memahami konsep ini dengan lebih baik. Berikut beberapa analogi array dalam kehidupan sehari-hari:

1. Kotak Barang

Bayangkan Anda memiliki kotak penyimpanan di rumah Anda. Kotak ini memiliki beberapa laci, dan setiap laci berisi jenis barang yang berbeda. Setiap laci dapat dianggap sebagai elemen dalam array, dan setiap jenis barang adalah nilai dalam elemen tersebut. Anda dapat dengan mudah mengakses barang tertentu dengan merujuk ke laci yang tepat.

2. Lemari Pakaian

Lemari pakaian Anda mungkin berisi beberapa rak atau bagian yang berisi pakaian Anda. Setiap rak dapat dianggap sebagai elemen dalam array, dan pakaian di setiap rak adalah nilai-nilai dalam elemen tersebut. Anda



dapat dengan cepat mencari pakaian tertentu dengan mengetahui rak yang sesuai.

3. Playlist Musik

Saat Anda membuat playlist musik di perangkat Anda, setiap lagu dalam playlist adalah elemen dalam array musik. Anda dapat mengatur lagu-lagu ini dalam urutan tertentu, dan Anda dapat dengan mudah memutar lagu-lagu sesuai urutan yang telah Anda tentukan.

Analogi-analogi ini membantu menjelaskan bagaimana array digunakan untuk mengatur dan mengelola data dalam kehidupan sehari-hari. Seperti dalam pemrograman, penggunaan array memungkinkan kita untuk menyimpan dan mengakses data dengan lebih efisien.

A. Array didalam Bahasa Python

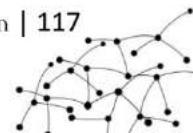
Pengertian array didalam bahasa Python sama halnya dengan array pada bahasa pemrograman yang lain. Dimana array merupakan variable tunggal yang digunakan untuk menyimpan beberapa nilai yang bisa diakses atau dimodifikasi dengan menggunakan indeks (Kurniawan, 2022).

Array didalam bahasa Python hampir sama dengan List yang ada didalam bahasa pemrograman tetapi juga mempunyai perbedaan. Dalam bahasa Python, "Array" dan "List" adalah dua konsep yang sering membingungkan, terutama karena dalam Python, "List" adalah struktur data yang digunakan untuk menyimpan koleksi elemen, sementara "Array" sendiri bukanlah struktur data bawaan seperti di beberapa bahasa pemrograman lainnya. Namun, list dapat digunakan untuk membuat struktur data yang

mirip dengan array. Berikut adalah tabel perbedaan antara List dan Array dalam bahasa Python.

Array	List
Terdiri dari elemen yang mempunyai tipe data yang sama	Untuk elemen pendirinya bisa berbeda-beda untuk tipe datanya
Untuk proses deklarasi perlu mengimpor sebuah modul	Untuk proses deklarasi tidak butuh untuk mengimpor modul eksplisit
Operasi aritmatika bisa langsung ditangani	Operasi aritmatika tidak bisa langsung ditangani
Lebih hemat memori karena hanya terdiri dari satu tipe data yang sama	Lebih boros memori karena terdiri dari banyak tipe data
Fleksibilitasnya masih kurang	Fleksibilitas besar

Jadi, dalam bahasa Python, List adalah struktur data yang lebih umum digunakan untuk menyimpan koleksi elemen dengan tipe data yang beragam, sementara NumPy array adalah struktur data yang lebih spesifik untuk operasi numerik dan biasanya memiliki tipe data yang seragam. Modul yang bisa kita pakai untuk membuat array didalam bahasa python adalah “*arrays*” dan “NumPy”. Untuk NumPy (Numerical Python) merupakan modul yang lebih disarankan didalam pembuatan array dikarenakan lebih fleksibel, memungkinkan operasi matematika, dan bisa untuk multidimensi.



B. Fungsi Array didalam Bahasa Python

Dalam bahasa Python, meskipun tidak ada tipe data array bawaan, tetapi dapat menggunakan array numerik menggunakan pustaka NumPy. Array NumPy sangat berguna dalam berbagai konteks, terutama ketika Anda perlu melakukan pemrosesan data numerik (Swastika, 2019). Berikut adalah beberapa kegunaan utama array NumPy dalam Python:

1. Komputasi Numerik: NumPy menyediakan array multidimensi yang efisien dan alat pemrosesan data numerik. Ini membuatnya sangat cocok untuk tugas-tugas komputasi numerik seperti aljabar linear, statistik, analisis data, dan perhitungan ilmiah.
2. Pemrosesan Data: NumPy memungkinkan Anda untuk dengan mudah melakukan operasi matematika pada seluruh array atau elemen-elemen array. Ini sangat berguna saat bekerja dengan dataset besar atau matriks numerik.
3. Ilmu Data (Data Science): NumPy adalah salah satu pustaka dasar dalam ekosistem ilmu data Python. Banyak pustaka ilmu data lainnya seperti pandas, scikit-learn, dan TensorFlow, menggunakan NumPy *arrays* sebagai dasar untuk pemrosesan data mereka.
4. Visualisasi Data: NumPy *arrays* sering digunakan sebagai input untuk pustaka visualisasi data seperti Matplotlib atau Seaborn. Anda dapat dengan mudah mengambil data dari array NumPy dan membuat grafik atau plot yang informatif.
5. Pemodelan Statistik: NumPy menyediakan banyak fungsi statistik yang berguna untuk menganalisis data,

seperti perhitungan rata-rata, median, deviasi standar, korelasi, dan lain-lain.

Secara keseluruhan NumPy array adalah alat yang cocok untuk pemrosesan data numerik dalam Python dan digunakan luas dalam berbagai bidang seperti ilmu pengetahuan data, ilmu pengetahuan komputer, rekayasa, ilmu fisika, dan banyak lagi.

