MENCOBA PYTHON

MENCOBA PYTHON
Belajar Python untuk Pemula

Rolly Maulana Awangga
Politeknik Pos Indonesia

Program Studi Sarjana Terapan Teknik Informatika.
Politeknik Pos Indonesia

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Mencoba Python / Rolly Maulana Awangga . . . [et al.].
p. cm.—(Wiley series in survey methodology)
"Wiley-Interscience."
Includes bibliographical references and index.
ISBN 0-471-48348-6 (pbk.)
1. Surveys—Methodology. 2. Social sciences—Research—Statistical methods. I. Groves, Robert M. II. Series.

HA31.2.S873 2007
001.4'33—dc22 2004044064
Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

Untuk Nusa Bangsa dan
Agama

CONTRIBUTORS

JURU KETIK, Fujitsu Laboratories Ltd., Fujitsu Limited, Atsugi, Japan

JURU KODING, Center for Solid State Electronics Research, Arizona State University, Tempe, Arizona

TUKANG BEBERES, Department of Electrical and Computer Engineering, University of Notre Dame, Notre Dame, South Bend, Indiana; formerly of Center for Solid State Electronics Research, Arizona State University, Tempe, Arizona

CONTENTS IN BRIEF

CONTENTS

LIST OF FIGURES

LIST OF TABLES

FOREWORD

This is the foreword to the book.

PREFACE

This is an example preface. This is an example preface. This is an example preface. This is an example preface.

R. K. WATTS

Durham, North Carolina
September, 2007

ACKNOWLEDGMENTS

From Dr. Jay Young, consultant from Silver Spring, Maryland, I received the initial push to even consider writing this book. Jay was a constant “peer reader” and very welcome advisor durying this year-long process.

To all these wonderful people I owe a deep sense of gratitude especially now that this project has been completed.

G. T. S.

ACRONYMS

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

GLOSSARY

NormGibbs	Draw a sample from a posterior distribution of data with an unknown mean and variance using Gibbs sampling.
pNull	Test a one sided hypothesis from a numberically specified posterior CDF or from a sample from the posterior
sintegral	A numerical integration using Simpson's rule

SYMBOLS

A Amplitude

& Propositional logic symbol

a Filter Coefficient

$\mathcal{B}$ Number of Beats

INTRODUCTION

CATHERINE CLARK, PHD.

Harvard School of Public Health
Boston, MA, USA

The era of modern began in 1958 with the invention of the integrated circuit by J. S. Kilby of Texas Instruments His first chip is shown in Fig. I. For comparison, Fig. I.2 shows a modern microprocessor chip.

This is the introduction. This is the introduction. This is the introduction. This is the introduction. This is the introduction. This is the introduction.

$$ABCDEF\alpha\beta\Gamma\Delta \sum_{def}^{abc} \tag{I.1}$$

REFERENCES

1. J. S. Kilby, "Invention of the Integrated Circuit," *IEEE Trans. Electron Devices,* **ED-23,** 648 (1976).
2. R. W. Hamming, *Numerical Methods for Scientists and Engineers*, Chapter N-1, McGraw-Hill, New York, 1962.
3. J. Lee, K. Mayaram, and C. Hu, "A Theoretical Study of Gate/Drain Offset in LDD MOSFETs" *IEEE Electron Device Lett.,* **EDL-7**(3). 152 (1986).

PART I

SUBMICRON SEMICONDUCTOR MANUFACTURE

CHAPTER 1

HOME

HOME

- python

Figure 1.1 Logo

1.1 python

Pemrograman python adalah bahasa pemrograman terpopuler di tahun 2016 menurut tiobe. Python juga memiliki sintak atau aturan penulisan code pemrograman. Salah satu bagian Home merupakan halaman pengantar untuk mempelajari python . sebelum ketahapan yang baru selain home ini pembaca memerlukan pengertian yang lain yaitu seperti enverinmoment setup, syntax dan lain lain, awal untuk penulis jelakan yaitu pengertian tentang class pada python untuk mengantarkan logika dan pengetahuan apaitu class.

