

ALGORITMA & PEMROGRAMAN

PYTHON SERIES

Penulis

IRWAN A. KAUTSAR, Ph.D



Program Studi Informatika
Fakultas Sains & Teknologi
Universitas Muhammadiyah Sidoarjo

Buku Ajar

Algoritma & Pemrograman

Python series

Crafted with ❤ by:

Irwan A. Kautsar, S.Kom., M.Kom., Ph.D

Program Studi Teknik Informatika

Fakultas Teknik

Universitas Muhammadiyah Sidoarjo

September 2017

ISBN: 978-979-3401-71-3

Buku Ajar

Algoritma & Pemrograman

Python series

Penulis : Irwan Alnarus Kautsar, S.Kom., M.Kom., Ph.D

Editor : LP3IK Universitas Muhammadiyah Sidoarjo

Penata Letak : Irwan Alnarus Kautsar, S.Kom., M.Kom., Ph.D

Desainer Cover : Stay@HomeMom

Redaksi:

Lembaga Pengkajian dan Pengembangan Pendidikan dan Al-Islam Kemuhammadiyahan (LP3IK)

Universitas Muhammadiyah Sidoarjo

Cetakan Pertama : Oktober 2017

Penerbit:

UMSIDA Press

© 2017 Universitas Muhammadiyah Sidoarjo

Kata Pengantar

‘Assalaamu’alaikum warahmatullaahi wabarakaaatuh.’

Puji syukur dipanjatkan kepada Allah Azza wa Jalla atas segala rahmat-NYA sehingga penulisan buku ajar Algoritma dan Pemrograman dapat terealisasikan. Penulis juga mengucapkan banyak terima kasih atas bantuan dari pihak yang telah berkontribusi memberikan sumbangsih baik moril maupun pemikiran.

Buku ajar Algoritma dan Pemrograman disusun dalam rangka menunjang proses kegiatan belajar mengajar yang memudahkan rekan pembaca sekalian mempelajari algoritma dan pemrograman, khususnya mahasiswa Program Studi Informatika, Fakultas Teknik Universitas Muhammadiyah Sidoarjo. Diharapkan adanya buku ajar ini, mahasiswa Program Studi Informatika mampu mengimplementasikan algoritma dengan membuat program sederhana menggunakan bahasa pemrograman Python. Penulis memilih bahasa pemrograman Python dalam modul buku ajar karena bahasa Python mudah dipelajari, cross-platform, multi device dan terpenting, kemudahannya dalam pengimplementasian algoritma.

Semoga buku ajar Algoritma dan Pemrograman dapat membantu menyelesaikan suatu permasalahan dengan melakukan pemecahan masalah berdasarkan pengimplementasian ilmu algoritma yang telah dipelajari. Penulis menyadari bahwa dalam penyusunan buku ajar ini terdapat kekurangan dan jauh dari kata sempurna. Oleh sebab itu, penulis berharap kritik, saran serta usulan demi perbaikan dan kesempurnaan buku ajar berikutnya. Mengingat tidak ada sesuatu yang sempurna tanpa saran yang membangun.

Sidoarjo, 01 Oktober 2017

Irwan A. Kautsar, S.Kom., M.Kom., Ph.D

surel: irwan@umsida.ac.id

Acknowledgement

Ucapan terima kasih kepada Rekan-rekan Lembaga pengkajian dan pengembangan pendidikan dan Al-Islam kemuhammadiyahan (LP3IK) Universitas Muhammadiyah Sidoarjo dalam memberikan kesempatan untuk menulis Buku Ajar mata kuliah Algoritma dan Pemrograman Python Series.

Daftar Isi

Kata Pengantar	4
Acknowledgement	5
Daftar Isi	6
Daftar Gambar	8
Daftar Tabel	11
Bab 1. Baca ini dulu	14
1.1. Instruksi dan Interaksi	14
1.2. Memasang Python	15
1.3. Penggunaan Python	18
Rangkuman	19
Soal/Evaluasi	19
Bab 2. Algoritma	21
2.1. Pentingnya Algoritma	21
1.2. Algoritma, Pseudocode dan Pemrograman	26
Rangkuman	30
Soal/Evaluasi	32
Bab 3. Pemrograman	34
3.1. Kenapa Python?	34
3.2. Variabel	37
3.3. Tipe Variabel	40
3.4. Conditional Statement	45
3.5. Looping	47
3.6. Procedure/Method	51
3.7. Recursive	52
3.8. I/O Process	53
3.9. Error Handling	56
Rangkuman	57
Soal/Evaluasi	58

Bab 4. Algoritma dan Pemrograman Pencarian (Searching)	60
4.1. Pencarian (Searching)	61
4.2. Sequential Search	66
4.3. Binary Search	70
Rangkuman	72
Soal/Evaluasi	72
Bab 5. Algoritma dan Pemrograman Pengurutan (Sorting)	74
5.1. Bubble Sort	74
5.2. Insertion Sort	77
5.3. Selection Sort	80
5.4. Merge Sort	88
5.5. Quick Sort	90
Bab 6. Optimasi Algoritma dan Pemrograman	93
6.1. Optimasi Algoritma dan Notasi Big O	93
6.2. Optimasi Pemrograman	95
6.3. Publikasi Ilmiah : Optimasi Algoritma Pengurutan dan Pencarian	100
Biodata Penulis	101
Bibliography	102

Daftar Gambar

Gambar 1.1. Tampilan Python intrepreter di sistem operasi Ms. Windows.	18
Gambar 1.2. Tampilan Python intrepreter di sistem operasi MacOS	18
Gambar 2.1. Mencari rute jarak terdekat menggunakan Google Maps	22
Gambar 2.2. Alur proses dari algoritma menjadi sebuah aplikasi	22
Gambar 2.3. Alur SDLC dari analisa hingga menjadi release	23
Gambar 2.4. Contoh gambar pixel pada pengenalan citra/image	24
Gambar 3.1. Syntax Java untuk menulis “Teh hangat satu.”	34
Gambar 3.2. Syntax C++ untuk menulis “Teh hangat satu.”	35
Gambar 3.3. Syntax Python untuk menulis “Teh hangat satu.”	35
Gambar 3.4. Proses swap variable pada bahasa pemrograman secara umum.	36
Gambar 3.5. Proses swap variable pada bahasa pemrograman Python.	36
Gambar 3.6. Contoh swap variable pada Python.	37
Gambar 3.7. Pendefinisian variabel di Python.	38
Gambar 3.8. contoh variable di Python.	39
Gambar 3.9. Output bilangan acak.	39
Gambar 3.10. Tipe Integer	40
Gambar 3.11. Tipe String	41
Gambar 3.12. Tipe Boolean	42
Gambar 3.13. Operator OR dan AND	42
Gambar 3.14. Penggunaan type ()	43
Gambar 3.15. Tipe List	44
Gambar 3.16. Penggunaan IF, ELIF dan ELSE.	45
Gambar 3.17. Kode penggunaan IF.	46
Gambar 3.18. Output Conditional Statement	46
Gambar 3.19. Flowchart perulangan For dan While.	47
Gambar 3.20. Perulangan For	48
Gambar 3.21. Output Perulangan For	48

Gambar 3.22. Output Perulangan While.	49
Gambar 3.23. Kode perulangan While.	50
Gambar 3.25. Penjumlahan menggunakan For.	53
Gambar 3.26. Penjumlahan menggunakan rekursif	53
Gambar 4.4. Tangkapan layar kode random100.py	64
Gambar 4.6. Hubungan waktu proses dengan jumlah data yang dicari.	65
Gambar 4.7. Sequential Search	67
Gambar 4.8. Kode dan Ouput Sequential Search.	68
Gambar 4.9. Sequential Search dengan sorted data.	69
Gambar 4.10. Sequential Search dengan item lebih besar dari midpoint.	70
Gambar 4.11. Sequential Search dengan angka kurang dari midpoint.	71
Gambar 4.12. Kode program Sequential Search.	71
Gambar 5.1. Satu fase perbandingan dalam Buble Sort.	75
Gambar 5.2. Kode program dan output script Buble Sort.	77
Gambar 5.3. Loop pada Insertion Sort.	78
Gambar 5.4. Kode program dan output pada Insertion Sort.	79
Gambar 5.5. Fase pertama dari Selection Sort.	81
Gambar 5.6. Array sementara dalam fase ke 1 algoritma Selection Sort.	83
Gambar 5.7. Fase ke 2 algoritma Selection Sort.	83
Gambar 5.8. Array sementara dalam fase ke 2 algoritma Selection Sort.	84
Gambar 5.9. Fase ke 3 algoritma Selection Sort.	84
Gambar 5.10. Array sementara dalam fase ke 3 algoritma Selection Sort.	85
Gambar 5.11. Array sementara dalam fase ke 4 algoritma Selection Sort.	85
Gambar 5.12. Array final dalam algoritma Selection Sort.	86
Gambar 5.13.Kode program algoritma Selection Sort.	87
Gambar 5.14. Ilustrasi algoritma Merge Sort.	88
Gambar 5.15. Kode program algoritma Merge Sort.	89
Gambar 5.16. Ilustrasi algoritma Quick Sort	90

Gambar 5.17. Kode program dan ouput dari implementasi algoritma Quick Sort	
91	
Gambar 6.2. Deret Fibonacci.	95
Gambar 6.3. Kode Program Deret Fibonacci	96
Gambar 6.4. Kode Program Deret Fibonacci dengan Dict.	97
Gambar 6.5. Output Kode Program Deret Fibonacci dengan Dict.	98
Gambar 6.6. Perbandingan kode tanpa dan dengan Dictionary.	99

Daftar Tabel

Tabel 3.1. Contoh manipulasi tipe data List.	45
Table 4.1. Waktu proses pencarian data dari 5 hingga 1 juta angka.	65
Tabel 6.1. Deret bilangan Fibonacci.	95
Tabel 6.2. Deret Fibonacci Angka Puluhan beserta waktu proses	97
Tabel. 6.3 Deret Fibonacci dengan Dict. beserta waktu proses	98

```
def dedicated(book):
    print book

dedicated("for my both sunshine . . .")
```

Bab 1

Baca ini dulu

Tujuan Instruksional :

Pada bab ini, dijelaskan bagaimana cara berinteraksi dengan modul bahan ajar ini.

Capaian Pembelajaran Mata Kuliah :

Mahasiswa diharapkan mampu mempersiapkan perangkat lunak yang dibutuhkan dengan tujuan untuk mengerjakan latihan yang ada di dalam modul bahan ajar ini.

Bab 1. Baca ini dulu

Buku ini didesain khusus sebagai bahan pendamping mata kuliah Algoritma dan Pemrograman. Namun, dapat digunakan oleh mahasiswa di luar bidang Informatika sebagai “makanan” pembuka dalam mengenal algoritma dan pemrograman.

1.1. Instruksi dan Interaksi

Sebelum memulai berinteraksi dengan buku ini, disarankan pembaca memperhatikan beberapa petunjuk ini :



Jika terdapat kode gambar globe, disarankan pembaca mencari referensi lebih lanjut melalui layanan search engine (Google, YouTube dan lain sebagainya).



Jika terdapat kode gambar pena, disarankan pembaca menulis pada sebuah kertas untuk mengerjakan/menggambarkan soal-soal yang diberikan.



Jika terdapat kode gambar notebook, disarankan pembaca menulis kode ke dalam sebuah text editor atau mencoba sendiri langkah-langkah dalam menulis kode sesuai dengan petunjuk yang diberikan.



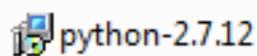
Jika terdapat kode gambar loudspeaker, menandakan untuk memperhatikan informasi yang berada disamping gambar tersebut.

1.2. Memasang Python

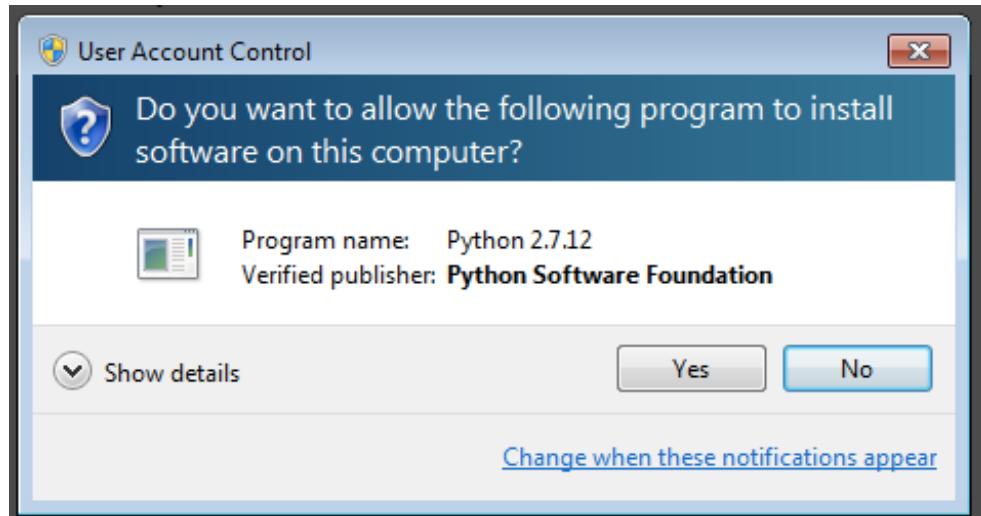
Buku ini menggunakan bahasa pemrograman Python yang dapat dijalankan di berbagai sistem operasi. Seperti Microsoft Windows, GNU/Linux OS, MacOS dan bahkan di perangkat mobile. Untuk pengguna MacOS dan GNU/Linux OS, Python sudah terinstall. Namun untuk pengguna Microsoft Windows, perlu di install dahulu. Berikut petunjuk menginstall Python di Microsoft Windows.

1. Unduh file Python installer di: <https://www.python.org/download/>.

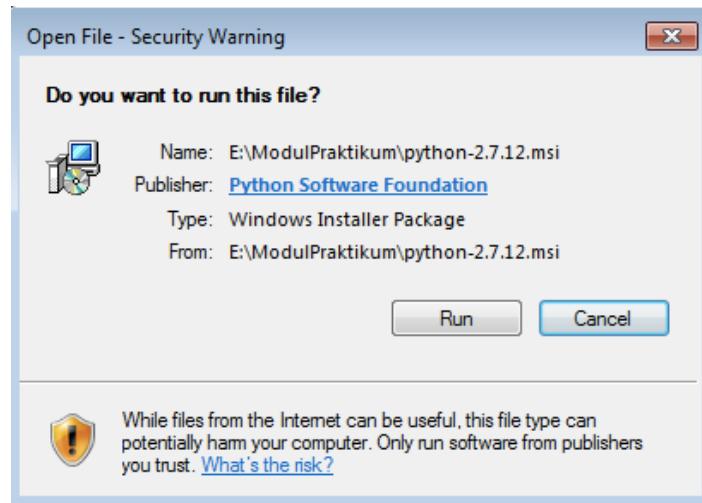
Pilih installer versi 2.7.x (saat modul ini dibuat, versi terakhir ialah 2.7.12).



2. Buka file installer tersebut. Pilih Yes.



3. Kemudian klik Run.

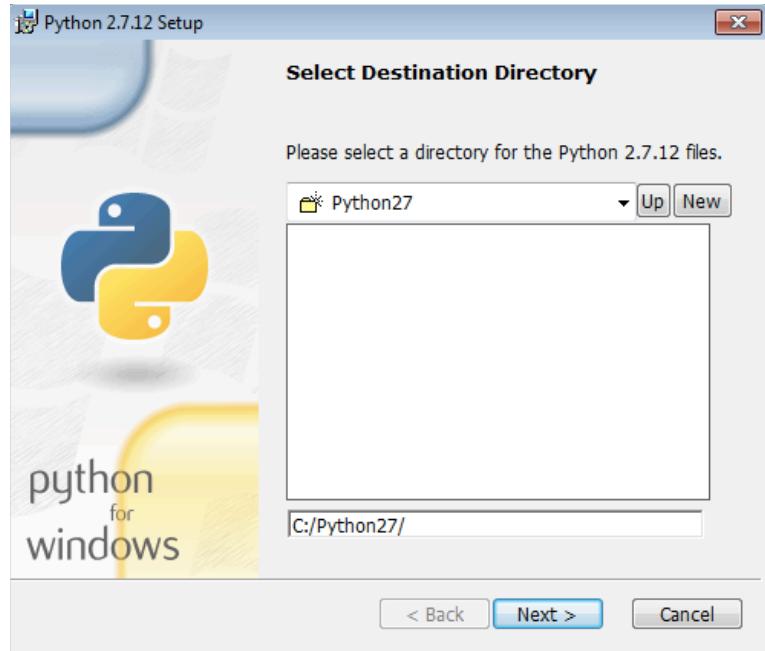


4. Pilih Next.

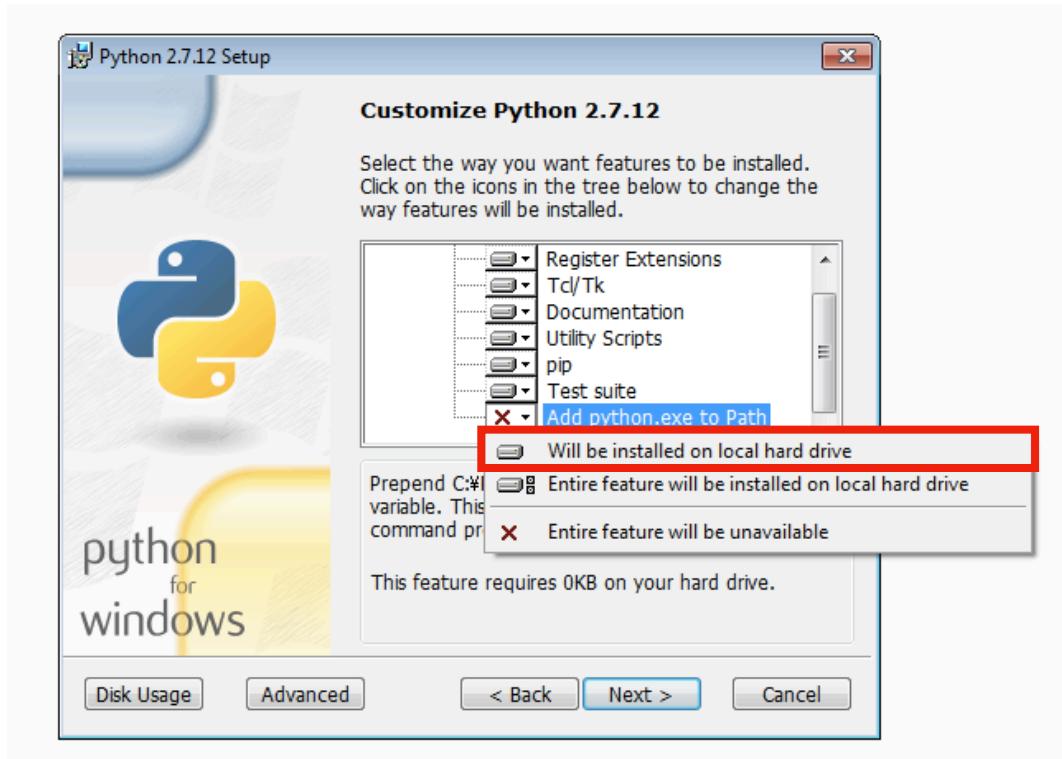


5. Selanjutnya tetap pilih Next. Kecuali anda ingin merubah tempat instalasi.

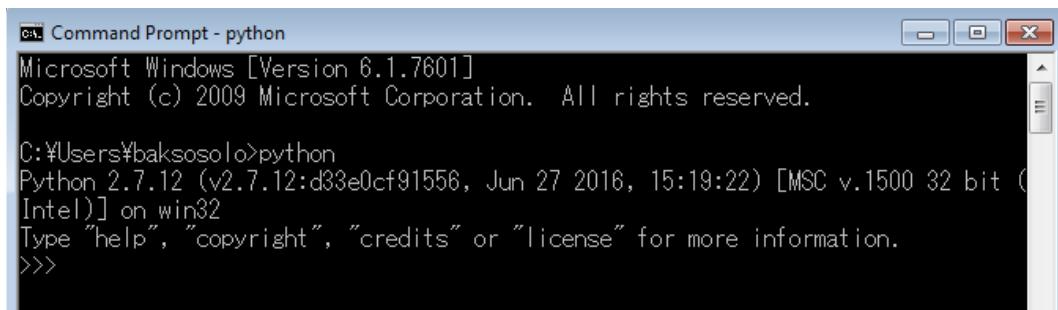
Tetapi saya sarankan klik Next.



6. Selanjutnya, ini yang **penting**! Agar Python dapat diakses lewat command prompt, maka opsi ini perlu dipilih.



7. Klik Finish untuk menutup instalasi. Dan disarankan untuk membooting komputer. Kemudian silahkan buka command prompt. Ketik Python. Jika muncul Python intrepreter seperti di gambar ini, maka Python sukses di install.



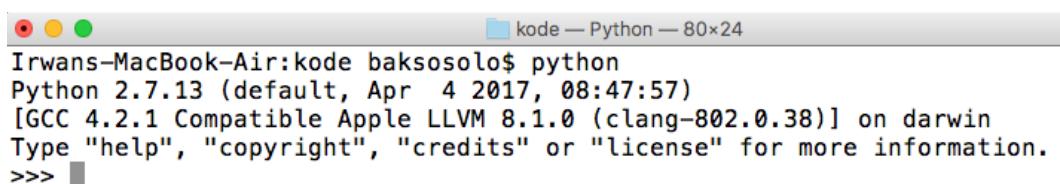
The screenshot shows a Microsoft Windows Command Prompt window titled "Command Prompt - python". The window displays the following text:
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\baksosolo>python
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

Gambar 1.1. Tampilan Python intrepreter di sistem operasi Ms. Windows.

1.3. Penggunaan Python

Untuk menggunakan bahasa Python terdapat 2 cara. Cara pertama ialah menggunakan Python intrepreter. Yaitu dengan cara membuka terminal atau yang disebut command prompt di Microsoft Windows, kemudian ketik Python.

Untuk tampilan di terminal MacOs ditunjukkan dalam gambar berikut:



The screenshot shows a Mac OS terminal window titled "kode — Python — 80x24". The window displays the following text:
Irwans-MacBook-Air:kode baksosolo\$ python
Python 2.7.13 (default, Apr 4 2017, 08:47:57)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.38)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>

Gambar 1.2. Tampilan Python intrepreter di sistem operasi Mac OS

Cara kedua menggunakan bahasa Python ialah menggunakan text editor, kemudian kode yang ditulis disimpan dengan akhiran file (ekstensi) .py. Selanjutnya file tersebut dieksekusi melalui terminal dengan perintah :

python nama_filenya.py

Untuk text editor, dapat menggunakan aplikasi gratis seperti:

1. Atom text editor. <https://atom.io>
2. Micro. <https://micro-editor.github.io>
3. Sublime Text: <http://www.sublimetext.com> (lisensi berbayar namun dapat dipakai secara gratis).
4. Light Table: <http://lighttable.com>

Rangkuman

Bab ini mempelajari bagaimana berinteraksi dengan bahan ajar dalam buku dan tutorial menginstall Python di laptop masing-masing.

Soal/Evaluasi

Python dapat pula digunakan dengan memanfaatkan intrepreter atau editor online, seperti pada alamat berikut: <https://www.python.org/shell/>.