C. Bagaimana cara menginisialisai Array dalam Python

Karena didalam Python tidak ada tipe data Array bawaan seperti di beberapa bahasa pemrograman lainnya (seperti C atau Java). Maka jika ingin bekerja dengan array numerik dan melakukan operasi matematika, dapat menggunakan pustaka NumPy, yang memungkinkan Anda membuat dan mengelola array numerik. Berikut adalah langkah-langkah untuk bekerja dengan NumPy *arrays* di Python:

1. Menginstal NumPy (jika belum terinstal):

Jika belum menginstal NumPy, maka dapat diinstalasi terlebih dahulu melalui terminal atau bash dengan menggunakan pip, perintah berikut:

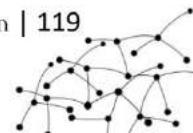
pip install numpy

A screenshot of a terminal window titled "bash". Inside the window, the command "pip install numpy" is being typed. In the top right corner of the terminal window, there is a "Copy code" button.

Gambar 1. Proses install numpy.

2. Impor NumPy:

Selanjutnya adalah mengimpor NumPy ke dalam program Python sebelum dapat menggunakannya:



```
import numpy as np
```



```
python Copy code  
import numpy as np
```

Gambar 2. Proses import numpy kedalam Python.

3. Membuat Array NumPy:

Setelah itu bisa dimulai untuk membuat array NumPy dengan berbagai cara. Beberapa contoh dasar adalah:

Membuat array dari daftar Python:

```
my_array = np.array([1, 2, 3, 4, 5])
```

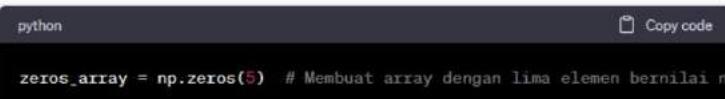


```
python Copy code  
my_array = np.array([1, 2, 3, 4, 5])
```

Gambar 3. Contoh pembuatan array dari daftar python.

Membuat array dengan nilai nol:

```
zeros_array = np.zeros(5) # Membuat array dengan lima elemen bernilai nol
```



```
python Copy code  
zeros_array = np.zeros(5) # Membuat array dengan lima elemen bernilai nol
```

Gambar 4. Contoh pembuatan array dengan nilai nol.

Membuat array dengan nilai satu:

```
ones_array = np.ones(3) # Membuat array dengan tiga elemen bernilai satu
```

```
python                                Copy code
ones_array = np.ones(3) # Membuat array dengan tiga elemen bernilai satu
```

Gambar 5. Contoh pembuatan array dengan nilai satu.

4. Melakukan Operasi pada Array NumPy:

Selain membuat operasi dasar array, dapat dilakukan juga untuk berbagai operasi matematika pada array NumPy seperti penambahan, pengurangan, perkalian, dan lainnya. Contoh:

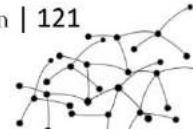
```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
# Penambahan elemen-elemen array
result = a + b # [5, 7, 9]
# Perkalian elemen-elemen array
result = a * b # [4, 10, 18]
```

```
python                                Copy code
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

# Penambahan elemen-elemen array
result = a + b # [5, 7, 9]

# Perkalian elemen-elemen array
result = a * b # [4, 10, 18]
```

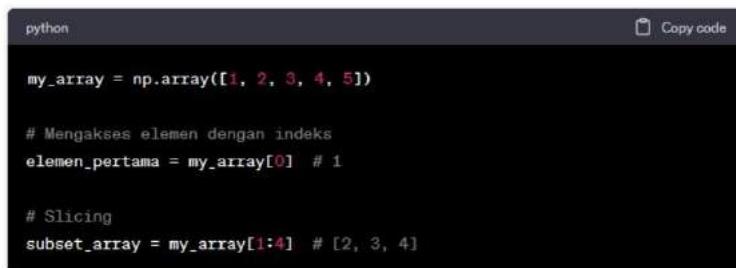
Gambar 6. Contoh operasi array pada numpy.



5. Mengakses Elemen dalam Array NumPy:

Dapat juga dilakukan untuk mengakses elemen-elemen dalam array NumPy menggunakan indeks, slicing, atau dengan menggunakan berbagai metode yang disediakan oleh NumPy.

```
my_array = np.array([1, 2, 3, 4, 5])  
# Mengakses elemen dengan indeks  
elemen_pertama = my_array[0] # 1  
# Slicing  
subset_array = my_array[1:4] #[2, 3, 4]
```



```
python  
Copy code  
  
my_array = np.array([1, 2, 3, 4, 5])  
  
# Mengakses elemen dengan indeks  
elemen_pertama = my_array[0] # 1  
  
# Slicing  
subset_array = my_array[1:4] # [2, 3, 4]
```

Gambar 7. Contoh akses elemen array pada numpy.

6. Mengubah Ukuran Array:

Berikut contoh untuk mengubah ukuran (shape) array NumPy dengan berbagai metode yang disediakan oleh NumPy. Ini termasuk reshape, resize, dan lain-lain.

```
my_array = np.array([1, 2, 3, 4, 5])
```

```
# Mengubah ukuran array  
  
reshaped_array = my_array.reshape((1, 5)) #  
Mengubah menjadi matriks baris 1x5
```



```
python  
  
my_array = np.array([1, 2, 3, 4, 5])  
  
# Mengubah ukuran array  
reshaped_array = my_array.reshape((1, 5)) # Mengubah menjadi matriks baris
```

Gambar 8. Contoh mengubah ukuran array pada numpy.

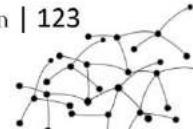
Diatas merupakan langkah-langkah dasar untuk bekerja dengan NumPy *arrays* di Python. NumPy adalah pustaka yang sangat kuat untuk pengolahan data numerik dan ilmu data dalam Python.

7. Jenis-jenis array dalam bahasa python

Pada umumnya untuk bekerja dengan kumpulan beberapa elemen didalam bahasa Python adalah dengan menggunakan list dan NumPy *arrays*. Berikut ini akan dijelaskan beberapa jenis array yang umum digunakan dalam Python:

- a. List: List adalah tipe data bawaan Python yang digunakan untuk menyimpan koleksi elemen. Ini adalah array dinamis yang dapat berisi elemen dengan tipe data yang berbeda. Misalnya:

```
my_list = [1, 2, 3, 4, 5]
```



```
python
my_list = [1, 2, 3, 4, 5]
Copy code
```

Gambar 9. Contoh tipe data list.