Nah class itu merupakan class yang didalam nya mempunyai metode yang sesuai dengan fungsinya, contoh di kehidupan nyata seperti kelas belajar sebagai class nya da isi dari kelas itu seperti bangku, sepidol, dan lain lain itu merupakan metode dari class dan metode itu berfungsi seperti fungsi itu sendiri seperti spidol untuk menulis itu contoh nya di sesuaikan dengan fungsi.

Pembuatan class pada python

Kita awali dengan sebuah kata kunci. Yaitu ”class ” yang kemudian di ikuti dengan ”nama class nya ” terkahir membuat kurung buka dan tutup serta membuat tanda titik dua ”() ” dan ”: ” kalo synax nya seperti ini.

namaClass()

untuk memanggil metode kita cukup menggunakan memanggil class yang kemudian di ikuti dengan pemanggilan nama metode yang tersedia di dalam class tersebut dengan di pisahkan oleh tanda titik seperti

namClass().namaMetode()

ingin lebih mudah kita tampung classnya dulu ke variable

penampung = namaClass()

penampung.namaMetode()

di dalam sebuah class yang dibuat biasanya terdapat init itu disediakan langsung oleh python nya jadi seperti ini

```
 class namaClass():
def init (self,parameter):
itu code program yang pertama kali kalian buat
def metode 1 (self,parameter):
isi metode
def metode 2 (self):
 isi metode
```

nah seperti itu lah kurang lebih. Setelah menjelaskan class kita akan menjelaskan variable seperti tadi penampung class, nah variable bisa di artikan sebagai huruf atau kata tujuan nya untuk mempermudah proses penulisan sebuah program.

Contoh dalam kehidupan nyata seperti gelas atau ember, ambil contoh ember kita ketahui bahwa ember bisa di isi dengan air, pasir, tanah dan lain lain, nah ember itu sebagai variable da nisi variable nya itu adalah air, tanah, pasir dan lain lain.

Contoh variable seperti ini

Variable= ”ini string atau teks ”

Variable2=12

Print(”nilai isi dari variable1 adalah: ”,variable1)

Print(”nilai atau isi dari variable2 adalah: ”,variable2)

Nah seperti itu contohnya

Ada beberapa hal yang harus di ketahui seperti input, data operation danlain lain seperti ini

- input
- data
- opration
- output
- condutional
- looping
- subroutine

1.1.1 Input

Tahapan ini merupakan proses memasukan data ke dalam proses komputer melalui peralatan input. Pada bahasa Python, untuk menerima masukan dari pengguna yaitu dengan menggunakan method input() dan raw _input().

1.1.2 Data

Data adalah bahan mentah yang akan diolah menjadi informasi sehingga dapat berguna dan dimanfaatkan oleh pengguna. Data dapat berupa variabel, konstanta, atau yang berisi bilangan, kalimat, dan lainnya. Tipe data berupa string, number, list, tuple, dan lainnya.

1.1.3 Operation

Operation adalah yang akan mengubah suatu nilai menjadi nilai lain. Yang termasuk operationatau yang biasa disebut dengan operator adalah operator aritmatika, operator assignment, dan lainnya.

1.1.4 Output

Output adalah menuliskan informasi yang ditampilkan dilayar, disk, atau ke salah satu unit I/O. Pada Python 2.0, untuk menampilkan output dengan menulis sintax print. Sedangkan pada Python 3.0 dengan menggunakan fungsi print().

1.1.5 Conditional

Merupakan jumlah perintah yang akan dijalankan jika kondisi tertentu sudah terpenuhi. Jika username dan password yang dimasukan benar, maka akan menampilkan halaman utama. Hal ini bisa disebut conditional. Pada conditional, Python menggunakan pernyataan if, else, dan elif.

1.1.6 Looping

Perintah yang akan berjalan beberapa kali, selama kondisi yang ditentukan atau kondisi yang terpenuhi. Pada looping ini, Python menggunakan pernyataan for dan while untuk melakukan perulangan.

1.1.7 Subroutine

Perintah yang bisa dijalankan dengan cara memanggil namanya. Sering disebut sebagai functionatau method. Pada bahasa pemrograman Python, untuk menggunakan function atau method yaitu dengan menggunakan pernyatan def nama _function().