Coba akses dan ketikkan : import this. Kemudian jawab pertanyaan ini. Apakah yang dimunculkan oleh intrepreter dengan memberikan perintah : import this ?

Bab 2

Algoritma

Tujuan Instruksional :

Pada bab ini, dijelaskan peran algoritma dalam ilmu pengetahuan, terutama dalam bidang informatika dan peran algoritma dalam mendukung perkembangan teknologi.

Capaian Pembelajaran Mata Kuliah :

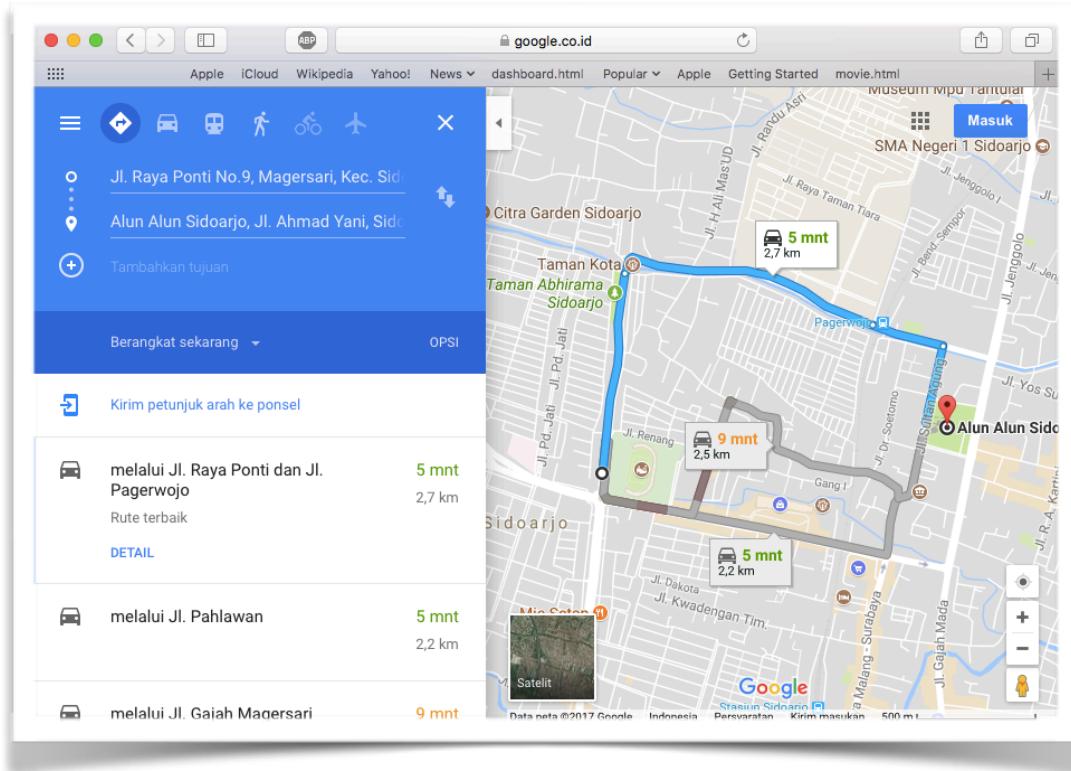
Mahasiswa diharapkan mampu memahami peran algoritma dan mampu menggunakan algoritma dalam menyelesaikan masalah yang ada pada lingkungan akademik.

Bab 2. Algoritma

2.1. Pentingnya Algoritma

Tahun 830 masehi seorang ilmuwan muslim bernama Al Khawarizmi menulis sebuah buku yang berjudul: “Al-Kitab al-mukhtasar fi hisab al-jabr wa'l-muqabala” (“The Compendious Book on Calculation by Completion and Balancing”. Dimana di dalam buku tersebut dijelaskan langkah-langkah menyelesaikan persoalan matematika dengan menjabarkan bahasa matematika secara abstrak (yang kemudian dikenal dengan aljabar). Seribu tahun kemudian(1821 Masehi), Charles Babbage mengadopsi cara Al Khawarizmi, penyelesaian masalah perhitungan sederhana dengan membangun sebuah mesin yang dikenal Babbage Engine, dibuat untuk menyelesaikan perhitungan fungsi polynomial. Seratus tahun kemudian yaitu pada tahun 1930, seorang ahli matematika bernama Alan Turing, mempublikasikan sebuah paper berjudul “On computable numbers, with an application to the Entscheidungsproblem” yang menjelaskan Turing Machine, dimana kemudian menjadi konsep dan teori dasar komputasi modern.

Dengan perkembangan teknologi saat ini, hampir dipastikan kita tiap hari menikmati hasil dari suatu perhitungan yang dikembangkan dari sebuah algoritma. Mulai dari mencari sebuah nama yang berada pada daftar kontak hingga melakukan belanja secara online melalui aplikasi yang ada di perangkat mobile. Termasuk didalamnya mencari rute dengan jarak terdekat dari Gelora Delta Sidoarjo (GOR) menuju Alun-alun seperti gambar 1.1.



Gambar 2.1. Mencari rute jarak terdekat menggunakan Google Maps

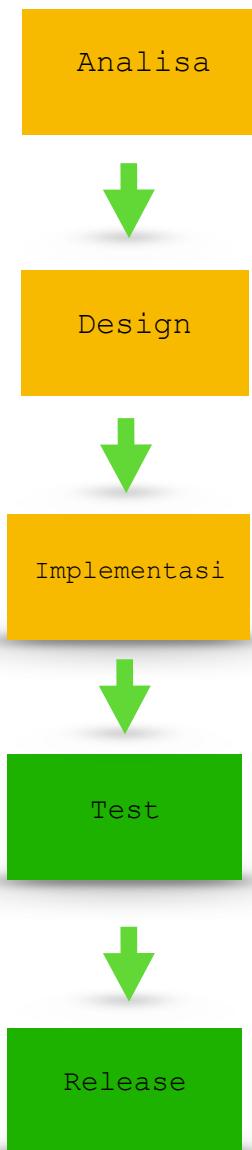
Dari sebuah algoritma menjadi suatu aplikasi yang digunakan dalam kehidupan sehari-hari, membutuhkan proses yang panjang. Sebuah algoritma perlu diterjemahkan menjadi sebuah Pseudocode. Dan dari Pseudocode perlu



diimplementasikan kepada bahasa pemrograman.

Gambar 2.2. Alur proses dari algoritma menjadi sebuah aplikasi

Sedangkan dalam literatur teori pengembangan perangkat lunak, siklus hidup perangkat lunak atau yang dikenal sebagai Software Development Life Cycle (SDCL) ialah sebagai berikut:

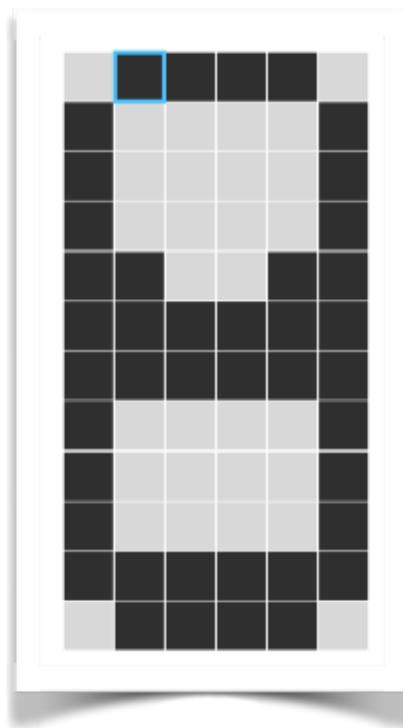


Gambar 2.3. Alur SDLC dari analisa hingga menjadi release

Dari gambar 1.2. dan 1.3 diatas, nampak kesamaan proses pengembangan aplikasi dari sebuah algoritma menjadi suatu aplikasi dengan alur siklus pengembangan aplikasi dari literatur Rekayasa Perangkat Lunak (Pressman 2005). Yang perlu diperhatikan ialah pembuatan algoritma, termasuk dalam

ranah analisa dan desain. Sedangkan untuk implementasi algoritma menjadi sebuah bahasa pemrograman, merupakan tahap implementasi di dalam SDLC.

Algoritma dibutuhkan untuk memberikan suatu keputusan yang cepat, berulang-ulang dan yang terpenting ialah keputusan tersebut memberikan jawaban mendekati kebenaran. Benar atau tidak dari output yang diberikan oleh suatu algoritma tergantung dari nilai/batasan yang ditentukan oleh pengembangnya. Komputer hanya mengeksekusi algoritma yang didesain oleh penulis program. Contoh pada aplikasi pengolahan citra digital (digital image processing) dalam menentukan nilai gambar dibawah ini merupakan karakter angka 8 atau huruf B.



Gambar 2.4. Contoh gambar pixel pada pengenalan citra/image

Mungkin diantara rekan pembaca ada yang berpendapat ini huruf B atau ada juga yang berpendapat ini angka 8.



Tanpa kita sadari, dalam keseharian kita juga telah mengimplementasikan algoritma. Sebagai contoh pada saat (maaf) buang air besar.

Silahkan tulis pada sebuah kertas, langkah-langkah yang perlu dilakukan secara berurutan ketika buang air besar.

Setelah ditulis, sesuaikan jawaban anda dengan langkah-langkah berikut :

1. Pergi menuju kamar mandi.
2. Membaca doa sebelum masuk kamar mandi.
3. Membuka pintu kamar mandi.
4. Menutup pintu kamar mandi.
5. Membuka pakaian.
6. Buang air besar.
7. Membersihkan diri.
8. Memakai pakaian kembali.
9. Membuka pintu kamar mandi.
10. Menutup pintu kamar mandi.
11. Membaca doa keluar dari kamar mandi.

Setidaknya ada 11 langkah dasar dalam buang air besar yang tanpa kita sadari telah melakukannya setiap hari. Yang perlu diperhatikan ialah, langkah-langkah tersebut harus dikerjakan secara berurutan. Dapat dibayangkan apabila kita lupa melakukan langkah nomor 4. :)

1.2. Algoritma, Pseudocode dan Pemrograman

Setelah memahami pentingnya mempelajari algoritma, selanjutnya kita perlu untuk mengetahui bagaimana proses menjadikan algoritma menjadi suatu aplikasi. Pertama, kita perlu mengetahui perbedaan antara Algoritma, Pseudocode dan Syntax Pemrograman. Cara yang paling mudah untuk mengetahui perbedaannya yaitu kita mencoba menyelesaikan masalah berikut: **“Bagaimana membuat sebuah program untuk menghitung sebuah luas bangunan datar?”**



Selanjutnya kita mencoba membuat sebuah Algoritma, Pseudocode dan Syntax Pemrograman untuk menghitung luas dari sebuah bangun datar. Misalnya kita memilih bangun datar persegi panjang. Contoh algoritma menghitung luas persegi panjang :

1. User memasukkan nilai panjang
2. User memasukkan nilai lebar
3. Komputer menghitung nilai luas persegi panjang yaitu $\text{panjang} \times \text{lebar}$
4. Komputer menampilkan nilai luas persegi panjang

Lalu untuk contoh Pseudocode dapat ditulis sebagai berikut:

1. Panjang \leftarrow input Panjang
2. Lebar \leftarrow input lebar
3. Luas persegi panjang \leftarrow panjang \times lebar
4. Tampilan luas persegi panjang

Dari Pseudocode tersebut, kita tulis pada bahasa pemrograman Python sebagai berikut :

```
luaspersegi.py
1 panjang = int(raw_input('masukkan nilai panjang: '))
2 lebar = int(raw_input('masukkan nilai lebar: '))
3
4 luas = panjang * lebar
5
6 print luas
7
```

Perlu disampaikan, bahwa Pseudocode dapat ditulis menjadi sebuah Fungsi/ Prosedur yang hampir mirip dengan Syntax Pemrograman :

Prosedur hitung Luas Persegi Panjang(panjang,lebar)
Input: panjang, lebar
Output: Luas persegi panjang

Langkah-langkah:

1. Meminta inputan dari user panjang dan lebar
2. Hitung luas = panjang x lebar
3. Tampilkan nilai luas.

Berdasar Pseudocode diatas, dapat ditulis ke dalam bahasa pemrograman Python dengan menggunakan sebuah prosedur pula. Seperti ditunjukkan pada tangkapan layar berikut :

```
hitungLuasPersegiPanjang.py
1 def hitungLuasPersegiPanjang(panjang,lebar):
2     luas = panjang * lebar
3     print "Luas persegi panjang:",luas
4
5     inputan_panjang = int(raw_input('Masukkan nilai panjang:'))
6     inputan_lebar = int(raw_input('Masukkan nilai lebar:'))
7
8     hitungLuasPersegiPanjang(inputan_panjang,inputan_lebar)
9
```

Dalam literasi-literasi buku pemrograman berbahasa Inggris, kita boleh menulis Pseudocode dengan bahasa Indonesia atau campuran(Indonesia dan Inggris). Tidak ada aturan khusus untuk menulis Pseudocode. Yang perlu diperhatikan, fungsi Pseudocode ialah menyampaikan ide langkah-langkah dengan jelas mengenai algoritma yang akan diimplementasi menjadi sebuah program/perangkat lunak/aplikasi.

Namun, contoh diatas belum menyelesaikan pertanyaan sebelumnya :

“Bagaimana membuat sebuah program untuk menghitung sebuah luas bangunan datar? ”.

Karena algoritma, Pseudocode tadi hanya menyelesaikan 1 bangun datar. Bagaimana dengan beberapa bangun datar lainnya? Hal ini perlu diperhatikan karena aplikasi yang dibuat harus yang aplikatif (dapat diimplementasikan) di semua objek. Termasuk semua objek bangun datar.

Dikarenakan setiap bangun datar berbeda rumusnya. Maka kita perlu berpikir bagaimana program kita dapat menghitung lebih dari satu rumus. Yaitu ke beberapa rumus bangun datar. Disinilah kita dituntut kreatif dalam menyelesaikan masalah. Kita harus berpikir bagaimana mendeskripsikan langkah-langkah (algoritma) untuk dapat menghitung bangun datar yang di input oleh user. Solusi sederhananya, kita berikan pertanyaan kepada user, mau menghitung bangun datar yang mana. Layaknya kita memilih menu makanan saat berada di restoran.

Contoh pengembangan algoritma untuk masalah ini sebagai berikut:

1. User memasukkan pilihan bangun datar.

2. **Jika** yang dipilih bangun datar :

a. Persegi panjang maka

1. User memasukkan nilai panjang dan

2. User memasukkan nilai lebar

3. Komputer menghitung nilai luas yaitu panjang x lebar

4. Komputer menampilkan nilai luas persegi panjang

b. Segitiga maka:

1. User memasukkan nilai alas

2. User memasukkan nilai tinggi

3. Komputer menghitung nilai luas yaitu $0.5 \times$ alas x tinggi

4. Komputer menampilkan nilai luas segitiga

c. Persegi maka:

1. User memasukkan nilai sisi

2. Komputer menghitung nilai luas yaitu sisi x sisi

3. Komputer menampilkan nilai luas persegi

d. Jajaran genjang maka:

1. User memasukkan nilai alas

2. User memasukkan nilai tinggi

3. Komputer menghitung nilai luas yaitu alas x tinggi

4. Komputer menampilkan nilai luas jajaran genjang

Bila dibandingkan dengan algoritma menghitung persegi panjang sebelumnya, maka algoritma baru diatas terdapat inputan tambahan untuk memilih bangun datar dan perhitungan luas bangun datarnya sesuai dengan pilihan user.

Fungsi Hitung-Luas-Persegi(sisi):

Input: sisi: nilai dari sisi

Output: Luas-Persegi yang dihitung dari rumus menghitung luas persegi

Step:

1. Hitung Luas-Persegi = sisi x sisi
2. Tampilkan Luas-Persegi

Dari contoh sederhana tersebut dapat disimpulkan bahwa:

1. Algoritma merupakan langkah-langkah berurutan untuk menyelesaikan masalah dan dapat ditulis dalam bahasa yang kita pahami.
2. Pseudocode menerjemahkan algoritma menjadi sebuah langkah-langkah yang nantinya ditulis dalam bahasa pemrograman.
3. Bahasa pemrograman mengimplementasi Pseudocode dan algoritma tersebut menjadi suatu alat hitung dalam bentuk aplikasi.

Rangkuman

Bab ini mempelajari peran algoritma dalam kehidupan sehari-hari serta bagaimana menuliskan algoritma menjadi sebuah Pseudocode. Kemudian Pseudocode diimplementasi menjadi sebuah kode pemrograman menggunakan bahasa Python.

Mengetahui algoritma merupakan elemen terpenting dalam suatu aplikasi, maka memahami algoritma merupakan hal yang wajib untuk dikuasai. Berikut hal-hal yang perlu diketahui dalam memahami kegunaan sebuah algoritma :

1. Algoritma bukan digunakan untuk mencari nilai yang benar dan absolut (pasti). Namun digunakan untuk mencari pendekatan yang meminimalisir kesalahan dan mendekati kebenaran. Kesalahan yang dihasilkan oleh algoritma masih bisa ditoleransi, asalkan dapat mengontrol, sejauh mana kesalahan tersebut (Cormen 2013).
2. Selain untuk membantu manusia dalam memberikan keputusan dan perhitungan, algoritma pada umumnya digunakan untuk 2 hal berikut : pengurutan dan pencarian. Dimana pengurutan dan pencarian ini merupakan dua hal yang tidak dapat dipisahkan.
3. Algoritma berbeda dengan Pseudocode dan kode program. Algoritma dapat ditulis dalam sebuah kertas secara abstrak, namun kode program ialah sebuah syntax yang harus mengikuti kaidah/aturan tertentu. Dapat pula dipahami bahwa kode program merupakan cara memberikan instruksi kepada komputer.
4. Algoritma dapat diartikan sebagai langkah-langkah dalam menyelesaikan suatu permasalahan/pekerjaan.

Soal/Evaluasi

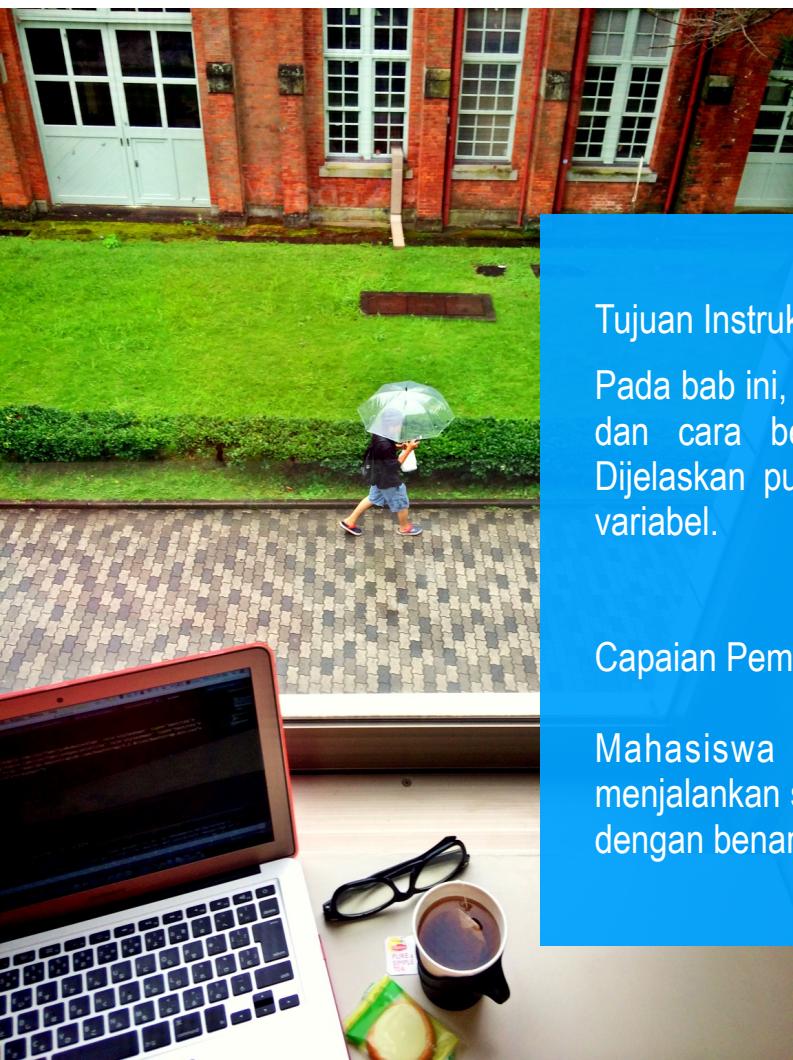
1. Jelaskan bagaimana memindahkan teh pada cangkir A ke cangkir B dan bagaimana pula memindahkan kopi pada cangkir B ke cangkir A. Silahkan tulis langkah-langkahnya di kertas.



2. Tuliskan algoritma dan Pseudocode dari pertanyaan berikut:
“Bagaimana menghitung keliling dari bangun datar berikut : persegi panjang, segitiga dan lingkaran.”
3. Tuliskan algoritma dari aplikasi yang bekerja pada sistem berikut :
Japan Automatic Train Crossover Gate. Link : <https://www.youtube.com/watch?v=vjdFSGPOdjA>

Bab 3

Pemrograman



Tujuan Instruksional :

Pada bab ini, dijelaskan latar belakang dipilihnya bahasa Python dan cara berinteraksi dengan Interpreter bahasa Python. Dijelaskan pula mengenai Variabel, Fungsi dan Tipe-tipe dari variabel.

Capaian Pembelajaran Mata Kuliah :

Mahasiswa mampu menggunakan Python Interpreter, menjalankan script Python dan memahami penggunaan variabel dengan benar sesuai tipe variabel tersebut.

Bab 3. Pemrograman

Kita mengetahui komputer dapat melakukan perhitungan secara persis dalam waktu yang singkat, namun perlu diingat bahwa bagaimanapun juga komputer bukanlah manusia. Sebuah komputer tidak dapat mengalahkan manusia.

Komputer tidak dapat mengambil keputusan sendiri, kecuali di komputer tersebut sudah di “program” untuk mengambil keputusan sendiri. Sehingga, keterbatasan komputer ialah mengambi inisiatif/keputusan di luar yang sudah diprogramkan. Agar komputer memberikan output seperti yang diharapkan, diperlukan pemrograman. Yaitu menulis baris-baris kode (syntax) dengan aturan tertentu tergantung bahasa pemrograman yang dipilih.

3.1. Kenapa Python?

3.1.1. Multi platform dan Multi device

Memahami perbedaan bahasa pemrograman bukan dengan cara mencari mana yang paling hebat bahasa pemrogramannya. Hal ini tidaklah tepat, karena setiap masing-masing bahasa pemrograman memiliki kelebihan dan kekurangan. Cara yang benar melihat perbedaan bahasa pemrograman adalah dengan memilih bahasa pemrograman yang sederhana dan mudah dipelajari untuk siapa saja yang baru belajar pemrograman. Python dipilih karena mudah dipelajari dan lebih efisien bagi programmer pemula. Kita lihat saja dalam menulis sebuah text dilayar “Teh hangat satu!”. Pada bahasa Java dituliskan sebagai berikut :

```
teh.java
1 public class HelloWorld {
2
3     public static void main(String[] args) {
4         System.out.println("Teh hangat satu.");
5     }
6 }
7
8
```

Gambar 3.1. Syntax Java untuk menulis “Teh hangat satu.”