- b. NumPy Array: NumPy adalah pustaka yang digunakan untuk bekerja dengan array numerik. NumPy array adalah jenis array yang sangat efisien dan digunakan luas dalam pemrosesan data numerik, ilmu data, dan komputasi ilmiah.

```
import numpy as np
my_array = np.array([1, 2, 3, 4, 5])
```

```
python
import numpy as np
my_array = np.array([1, 2, 3, 4, 5])
Copy code
```

Gambar 10. Contoh numpy array.

- c. Array NumPy multi dimensi: Ini adalah array NumPy dengan dua dimensi atau lebih. Mereka digunakan untuk mewakili data dalam bentuk tabel, seperti dataset ilmu data atau gambar berwarna.

```
import numpy as np
my_3d_array = np.array([[1, 2], [3, 4]], [[5, 6], [7, 8]])
```

```
python
import numpy as np
my_3d_array = np.array([[1, 2], [3, 4]], [[5, 6], [7, 8]])
Copy code
```

Gambar 11. Contoh array multi dimensi.

- d. Array Karakter (String): membuat array yang berisi karakter atau string dalam Python. Ini berguna untuk pemrosesan teks.

char_array = np.array(['a', 'b', 'c', 'd'])

```
python Copy code  
char_array = np.array(['a', 'b', 'c', 'd'])
```

Gambar 12. Contoh array karakter.

- e. Array Bilangan Kompleks: NumPy juga mendukung array yang berisi bilangan kompleks.

complex_array = np.array([1 + 2j, 3 - 4j, 5 + 6j])

```
python Copy code  
complex_array = np.array([1 + 2j, 3 - 4j, 5 + 6j])
```

Gambar 13. Contoh array bilangan array.

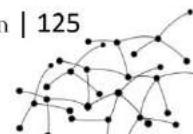
- f. Array *Ragged*: Ini adalah jenis array yang elemennya bisa memiliki panjang yang berbeda. Mereka sering digunakan dalam pemrosesan teks dan data berstruktur.

ragged_array = [[1, 2], [3, 4, 5], [6]]

```
python Copy code  
ragged_array = [[1, 2], [3, 4, 5], [6]]
```

Gambar 14. Contoh array *ragged*.

Pilihan jenis array yang tepat tergantung pada kebutuhan Anda dalam pengembangan perangkat lunak. Python sangat fleksibel dan dapat menangani berbagai jenis array sesuai dengan kebutuhan aplikasi yang diinginkan.



12

Algoritma Rekursi

Materi paling menarik dalam algoritma pemrograman di bahasa pemrograman apapun adalah pembahasan mengenai rekursif. Biasanya materi ini diberikan di chapter terakhir dalam buku pemrograman, dengan kata lain rekursif ini adalah materi puncak dalam algoritma pemrograman.

Rekursif itu sendiri adalah teknik menyelesaikan persoalan dimana di dalamnya mengandung definisi persoalan itu sendiri. Di dalam dunia pemrograman, fungsi rekursif merupakan sebuah metode perulangan yang bersifat non-iterasi. Sebenarnya fungsi rekursif hanyalah sebuah fungsi biasa seperti fungsi def pada umumnya. Dia bisa dipanggil, bisa menerima parameter, bisa mengembalikan nilai, dan lain sebagainya. Hanya saja, sesuai namanya, fungsi rekursif itu bersifat rekursif. Karena ia memanggil dirinya sendiri sehingga menimbulkan efek perulangan. Perulangan ini bisa berhenti ketika kondisi tertentu tercapai, atau bisa juga bersifat tak terbatas, atau mungkin bahkan bisa menimbulkan error karena pemanggilan fungsi yang tak ada habisnya.

Dalam keseharian, rekursif ini dianalogikan dengan 2 cermin yang sedang berhadapan. Cermin-cermin tersebut saling memantulkan satu sama lain, sehingga hasil pantulan satu cermin akan dipantulkan lagi dan lagi secara terus menerus. Akhirnya hal tersebut menimbulkan efek cermin di dalam cermin di dalam cermin di dalam cermin di dalam cermin dan seterusnya seperti yang terlihat pada Gambar berikut.



Gambar 1. Analogi Rekursi dengan cermin

A. Kelebihan Rekursi

Algoritma rekursi memiliki beberapa kelebihan dibandingkan dengan menggunakan looping biasa, diantaranya adalah:

1. Efisiensi kode: Dalam beberapa kasus, penggunaan fungsi rekursif dapat menghasilkan kode yang lebih



- efisien dan lebih mudah dibaca dibandingkan dengan penggunaan loop.
2. Fleksibilitas: Fungsi rekursif memungkinkan pengguna untuk menghasilkan program yang lebih fleksibel dan dapat menangani masalah yang kompleks.
 3. Kebutuhan memori rendah: Fungsi rekursif membutuhkan memori yang lebih sedikit dibandingkan dengan loop yang mengulang tugas yang sama.

B. Mendefinisikan Rekursi

Fungsi rekursif terdiri dari dua bagian penting yaitu:

1. Basis

Basis adalah bagian yang berisi nilai fungsi yang terdefinisi secara eksplisit. Bagian ini juga sekaligus menghentikan rekursif dan memberikan sebuah nilai yang terdefinisi pada fungsi rekursif.

2. Rekurens

Bagian ini mendefinisikan fungsi dalam terminologi dirinya sendiri. Berisi kaidah untuk menemukan nilai fungsi pada suatu input dari nilai-nilai lainnya pada input yang lebih kecil.