Fungsi def dalam python

Penggunaan fungsi tanpa parameter

Command=fungsi()

Deklarasi command= def fungsi()

Pemanggilan fungsi, parameter sesuai dengan kata kunci seperti tadi class

Command= fungsi(arg=1, arg2=2)

Deklarasi command def fungsi (arg2,arg2)

Pemanggilan fungsi, parameter sesuai dengan posisi

Command= fungsi()

Deklarasi command def fungsi (x)

Pemanggilan fungsi parameter sesuai dengan argument posisional tuple

Command= fungsi((1,2).(1,3))

Deklarasi command def fungsi (*args)

Pemanggilan fungsi, parameters ssesuia argument kata kunci dictionary

Command= fungsi (bahasa= python,versi=2.2)

Deklarasi command = def fungsi (**args)

Itu adalah cara memanggil dalam code python pemrograman jadi ada nenerapa fungsi yang dibutuhkan penulisan dengan tepat maka sebab itu dengan sebab itu buaat lah penulisan yang mudah. Karena pemrograma python sangan sensitive bila ada kesalahan sedikit di penulisan atau symbol yang tertinggal.

1.1.8 Sejarah

Bahasa pemrograman Python adalah bahasa yang dibuat oleh seorang keturunan Belanda yaitu Guido van Rossum. Awalnya, pembuatan bahasa pemrograman ini adalah untuk membuat skrip bahasa tingkat tinggi pada sebuah sistem operasi yang terdistribusi Amoeba. Python telah digunakan oleh beberapa pengembang dan bahkan digunakan oleh beberapa perusahaan untuk pembuatan perangkat lunak

komersial.

Pemrograman bahasa python ini adalah pemrogram gratis atau freeware, sehingga dapat dikembangkan, dan tidak ada batasan dalam penyalinannya dan mendistribusikan.

Dukungan Komunitas yang Aktif

Python adalah salah satu pemrograman yang terus berkembang dan bertahan dikarenakan dukungan komunitas yang aktif diseluruh dunia. Banyak forum-forum ataupun blogger-blogger yang sering membagi pengalaman dalam menggunakan python. Hal ini memudahkan bagi pengguna pemula maupun pengembang untuk bertanya dan sharing tentang ilmu pemrograman ini.

Kelebihan dan Kekurangan

Kelebihan :

1. Tidak ada tahapan kompilasi dan penyambungan (link) sehingga kecepatan perubahan pada masa pembuatan sistem aplikasi meningkat.

2. Tidak ada deklarasi tipe data yang merumitkan sehingga program menjadi lebih sederhana, singkat, dan fleksible.

3. Manajemen memori otomatis yaitu kumpulan sampah memori sehingga dapat menghindari pencacatan kode.

4. Tipe data dan operasi tingkat tinggi yaitu kecepatan pembuatan sistem aplikasi menggunakan tipe objek yang telah ada.

5. Pemrograman berorientasi objek.

6. Pelekatan dan perluasan dalam C.

7. Terdapat kelas, modul, eksepsi sehingga terdapat dukungan pemrograman skala besar secara modular.

8. Pemuatan dinamis modul C sehingga ekstensi menjadi sederhana dan berkas biner yang kecil

9. Pemuatan kembali secara dinamis modul phyton seperti memodifikasi aplikasi tanpa menghentikannya.

10. Model objek universal kelas Satu.

11. Konstruksi pada saat aplikasi berjalan.

12. Interaktif, dinamis dan alamiah.

13. Akses hingga informasi interpreter.

14. Portabilitas secara luas seperti pemrograman antar platform tanpa ports.

15. Kompilasi untuk portable kode byte sehingga kecepatan eksekusi bertambah dan melindungi kode sumber.

16. Antarmuka terpasang untuk pelayanan keluar seperti perangkat Bantu system, GUI, persistence, database, dll.

Kekurangan :

1. Beberapa penugasan terdapat diluar dari jangkauan python, seperti bahasa pemrograman dinamis lainnya, python tidak secepat atau efisien sebagai statis, tidak seperti

bahasa pemrograman kompilasi seperti bahasa C.