Sedangkan bahasa C++ sebagai berikut :

```
teh.cpp
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "Teh hangat satu." << endl;
8     return 0;
9 }
10
```

Gambar 3.2. Syntax C++ untuk menulis “Teh hangat satu.”

Untuk bahasa Python, cukup ditulis sebagai berikut :

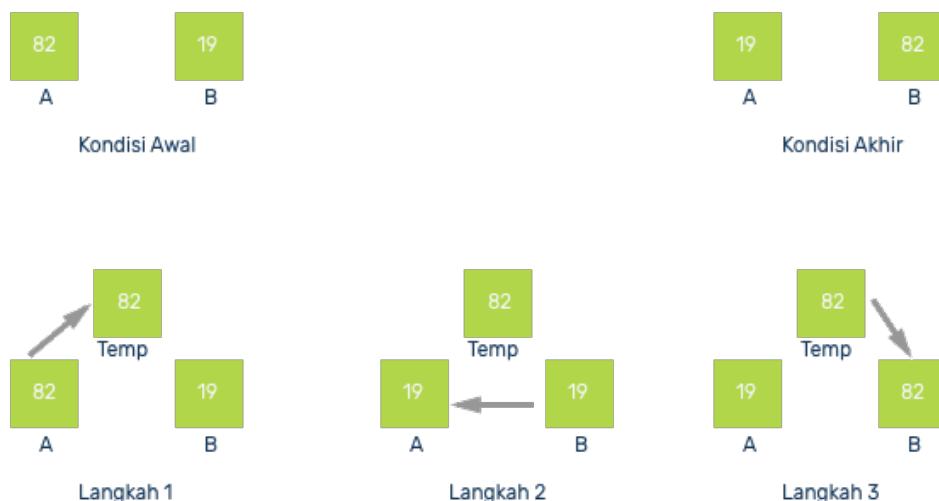
```
teh.py
1 print 'Teh hangat satu.'
2
3
```

Gambar 3.3. Syntax Python untuk menulis “Teh hangat satu.”

Python dipilih karena syntax bahasa pemrograman Python begitu sederhana dan memudahkan orang yang baru belajar membuat program. Terlebih Python dapat di install dihampir semua sistem operasi yang umum digunakan kalangan mahasiswa. Yaitu GNU/Linux, MacOS dan Microsoft Windows. Juga tersedia aplikasi Python untuk Android (<http://www.qpython.com>) dan IOS (<http://pythonforios.com/>).

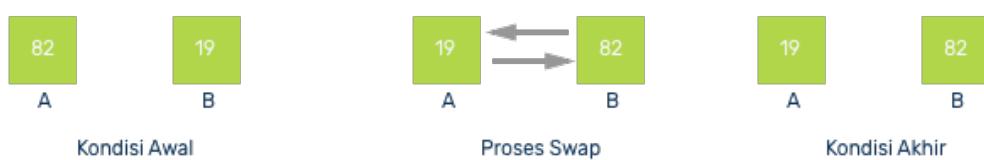
3.1.2. Simultan Swap

Di dalam bahasa pemrograman, pada umumnya untuk memindahkan nilai dari satu variabel ke variabel lainnya, membutuhkan 3 langkah. Jika ingin memindahkan nilai variabel dari A ke B, biasanya membutuhkan satu lagi variabel. Sebut saja variabel Temp. Langkah pertama, Nilai A dimasukkan dalam variable Temp. Langkah kedua, nilai B dimasukkan dalam nilai A. Langkah ketiga, nilai B. Ketiga langkah tersebut di ilustrasikan pada gambar berikut :



Gambar 3.4. Proses swap variable pada bahasa pemrograman secara umum.

Sedangkan di dalam Python, memungkinkan untuk secara simultan meng-swap variable dari A ke B.



Gambar 3.5. Proses swap variable pada bahasa pemrograman Python.

```
>>> A=82
>>> B=19
>>> A
82
>>> B
19
>>> A,B=B,A
>>> A
19
>>> B
82
```

Gambar 3.6. Contoh swap variable pada Python.

Pada Gambar 3.6. kita definisikan sebuah variabel A dengan angka 82. Dan sebuah variabel B dengan angka 19. Lalu dicetak nilai A dan B, dengan nilai masing-masing 82 dan 19. Kemudian dilakukan proses swap A,B=B,A, nilai A dan B berubah menjadi 19 dan 82.

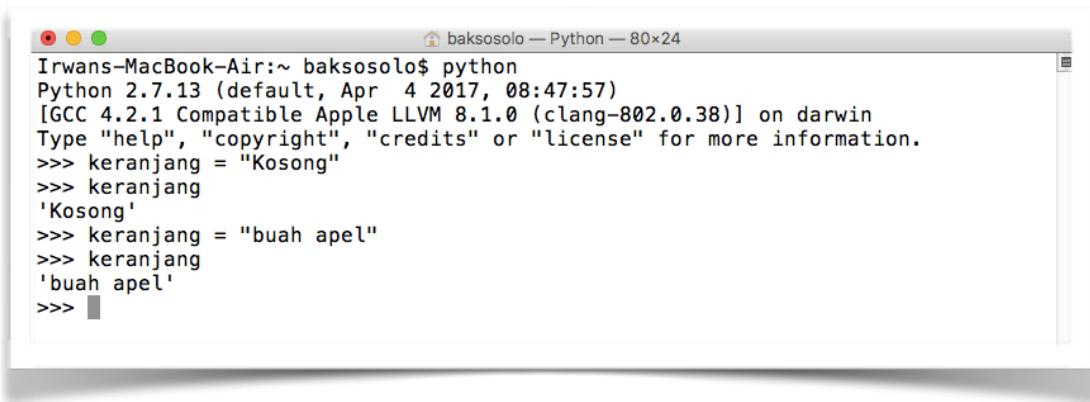


Untuk memahami cara kerja swap variable di Python. Silahkan coba menyelesaikan gambar 3.5 dengan mengikuti langkah-langkah seperti di gambar 3.6 dengan menggunakan Python Intrepreter.

3.2. Variabel

Variabel digunakan untuk menyimpan inputan dari user dan menyimpan hasil perhitungan yang kemudian dijadikan output/informasi bagi user. Untuk memahami penggunaan variabel, silahkan ikuti langkah-langkah berikut dan perhatikan gambar :

1. Buka terminal dan masuk ke Python intrepreter.
2. Buat variabel “keranjang” (tanpa tanda kutip) dan isikan dengan nilai “Kosong” (menggunakan tanda kutip).
3. Kemudian kita panggil variabel keranjang. Dan akan ditampilkan nilai “Kosong”. Hal ini menunjukkan bahwa kita sudah memberikan nilai “Kosong” pada variabel keranjang.
4. Kemudian kita masukkan “buah apel” ke dalam variabel keranjang.
5. Selanjutnya, jika kita panggil variabel keranjang, maka akan ditampilkan “buah apel”, tidak lagi “Kosong”.



```
Irwans-MacBook-Air:~ baksosolo$ python
Python 2.7.13 (default, Apr  4 2017, 08:47:57)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.38)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> keranjang = "Kosong"
>>> keranjang
'Kosong'
>>> keranjang = "buah apel"
>>> keranjang
'buah apel'
>>>
```

Gambar 3.7. Pendefinisian variabel di Python.

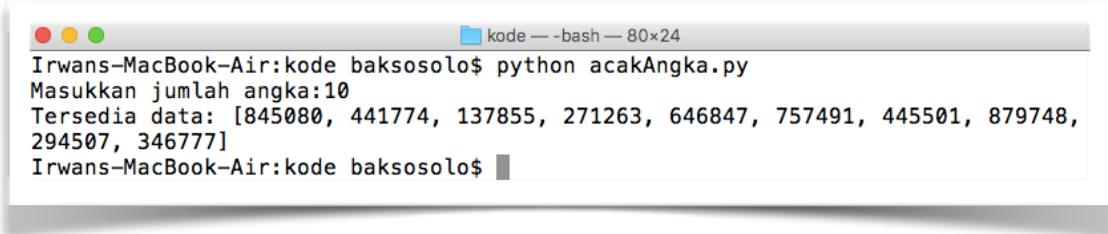
Dalam bahasa pemrograman apapun, variabel pasti dibutuhkan. Variabel merupakan alat bantu untuk menyimpan inputan dari user dan digunakan untuk proses perhitungan. Coba lihat kode berikut :



```
acakAngka.py
1 import random
2
3 def acakAngka(jml):
4     himpunan = random.sample(xrange(0,1000000),jml)
5     print "Tersedia data:",himpunan
6
7 jumlah= int(raw_input('Masukkan jumlah angka:'))
8
9 acakAngka(jumlah)
10
```

Gambar 3.8. contoh variable di Python.

Pada kode diatas, dapat dilihat bahwa kita mendefinisikan sebuah variabel bernama jumlah. Variabel jumlah digunakan untuk menampung inputan dari user. Jika dimasukkan angka 10, maka program diatas akan memanggil 10 angka acak. Seperti ditunjukkan pada gambar berikut :



```
Irwans-MacBook-Air:kode baksosolo$ python acakAngka.py
Masukkan jumlah angka:10
Tersedia data: [845080, 441774, 137855, 271263, 646847, 757491, 445501, 879748,
294507, 346777]
Irwans-MacBook-Air:kode baksosolo$
```

Gambar 3.9. Output bilangan acak.

Walaupun pada Python dimungkinkan tidak menggunakan variabel dalam melakukan proses perhitungan, namun data hasil perhitungan perlu ditampung dalam sebuah variabel agar hasil perhitungan tersebut dapat diolah menjadi suatu informasi dan digunakan pada proses perhitungan selanjutnya.

Untuk penamaan variabel, dalam menggunakan bahasa pemrograman Python, perlu memperhatikan beberapa aturan berikut, yaitu :

1. Penamaan variabel dapat menggunakan kombinasi dari huruf A-Z, atau a-z. Lambang angka dari 0 hingga 9. Sebagai contoh : LontongB4l4p = "7500".
 2. Tidak boleh mengandung spasi. Contoh : nasi goreng = 'enak', Seharusnya, nasigoreng = 'enak' atau nasi_goreng='enak'.
 3. Tidak boleh hanya angka. Contoh yang salah : 1999 = "tahun_lahirku".
 4. Angka tidak boleh digunakan pada awal penamaan variabel.
Misal := "kertas". Seharusnya Media34 = "kertas".
- 5.Tidak boleh menggunakan kata yang termasuk built in function Python. Misal kata : import, main, def dan lainnya. Detail built in function dapat dilihat disini : <https://docs.python.org/2/library/functions.html> (Python versi 2) dan <https://docs.python.org/3/library/functions.html> (Python versi 3).

3.3. Tipe Variabel

Berikut dijelaskan beberapa Tipe Variabel.

3.3.1. Integer



```
persegi.py
1 Panjang = 90
2 Lebar = 70
3 print "Luas persegi panjang tersebut ialah:", Panjang * Lebar
4
```

Gambar 3.10. Tipe Integer

Tipe integer digunakan dalam menyimpan bilangan bulat. Misalnya variabel panjang pada script diatas disimpan dengan nilai 90. Sedangkan lebar = 70. Maka output dari program diatas ialah hasil dari perhitungan panjang dikalikan lebar (Gambar 2.1). Perlu diketahui di dalam Python, tipe variabel biasanya tidak perlu didefinisikan seperti kode diatas. Akan tetapi, disarankan untuk

memberikan fungsi “int” dalam sebuah nilai variabel. Contoh int(90) atau int(80).

3.3.2. String

Tipe string, merupakan tipe yang menyimpan karakter yang ada pada semua tombol keyboard. Yaitu, A-Z, 0-9, <>_+=!@#\$%^&*. Juga bisa menyimpan karakter linguistik negara lain. Contoh: 熊本大学. Untuk mendefinisikan variabel string, cukup memberikan tanda petik satu/dua pada nilai yang dimasukkan. Sebagai contoh: kota='sidoarjo' atau kota = “sidoarjo”. Jika yang digunakan di awal petik satu, maka akhirannya juga harus menggunakan petik satu juga.

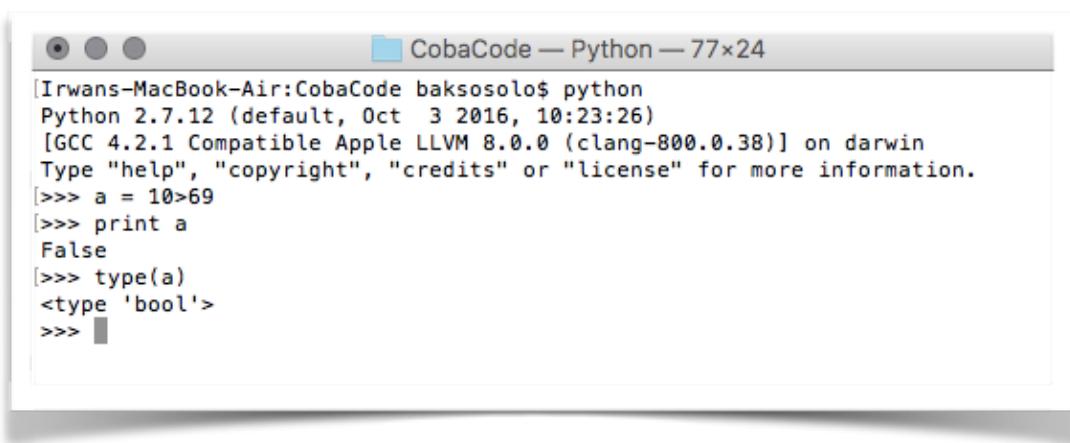
Perhatikan gambar berikut:

```
>>> kota = 'Sidoarjo'  
>>> type(kota)  
<type 'str'>  
>>> campus = '熊本大学'  
>>> type(campus)  
<type 'str'>  
>>>
```

Gambar 3.11. Tipe String

3.3.3. Boolean

Tipe boolean merupakan tipe yang hanya mengeluarkan output nilai True or False saja. Contoh: Jika kita memberikan sebuah variabel a, dengan nilai $10 > 69$, akan menghasilkan nilai salah (False). Dikarenakan pernyataan 10 lebih besar dari 69 ialah pernyataan yang bernilai salah (False).

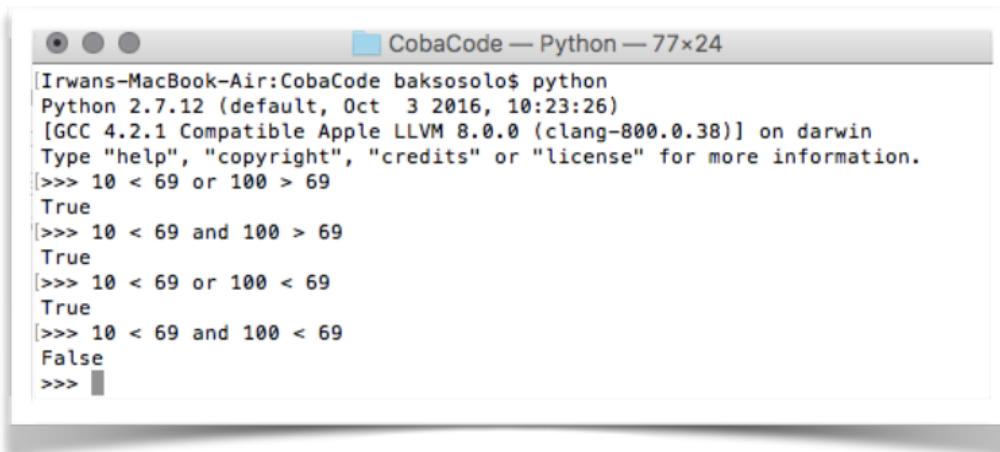


```
Irwans-MacBook-Air:CobaCode baksosolo$ python
Python 2.7.12 (default, Oct  3 2016, 10:23:26)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.38)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 10>69
>>> print a
False
>>> type(a)
<type 'bool'>
>>>
```

Gambar 3.12. Tipe Boolean

Tipe boolean dikenal untuk membandingkan operator AND dan OR. Masih ingat perbandingan tersebut di pelajaran matematika Sekolah Dasar ?

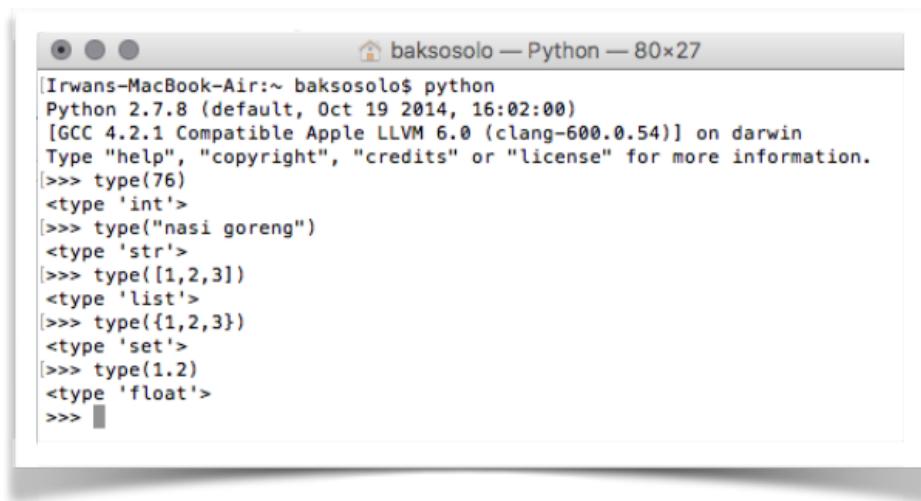
Operator OR akan bernilai True selama salah satu variabel yang dibandingkan bernilai True. Sebaliknya, operator AND akan bernilai True jika kedua variabel yang dibandingkan bernilai True.



```
Irwans-MacBook-Air:CobaCode baksosolo$ python
Python 2.7.12 (default, Oct  3 2016, 10:23:26)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.38)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 10 < 69 or 100 > 69
True
>>> 10 < 69 and 100 > 69
True
>>> 10 < 69 or 100 < 69
True
>>> 10 < 69 and 100 < 69
False
>>>
```

Gambar 3.13. Operator OR dan AND

Yang menarik di Python ialah tipe dari variabel Python tergantung dari tipe value yang diberikan di variabel. User dapat mengetahui tipe dari variabel dengan menanyakan kepada Python intrepreter seperti pada gambar berikut :



```
Irwans-MacBook-Air:~ baksosolo$ python
Python 2.7.8 (default, Oct 19 2014, 16:02:00)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.54)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> type(76)
<type 'int'>
>>> type("nasi goreng")
<type 'str'>
>>> type([1,2,3])
<type 'list'>
>>> type({1,2,3})
<type 'set'>
>>> type(1.2)
<type 'float'>
>>>
```

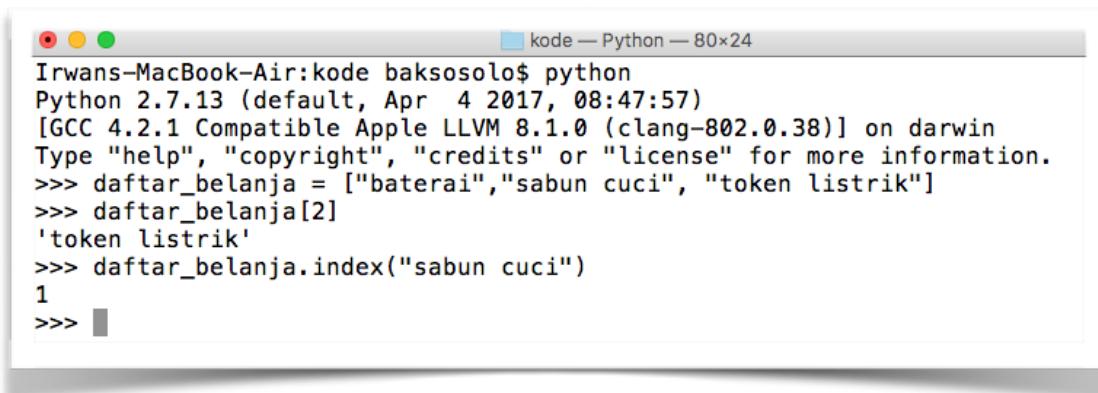
Gambar 3.14. Penggunaan type ()

3.3.4. List

Tipe list merupakan sebuah daftar variabel dalam satu himpunan. Bingung definisinya? Bayangkan saja anda disuruh ibu belanja. Apa yang anda lakukan? Tentunya membuat daftar belanja tadi. Contoh : baterai, sabun cuci dan token listrik. Pendefinisian daftar belanjaan tadi di dalam Python ialah sebagai berikut :

```
daftar_belanja = ["baterai", "sabun cuci", "token listrik"]
```

Dimana dapat ditulis dalam Python intrepreter seperti pada gambar berikut :



```
Irwans-MacBook-Air:kode baksosolo$ python
Python 2.7.13 (default, Apr  4 2017, 08:47:57)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.38)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> daftar_belanja = ["baterai","sabun cuci", "token listrik"]
>>> daftar_belanja[2]
'token listrik'
>>> daftar_belanja.index("sabun cuci")
1
>>>
```

Gambar 3.15. Tipe List

Kemudian jika kita ingin memanggil data pada list tersebut, kita menggunakan perintah: `daftar_belanja[2]`, maka data yang muncul ialah data pada indeks ke 2, yaitu “token listrik”. Sedangkan apabila kita ingin mengetahui posisi indeks dari sebuah data tersebut, kita menggunakan perintah : `daftar_belanja.index("sabun cuci")`. Maka outputnya ialah 1. Yang perlu diingat bahwa, indeks dalam Python dimulai dari angka 0, bukan angka 1.

Perlu diketahui pula, tipe List dalam Python dapat dipilih dan dimanipulasi elemennya. Misalnya kita mempunyai sebuah himpunan angka: 1,3,5,7,9. Maka kita definisikan angka tersebut menjadi list yaitu dengan cara : `himpunan = [1,3,5,7,9]`. Adapun manipulasi variabel tipe List yang bisa kita lakukan ialah sebagai berikut :

Tabel 3.1. Contoh manipulasi tipe data List.