Misalkan kita ingin mendefinisikan rekursi untuk penyelesaian faktorial. Penyelesaian faktorial harus dilakukan secara iteratif, dimana:

$$n! = (n)(n-1)(n-2) \dots (1)$$

$$1! = 1$$

$$0! = 1$$

Dengan menggunakan rekursi, faktorial dapat didefinisikan dengan rumus sebagai berikut:

$$f(n) = \begin{cases} 1, & n = 0, n = 1 \\ nf(n - 1), & n > 1 \end{cases}$$

Basis
Rekurens

C. Fase pada Rekursi

Proses penyelesaian masalah menggunakan rekursi mengalami dua fase yaitu fase awal dan fase balik.

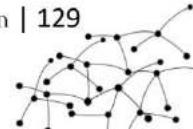
1. Fase Awal

Pada fase ini, masing-masing proses memanggil dirinya sendiri. Fase akan berhenti Ketika sudah mencapai kondisi basis. Contoh dalam penyelesaian faktorial dari n , kondisi basisnya adalah $n=0$, dan $n=1$. Pada kondisi tersebut fungsi sudah tidak memanggil dirinya sendiri dan kembali pada sebuah nilai (dalam kasus ini nilai =1).

2. Fase Balik

Pada fase ini, fungsi sebelumnya akan dikunjungi lagi. Fase ini berlanjut sampai pemanggilan awal, hingga secara lengkap proses telah berjalan.

Berikut adalah gambaran fase rekursi untuk penyelesaian faktorial:



$F(4) = 4 \times F(3)$	Fase
$F(3) = 3 \times F(2)$	Awal
$F(2) = 2 \times F(1)$	
$F(1) = 1$ (kondisi basis)	

$F(2) = (2) \times (1)$	Fase
$F(3) = (3) \times (2)$	Balik
$F(4) = (4) \times (6)$	
24 rekursi lengkap	

D. Membuat Fungsi Rekursi dengan Phyton

Cara membuat fungsi rekursi sama dengan membuat fungsi biasa, hanya saja di dalamnya terdapat pemanggilan fungsi itu sendiri. Contoh sederhananya bisa dilihat dalam potongan program berikut:

```
def halo_dunia():
    print('Halo dunia!')
    # panggil dirinya sendiri
    halo_dunia() # <-- rekursifitas
```

memanggil fungsi hello_dunia untuk pertama kali

```
halo_dunia()
```

Dalam program tersebut terdapat fungsi halo_dunia yang di dalamnya terdapat pemanggilan fungsi halo_dunia itu sendiri. Ketika fungsi halo_dunia dilakukan pemanggilan untuk pertama kali, maka fungsi tersebut

akan memanggil ulang dirinya sendiri secara terus menerus sampai memenuhi kondisi untuk berhenti. Hasil yang didapatkan setelah program di atas dijalankan adalah sebagai berikut:

...

Halo dunia!

Halo dunia!

Halo dunia!

Fatal Python error: `_Py_CheckRecursiveCall`: Cannot recover

from stack overflow.

Python runtime state: initialized

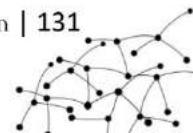
Program akan mencetak kalimat “Halo dunia!” secara terus menerus selama fungsi tersebut dipanggil. Namun perlu diperhatikan, dalam tampilan keluaran tersebut terdapat error sebagai berikut:

Fatal Python error: `_Py_CheckRecursiveCall`: Cannot recover

from stack overflow.

Python runtime state: initialized

Error tersebut disebabkan karena perulangan rekursif sudah dipanggil terlalu banyak dan tidak ada kondisi untuk menghentikan prosesnya. Oleh karena itu, sistem menghentikannya secara paksa. Perlu diketahui kode program di atas berpotensi bisa membuat PC kita menjadi tidak berjalan sebagaimana mestinya karena efek perulangan rekursif tak terbatas (meskipun sebenarnya



interpreter python telah membatasi jumlah kedalaman rekursif sebelum PC kehilangan kendali).

E. Studi Kasus 1 (Menampilkan Faktorial)

Faktorial dari bilangan n adalah perkalian bilangan positif dari angka 1 sampai bilangan itu sendiri. Bilangan faktorial sendiri biasa disimbolkan dengan tanda seru (!). Sebagai contoh nilai faktorial dari 5 yaitu

$$5! = 5 * 4 * 3 * 2 * 1$$

$$5! = 120$$

Seperti yang telah dibahas sebelumnya rumus untuk faktorial adalah sebagai berikut:

$$f(n) = \begin{cases} 1, & n = 0, n = 1 \\ nf(n - 1), & n > 1 \end{cases}$$

Maka untuk perhitungan 5! langkahnya adalah

$$5! = 5 * 4!$$

$$\rightarrow 4! = 4 * 3!$$

$$\rightarrow 3! = 3 * 2!$$

$$\rightarrow 2! = 2$$

Sebelum membuat program dengan rekursif, mari kita coba membuatnya dengan looping for terlebih dahulu. Logikanya sangat sederhana, kita buat satu variabel untuk menyimpan hasil faktorial, dan kita isi dengan nilai awal 1. Setelah itu kita akan kalikan terus menerus dengan angka berikutnya sampai mencapai angka n itu sendiri, seperti berikut:

```
n = int(input('Masukkan nilai n: '))
```

```
faktorial = 1
```

```
for i in range(1, n + 1):
    faktorial *= i
    print(f'{n}! = {faktorial}')
```

Atau, kita juga bisa memulai perulangannya dari angka 2 dari pada angka 1, karena angka 1 sudah kita set sebagai nilai awal variabel faktorial.

```
for i in range(2, n + 1):
    faktorial *= i
```

Program di atas akan menghasilkan keluaran sebagai berikut:

Masukkan nilai n: 5

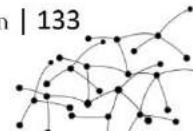
$5! = 120$

Dari logika looping di atas, kita bisa mengubah langkah-langkah tersebut dalam rekursif seperti ini:

```
n = int(input('Masukkan nilai n: '))
def hitung_faktorial (n):
    if n > 2:
        return n * hitung_faktorial(n - 1)
    return 2
faktorial = hitung_faktorial(n)
print(f'{n}! = {faktorial}')
```

maka output yang akan kita dapat adalah sebagai berikut:

Masukkan nilai n: 5



$5! = 120$

Masukkan nilai n: 10

$10! = 3628800$

Selain solusi di atas, ada pilihan solusi yang jauh lebih sederhana yaitu menggunakan fungsi yang sudah disediakan dalam python berupa fungsi factorial() yang berada pada modul math. Untuk menggunakan fungsi tersebut kita bisa melakukannya dengan cara berikut:

```
import math  
n = int(input('Masukkan nilai n: '))  
faktorial = math.factorial(n)  
print(f'{n}! = {faktorial}')
```

Adapun output yang dihasilkan adalah seperti berikut:

Masukkan nilai n: 5

$5! = 120$

Masukkan nilai n: 10

$10! = 3628800$

F. Studi Kasus 2 (Menghitung Fibonacci)

Fibonacci adalah suatu deret bilangan yang mana tiap angkanya adalah hasil penjumlahan dari dua angka sebelumnya dan dua anggota pertama dari deret fibonacci selalu 0 dan 1. Berikut adalah contoh deret fibonacci:

0 1 1 2 3 5 8 13 21 34

Dari pola deret di atas dapat kita ambil rumus sebagai berikut:

$$f(x) = \begin{cases} 0, & n = 0 \text{ Basis} \\ 1, & n = 1 \text{ Basis} \\ f(n - 1) + f(n - 2), & n > 1 \text{ Rekuren} \end{cases}$$

Sebelum membuat program dengan rekursi, mari kita simulasikan terlebih dahulu, setiap iterasi pemanggilan fungsi harus mengembalikan setidaknya satu list dengan satu anggota dengan program sebagai berikut:

```
def fibonacci(n):
```

```
    return [n]
```

kemudian, lakukan pemanggilan dan cetak secara berurutan seperti ini:

```
print(fibonacci(1))
```

```
print(fibonacci(2))
```

```
print(fibonacci(3))
```

```
print(fibonacci(4))
```

maka, output yang kita dapat adalah

```
[1]
```

```
[2]
```

```
[3]
```

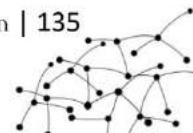
```
[4]
```

Sedangkan output yang kita inginkan adalah

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

Kita bisa menggabungkan dua buah list menjadi satu dengan operator `+`. Sehingga kode program kita akan terlihat seperti ini:

```
def fibonacci (n):
```



```
if n < 1:  
    return [n]  
    return fibonacci(n - 1) + [n]  
print(fibonacci(5))
```

Output yang didapat adalah:

```
[0, 1, 2, 3, 4, 5]
```

Secara singkat, fungsi rekursif di atas akan mengembalikan list sebelum n ditambah dengan list yang berisi n itu sendiri. Kalau dibongkar, ia melakukan penjumlahan list sebanyak n + 1:

```
[0] + [1] + [2] + [3] + [4] + [5]
```

```
# hasil [0, 1, 2, 3, 4, 5]
```

Kita menginputkan n dengan nilai 5, dan yang kita dapatkan adalah 6 item (karena perulangannya kita buat dari 0).

Kita berhasil membuat fungsi rekursif yang mengembalikan sebuah list. Ini adalah pondasi alur kita. Langkah berikutnya agar alur program menjadi lebih mudah untuk dipahami, mari kita buat beberapa variabel.

```
def fibonacci (n):  
    if n < 1:  
        return [n]  
    listSebelumN = fibonacci(n - 1)  
    angka1 = listSebelumN[-2] if len(listSebelumN) > 2  
    else 0
```

```
angka2 = listSebelumN[-1] if len(listSebelumN) > 2
else 1
print('listSebelumN', listSebelumN)
print(f'angka1: {angka1}, angka2: {angka2}')
return listSebelumN + [n]
```

Jika dijalankan, program di atas akan menghasilkan output:

```
listSebelumN [0]
angka1: 0, angka2: 1
listSebelumN [0, 1]
angka1: 0, angka2: 1
listSebelumN [0, 1, 2]
angka1: 1, angka2: 2
listSebelumN [0, 1, 2, 3]
angka1: 2, angka2: 3
listSebelumN [0, 1, 2, 3, 4]
angka1: 3, angka2: 4
[0, 1, 2, 3, 4, 5]
```

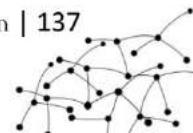
Langkah kita selanjutnya yaitu menentukan nilai n berdasarkan hasil penjumlahan 2 angka sebelumnya. Sehingga, kita hanya perlu mengubah return menjadi:

```
return listSebelumN + [angka1 + angka2]
```

maka keluarannya akan berubah menjadi seperti ini:

```
listSebelumN [0]
```

```
angka1: 0, angka2: 1
```



```
listSebelumN [0, 1]
angka1: 0, angka2: 1
listSebelumN [0, 1, 1]
angka1: 1, angka2: 1
listSebelumN [0, 1, 1, 2]
angka1: 1, angka2: 2
listSebelumN [0, 1, 1, 2, 3]
angka1: 2, angka2: 3
[0, 1, 1, 2, 3, 5]
```

G. Studi Kasus 3 (Menghitung Bilangan Berpangkat)

```
def pangkat(x,y):
if y == 0:
    return 1
else:
    return x * pangkat(x,y-1)
x = int(input("Masukan Nilai X : "))
y = int(input("Masukan Nilai Y : "))
print("%d      dipangkatkan      %d      =      %d"      %
(x,y,pangkat(x,y)))
```

Fungsi pangkat() di atas berfungsi untuk menghitung bilangan x yang di pangkatkan y. Dalam fungsi tersebut terdapat pemanggilan dirinya sendiri. Untuk setiap nilai x dan y yang di masukan pengguna akan dikirim ke fungsi pangkat() melalui parameter variabel x dan y.

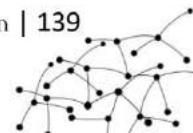
Selama nilai y bukan 0 maka fungsi pangkat() akan terus memanggil dirinya sendiri, dan nilai y akan terus dikurangi 1 ($y-1$) sampai kondisi bisa terpenuhi dan perulangan dihentikan.