Himpunan = [1,3,5,7,9]

Perintah di Python	Output	Keterangan
len(himpunan)	5	Menampilkan banyaknya data dalam satu himpunan
9 in himpunan	TRUE	Mencari tahu apakah 9 termasuk di himpunan
himpunan[3]	7	Memilih data di index yang ke 3.
himpunan[3:]	[7,9]	Menampilkan data yang tersisa, setelah dipotong 3 angka dari kiri
himpunan[-3:]	[5,7,9]	Memilih 3 data dari kanan
himpunan[:3]	[1,3,5]	Menampilkan 3 data dari kiri
himpunan[:-3]	[1,3]	Menampilkan data yang tersisa, setelah dipotong 3 angka dari kanan

3.4. Conditional Statement

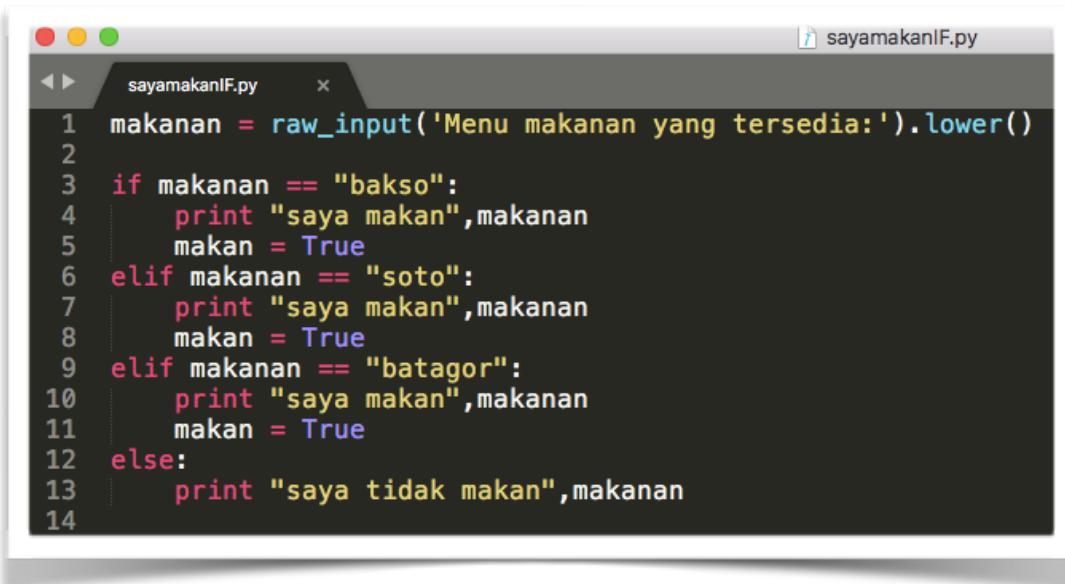
Salah satu yang sering dijumpai dalam pemrograman ialah pernyataan bersyarat (Conditional Statement). Yaitu memerintahkan komputer untuk memberikan output tertentu sesuai dengan kondisi/syarat yang tidak ditentukan. Bentuk umum dari Conditional Statement ini ialah:

```

if (kondisi A terpenuhi):
    melakukan operasi jika kondisi A terpenuhi
elif (kondisi B terpenuhi):
    melakukan operasi jika kondisi A terpenuhi
else:
    melakukan operasi jika kondisi A dan B TIDAK terpenuhi
  
```

Gambar 3.16. Penggunaan IF, ELIF dan ELSE.

Sebuah komputer akan menampilkan pesan “Saya akan makan” jika yang dihidangkan ialah bakso, soto dan batagor. Jika selain makanan tersebut maka ditampilkan pesan “Saya tidak makan.”. Untuk keadaan ini, dapat dicontohkan dengan kode berikut :



```
sayamakanIF.py
1 makanan = raw_input('Menu makanan yang tersedia:').lower()
2
3 if makanan == "bakso":
4     print "saya makan",makanan
5     makan = True
6 elif makanan == "soto":
7     print "saya makan",makanan
8     makan = True
9 elif makanan == "batagor":
10    print "saya makan",makanan
11    makan = True
12 else:
13     print "saya tidak makan",makanan
14
```

Gambar 3.17. Kode penggunaan IF.

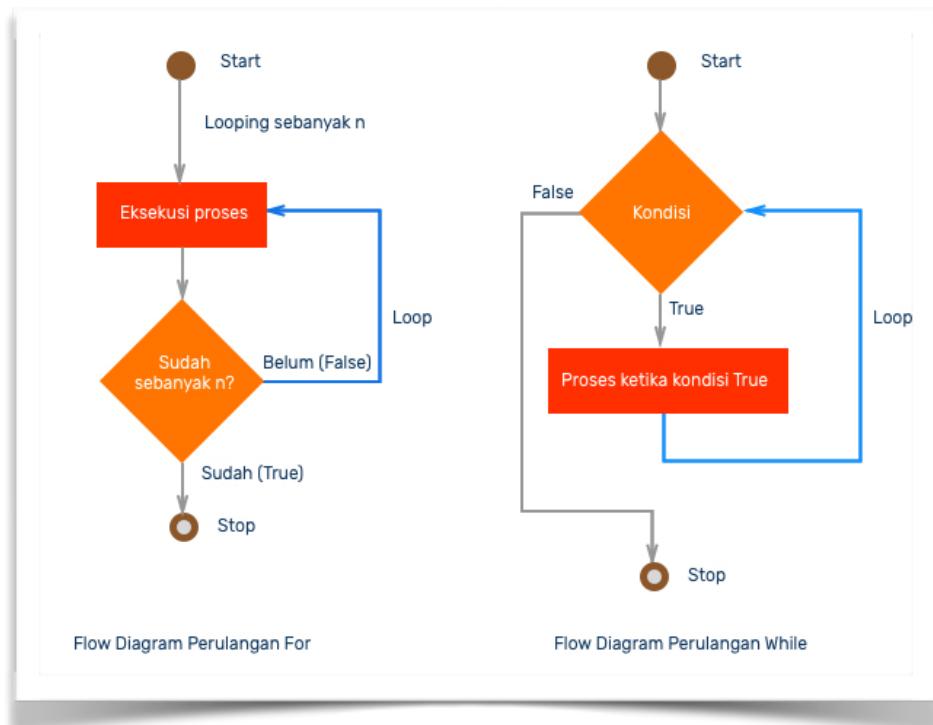
Maka, output dari kode diatas akan menampilkan “saya makan ...” ketika kata kunci bakso, soto dan batagor ditulis dalam inputan tersebut. Seperti yang ditunjukkan gambar berikut :

```
Irwans-MacBook-Air:kode baksosolo$ python sayamakanIF.py
Menu makanan yang tersedia:soto
saya makan soto
Irwans-MacBook-Air:kode baksosolo$ python sayamakanIF.py
Menu makanan yang tersedia:bakso
saya makan bakso
Irwans-MacBook-Air:kode baksosolo$ python sayamakanIF.py
Menu makanan yang tersedia:mie ayam
saya tidak makan mie ayam
```

Gambar 3.18. Output Conditional Statement

3.5. Looping

Dalam memproses suatu perhitungan, terkadang kita membutuhkan sebuah perulangan. Di dalam Python terdapat dua jenis perulangan, yaitu For dan While.



Gambar 3.19. Flowchart perulangan For dan While.

3.5.1. Perulangan For

Perulangan For digunakan apabila kita ingin melakukan perulangan tanpa kondisi tertentu (gambar 3.19). Dan looping for akan berhenti apabila perulangan sudah mencapai n looping yang ditentukan. Misalnya, kita hanya melakukan perulangan sebanyak $n=10$ kali untuk menuliskan kata “saya ingin makan bakso!”.

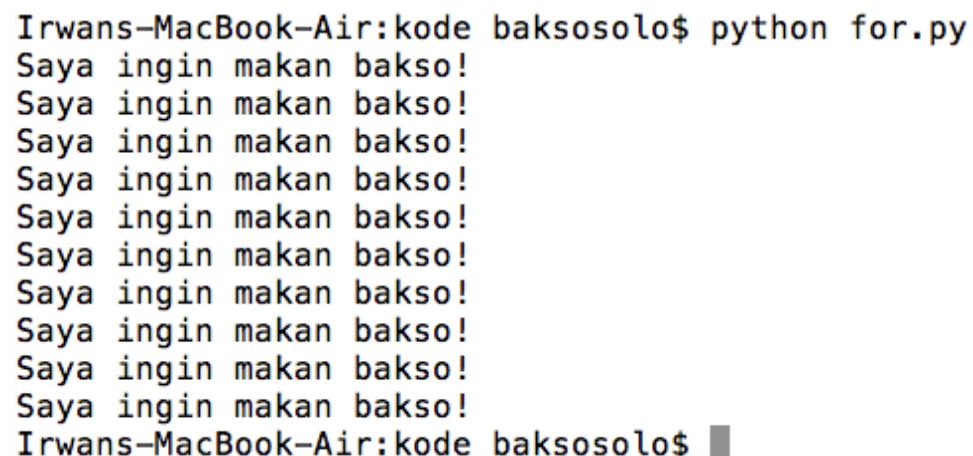
Maka syntax kodennya ialah :



```
for.py
for i in range(10):
    print "Saya ingin makan bakso!"
```

Gambar 3.20. Perulangan For

Jika kode diatas dijalankan, akan menampilkan perulangan “Saya ingin makan bakso!” sebanyak 10 kali. Dimana looping akan berhenti setelah melakukan 10 kali looping.



```
Irwans-MacBook-Air:kode baksosolo$ python for.py
Saya ingin makan bakso!
Irwans-MacBook-Air:kode baksosolo$
```

Gambar 3.21. Output Perulangan For

3.5.2. Perulangan While

Berbeda dengan perulangan For, perulangan While digunakan ketika ingin mengulang suatu proses perhitungan dengan sebuah kondisi tertentu (gambar 19). Untuk memahami perulangan While, perhatikan studi kasus berikut : Kita ingin membuat program yang terus-menerus menanyakan user apakah user sudah kenyang apa belum. Jika dijawab sudah kenyang, maka looping berhenti. Contoh output kodennya sebagai berikut :



```
Irwans-MacBook-Air:kode baksosolo$ python while.py
Apakah anda sudah kenyang?(Y/T):t
Status: belum kenyang.
Apakah anda sudah kenyang?(Y/T):t
Status: belum kenyang.
Apakah anda sudah kenyang?(Y/T):T
Status: belum kenyang.
Apakah anda sudah kenyang?(Y/T):A
Mohon isi dengan Y atau T.
Apakah anda sudah kenyang?(Y/T):B
Mohon isi dengan Y atau T.
Apakah anda sudah kenyang?(Y/T):y
Status: sudah kenyang.
Irwans-MacBook-Air:kode baksosolo$
```

Gambar 3.22. Output Perulangan While.

Nah, bagaimana membuat programnya? Kita butuh program yang akan berulang menanyakan “Apakah anda sudah kenyang?”. Dan ketika user menginputkan huruf Y/y maka perulangan akan berhenti.

Kita mulai dengan mendefinisikan kondisi perulangan dengan nilai kenyang =

False. Artinya, selama variabel “kenyang” bernilai False, perulangan terus dijalankan.

Sebagai informasi, dengan mengkondisikan suatu variabel sesuai dengan kondisi perulangan While, maka oleh Python dianggap True. Walaupun yang kita setting nilai variabel tersebut adalah False. Jika kondisinya bernilai True (seperti di gambar 3.19 kondisi while bergaris biru), maka perulangan terus dilakukan. Perulangan tersebut akan berhenti apabila kondisi nilai variabel kenyang tidak sesuai dengan kondisi While yang kita definisikan. Untuk itu, agar looping/perulangan ini berhenti, kita perlu definisikan nilai kenyang menjadi True. Nah, ketika user mengisi pertanyaan “Apakah anda sudah kenyang?”, dengan nilai Y/y maka nilai kenyang kita setting menjadi True dan hal ini menyebabkan ketidaksesuaian dengan kondisi prasyarat While maka perulangan akan berhenti. Perhatikan kode berikut ini.

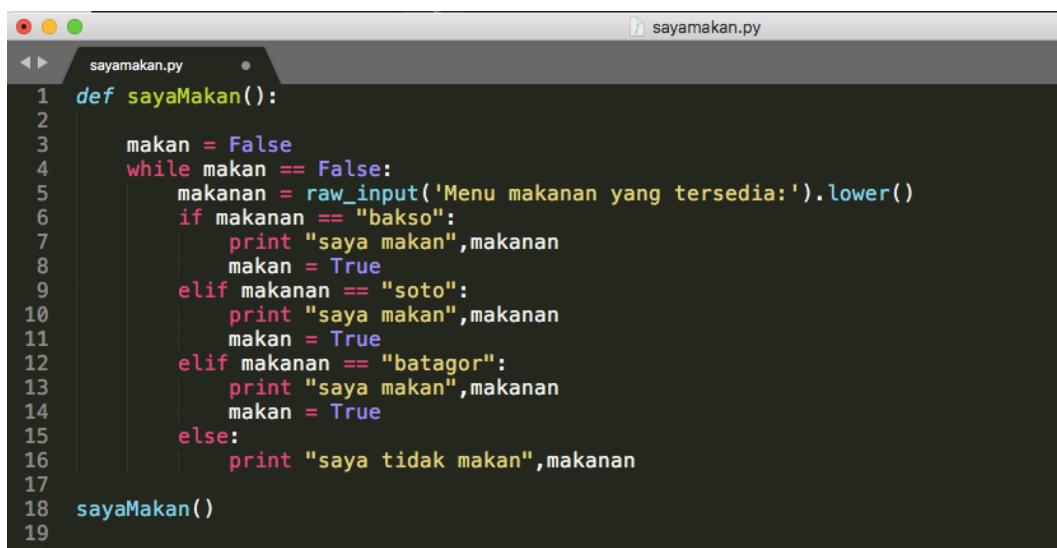


```
while.py
1  kenyang = False
2
3  while kenyang == False:
4      status = raw_input("Apakah anda sudah kenyang?(Y/T):")
5      if status.lower() == "y":
6          print "Status: sudah kenyang."
7          kenyang = True
8      elif status.lower() == "t":
9          print "Status: belum kenyang."
10     else:
11         print "Mohon isi dengan Y atau T."
12
```

Gambar 3.23. Kode perulangan While.

3.6. Procedure/Method

Suatu pemrograman yang baik ialah tidak berulang-ulang menulis kode program. Sehingga baris kode yang ditulis menjadi tidak terlalu banyak.



```
sayamakan.py
1 def sayaMakan():
2
3     makan = False
4     while makan == False:
5         makanan = raw_input('Menu makanan yang tersedia:').lower()
6         if makanan == "bakso":
7             print "saya makan",makanan
8             makan = True
9         elif makanan == "soto":
10            print "saya makan",makanan
11            makan = True
12        elif makanan == "batagor":
13            print "saya makan",makanan
14            makan = True
15        else:
16            print "saya tidak makan",makanan
17
18    sayaMakan()
19
```

Gambar 3.24. Metode di Python.

Silahkan perhatikan kode pada gambar 3.17 dan gambar 3.24. Dan jalankan kode tersebut. Kode pada gambar 3.17, tidak menggunakan sebuah metode. Berbeda dengan kode pada gambar 3.19 yang menggunakan metode. Perbedaannya ialah adanya deklarasi metode dengan bentuk umum :

def nama Metode(variabel yang dibutuhkan):
proses yang dijalankan metode

Untuk memanggil metode tersebut, cukup dengan menuliskan nama metode yang telah didefinisikan sebelumnya (baris ke 18 pada gambar 3.24). Dengan menggunakan metode di Python, kode lebih mudah dibaca dan terstruktur.

3.7. Recursive

Arti dari rekursif ialah menyelesaikan masalah dengan memanggil metodenya sendiri. Bingung? Bagaimana maksudnya? Coba perhatikan List berikut :

Himpunan = [1,3,5,7,9]. Bagaimana membuat program untuk menjumlahkan semua elemen himpunan? Cara manual kita menghitungnya dengan cara berikut :

jumlah_himpunan = (1+3) + 5 + 7 + 9

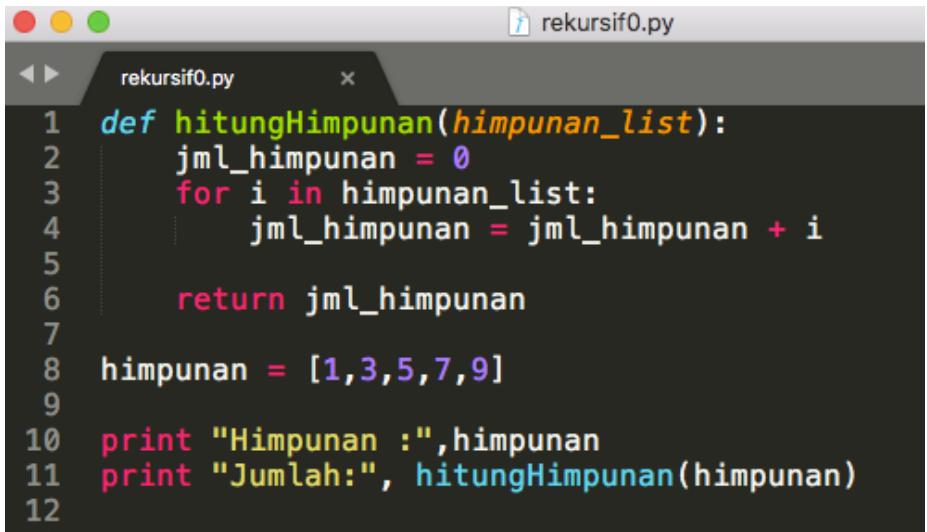
jumlah_himpunan = (4 + 5) + 7 + 9

jumlah_himpunan = (9 + 7) + 9

jumlah_himpunan = (16 + 9)

jumlah_himpunan = 25

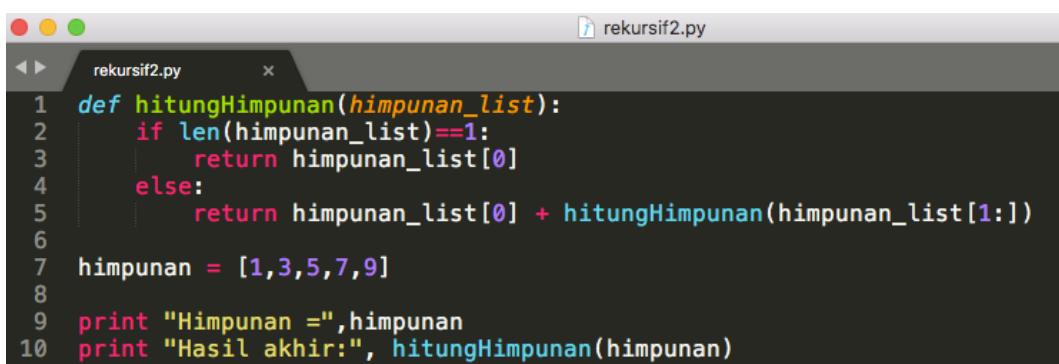
Jika diimplementasikan menjadi program ialah sebagai berikut :



```
rekursif0.py
1 def hitungHimpunan(himpunan_list):
2     jml_himpunan = 0
3     for i in himpunan_list:
4         jml_himpunan = jml_himpunan + i
5
6     return jml_himpunan
7
8 himpunan = [1,3,5,7,9]
9
10 print "Himpunan :", himpunan
11 print "Jumlah:", hitungHimpunan(himpunan)
12
```

Gambar 3.25. Penjumlahan menggunakan For.

Kode diatas, menyelesaikan perhitungan dengan menjumlahkan elemen satu dengan elemen berikutnya pada setiap perulangan. Waktu proses penjumlahan dengan cara ini bergantung pada jumlah data yang ada dalam suatu himpunan tersebut. Agar tidak melakukan perulangan, dapat menggunakan rekursif seperti kode berikut :



```
rekursif2.py
1 def hitungHimpunan(himpunan_list):
2     if len(himpunan_list)==1:
3         return himpunan_list[0]
4     else:
5         return himpunan_list[0] + hitungHimpunan(himpunan_list[1:])
6
7 himpunan = [1,3,5,7,9]
8
9 print "Himpunan =",himpunan
10 print "Hasil akhir:", hitungHimpunan(himpunan)
```

Gambar 3.26. Penjumlahan menggunakan rekursif

Dapat dilihat pada gambar 3.26. penjumlahan elemen himpunan tidak menggunakan perulangan For. Akan tetapi dengan memanggil metode itu sendiri (kode baris ke 5).

3.8. I/O Process

Python dapat menerima inputan dari User ketika script .py dijalankan. Ada dua macam inputan yaitu raw_input dan input. raw_input akan membaca yang dimasukkan user dan dibaca sebagai string. Berbeda dengan input, input akan memproses inputan user sesuai ekspresi yang diinputkan. Sebagai contoh apabila pada raw_input dimasukkan 5^*5 maka 5^*5 ini dianggap string saja.

Jika dimasukkan pada input $5*5$, maka akan ditampilkan hasil dari ekspresi $5*5$.

```
>>> masukan = raw_input("enter:")
enter:5*5
>>> masukan
'5*5'
>>> masukan = input("enter:")
enter:5*5
>>> masukan
25
```

Gambar 3.27. Perbedaan raw_input dan input.

Untuk membaca inputan dari sebuah file, dapat menggunakan perintah open.

Seperti gambar kode berikut :

The screenshot shows three terminal windows labeled A, B, and C. Window A (top) shows the command `python openMakan.py` being run, followed by the output "Saya ingin makan soto". Window B (middle) shows the Python code `filetxt = open("makan.txt","r")` and `print filetxt.read()`. Window C (bottom) shows the contents of the file `makan.txt`, which is "Saya ingin makan soto".

```
Irwans-MacBook-Air:kode baksosolo$ python openMakan.py
Saya ingin makan soto
Irwans-MacBook-Air:kode baksosolo$
```

```
openMakan.py
```

```
1 filetxt = open("makan.txt","r")
2 print filetxt.read()
```

```
makan.txt
```

```
1 Saya ingin makan soto
2
```

Gambar 3.28. Kode Read File

Pada tangkapan layar diatas, terdapat sebuah file text bernama `makan.txt` (gambar 3.28 bagian C). Kemudian dengan perintah `open` dan opsi “`r`” untuk `read`. Selanjutnya dengan perintah `print` menampilkan isi dari file text tersebut

(gambar 3.28 bagian B). Gambar 3.28 bagian A menampilkan output dari hasil eksekusi kode tersebut (gambar 3.28 bagian B).

Tidak hanya membaca file, dapat juga menulis pada file tersebut. Dengan memberikan opsi “w” pada saat open file. Serta memberi perintah :

```
file.write("text yang ingin ditulis")
```

Namun perlu diperhatikan ketika memberikan opsi “w”, isi text sebelumnya akan direplace dengan text inputan terbaru. Silahkan perhatikan gambar 3.29 berikut.

The image shows three terminal windows (A, B, and C) illustrating the write operation:

- Terminal A:** Shows the command `python writeMakan.py` being run, resulting in the output "Saya ingin makan bakso".
- Terminal B:** Shows the Python script `writeMakan.py` which contains:

```
1 file = open("makan.txt","w")
2 file.write("Saya ingin makan bakso")
3
4 file = open("makan.txt","r")
5 print file.read()
```
- Terminal C:** Shows the contents of the file `makan.txt`, which contains the single line "Saya ingin makan bakso".

Gambar 3.29. Kode Write File

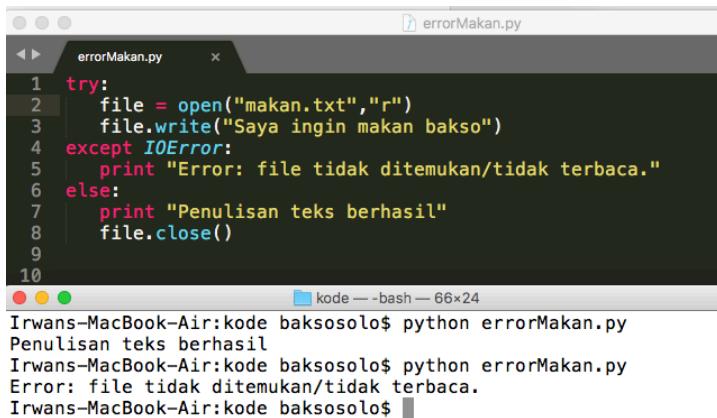
Pada tangkapan layar diatas, terdapat sebuah file text bernama makan.txt (gambar 3.28 bagian C) telah berubah isinya sesuai dengan perintah file.write (kode baris ke 2 pada gambar 3.29 bagian B). Kemudian dengan perintah open dan opsi “r” untuk menampilkan perubahan yang dilakukan. Selanjutnya dengan perintah print menampilkan isi dari file text tersebut (kode baris ke 5 pada gambar 3.29 bagian B). Gambar 3.28 bagian A menampilkan output dari hasil eksekusi kode tersebut (gambar 3.28 bagian B).

3.9. Error Handling

Yang perlu untuk dipelajari dalam pemrograman selanjutnya ialah bagaimana menangani adanya error saat eksekusi program dijalankan. Bentuk umum dari penanganan error ialah sebagai berikut :

```
try:  
    sebuah prosedure disini  
  
except Exception1:  
    Jika ada Eksepsi 1, maka eksekusi kode ini  
except Exception2:  
    Jika ada Eksepsi 1, maka eksekusi kode ini  
.....  
else:  
    Jika tidak ada exception, maka eksekusi kode ini
```

Contoh kode programnya sebagai berikut :



```
errorMakan.py
1 try:
2     file = open("makan.txt","r")
3     file.write("Saya ingin makan bakso")
4 except IOError:
5     print "Error: file tidak ditemukan/tidak terbaca."
6 else:
7     print "Penulisan teks berhasil"
8     file.close()
9
10
```

Irwans-MacBook-Air:kode baksosolo\$ python errorMakan.py
Penulisan teks berhasil
Irwans-MacBook-Air:kode baksosolo\$ python errorMakan.py
Error: file tidak ditemukan/tidak terbaca.
Irwans-MacBook-Air:kode baksosolo\$

Gambar 3.30. Kode Error Handling.

Pada kode diatas, baris ke 2, opsi membaca file diberi opsi “w”. Maka akan muncul pesan penulisan teks berhasil. Namun apabila diberi opsi “r”, dimana opsi “r” ini mode read only, maka apabila user mencoba untuk menulis di file, maka program akan menangkap Error input/output tersebut, dan menampilkan pesan yang sudah kita buat.

Error handling ini sangat berguna untuk mencegah program yang dibuat memberikan respon yang dapat mengeksplorasi bug pada program kita. Serta mencegah program kita untuk mengalami crash (membeku atau menutup dengan sendirinya).

Error handling dapat pula digunakan oleh developer (pengembang aplikasi) untuk memberikan label unik pada suatu prosedur. Sehingga label ini sangat berguna untuk debugging (debugging = proses mencari bug).

Rangkuman

Bab ini mempelajari mengenai aturan-aturan dari Syntax Pemrograman dengan bahasa Python. Dipelajari pula aturan-aturan dalam memberikan nama variabel.

Dimana di dalam dunia pemrograman, perlu mengetahui pola-pola penggunaan Variabel, Conditional Statement, Looping, Rekursif dan Error Handling.

Soal/Evaluasi

1. Tuliskan algoritma, pseudocode dan kode program untuk menentukan nilai maksimal dan minimal dari suatu himpunan yang terdiri dari 8 angka random.
2. Tuliskan algoritma, pseudocode dan kode program untuk menentukan bilangan ganjil atau bilangan genap.
3. Tuliskan algoritma, pseudocode dan kode program untuk menghitung nilai faktorial. Misalnya $3!=3 \times 2 \times 1=6$. $5!=5 \times 4 \times 3 \times 2 \times 1=120$.
4. Tuliskan kode program yang mengimplementasikan rumus fibonacci secara rekursif.

Bab 4

Algoritma dan Pemrograman Pencarian (Searching)

Tujuan Instruksional:

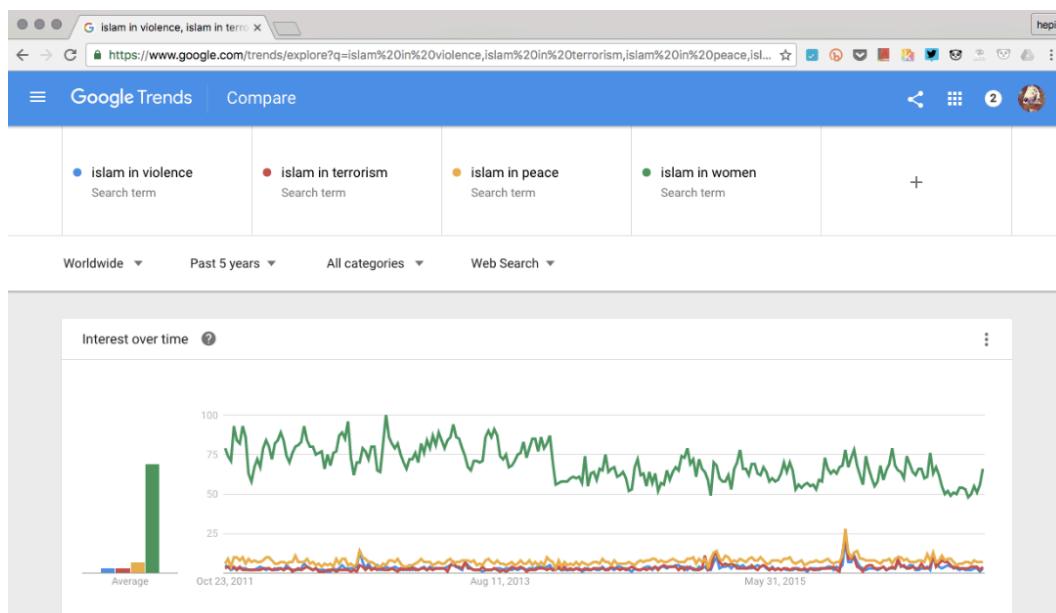
Pada bab ini, dijelaskan mengenai algoritma pencarian. Algoritma pencarian akan lebih cepat apabila data sudah terurutkan. Untuk itu, algoritma pencarian sangat berhubungan dengan algoritma pengurutan. Untuk algoritma pencarian, akan dijelaskan melalui Algoritma Sequential Search dan Binary Search.

Capaian Pembelajaran Mata Kuliah :

Mahasiswa diharapkan mampu memahami algoritma pencarian dengan menggambarkan langkah-langkah algoritma pencarian melalui sebuah flowchart dan menuliskan langkah tersebut menjadi sebuah Pseudocode.

Bab 4. Algoritma dan Pemrograman Pencarian (Searching)

Salah satu fitur vital dalam menggunakan sumber daya perangkat komputer ialah fitur pencarian. Pada era saat ini, fitur pencarian hampir terdapat dalam kegiatan sehari-hari. Seperti pengambilan uang tunai di mesin ATM. Mencari berita dengan kata kunci tertentu di mesin pencari seperti Duck-Duck Go, Google dan Yahoo. Atau hanya sekedar melihat status pertemanan maya di media sosial. Kegiatan tersebut dipastikan mengimplementasikan algoritma pencarian.



Gambar diatas menunjukkan kata kunci tertentu yang populer di mesin pencari Google dengan keyword “Islam in violence”, “Islam in terrorism”, “Islam in peace” dan “Islam in women”. Gambar tersebut menunjukkan bahwa pengguna internet lebih tertarik dengan kata kunci islam in peace dan islam in women dari pada islam in violence dan islam in terrorism. Tampilan grafis ini dapat dimunculkan karena adanya algoritma pencarian.

4.1. Pencarian (Searching)



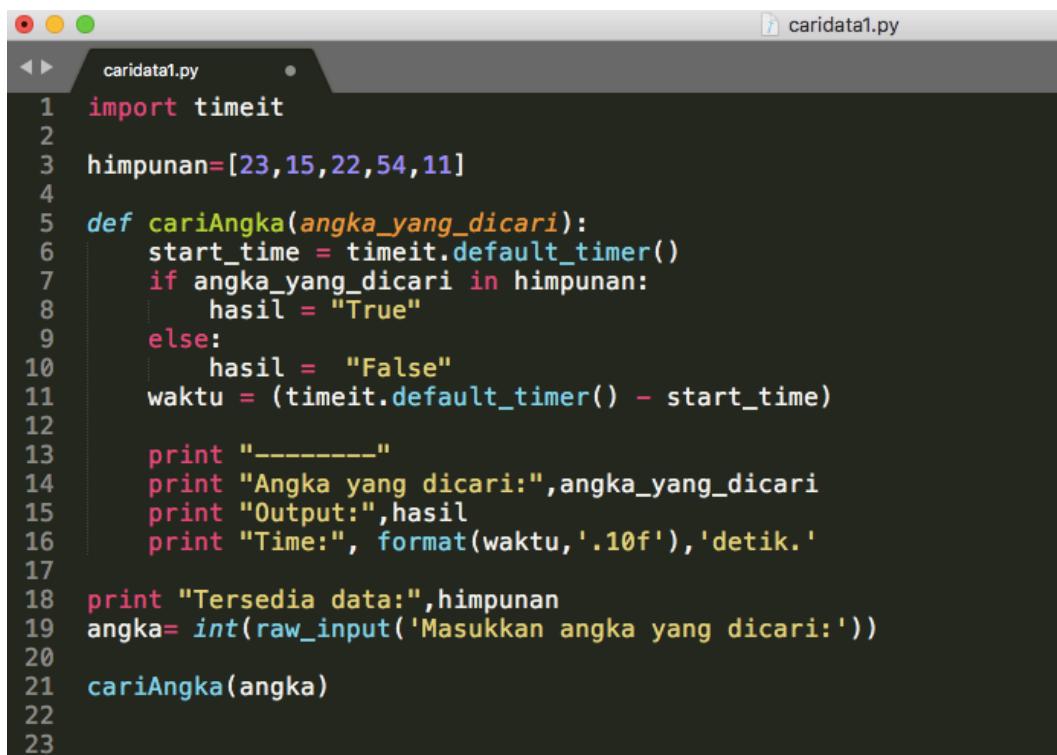
Silahkan buka Python intrepreter. Dan jalankan perintah seperti di gambar 4.1

Algoritma pencarian ialah algoritma yang digunakan untuk mencari data/nilai/angka tertentu dalam suatu himpunan. Hasil dari algoritma ini ialah nilai benar dan salah. Komputer akan memberikan output benar/True apabila data/nilai/angka yang dicari terdapat dalam himpunan tersebut. Sebaliknya komputer akan memberikan output False apabila data yang dicari tidak ada dalam himpunan tersebut. Contoh sederhana dalam Python untuk mencari suatu angka ialah:

```
>>> 19 in [23,15,22,54,11]
False
>>> 11 in [23,15,22,54,11]
True
>>>
```

Gambar 4.1. Eksekusi pencarian suatu angka pada suatu index.

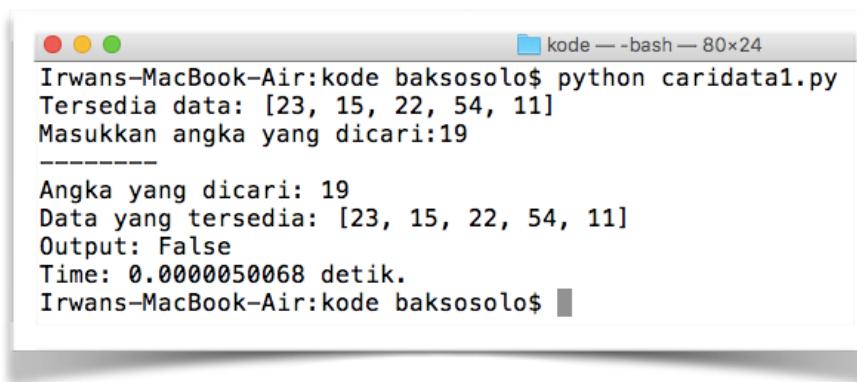
Contoh diatas ialah contoh sederhana sekali. Dikatakan sederhana karena tidak menggunakan algoritma dengan data yang tidak banyak. Sehingga eksekusi pencarian tersebut terlihat singkat. Untuk mengetahui waktu yang dibutuhkan untuk mengeksekusi pencarian di suatu himpunan yang terdiri dari angka [23,15,22,54,11], dapat ditulis dalam program sebagai berikut :



```
caridata1.py
1 import timeit
2
3 himpunan=[23,15,22,54,11]
4
5 def cariAngka(angka_yang_dicari):
6     start_time = timeit.default_timer()
7     if angka_yang_dicari in himpunan:
8         hasil = "True"
9     else:
10        hasil = "False"
11    waktu = (timeit.default_timer() - start_time)
12
13    print "-----"
14    print "Angka yang dicari:",angka_yang_dicari
15    print "Output:",hasil
16    print "Time:", format(waktu,'.10f'),'detik.'
17
18 print "Tersedia data:",himpunan
19 angka= int(raw_input('Masukkan angka yang dicari:'))
20
21 cariAngka(angka)
22
23
```

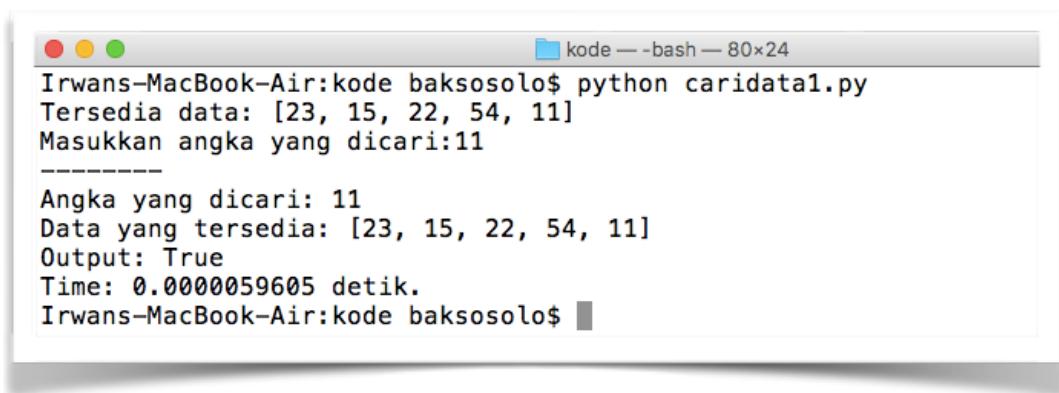
Gambar 4.2. Tangkapan layar kode caridata1.py

Ketika kode caridata1.py dijalankan, kita perlu memasukkan angka yang dicari seperti pada gambar 4.1. Pada saat mengeksekusi kode caridata1.py, diperoleh waktu 0,0000050068 detik untuk mencari angka 19 dan bernilai False (Gambar 4.3.A). Serta membutuhkan waktu 0,0000059605 detik untuk mencari angka 11 dan bernilai True (Gambar 4.3.B).



```
Irwans-MacBook-Air:kode baksosolo$ python caridata1.py
Tersedia data: [23, 15, 22, 54, 11]
Masukkan angka yang dicari:19
-----
Angka yang dicari: 19
Data yang tersedia: [23, 15, 22, 54, 11]
Output: False
Time: 0.0000050068 detik.
Irwans-MacBook-Air:kode baksosolo$
```

Gambar 4.3.A. Mencari angka 19 dan bernilai False.



```
Irwans-MacBook-Air:kode baksosolo$ python caridata1.py
Tersedia data: [23, 15, 22, 54, 11]
Masukkan angka yang dicari:11
-----
Angka yang dicari: 11
Data yang tersedia: [23, 15, 22, 54, 11]
Output: True
Time: 0.0000059605 detik.
Irwans-MacBook-Air:kode baksosolo$
```

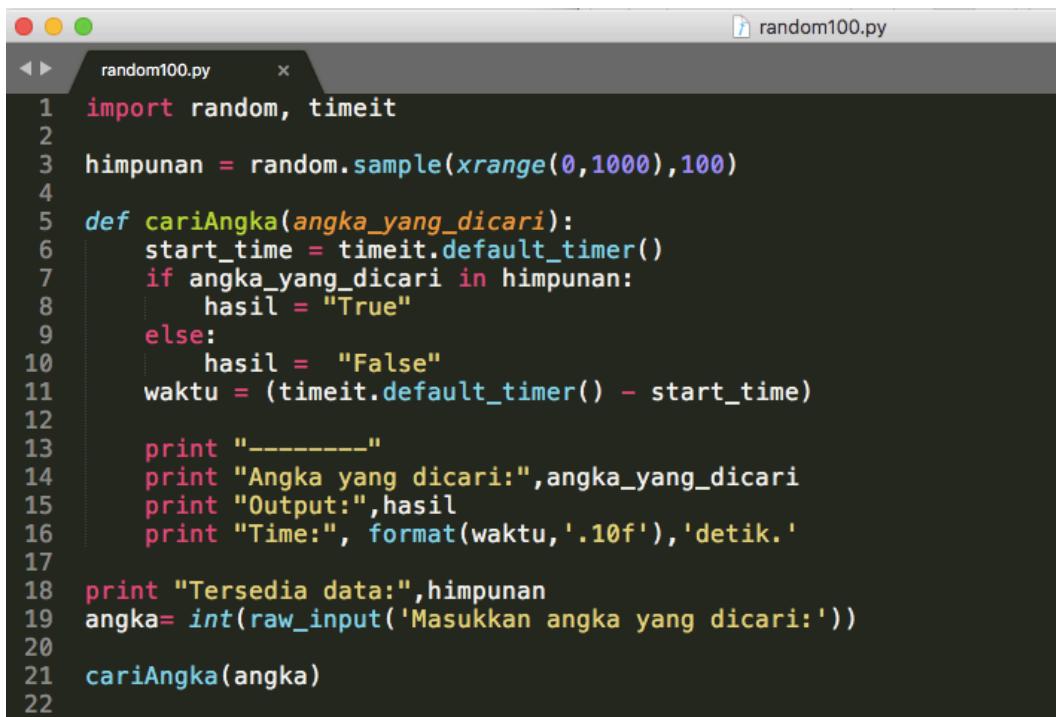
Gambar 4.3.B. Mencari angka 11 dan bernilai True.

Maka timbul pertanyaan. Jika mencari data tanpa algoritma saja begitu cepat, maka untuk apa algoritma? Pertanyaan ini dapat dijawab dengan menggunakan kode caridata1.py dengan mencari data dari suatu himpunan yang terdiri dari 100 data/angka. Namun kode caridata1.py perlu dimodifikasi dengan mendefinisikan variabel himpunan diisi oleh nilai acak/random. Untuk mendapatkan nilai random sebanyak 100 angka dan angka yang di random berkisar dari angka 0 hingga 100, serta tidak ada perulangan(angka yang dirandom hanya 1 kali munculnya), maka dapat menggunakan modul Python bernama **random** dengan cara memanggilnya sebagai berikut :

```
random.sample(xrange(angka_acak_awal,angka_acak_akhir),banyaknya_data)
```



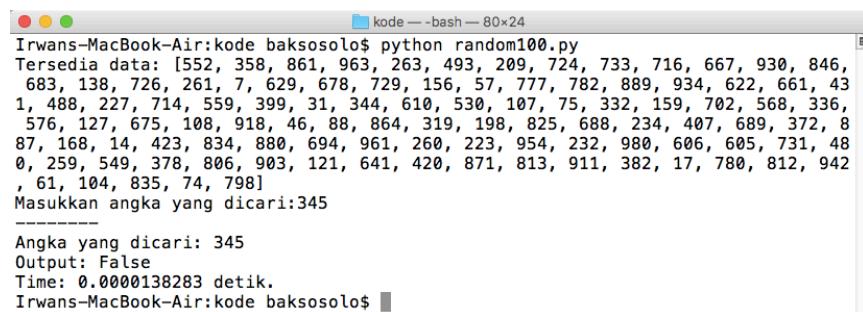
Silahkan buka text editor. Dan copy seluruh kode yang ada pada gambar 4.1. Pada baris ke 1, tambahkan modul random. Dan baris ke 3 variable himpunan dimodifikasi dengan membangkitkan 100 angka acak. Simpan dengan nama random100.py. Lihat gambar 4.4.



```
random100.py
1 import random, timeit
2
3 himpunan = random.sample(xrange(0,1000),100)
4
5 def cariAngka(angka_yang_dicari):
6     start_time = timeit.default_timer()
7     if angka_yang_dicari in himpunan:
8         hasil = "True"
9     else:
10        hasil = "False"
11    waktu = (timeit.default_timer() - start_time)
12
13    print "-----"
14    print "Angka yang dicari:",angka_yang_dicari
15    print "Output:",hasil
16    print "Time:", format(waktu,'.10f'),'detik.'
17
18 print "Tersedia data:",himpunan
19 angka= int(raw_input('Masukkan angka yang dicari:'))
20
21 cariAngka(angka)
22
```

Gambar 4.4. Tangkapan layar kode random100.py

Dengan menjalankan kode random100.py, dapat dilihat, untuk mencari suatu data pada sebuah himpunan dengan 100 angka random, diperlukan waktu yang lebih banyak yaitu 0,0000138283 (Gambar 4.5.). Walaupun yang kita rasakan masih terasa singkat, namun apabila dibandingkan dengan perbedaan waktu pencarian data pada Gambar 4.3A-B, Gambar 4.5, menunjukkan kenaikan waktu proses yang cukup signifikan.



```
Irwans-MacBook-Air:kode baksosolo$ python random100.py
Tersedia data: [552, 358, 861, 963, 263, 493, 209, 724, 733, 716, 667, 930, 846,
683, 138, 726, 261, 7, 629, 678, 729, 156, 57, 777, 782, 889, 934, 622, 661, 43
1, 488, 227, 714, 559, 399, 31, 344, 610, 530, 107, 75, 332, 159, 702, 568, 336,
576, 127, 675, 108, 918, 46, 88, 864, 319, 198, 825, 688, 234, 407, 689, 372, 8
87, 168, 14, 423, 834, 880, 694, 961, 260, 223, 954, 232, 980, 606, 605, 731, 48
0, 259, 549, 378, 806, 903, 121, 641, 420, 871, 813, 911, 382, 17, 780, 812, 942
, 61, 104, 835, 74, 798]
Masukkan angka yang dicari:345
-----
Angka yang dicari: 345
Output: False
Time: 0.0000138283 detik.
Irwans-MacBook-Air:kode baksosolo$
```

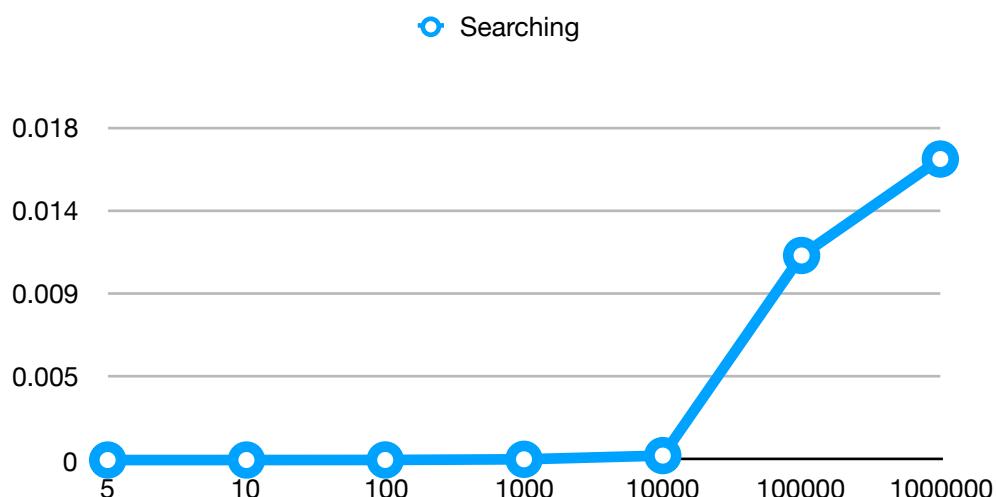
Gambar 4.5. Tangkapan layar hasil eksekusi file random100.py

Kenaikan waktu proses yang dibutuhkan dalam mencari suatu angka yang naik secara signifikan tersebut, mengikuti banyaknya jumlah data pada suatu himpunan. Setelah mencoba suatu nilai dari data yang berjumlah 5 hingga 1 juta data. Diperoleh waktu berikut :

Table 4.1. Waktu proses pencarian data dari 5 hingga 1 juta angka.