Masukkan Nilai X : 5

Masukkan Nilai Y : 2

5 dipangkatkan 2= 25

Nilai x yang dimasukan adalah 5 dan y=2 ketika 5 di pangkatkan 2 maka hasilnya 25.



DAFTAR PUSTAKA

- Addition, D. and Floating, M. (no date) 'Arithmetic Operators in Python Exercises: • Type the following at the prompt and then execute the command , observe what you get', pp. 14–21.
- Beazley, David M. (2009). "Python Essential Reference." Addison-Wesley Professional.
- Downey, B. Allan (2015) *Think Python, 2nd Edition*. O-Rilley Media, Inc.
- Drake, Fred L. (2016). "Python 101." CreateSpace Independent Publishing Platform.
- Dr. Joseph Teguh Santoso, S.Kom, M. K. (2022) *Proyek Coding dengan Python*. Semarang: Yayasan Prima Agus Teknik.
- Herho, S. H. S. (2017) 'Tutorial Pemrograman Python 2 Untuk Pemula', *WCPL Press*, pp. 1–140.
- Jubilee Enterprise (2019) *Python untuk programmer pemula*.
- Javatpoint.com (2023) *javatpoint.com*. Available at: <https://www.javatpoint.com/python-data-types> (Accessed: 20 September 2023).
- Kuhlman, D. (2013) 'A Python Book', *A Python Book*, pp. 1–227.
- Kurniawan, D., 2022. *Pengenalan Machine Learning dengan Python*. Elex Media Komputindo.
- Langtangen, Hans Petter. (2016). "A Primer on Scientific Programming with Python." Springer.

Python Documentation. <https://docs.python.org/3/>

Lambert, K.A. (2017) *Fundamentals of Python: First Programs Second Edition.*

McKinney, Wes. (2017). "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython." O'Reilly Media.

Mattes, E., 2015, *Python crash course : a hands-on, project-based introduction to programming*, no strarch press, San Francisco.

Munir, no date; *Fungsi Rekursif*, no date; Brassard, 1996; Septian, 2013; Asadi, 2016; Rubio, 2018; Gowrishankar, 2019; Ma'arif, 2020; Sintiari, 2023

Martelli, Alex et al. (2005). "Python in a Nutshell." O'Reilly Media.

Malhotra, D.M. and N. (2020) *Data Structures and Program Design Using Python*.

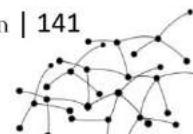
Mambang, Dona Marleny, F., & Zulfadhilah, M. (2022). *Algoritma Pemrograman Menggunakan Python I* (Issue September). <https://www.researchgate.net/publication/363769056>

Operators, A. *et al.* (no date) 'Python Arithmetic Operators : Python Comparison Operators : Python Assig nment Operators':

Paul Barry (2016) *Head First Python: A Brain-Friendly Guide*. O'Reilly Media.

Pilgrim, Mark. (2004). "Dive Into Python." Apress.

Peter Wentworth, J. E. and Meyers, A. B. D. and C. (2012) *How to Think Like a Computer Scientist: Learning with*



Python 3 The way of the program.

Ramalho, L., 2022, *Fluent Python : clear, concise, and effective programming*, 2nd edn., O'Reilly Media, Inc, United State of America.

RevoU (2023) *9 Library Python Terbaik untuk Data Analytics*. Available at: <https://revou.co/panduan-teknis/library-python> (Accessed: 25 September 2023).

Ryane Puspa (2021) *Kelebihan Bahasa Pemrograman Python*. Available at: <https://academy.alterra.id/blog/kelebihan-bahasa-pemrograman-python/> (Accessed: 25 September 2023).

Swastika, W., 2019. *Pengantar Algoritma dan Penerapannya pada Python*. Ma Chung Press.

Summerfield, Mark. (2009). "Programming in Python 3: A Complete Introduction to the Python Language." Addison-Wesley Professional.

Stevens, T. J. and Boucher, W. (2015) *Python basics, Python Programming for Biology*. doi: 10.1017/cbo9780511843556.004.

Sugiana, O. (2002) 'Membuat Aplikasi Bisnis Menggunakan bahasa Python dan database berbasis SQL.'

Stevens, T.J. and Boucher, W. (2015) *Python basics, Python Programming for Biology*. Available at: <https://doi.org/10.1017/cbo9780511843556.004>.

van Rossum, Guido, and Drake, Fred L. (2009). "Python 3 Reference Manual." CreateSpace.

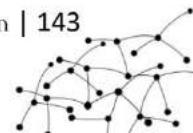
Varoquaux, Gaël et al. (2015). "Python Scientific Lecture Notes." <https://scipy-lectures.org/>.

VanderPlas, Jake. (2016). "Python Data Science Handbook."
O'Reilly Media.

<https://belajarpython.com/tutorial/operator-python/>

<https://www.python.org/>

(MKom et al., 2022)





TENTANG PENULIS



Sri Tria Siska, S.Kom., M.Kom., lahir di Batusangkar, 19 April 1992. Penulis menyelesaikan Pendidikan Sarjana S-1 Sistem Informasi di Universitas Putra Indonesia “YPTK” Padang dan pendidikan Megister S-2 Sistem Informasi di Universitas Putra Indonesia “YPTK” . Penulis adalah Dosen Program Studi D-3 Teknik Komputer di STT Payakumbuh. Penulis juga mengabdi sebagai Tutor Online di Universitas Terbuka pada matakuliah Pengantar Sistem Informasi dan Algoritma dan Pemograman. Penulis juga mengampu beberapa matakuliah lainnya di STT Payakumbuh yaitu Sistem Operasi, Sistem Basis Data, Teknologi Informasi dan Komunikasi dan Algoritma dan Pemograman Dasar, Interaksi Manusia dan Internet Of Things, Arsitektur dan Organisasi Komputer. Penulis juga aktif melakukan penelitian di Bidang Data Mining dan Information Systems. Penulis dapat dihubungi melalui email sritriasiska@gmail.com.