Jumlah Data	Waktu Proses (detik)
5	0.0000019073
10	0.000002861
100	0.0000059605
1.000	0.0000419617
10.000	0.0002419949
100.000	0.0110569
1.000.000	0.0162689686

Dan kemudian seperti yang ditunjukkan oleh grafis pada Gambar.



Gambar 4.6. Hubungan waktu proses dengan jumlah data yang dicari.

Grafis tersebut diperoleh setelah mencoba mencari angka dari suatu himpunan dengan nilai benar/salah pada jumlah elemen himpunan dari 5,10,100,1.000,10.000,100.000 dan 1.000.000 angka. Dari data tabel dan gambar tersebut dapat disimpulkan bahwa, makin banyak angka dari suatu himpunan maka akan semakin banyak waktu proses yang dibutuhkan. Disinilah letak peran dan fungsi sebuah algoritma. Yaitu mempercepat waktu proses jumlah data yang besar dengan mempercepat proses pencarian melalui langkah-langkah yang kreatif, terstruktur dan solutif. Disebut langkah-langkah kreatif karena membuat suatu algoritma seperti membuat karya desain.

4.2. Sequential Search

Sequential search ialah sebuah algoritma yang mencari nilai/angka dari sebuah himpunan(baca: array), berurutan mulai dari data pada urutan pertama hingga terakhir. Algoritmanya sebagai berikut :

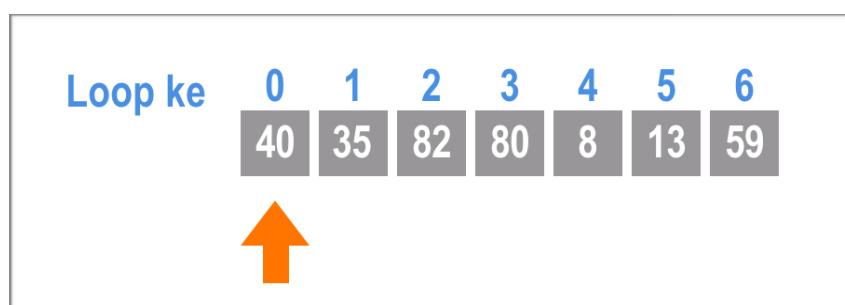
1. Membaca array/list himpunan.
2. Membaca inputan data yang dicari.
3. Perulangan dimulai dari posisi/index pertama hingga terakhir :
 - 3.1. Membandingkan data pada himpunan dengan data inputan yang dicari.
 - 3.2. Jika data yang dibandingkan sama, maka program memberikan respon True. Perulangan berhenti.
 - 3.3. Jika tidak ditemukan, dilanjutkan dengan perulangan berikutnya.
4. Output : status datanya bertemu True atau False.

Untuk Pseudocode dari algoritma diatas ialah:

```
Procedure Sequential-Search(H, i) :  
Input: H=Array data yang dicari  
Output: True/False  
hasil = False  
Loop 0 hingga panjang(himpunan)  
    IF himpunan[i] = x:  
        Hasil = True  
        Loop stop  
  
    ELSE himpunan[i] != x:  
        lanjutkan loop.
```

Show hasil

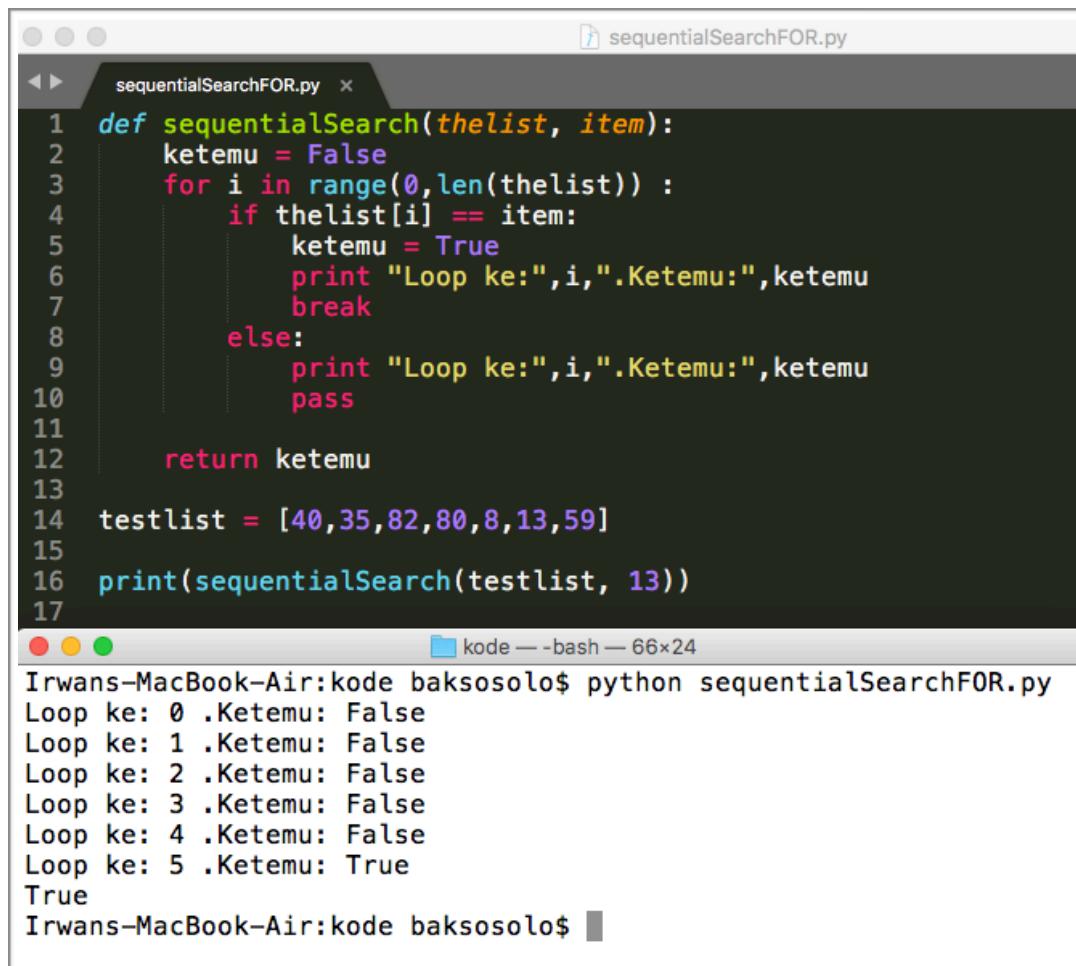
Jika terdapat suatu himpunan/array: [40,35,82,80,8,13,59] dan data yang dicari ialah angka 13, maka ilustrasi sequential searchnya dijelaskan pada gambar berikut :



Gambar 4.7. Sequential Search

Perulangan akan mencari dari loop pertama (indeks 0) hingga loop sebanyak data di himpunan. Namun karena data yang dicari adalah 13, maka perulangan

akan berhenti di loop indeks ke 5 dan memberikan status true. Perhatikan kode dan output berikut :



The screenshot shows a terminal window titled "sequentialSearchFOR.py". The code within the window is as follows:

```
sequentialSearchFOR.py
1 def sequentialSearch(thelist, item):
2     ketemu = False
3     for i in range(0,len(thelist)) :
4         if thelist[i] == item:
5             ketemu = True
6             print "Loop ke:",i,".Ketemu:",ketemu
7             break
8         else:
9             print "Loop ke:",i,".Ketemu:",ketemu
10            pass
11
12     return ketemu
13
14 testlist = [40,35,82,80,8,13,59]
15
16 print(sequentialSearch(testlist, 13))
17
```

Below the code, the terminal prompt is "Irwans-MacBook-Air:kode baksosolo\$". The output of the command "python sequentialSearchFOR.py" is displayed, showing the step-by-step search process:

```
Irwans-MacBook-Air:kode baksosolo$ python sequentialSearchFOR.py
Loop ke: 0 .Ketemu: False
Loop ke: 1 .Ketemu: False
Loop ke: 2 .Ketemu: False
Loop ke: 3 .Ketemu: False
Loop ke: 4 .Ketemu: False
Loop ke: 5 .Ketemu: True
True
Irwans-MacBook-Air:kode baksosolo$
```

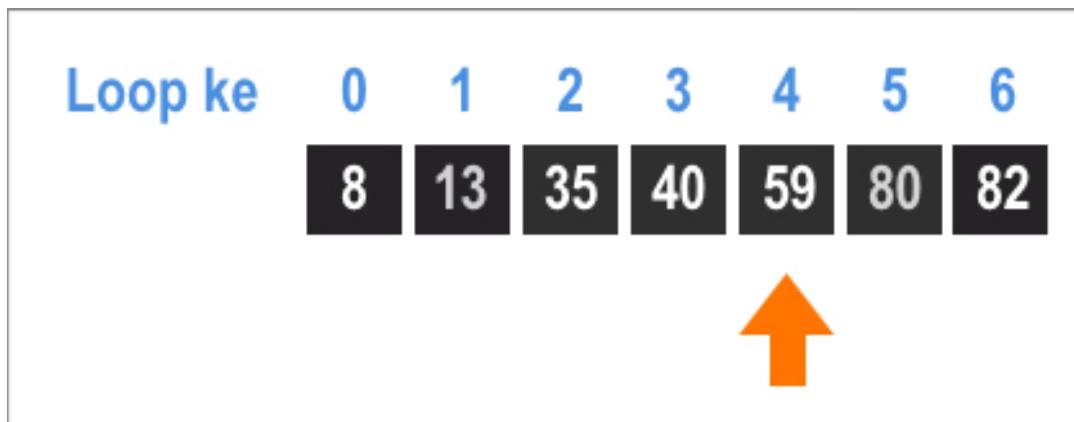
Gambar 4.8. Kode dan Ouput Sequential Search.

Yang perlu diingat dan menjadi ciri khas dari sequential search ialah kata-kata sequential itu sendiri. Arti sequential disini ialah urutan. Yaitu mencari data dengan mencari secara berurutan dari data pertama hingga terakhir.



Jika melihat dengan seksama yang dilakukan oleh sequential search, data awal yang dicari tidaklah berurutan. Sehingga untuk dinyatakan data TIDAK ADA/ False, maka algoritma harus mencari hingga akhir himpunan/array.

Keadaan ini berbeda apabila data dalam himpunan tersebut sudah terurutkan. Maka algoritma dari sequential search dapat diimprovisasi dengan berhenti looping apabila data yang dibandingkan tersebut lebih besar daripada angka yang dicari. Perhatikan gambar berikut:



Gambar 4.9. Sequential Search dengan sorted data.

Jika angka yang dicari di dalam himpunan ialah 43. Maka loop dapat berhenti pada loop ke 4 (Gambar 4.9). Karena jika data tersebut sudah terurut, maka bisa dipastikan angka 43 tidak akan berada di posisi setelah angka 59. Bukankah angka-angka setelah 59 pasti lebih besar dan atau sama dengan 59?

Untuk itu, algoritma searching akan lebih efektif (lebih cepat memberikan hasil) apabila mencari pada data yang telah terurutkan.

Untuk lebih memahami bagaimana algoritma pengurutan dalam suatu array, dijelaskan pada Bab. 5



4.3. Binary Search

Binary search hanya dapat bekerja pada array data yang sudah terurutkan. Prinsip kerja Binary search ialah mencari posisi tengah pada array tersebut. Apabila angka yang dicari lebih besar dari angka pada posisi tengah, maka proses pencarian dimulai dari posisi tengah tersebut hingga akhir. Namun sebaliknya apabila angka yang dicari kurang dari nilai tengah, maka proses pencarian ini dilakukan dari indeks ke 0 hingga posisi tengah tersebut. Untuk lebih jelasnya, perhatikan gambar berikut :



Gambar 4.10. Sequential Search dengan item lebih besar dari midpoint.

Pada gambar diatas (gambar 4.10), ditentukan dahulu posisi tengah sebuah array yang bisa kita sebut midpoint. Dalam gambar diatas, posisi midpoint ialah 40. Jika yang dicari ialah angka 43 dan karena 43 lebih besar dari 40, maka proses pencarian dimulai dari posisi setelah midpoint (40) hingga akhir himpunan/array. Namun sebaliknya, jika angka yang dicari ialah 13, maka proses pencarian

dimulai dari indeks awal hingga posisi midpoint (40). Seperti yang diilustrasikan pada gambar 4.11 dibawah ini.



Gambar 4.11. Sequential Search dengan angka kurang dari midpoint.

Kode program dari binary search untuk mencari angka 67 dapat ditulis seperti gambar berikut:

```
binarySearch.py
1 def binarySearch(thelist, item):
2     awal = 0
3     akhir = len(thelist)-1
4     ketemu = False
5
6     while awal<=akhir and not ketemu:
7         midpoint = (awal + akhir)/2
8
9         if thelist[midpoint] == item:
10            ketemu = True
11        else:
12            if item > thelist[midpoint]:
13                awal = midpoint+1
14            else:
15                akhir = midpoint-1
16
17    return ketemu
18
19 himpunan = [8,13,35,40,59,80,82]
20
21 print(binarySearch(himpunan, 67))
```

Gambar 4.12. Kode program Sequential Search.

Rangkuman

Bab ini mempelajari bagaimana kerja algoritma searching akan lebih efisien apabila data yang ada pada suatu array telah diurutkan. Untuk itu, algoritma searching akan dijalankan setelah sebuah data diurutkan dengan algoritma sorting. Disini pentingnya memahami dahulu algoritma searching karena yang nantinya dipakai oleh user ialah algoritma searching bukan algoritma sorting.

Soal/Evaluasi

1. Tuliskan algoritma dan pseudocode dari algoritma Sequential Search dan Binary Search.
2. Silahkan latihan dengan memodifikasi algoritma binary search menggunakan rekursif.

Bab 5

Algoritma dan Pemrograman Pengurutan (Sorting)



Tujuan Instruksional:

Pada bab ini, dijelaskan mengenai algoritma dasar yaitu algoritma pengurutan, atau yang dikenal dengan sorting algorithm. Macam algoritma yang akan dijelaskan yaitu: Bubble Sort, Insertion Sort, Selection Sort, Merge Sort dan Quick Sort.

Capaian Pembelajaran Mata Kuliah :

Mahasiswa diharapkan memahami algoritma pengurutan dengan mampu menggambarkan langkah-langkah algoritma pengurutan melalui sebuah flowchart dan menulis langkah tersebut menjadi sebuah Pseudocode.

Bab 5. Algoritma dan Pemrograman Pengurutan (Sorting)

5.1. Bubble Sort

Bubble sort ialah algoritma pengurutan yang paling sederhana dan mudah untuk dipelajari. Namun, bubble sort memakan waktu proses yang paling lama dibandingkan dengan algoritma sorting lainnya. Karena algoritma bubble sort membandingkan satu per satu value dalam suatu himpunan ke dalam satu fase perbandingan. Jika ada n item dalam sebuah himpunan, maka dalam satu fase perbandingan akan dilakukan $n-1$ proses. Jika ada himpunan angka [40, 35, 82, 80, 8, 13, 59] dimana dalam himpunan tersebut terdapat 7 angka, maka dalam 1 fase perbandingan, dilakukan $(n-1)$ yaitu 6 kali perbandingan.

Apakah yang dibandingkan? Jawabannya ialah membandingkan dua angka dimana posisi angka yang paling besar dipindah sebelah kanan dan posisi angka yang lebih kecil dipindah sebelah kiri. Misalnya angka yang berada pada indeks ke nol dengan angka yang berada pada indeks ke 1 dalam himpunan [40, 35, 82, 80, 8, 13, 59] yaitu 40 dan 35. Dapat dilihat bahwa angka 40 lebih besar daripada 35, maka posisi angka 40 akan bertukar dengan posisi angka 35. Satu fase ini akan berhenti apabila angka yang terbesar sudah ada di posisi/urutan paling belakang. Untuk lebih jelasnya, 1 fase perbandingan dapat di ilustrasikan dalam gambar berikut :



Gambar 5.1. Satu fase perbandingan dalam Buble Sort.

Ciri khas dalam proses pengurutan data menggunakan algoritma Bubble Sort ialah seperti prinsip Bubble(busa sabun) yaitu angka yang paling besar, diposisikan paling atas/paling kanan. Hal ini dianalogikan seperti busa sabun/ gelembung udara yang didalam air posisinya akan selalu berada diatas.

Jumlah looping dalam 1 fase serta jumlah fase itu sendiri tergantung dari jumlah elemen data yang diurutkan. Contoh pada gambar diatas, terdapat 7 elemen data. Maka dalam 1 fase perbandingan, terdapat 6 kali looping. Dimana hal ini diperoleh dari $n-1=7-1=6$ looping.

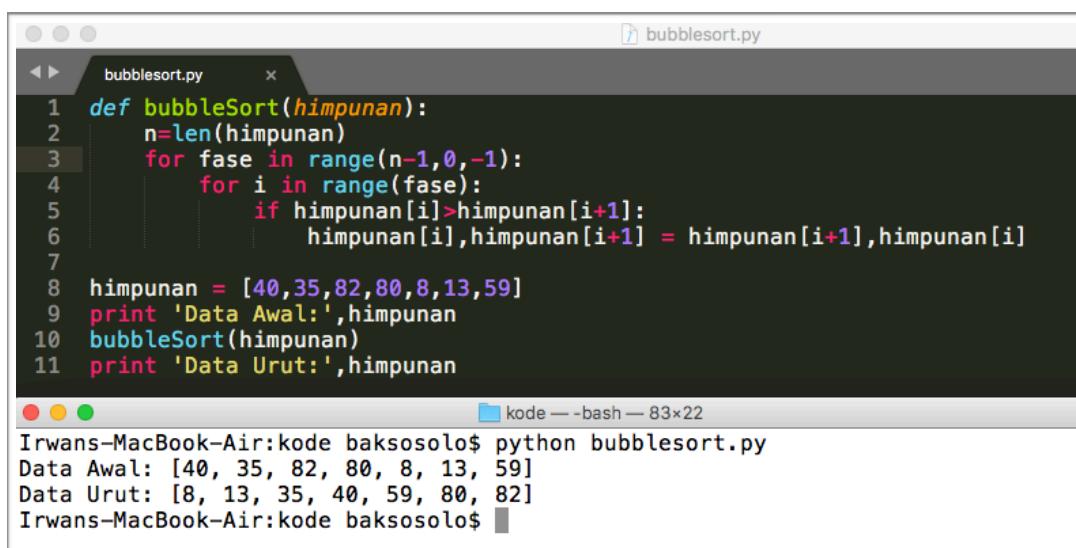
Algoritma bubble sort dapat ditulis sebagai berikut :

1. Mendefinisikan jumlah fase, yaitu dari fase 1 sampai n. Dimana n merupakan jumlah elemen dalam himpunan.
2. Tiap fase membandingkan dua bilangan pada urutan pertama dan kedua dan seterusnya, hingga angka yang paling besar berada pada posisi paling akhir.
3. Jika angka yang disebelah kiri lebih besar dari angka yang di sebelah kanan, maka dilakukan pertukaran posisi (swap). Jika sebaliknya, maka tidak ada pertukaran dan perulangan dilanjutkan ke fase berikutnya.
4. Melakukan perbandingan secara berulang pada langkah ke 2, dari urutan pertama hingga urutan sebelum angka yang terbesar.
5. Berhenti melakukan perulangan dan perbandingan apabila sudah tidak ada angka yang perlu di swap dalam satu fase. Selanjutnya, diulangi dari langkah ke 2.

Pseudocode bubble sort algorithm.

```
procedure bubbleSort (H)
Input: H = Array sebuah himpunan
n = length (A)
repeat
    swapped = false
    for i = 0 to n-1 inclusive do
        if A[i] > A[i+1] then
            swap( A[i+1], A[i] )
            swapped = true
        end if
    end for
    until not swapped
end procedure
```

Dari Pseudocode tersebut, kita bisa melihat bahwa untuk mengimplementasikan bubble sort, dibutuhkan 2 buah perulangan. Perulangan untuk mencari angka terbesar dan perulangan untuk melakukan swap/pertukaran posisi apabila ditemukan angka terbesar. Dimana kode program dan tampilan algoritma Bubble Sort ketika dieksekusi sebagai berikut :



```
bubblesort.py
1 def bubbleSort(himpunan):
2     n=len(himpunan)
3     for fase in range(n-1,0,-1):
4         for i in range(fase):
5             if himpunan[i]>himpunan[i+1]:
6                 himpunan[i],himpunan[i+1] = himpunan[i+1],himpunan[i]
7
8 himpunan = [40,35,82,80,8,13,59]
9 print 'Data Awal:',himpunan
10 bubbleSort(himpunan)
11 print 'Data Urut:',himpunan
```

Irwans-MacBook-Air:kode baksosolo\$ python bubblesort.py
Data Awal: [40, 35, 82, 80, 8, 13, 59]
Data Urut: [8, 13, 35, 40, 59, 80, 82]
Irwans-MacBook-Air:kode baksosolo\$

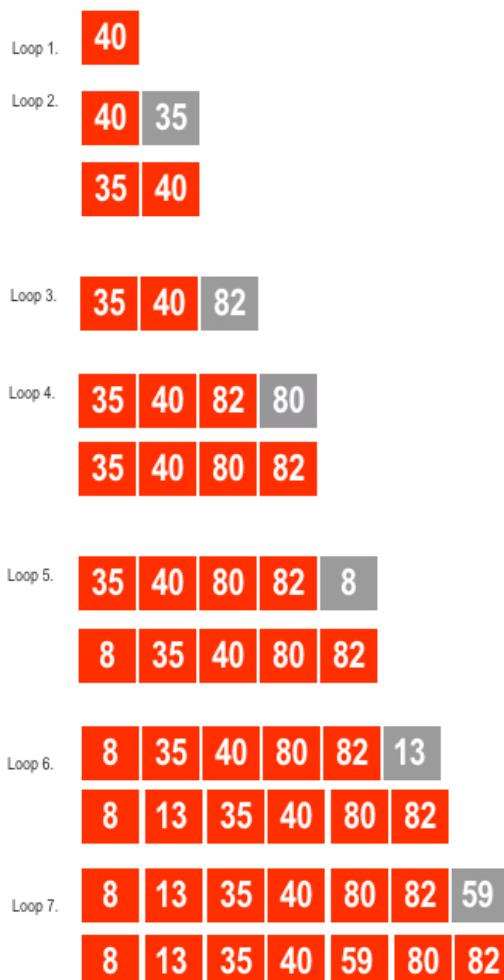
Gambar 5.2. Kode program dan output script Bubble Sort.