Hariyadi, S.Kom., M.Kom, Lahir di Muaro Labuh, 21 Juni 1989 Menyelesaikan Studi S1 Sistem Komputer Universitas Putra Indonesia “YPTK” Tahun 2012, Lulus S2 di Program Studi Magister Ilmu Komputer di Perguruan Tinggi yang sama Tahun 2015. Saat ini adalah Dosen tetap di Program Studi Teknik Elektro Universitas Muhammadiyah Sumatera Barat (UM Sumbar) Sejak 2016 dan menjabat Sebagai

Wakil Dekan Fakultas Teknik UM Sumatera Barat sejak Tahun 2020 yang sebelumnya menjadi Kepala UPT. Puskom UM Sumbar. Dosen pengampu Mata Kuliah Sistem Digital, Algoritma Pemograman, Jaringan Komputer. Sebelumnya pernah Menerbitkan buku dengan Judul Dokumentasi Keperawatan Pada Poliklinik Gigi (Kajian Manual dan Komputerisasi), Buku ICT dan Perkembangan Pendidikan Islam, Kewirausahaan Digital dan editor di buku Media - Media Pembelajaran. Saat ini Penulis Sedang menempuh Pendidikan S3 di Universitas Negeri Padang.



Alfry Aristo Jansen Sinlae, S.Kom., M.Cs.
Penulis mendapatkan gelar Sarjana Komputer (S.Kom) pada tahun 2010 dari Program Studi Teknik Informatika, Fakultas Teknologi Informasi (FTI), Universitas Kristen Satya Wacana (UKSW) Salatiga. Kemudian melanjutkan pendidikan S2 pada Fakultas Teknologi Informasi, Program Studi Magister Sistem Informasi, Universitas Kristen Satya Wacana dan telah menyelesaikan jenjang pendidikan Master dengan gelar Master of Computer Science (M.Cs) pada tahun 2012. Adapun bidang ilmu yang ditekuni penulis adalah Sistem Informasi, Sistem Pakar, Sistem Pendukung Keputusan, dan Database. Saat ini penulis aktif bekerja sebagai staff pengajar pada Program Studi Ilmu Komputer, Fakultas Teknik, Universitas Katolik Widya Mandira Kupang. Penulis juga aktif terlibat dalam menghasilkan karya ilmiah yang diterbitkan pada Jurnal Internasional Terindeks Scopus, Jurnal Nasional Terakreditasi, dan Jurnal Nasional. Selain itu, aktif pula dalam kolaborasi menghasilkan tulisan untuk diterbitkan dalam buku ber-ISBN.



Buku ini adalah salah satu karya dan kedepannya secara konsisten akan disusul dengan buku-buku berikutnya. Pembahasan dalam buku yang ditulis ini semata-mata untuk berbagi ilmu pengetahuan.



Nova Tri Romadloni, M.Kom., Lahir pada tahun 1996 di Kabupaten Karanganyar, Jawa Tengah. Bekerja sebagai seorang Dosen tetap pada Prodi Informatika, Fakultas Sains dan Teknologi, Universitas Muhammadiyah Karangay. Menyelesaikan studi D3 dengan program studi Manajemen Informatika di Universitas Bina Sarana Informatika. Dimana pada saat menjadi mahasiswa mulai semester 4 menjadi Asistan Lab selama 2 Tahun. Kemudian melanjutkan S1 dengan program studi Sistem Informasi dan Melanjutkan S2 dengan program studi Ilmu Komputer pada Universitas Nusa Mandiri Jakarta, yang Lulus pada tahun 2019 dengan konsentrasi Data Mining. Mata kuliah yang pernah diampu diantara lain, Logika Informatika, Praktikum Algoritma dan Struktur Data, Sistem Basis Data, Praktikum Sistem Basis Data, Pengenalan Web dan Praktikum Pengenalan Web.



Dini Nurmalasari lahir di Garut Jawa Barat telah menempuh Pendidikan S1 di Teknik Informatika UIN Suska Riau, kemudian melanjutkan kuliah S2 Informatika di ITB Bandung. Saat ini mengajar pada bidang Data Engineering dan Data Science di Jurusan Teknologi Informasi Politeknik Caltex Riau sejak tahun 2004. Beberapa sertifikasi yang dimiliki saat

ini diantaranya Certified Big Data Foundation dari Cloud Credential Council, Certified Data Science Practitioner, sertifikasi BNSP untuk skema System Analyst dan Asesor Kompetensi, Oracle Certified Associate SQL, serta Python Programming Certified. Fokus penelitian yang dilakukan saat ini terkait implementasi data warehouse dan business intelligence, text mining dengan focus pada ekstraksi fitur dan visualisasi data, serta beberapa data science dan pengembangan aplikasi.



Heni Rachmawati lahir di Pekanbaru Riau, menempuh Pendidikan S1 Teknik Informatika Telkom University, kemudian melanjutkan S2 di Teknik Elektro Institut Sepuluh Nopember. Saat ini mengajar pada bidang programming dan rekayasa perangkat lunak di Jurusan Teknologi Informasi Politeknik Caltex Riau sejak tahun 2007. Beberapa sertifikasi yang di miliki diantaranya Python Programming, Agile dan COBIT 2019. Fokus penelitian yang dilakukan saat ini pada information system for health, education, dan smart government.



Wenda Novayani lahir di Taluk Kuantan Riau, menempuh Pendidikan D3 di Teknik Komputer Politeknik Caltex Riau, kemudian melanjutkan D4 di Teknik Elektro ITB Bandung, dan melanjutkan S2 di Teknologi Informasi UGM Yogyajarta. Saat ini mengajar pada bidang programming dan multimedia di Jurusan Teknologi Informasi Politeknik Caltex



Riau sejak tahun 2007. Beberapa sertifikasi yang di miliki diantaranya Python Programming dan Animasi. Fokus penelitian yang di lakukan saat ini pada Game Based Learning(GBL), Virtual Reality, Animasi untuk mendukung Smart Learning.