5.2. Insertion Sort

Insertion Sort merupakan algoritma pengurutan yang mengurutkan data dengan mengambil 1 data pada array, kemudian menambahkan (insert) data pada indeks selanjutnya, lalu diurutkan. Jika terdapat himpunan/array [40, 35, 82, 80, 8, 13, 59] maka data diurutkan dengan mengambil angka pada indeks pertama. Dan kemudian diurutkan. Penjelasan untuk tiap-tiap loopnya sebagai berikut :

Loop 1 : [40] diasumsikan sebagai sebuah array baru dan sudah terurutkan.

Loop 2: Menambahkan 35. Sehingga terdapat array baru [40,35]. Dan kemudian diurutkan. Sehingga hasil akhir loop 1 ialah array baru yang sudah terurutkan [35,40]. Dan dilanjutkan pada loop selanjutnya dengan menambahkan elemen baru dan kemudian diurutkan. Penggunaan Loop pada Insertion Sort di ilustrasikan pada gambar dibawah ini.



Gambar 5.3. Loop pada Insertion Sort.

Untuk kode program dan algoritma Insertion Sort saat dieksekusi ditunjukkan gambar dibawah ini:

The screenshot shows a terminal window with two tabs: 'insertionSort.py' and 'kode — bash — 83x22'. The 'insertionSort.py' tab contains the following Python code:

```
1 def insertionSort(thelist):
2     for i in range(1,len(thelist)):
3         valueInserted = thelist[i]
4         position = i
5
6         while position>0 and thelist[position-1]>valueInserted:
7             thelist[position] = thelist[position-1]
8             position = position-1
9
10        thelist[position] = valueInserted
11
12
13 himpunan = [40,35,82,80,8,13,59]
14 print 'Data Awal:',himpunan
15 insertionSort(himpunan)
16 print 'Data Urut:',himpunan
```

The 'kode — bash — 83x22' tab shows the output of running the script:

```
Irwans-MacBook-Air:kode baksosolo$ python insertionSort.py
Data Awal: [40, 35, 82, 80, 8, 13, 59]
Data Urut: [8, 13, 35, 40, 59, 80, 82]
Irwans-MacBook-Air:kode baksosolo$
```

Gambar 5.4. Kode program dan output pada Insertion Sort.

5.3. Selection Sort

Agar mudah mengingat atau memahami algoritma Selection Sort, kita perlu perhatikan nama dari sort ini. Yaitu “Selection”. Dinamakan Selection Sort karena untuk mengurutkan sesuatu yang paling besar, kita perlu memilih (select) item pada awal indeks/urutan, dimana kemudian item tersebut kita anggap mempunyai nilai yang paling besar. Selanjutnya seperti pada bubble sort, item satu dan lainnya dibandingkan dengan mencari nilai yang lebih besar. Jika terdapat item yang lebih besar dari item yang dianggap paling besar tadi, maka indeks posisi item yang lebih besar ditukar dengan posisi nilai item yang dibandingkan tersebut.

Yang membedakan Selection Sort dan Bubble Sort ialah pada algoritma Selection Sort yang di swap ialah posisi angka yang paling besar. Namun jika perulangan sudah pada item posisi terakhir, maka item yang ada pada posisi terakhir akan di swap dengan item yang dianggap paling besar. Sehingga dalam satu fase perbandingan, angka yang terbesar akan berada sisi paling kanan/ posisi yang paling akhir. Ilustrasi dari selection sort untuk mengurutkan data [40,35,82,80,8,13,59] dijelaskan pada gambar 5.5.

Selection Sort : Fase ke 1.

Satu fase perbandingan						
40	35	82	80	8	13	59
40	35	82	80	8	13	59
40	35	82	80	8	13	59
40	35	82	80	8	13	59
40	35	82	80	8	13	59
40	35	82	80	8	13	59

Loop 1.
Loop ke 2. $82 > 40 = \text{True}$.
Angka Terbesar saat ini menjadi 82.
Loop ke 3. $80 > 82 = \text{False}$.
Angka Terbesar tetap pada angka 82.
Loop ke 4. $8 > 82 = \text{False}$.
Angka Terbesar tetap pada angka 82.
Loop ke 5. $82 > 13 = \text{True}$.
Angka Terbesar tetap pada angka 82.
Loop ke 6. $82 > 59 = \text{True}$.
Angka Terbesar tetap pada angka 82.

Gambar 5.5. Fase pertama dari Selection Sort.

Gambar diatas menunjukkan satu fase perbandingan dengan jumlah 6 loop tiap 1 fase perbandingan. Output dari satu fase perbandingan ialah angka yang terbesar berada di posisi paling kanan. Fase pertama dalam selection sort dijelaskan secara detail sebagai berikut :

Fase 1. Max = himpunan[0] = 40

Loop 1: $35 > 40 = \text{FALSE}$

Dipilih angka dengan indeks posisi = 0, dianggap paling besar. Dalam hal ini, 40 dianggap paling besar. Kemudian dibandingkan dengan item berikutnya. Dalam hal ini komputer menilai apakah 35 lebih besar dari pada 40? Karena hasil perbandingan ini bernilai False, maka angka 40 dianggap yang paling besar. Dan tidak perlu tukar posisi antara angka 40 dan 35.

Loop 2. $82 > 40 = \text{True}$

Berpindah pada item berikutnya yaitu angka 80. Dalam hal ini, $80 > 40$ bernilai True. Sehingga acuan angka yang dianggap paling besar tidak lagi 40. Namun angka yang dianggap maksimum berubah dari 40 menjadi 80. Pada loop 2 ini, posisi angka 40 dan 80 tidak diperlukan swap. Namun perbandingan angka terbesar tidak lagi menggunakan 40 tetapi menggunakan 80.

Loop 3. $80 > 82 = \text{False}$. Tidak diperlukan swap, nilai maksimal tetap.

Loop 4. $8 > 82 = \text{False}$. Tidak diperlukan swap, nilai maksimal tetap.

Loop 5. $13 > 82$. False. Tidak diperlukan swap, nilai maksimal tetap.

Loop 6. $59 > 82$. False. nilai maksimal tetap.

Setelah loop ke 6. Posisi nilai maksimal ditukar dengan indeks yang paling akhir. Dalam hal ini yaitu 82 sebagai maks, diswap dengan 59. Sehingga nilai maks sudah berada di indeks paling belakang. Di ilustrasikan pada gambar berikut :



Gambar 5.6. Array sementara dalam fase ke 1 algoritma Selection Sort.

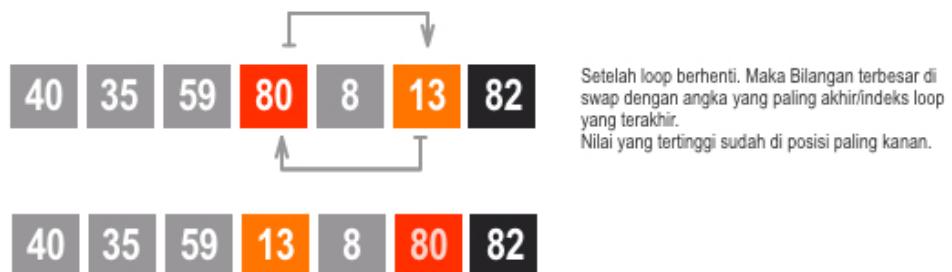
Selection Sort : Fase ke 2.



Gambar 5.7. Fase ke 2 algoritma Selection Sort.

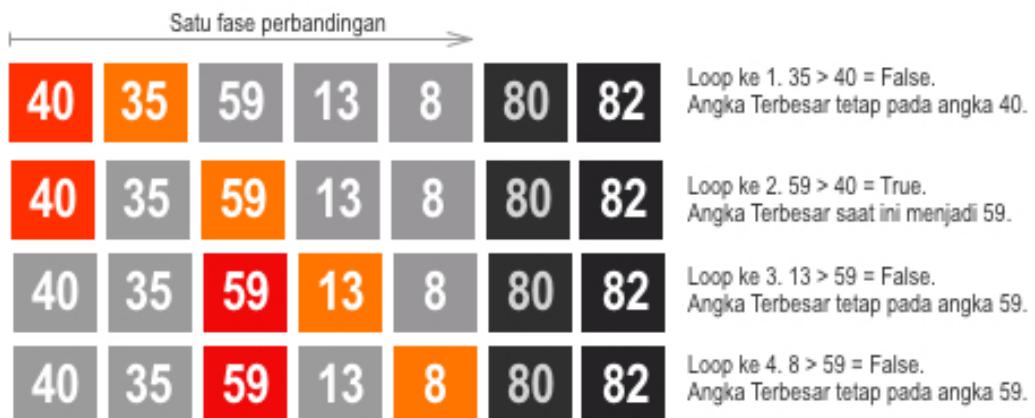
Proses diatas tadi belum selesai. Seperti dalam gambar Gambar 5.6, fase pertama dari selection sort memiliki himpunan/array sementara dengan deretan berikut : [40,35,59,80,8,13,82]. Dimana array ini masih belum urut. Untuk itu dalam 1 fase perbandingan selanjutnya, dilakukan lagi looping, namun dari indeks awal

hingga indeks n (jumlah data) -1 (gambar 5.7). Yaitu dari angka 40 hingga 13 (gambar 5.6). Karena angka 82 yang berada di indeks paling belakang merupakan item dengan nilai maksimal/terbesar. Tidak perlu lagi di swap dengan item lainnya. Setelah loop ke 5. Nilai maksimal (80) di swap posisinya dengan angka 13.



Gambar 5.8. Array sementara dalam fase ke 2 algoritma Selection Sort.

Selection Sort : Fase ke 3.



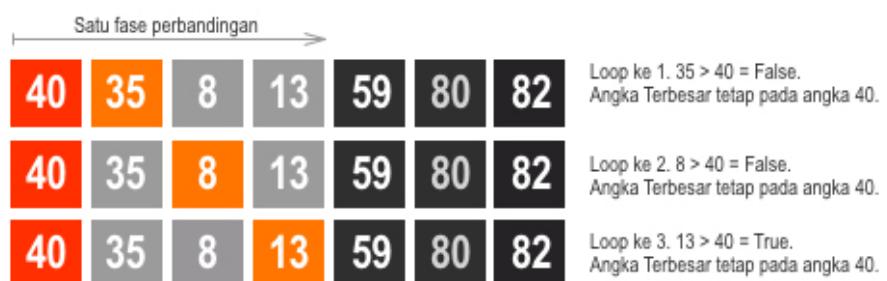
Gambar 5.9. Fase ke 3 algoritma Selection Sort.

Setelah dilakukannya Loop ke 4. Maka angka maksimal di swap dengan angka di indeks terakhir.



Gambar 5.10. Array sementara dalam fase ke 3 algoritma Selection Sort.

Selection Sort : Fase ke 4.



Gambar 5.11. Array sementara dalam fase ke 4 algoritma Selection Sort.

Dari hasil array sementara pada fase ke 4. Nilai awal berubah menjadi 13. Dan pada fase selanjutnya, angka yang dianggap maksimal tidak lagi 40, melainkan 13.

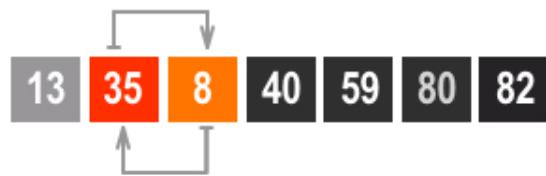
Selection Sort : Fase ke 5 dan ke 6.

Satu fase perbandingan

13	35	8	40	59	80	82
13	35	8	40	59	80	82

Loop ke 1. $35 > 13 = \text{True}$.
Angka Terbesar pada angka 35.

Loop ke 2. $8 > 35 = \text{False}$.
Angka Terbesar saat ini tetap pada angka 35.



Setelah loop berhenti. Maka Bilangan terbesar di swap dengan angka yang paling akhir/index loop yang terakhir.

13	8	35	40	59	80	82
----	---	----	----	----	----	----

Nilai yang tertinggi sudah di posisi paling kanan.

Satu fase perbandingan

13	8	35	40	59	80	82
----	---	----	----	----	----	----

Loop ke 1. $35 > 40 = \text{False}$.
Angka Terbesar tetap pada angka 40.



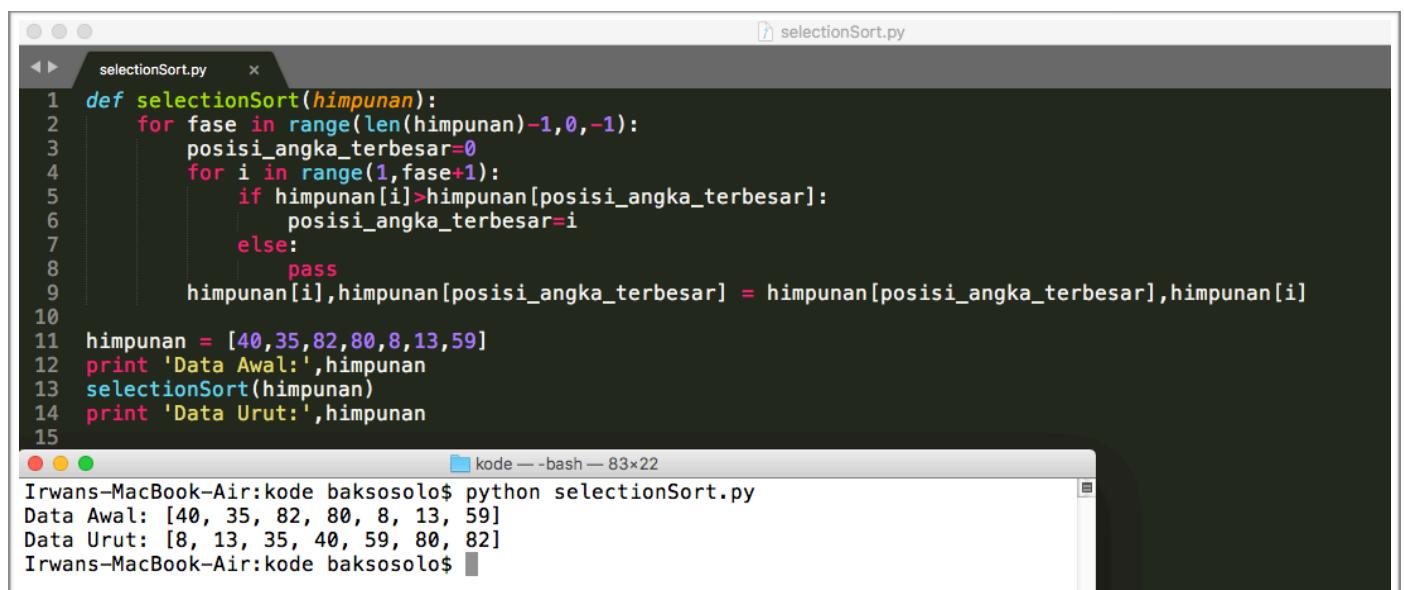
Setelah loop berhenti. Maka Bilangan terbesar di swap dengan angka yang paling akhir/index loop yang terakhir.

8	13	35	40	59	80	82
---	----	----	----	----	----	----

Nilai yang tertinggi sudah di posisi paling kanan.

Gambar 5.12. Array final dalam algoritma Selection Sort.

Hasil akhir array yang sudah tersorting ditunjukkan setelah proses swap antara angka 13 dan 8 pada gambar 5.12. Sedangkan untuk kode program yang mengimplementasikan Selection Sort dan mengurutkan array [40,35,59,80,8,13,82] ditampilkan pada gambar berikut :



The screenshot shows a Mac OS X desktop environment. In the foreground, a terminal window titled 'kode — bash — 83x22' displays the output of a Python script. The script's name is 'selectionSort.py'. The terminal output shows:

```
Irwans-MacBook-Air:kode baksosolo$ python selectionSort.py
Data Awal: [40, 35, 82, 80, 8, 13, 59]
Data Urut: [8, 13, 35, 40, 59, 80, 82]
Irwans-MacBook-Air:kode baksosolo$
```

In the background, a code editor window titled 'selectionSort.py' shows the Python code for the Selection Sort algorithm. The code defines a function 'selectionSort' that takes a list 'himpunan' as input. It iterates through the list from the last element to the first. For each element, it finds the index of the maximum value in the remaining unsorted portion of the list and swaps it with the current element.

```
selectionSort.py
1 def selectionSort(himpunan):
2     for fase in range(len(himpunan)-1,0,-1):
3         posisi_angka_terbesar=0
4         for i in range(1,fase+1):
5             if himpunan[i]>himpunan[posisi_angka_terbesar]:
6                 posisi_angka_terbesar=i
7             else:
8                 pass
9         himpunan[i],himpunan[posisi_angka_terbesar] = himpunan[posisi_angka_terbesar],himpunan[i]
10
11 himpunan = [40,35,82,80,8,13,59]
12 print 'Data Awal:',himpunan
13 selectionSort(himpunan)
14 print 'Data Urut:',himpunan
15
```

Gambar 5.13.Kode program algoritma Selection Sort.

5.4. Merge Sort

Kunci dari Merge Sort ini ialah strategi divide, conquer dan combine. Masih ingat pelajaran sejarah waktu di sekolah dasar? Masih ingat politik penjajah dalam merusak keamanan negeri ini? Penjajah menggunakan politik pecah belah(adu domba). Hal ini mirip dengan algoritma Merge Sort. Merge Sort membagi elemen-elemen dalam himpunan bilangan yang akan diurutkan menjadi bagian-bagian kecil. Kita mulai belajar dengan array data yang kecil. Misalnya $A = [2,4,3,1]$. Maka array A perlu dipecah menjadi dua dan hingga terkecil.



Gambar 5.14. Ilustrasi algoritma Merge Sort.

Divide, Conquer dan Combine, maka untuk implementasi Merge Sort, dibutuhkan algoritma untuk memecah(divide), mengurutkan(conquer) serta menggabungkan array yang dipecah tadi menjadi suatu array baru(combine). Disini diperlukan metode yang berfungsi sebagai divide, conquer dan combine

dari Merge Sort itu sendiri. Untuk kode program yang menunjukkan ketiga bagian tersebut dalam implementasi algoritma Merge Sort pada array [40,35,59,80,8,13,82] ditunjukkan melalui gambar berikut ini :

The screenshot shows a terminal window with two parts. The top part is a code editor with the file 'easymergeSort.py' open, displaying Python code for Merge Sort. The bottom part is a terminal window titled 'kode — bash — 83x22' showing the execution of the script with input and output.

```
 1 import sys
 2
 3 def divide(thelist, first, last):
 4     if first < last:
 5         middle = (first + last)//2
 6         divide(thelist, first, middle)
 7         divide(thelist, middle+1, last)
 8         combine(thelist, first, middle, last)
 9
10 def combine(thelist, first, middle, last):
11     L = thelist[first:middle+1]
12     R = thelist[middle+1:last+1]
13     L.append(sys.maxsize)
14     R.append(sys.maxsize)
15     i = j = 0
16
17     for k in range (first, last+1):
18         if L[i] <= R[j]:
19             thelist[k] = L[i]
20             i += 1
21         else:
22             thelist[k] = R[j]
23             j += 1
24
25
26 def mergeSort(thelist):
27     divide(thelist, 0, len(thelist)-1)
28
29 himpunan = [40,35,82,80,8,13,59]
30 print 'Data Awal:',himpunan
31 mergeSort(himpunan)
32 print 'Data Urut:',himpunan
```

Irwans-MacBook-Air:kode baksosolo\$ python easymergeSort.py
Data Awal: [40, 35, 82, 80, 8, 13, 59]
Data Urut: [8, 13, 35, 40, 59, 80, 82]
Irwans-MacBook-Air:kode baksosolo\$

Gambar 5.15. Kode program algoritma Merge Sort.

5.5. Quick Sort

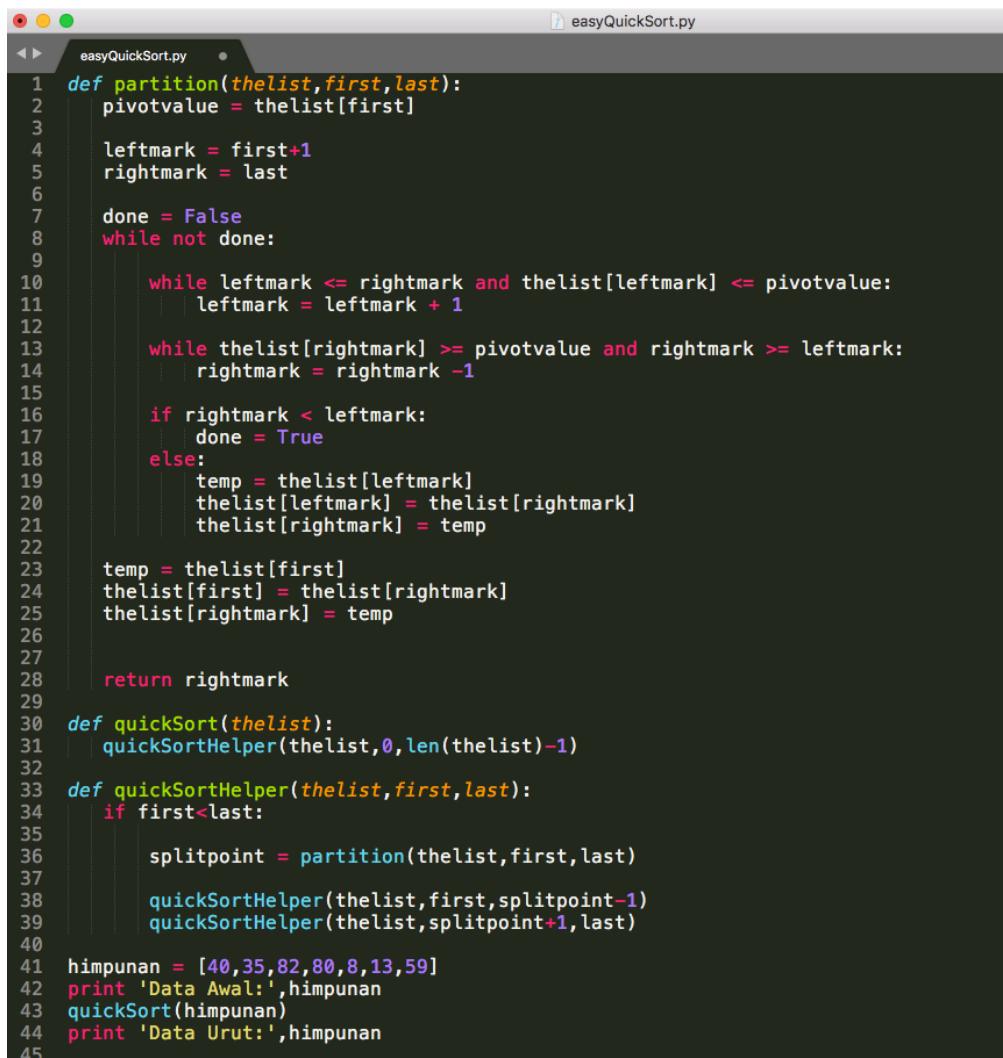
Quick Sort menerapkan prinsip yang sama dengan Merge Sort. Yaitu Divide, Conquer dan Combine. Namun, Quick Sort tidak membuat array baru seperti Merge Sort. Namun, membagi array setelah secara random memilih sebuah data di Array tersebut secara acak, kemudian dijadikan pivot.

Setelah memilih pivot, Quick Sort melakukan divide, dengan menukar posisi data di elemen, dengan yang lebih kecil dari pivot berada di sebelah kiri, yang lebih besar di sebelah kanan pivot. Selanjutnya di Conquer(diurutkan) masing-masing array sebelum pivot, dan setelah pivot. Seanjutnya di combine (Array tersebut digabungkan). QuickSort dapat diilustrasikan oleh gambar berikut:



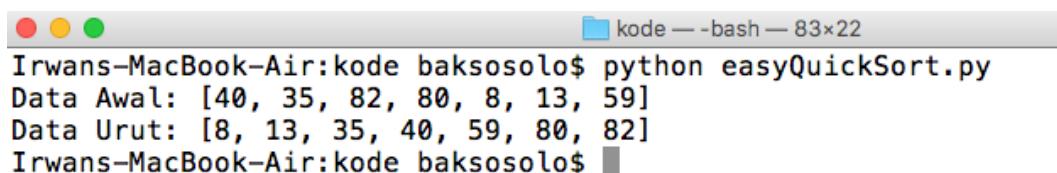
Gambar 5.16. Ilustrasi algoritma Quick Sort

Sedangkan untuk kode programmnya sebagai berikut.



```
easyQuickSort.py
```

```
1 def partition(thelist,first,last):
2     pivotvalue = thelist[first]
3
4     leftmark = first+1
5     rightmark = last
6
7     done = False
8     while not done:
9
10         while leftmark <= rightmark and thelist[leftmark] <= pivotvalue:
11             leftmark = leftmark + 1
12
13         while thelist[rightmark] >= pivotvalue and rightmark >= leftmark:
14             rightmark = rightmark -1
15
16         if rightmark < leftmark:
17             done = True
18         else:
19             temp = thelist[leftmark]
20             thelist[leftmark] = thelist[rightmark]
21             thelist[rightmark] = temp
22
23     temp = thelist[first]
24     thelist[first] = thelist[rightmark]
25     thelist[rightmark] = temp
26
27
28     return rightmark
29
30 def quickSort(thelist):
31     quickSortHelper(thelist,0,len(thelist)-1)
32
33 def quickSortHelper(thelist,first,last):
34     if first<last:
35
36         splitpoint = partition(thelist,first,last)
37
38         quickSortHelper(thelist,first,splitpoint-1)
39         quickSortHelper(thelist,splitpoint+1,last)
40
41 himpunan = [40,35,82,80,8,13,59]
42 print 'Data Awal:',himpunan
43 quickSort(himpunan)
44 print 'Data Urut:',himpunan
45
```



```
Irwans-MacBook-Air:kode baksosolo$ python easyQuickSort.py
Data Awal: [40, 35, 82, 80, 8, 13, 59]
Data Urut: [8, 13, 35, 40, 59, 80, 82]
Irwans-MacBook-Air:kode baksosolo$
```

Gambar 5.17. Kode program dan ouput dari implementasi algoritma Quick Sort

Bab 6

Optimasi Algoritma dan Pemrograman



Tujuan Instruksional:

Pada bab ini, dijelaskan mengenai beberapa ulasan mengenai optimasi mendesain sebuah algoritma dan mengimplementasi algoritma tersebut kepada sebuah pemrograman

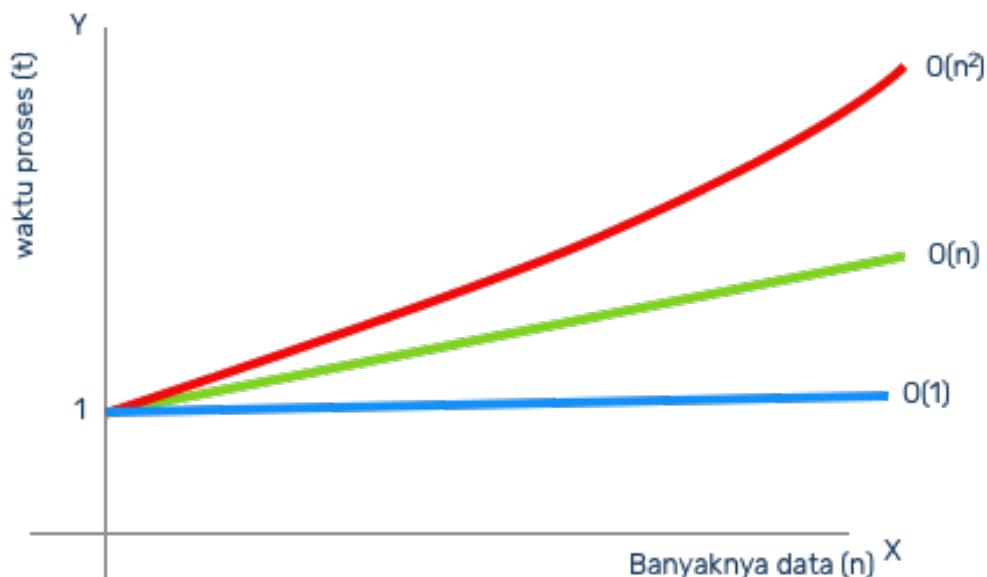
Capaian Pembelajaran Mata Kuliah :

Mahasiswa diharapkan memahami bagaimana mengoptimasi sebuah algoritma pencarian dan pengurutan serta mampu melihat perkembangan penelitian dalam ranah algoritma dan pemrograman

Bab 6. Optimasi Algoritma dan Pemrograman

6.1. Optimasi Algoritma dan Notasi Big O

Algoritma dapat dioptimasi dengan mempercepat waktu prosesnya. Makin besar data yang diolah, makin cepat proses perhitungannya dan makin cepat menampilkan informasi maka makin efektif algoritma tersebut. Efektifitas algoritma selalu dibandingkan dengan jumlah data (bisa disimbolkan sumbu X) dan waktu prosesnya (disimbolkan sumbu Y).



Gambar 6.1. Macam Grafis Notasi big O.

Untuk mengestimasi waktu proses dari suatu algoritma, digunakan notasi O (baca: big-oh) dan notasi O bukan digunakan untuk mengukur secara tepat waktu proses suatu algoritma. Optimasi yang paling dicari dan ideal ialah $O(1)$, karena data makin besar, waktu proses eksekusi tetap (warna biru).

Optimasi yang masih dapat diterima (acceptable) ialah Jika waktu proses dinotasikan t untuk memproses dataset (yang dinotasikan) n , maka waktu proses yang dibutuhkan untuk memproses satu satuan “luas” dataset diperkirakan memakan waktu :

$$t(n) = O(n^2)$$

Maka waktu yang dibutuhkan untuk mengeksekusi sebuah dataset yang naik sepuluh kali lipat (1,10,100,1000, dan seterusnya) dataset, maka waktu eksekusinya meningkat 100 kali lipat. Maka dalam garis berwarna merah pada gambar menunjukkan bahwa, makin besar data yang diproses, maka makin butuh waktu proses.

Lalu yang menjadi pertanyaan. Kenapa algoritma perlu untuk di optimasi? Jawabnya agar efisiensi pada proses perhitungan. Apabila algoritma tersebut dapat memberikan proses perhitungan yang cepat, namun membutuhkan resource (baca:spesifikasi komputer yang high end), maka algoritma tersebut menjadi sulit untuk diimplementasikan di sebuah/single mesin web server. (untuk kasus ini, muncullah peran Grid Computing yang mendistribusikan resource kepada beberapa komputer untuk memproses suatu algoritma).

Terlebih apabila algoritma tersebut dibutuhkan untuk kebutuhan sehari-hari, seperti melakukan enkripsi data pada paket data yang dikirim oleh browser ketika kita melakukan transaksi online. Untuk itu masih terbuka peluang penelitian untuk bagaimana mengoptimasikan sebuah algoritma.

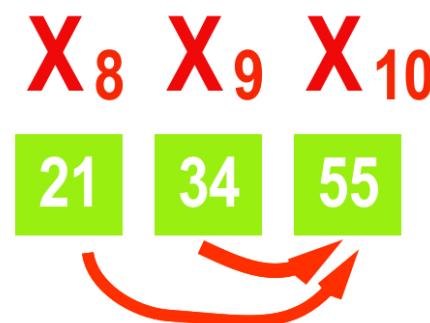
6.2. Optimasi Pemrograman

Selain algoritma, bagaimana mengimplementasikan algoritma tersebut juga perlu untuk di optimasi. Sebagai contoh. Masih ingat algoritma Fibonacci? Yaitu sebuah algoritma yang memberikan deret angka sebagai berikut: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,...n. Dimana deretan angka yang diperoleh dari:

Tabel 6.1. Deret bilangan Fibonacci.

n =	0	1	2	3	4	5	6	7	8	9	10
Xn =	0	1	1	2	3	5	8	13	21	34	?

Dimana $X_n = X(n - 1) + X(n - 2)$. Misalnya $X_3 = X_2 + X_1 = 1 + 1 = 3$.



$$X_{10} = X_9 + X_8$$

Gambar 6.2. Deret Fibonacci.

Lalu bagaimana dengan X_{10} ? Jawabnya: $X_{10} = X_9 + X_8 = 21 + 34 = 55$.

Kemudian kita mencoba untuk membuat kode program untuk menghitung deret fibonacci tersebut. Yaitu:

```
fib.py
1 import timeit
2
3 def fib(n):
4     if n < 2:
5         return n
6     else:
7         return fib(n-1)+fib(n-2)
8
9 number = int(raw_input("Number:"))
10 start_time = timeit.default_timer()
11 number=fib(number)
12 time = (timeit.default_timer() - start_time)
13
14 print "Output:",number
15 print "Time:", format(time,'.10f')
16
```

```
Irwans-MacBook-Air:kode baksosolo$ python fib.py
Number:3
Output: 2
Time: 0.0001039505
Irwans-MacBook-Air:kode baksosolo$ python fib.py
Number:8
Output: 21
Time: 0.0000610352
Irwans-MacBook-Air:kode baksosolo$ python fib.py
Number:9
Output: 34
Time: 0.0000948906
Irwans-MacBook-Air:kode baksosolo$ python fib.py
Number:10
Output: 55
Time: 0.0001790524
Irwans-MacBook-Air:kode baksosolo$
```

Gambar 6.3. Kode Program Deret Fibonacci

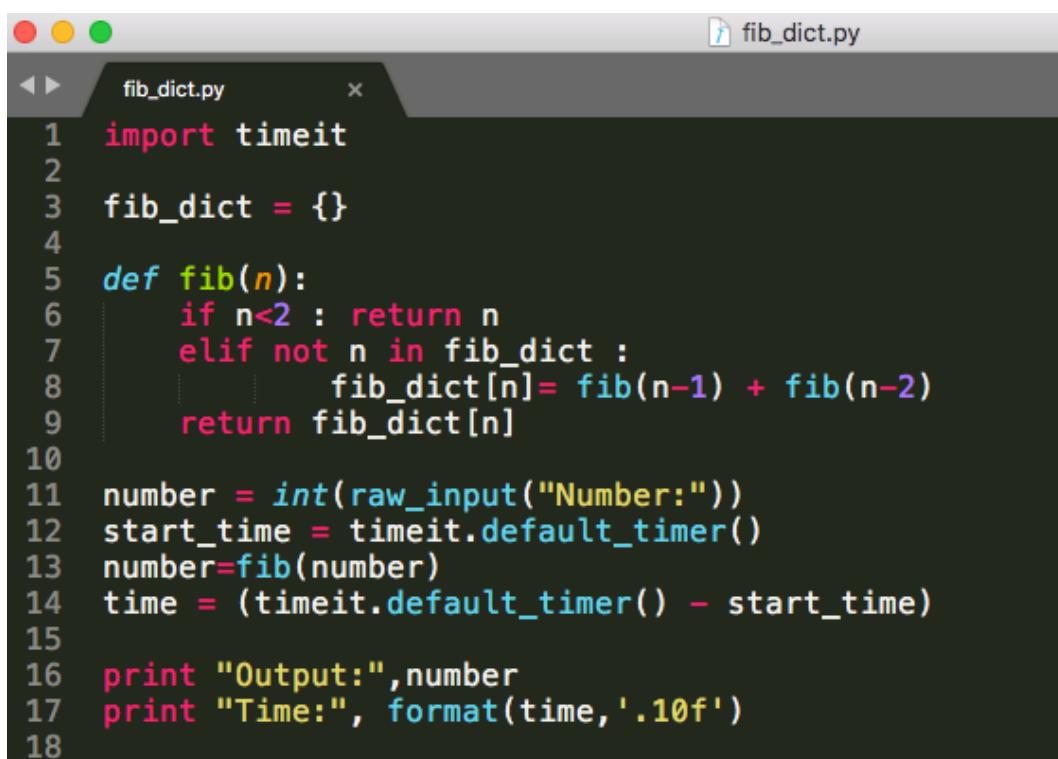
Dari tangkapan layar diatas, diketahui waktu eksekusi untuk menghitung deret fibonacci pada angka tertentu. Dan makin besar angka, makin besar waktu yang dibutuhkan untuk memproses jumlah deret Fibonacci tersebut.

Dengan program yang sama, silahkan masukkan inputan (untuk menghitung) deret Fibonacci dari angka: 25,30,35,40,50 dan 100. Silahkan masukkan waktu yang diperoleh dalam tabel berikut:

Tabel 6.2. Deret Fibonacci Angka Puluhan beserta waktu proses

n =	25	30	35	40	50	100
Xn =	75025	832040	9227465	102334155	terminated	terminated
T(detik)	0.0929	0.9664	11.090	119.342	terminated	terminated

Dapat dilihat dari waktu proses tersebut, bahwa makin besar n, makin besar butuh waktu proses. Terutama menghitung deret Fibonacci dengan n= 50 dan n=100 (Komputer saya tidak dapat memberikan . Oleh karena itu, optimasi kode untuk mempercepat proses perhitungan tersebut perlu dilakukan. Bandingkan kode pada gambar 6.3 dengan kode berikut:



```

fib_dict.py

1 import timeit
2
3 fib_dict = {}
4
5 def fib(n):
6     if n<2 : return n
7     elif not n in fib_dict :
8         fib_dict[n]= fib(n-1) + fib(n-2)
9     return fib_dict[n]
10
11 number = int(raw_input("Number:"))
12 start_time = timeit.default_timer()
13 number=fib(number)
14 time = (timeit.default_timer() - start_time)
15
16 print "Output:",number
17 print "Time:", format(time,'.10f')
18

```

Gambar 6.4. Kode Program Deret Fibonacci dengan Dict.

Output dari program tersebut ialah:

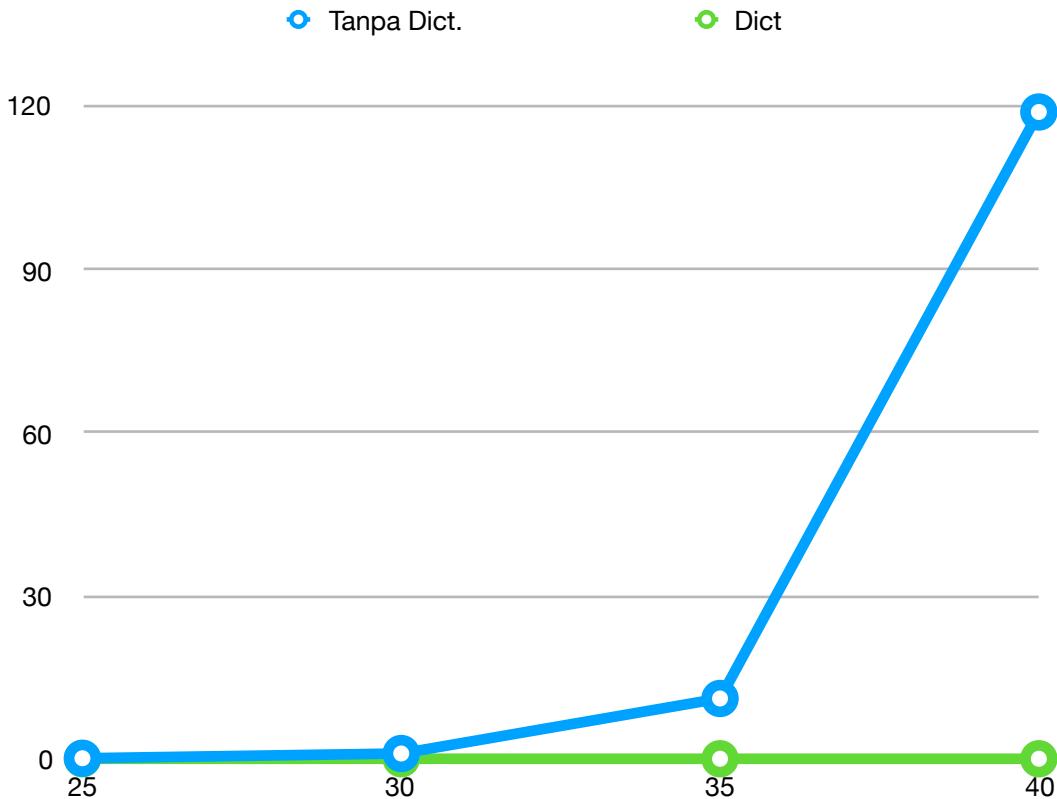
```
Irwans-MacBook-Air:kode baksosolo$ python fib_dict.py
Number:25
Output: 75025
Time: 0.0001659393
Irwans-MacBook-Air:kode baksosolo$ python fib_dict.py
Number:30
Output: 832040
Time: 0.0000910759
Irwans-MacBook-Air:kode baksosolo$ python fib_dict.py
Number:35
Output: 9227465
Time: 0.0001997948
Irwans-MacBook-Air:kode baksosolo$ python fib_dict.py
Number:40
Output: 102334155
Time: 0.0002419949
Irwans-MacBook-Air:kode baksosolo$ python fib_dict.py
Number:50
Output: 12586269025
Time: 0.0002820492
Irwans-MacBook-Air:kode baksosolo$ python fib_dict.py
Number:100
Output: 354224848179261915075
Time: 0.0005378723
```

Gambar 6.5. Output Kode Program Deret Fibonacci dengan Dict.

Dan setelah dimasukkan dalam tabel perhitungan

Tabel. 6.3 Deret Fibonacci dengan Dict. beserta waktu proses

n =	25	30	35	40	50	100
Xn =	75025	832040	9227465	102334155	12586269025	354224848179261915075
T(detik)	0.0001659393	0.0000910759	0.0001997948	0.0002419949	0.0002820492	0.0005378723



Gambar 6.6. Perbandingan kode tanpa dan dengan Dictionary.

Dapat dibandingkan bahwa waktu proses kode pada gambar 6.4 lebih cepat. Kode tersebut dioptimasi dengan menyimpan hasil deret Fibonacci ke dalam sebuah Dictionary Python. Sehingga untuk proses perhitungan pada selanjutnya, komputer tidak perlu menghitung dari awal. Cara ini hanyalah contoh kecil untuk optimasi kode pemrograman. Contoh yang lain, pembaca perlu memperbanyak membaca literasi-literasi berkenaan optimasi kode pemrograman.

6.3. Publikasi Ilmiah : Optimasi Algoritma Pengurutan dan Pencarian

Untuk lebih memahami teori tentang algoritma, sangat disarankan untuk membaca publikasi ilmiah yang membahas algoritma baru untuk melakukan proses pencarian dan perhitungan. Sebagai contoh publikasi yang berjudul : Fast Algorithms for Sorting and Searching Strings (Bentley and Sedgewick 1997)¹. Serta tidak ada salahnya apabila membaca publikasi terhadap penelitian yang dilakukan 20 tahun kemudian (2017). Yaitu publikasi berkenaan dengan pengembangan algoritma Insertion Sort, yang berjudul: Bidirectional Conditional Insertion Sort algorithm; An efficient progress on the classical insertion sort (Mohammed, Amrahov et al. 2017)².

Agar lebih mudah mempelajari/membaca publikasi ilmiah, cukup rangkum beberapa point berikut:

1. Latar belakang yang menjadi perhatian di publikasi tersebut.
2. Jika mengajukan permasalahan, bagaimana peneliti menyelesaikan permasalahan tersebut?
3. Bagaimana penelitian sebelumnya yang dilakukan oleh penulis tersebut atau peneliti lainnya?
4. Bagaimana penelitian tersebut dilakukan oleh penulis. Menggunakan metode apa? Apa yang berbeda dari peneliti lainnya?
5. Bagaimana hasil dari eksperimen tersebut.
6. Bagaimana Future Works (hal-hal yang perlu/dapat dikembangkan) pembaca/peneliti lainnya.

Keenam pertanyaan ini penulis sampaikan berdasar pengamalan pribadi penulis dalam membaca publikasi ilmiah yang akan menjadi rujukan penelitian selanjutnya.

¹ <http://www.cs.princeton.edu/~rs/strings/paper.pdf>

² <https://arxiv.org/pdf/1608.02615.pdf>

Biodata Penulis



Irwan A. Kautsar, pria kelahiran tahun 1982, menyelesaikan Studi S1 di Teknik Informatika ITS Surabaya (2004-2008); S2 Teknik Informatika ITS Surabaya (2009-2012) dan S3 di Kumamoto University, Kumamoto, Jepang (2012-2016). Saat ini aktif mengajar di Prodi Informatika Fakultas Teknik Universitas Muhammadiyah Sidoarjo. Bidang penelitian penulis ialah Network Security, Open System, Sistem Terdistribusi Konten Pembelajaran (Distributed Learning Object), Desentralisasi Jaringan Komputer dan Web Services. Publikasi yang dilakukan penulis dapat dilihat pada laman berikut: <http://hepidad.github.io/pubs>. Penulis dapat dijangkau melalui surel irwan@umsida.ac.id.

Bibliography

- Bentley, J. L. and R. Sedgewick (1997). Fast algorithms for sorting and searching strings. Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics.
- Cormen, T. H. (2013). Algorithms unlocked, Mit Press.
- Mohammed, A. S., §. E. Amrahov and F. V. Çelebi (2017). "Bidirectional Conditional Insertion Sort algorithm; An efficient progress on the classical insertion sort." Future Generation Computer Systems **71**: 102-112.
- Pressman, R. S. (2005). Software engineering: a practitioner's approach, Palgrave Macmillan.

Buku ajar Algoritma dan Pemrograman disusun dalam rangka menunjang proses kegiatan belajar mengajar yang memudahkan rekan pembaca sekalian mempelajari algoritma dan pemrograman, khususnya mahasiswa Program Studi Informatika, Fakultas Sains dan Teknologi Universitas Muhammadiyah Sidoarjo. Diharapkan adanya buku ajar ini, mahasiswa Program Studi Informatika mampu mengimplementasikan algoritma dengan membuat program sederhana menggunakan bahasa pemrograman Python. Penulis memilih bahasa pemrograman Python dalam modul buku ajar karena bahasa Python mudah dipelajari, cross-platform, multi device dan terpenting, kemudahannya dalam pengimplementasian algoritma.



Irwan A. Kautsar, saat ini aktif mengajar di Prodi Informatika Fakultas Sains dan Teknologi Universitas Muhammadiyah Sidoarjo. Bidang penelitian penulis ialah Network Security, Open System, Sistem Terdistribusi dan

Konten Pembelajaran (Distributed Learning Object), Desentralisasi Jaringan Komputer dan Web Services. Publikasi yang dilakukan penulis dapat dilihat pada laman berikut: <http://hepidad.github.io/pubs>.

Cover Design: Stay@HomeMom

ISBN 978-979-3401-71-3

A standard linear barcode representing the ISBN number 978-979-3401-71-3.

9 789793 401713