Penulis bernama lengkap **Trizaurah Armiani, S.Kom., M.Sc** yang lahir di Palembang, 22 Januari 1994, ia adalah anak kedua dari lima bersaudara. Ia adalah alumnus dari *National Taipei University and Technology* Jurusan *Computer Science* dan sekarang mengabdi sebagai Dosen Jurusan Manajemen Informatika pada Politeknik Negeri Sriwijaya.

Selain mengajar, ia juga melakukan penelitian pada bidang *data mining, artificial intelligence* dan *data mining*.



I Wayan Rangga Pinastawa merupakan lulusan Sarjana dan juga Magister pada bidang Komputer di Universitas AMIKOM Yogyakarta. Sempat memulai karir di Bank Danamon Serta menjadi SEO di Jawapos.com sekarang penulis berprofesi sebagai dosen Informatika di Fakultas Ilmu Komputer UPN Veteran jakarta. Penulis memiliki hoby membuat dan menulis blog. Serta memiliki fokus utama penelitian pada bidang Machine Learning, Data Mining, Decision Support System dan Search Engine Optimization.



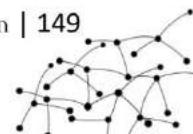
Teuku Radillah (Tera) merupakan seorang praktisi dibidang komputer. Kegemaran penulis adalah mempelajari dan menulis program – program Komputer Seperti, Visual Studio.Net, PHP, Java, Kotlin, Android dan bahasa nirprosedural *Structured Query Languange* (SQL), telah menjadikan penulis seorang programmer komputer, dimana karya – karya penulis telah digunakan oleh instansi pemerintahan maupun swasta.



Fithri Selva Jumeilah, S.Kom., M.T.I lahir di Baturaja, 4 Mei 1990. Ia adalah seorang dosen di Politeknik Negeri Sriwijaya lebih tepatnya di Program studi Teknologi Informatika Multimedia Digital Jurusan Teknik Komputer. Sejak 2012 ia sudah menjadi dosen di berbagai perguruan tinggi. Mata kuliah yang biasa diampu adalah matakuliah Motion Grafik, 3D Modeling, Multimedia Interaktif dan Pemrograman. Gelar Sarjana Komputer diperoleh dari Univeristas Sriwijaya Jurusan Teknik Informatika dan untuk gelar Magister Teknologi Informasi didapat dari Universitas Indonesia. Saat ini ia telah banyak menghasilkan penelitian yang terindex sinta.



Mohammad Robihul Mufid atau biasa dipanggil dengan Mufid. Lahir di Gresik, 22 Agustus 1994. Dengan berlatar belakang Pendidikan mulai dari D4 Teknik Informatika dan S2 Teknik Informatika dan Komputer di Politeknik



Elektronika Negeri Surabaya.

Merupakan salah satu penulis dari buku ini yang berfokus untuk memberikan pemahaman tentang bagaimana mengimplementasikan sebuah program dengan menggunakan bahsa Python. Khususnya bagaimana mengimplementasikan sebuah array didalam Bahasa Python.



Saniyatul Mawaddah, lahir di Lamongan, 15 Maret 1993. Menyelesaikan Pendidikan dasar di SD Dinoyo II pada tahun 2005. Kemudian menyelesaikan Pendidikan SMP dan SMA di SMP Negeri 1 Lamongan pada tahun 2008 dan SMA Negeri 2 Lamongan pada tahun 2011. Pada tahun tersebut Saniya, panggilan akrabnya, menempuh Pendidikan diploma IV di Politeknik Elektronika Negeri Surabaya dan lulus tahun 2015. Setelah lulus, Saniya sempat bekerja sebagai pengajar Akademi Komunitas Negeri Lamongan sambil menempuh Pendidikan S2 nya di Institut Teknologi Sepuluh Nopember Surabaya pada tahun 2017-2019. Setelah lulus S2, Saniya bekerja sebagai ASN Dosen di Politeknik Elektronika Negeri Surabaya PSDKU Lamongan hingga sekarang.

Algoritma PEMROGRAMAN



Python adalah salah satu bahasa yang paling populer di dunia, selain mudah dipelajari, Python juga dapat digunakan untuk berbagai macam proyek, mulai dari pengembangan aplikasi web sampai analisis data.

Buku ini dirancang untuk memudahkan para pembaca memahami Algoritma Pemrograman Phyton secara lebih mudah dan terstruktur yang terdiri dari 13 Bab. Bab pertama membahas tentang Konsep Dasar Algoritma, Bab kedua Notasi Algoritma, Bab ketiga Pemrograman Phyton, Bab keempat Operasi

Dasar Phyton, Bab kelima Operator-operator Phyton, Bab keenam List, Dictionary, Tuple, Bab ketujuh Tipe Data, Variable, Konstanta, Dan Nilai, Bab kedelapan Percabangan, Bab kesembilan Library,

Bab kesepuluh Algoritma Perulangan, Bab kesebelas Fungsi, Bab kedua belas Array, dan Bab ketiga belas Algoritma Rekursi. Buku ini dilengkapi dengan latihan soal dan studi kasus yang membantu pemahaman pembaca.

Dengan perkembangan teknologi yang begitu cepat, pemrograman menjadi keahlian yang sangat dibutuhkan di berbagai bidang pekerjaan.

Bahasa pemrograman Python menjadi salah satu bahasa yang paling populer digunakan di dunia pemrograman. Buku ini dapat dijadikan sebagai sumber bacaan bagi mahasiswa dan siapa pun yang ingin mempelajari pemrograman dengan Python. Melalui buku ini mahasiswa yang akan melakukan penelitian dan menggunakan Python sebagai Bahasa pemrograman bisa memperoleh referensi yang mudah dipahami untuk menuliskan kode programnya.

ISBN 978-623-09-6020-8

A standard linear barcode representing the ISBN number.

9 786230 960208



PT Penerbit Penamuda Media
Godean, Yogyakarta
085700592256
@penamuda_media
penamuda.com