2. Disebabkan python merupakan interpreter, python bukan merupakan perangkat bantu terbaik untuk pengantar komponen performa kritis.

3. Python tidak dapat digunakan sebagai dasar bahasa pemrograman implementasi untuk beberapa komponen, tetapi dapat bekerja dengan baik sebagai bagian depan skrip antarmuka untuk mereka.

4. Python memberikan efisiensi dan fleksibilitas tradeoff by dengan tidak memberikannya secara menyeluruh. Python menyediakan bahasa pemrograman optimasi untuk kegunaan, bersama dengan perangkat bantu yang dibutuhkan untuk diintegrasikan dengan bahasa pemrograman lainnya.

5. Banyak terdapat referensi lama terutama dari pencarian google, python adalah pemrograman yang sangat lambat. Namun belum lama ini ditemukan bahwa Google, Youtube, DropBox dan beberapa software sistem banyak menggunakan Python.

6. Kini Python menjadi salah satu bahasa pemrograman yang populer digunakan oleh pengembangan web, aplikasi web, aplikasi perkantoran, simulasi, dan masih banyak lagi. Hal ini disebabkan karena Python bahasa pemrograman yang dinamis dan mudah dipahami.

7. Selain itu, sekarang telah tersedia berbagai situs kursus yang bagus untuk mempelajari bahasa pemrograman Python ini sehingga pembaca maupun developer pemula yang akan mempelajari bahasa ini akan menjadi lebih mudah karena dapat berlatih dimanapun dan kapanpun selama terhubung dengan Internet.

8. Menariknya, berbagai situs kursus gratis ini menawarkan metode pembelajaran yang interaktif sehingga mudah di-

mengerti oleh pesertanya.

Python merupakan pemograman yang tidak pernah di compile secara full. Jika kamu sudah menyelesaikan programnya dan kamu ingin mengirim ke teman atau di bagikan ke internet maka teman atau orang lain dapat mengubah kode di program kamu karena program di buka di notepad, python akan tetap berbentuk kode yang sama tidak acak acakan sehingga orang lain dapat memahami pemograman yang kamu buat.

Python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

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

Ketik teks berikut pada prompt Python dan tekan Enter:

print ”Hello, Python!”

Jika Anda menjalankan versi baru Python, Anda perlu menggunakan pernyataan cetak dengan tanda kurung seperti pada cetak (”Halo, Python!”) ;. Namun dengan versi Python 2.4.3, ini menghasilkan hasil sebagai berikut:

Hello, Pyhton!

Pemrograman Mode Script

Memohon interpreter dengan parameter script memulai eksekusi script dan berlanjut sampai script selesai. Saat skrip selesai, juru bahasa tidak lagi aktif.

Mari kita tuliskan program Python sederhana dalam sebuah naskah. File Python memiliki ekstensi .py. Ketik kode sumber berikut di file

Objek Dengan Python, seperti semua bahasa berorientasi objek, ada kumpulan kode dan data yang disebut objek, yang biasanya mewakili potongan dalam model konseptual suatu sistem.

Objek dengan Python dibuat (yaitu, instantiated) dari template yang disebut kelas (yang akan dibahas kemudian, sebanyak bahasa dapat digunakan tanpa memahami kelas). Mereka memiliki atribut, yang mewakili berbagai potongan kode dan data yang membentuk objek. Untuk mengakses atribut, seseorang menuliskan nama objek yang diikuti oleh suatu periode (selanjutnya disebut titik), diikuti dengan nama atribut.

Contohnya adalah atribut 'atas' dari string, yang mengacu pada kode yang mengembalikan salinan string di mana semua huruf adalah huruf besar. Untuk mendapatkan ini, perlu untuk memiliki cara untuk merujuk ke objek (dalam contoh berikut, jalan adalah string literal yang membangun objek).

Paradigma : Multi-paradigm: object-oriented, imperative, functional, procedural, reflective Muncul Tahun : 1991 Perancang : Guido van Rossum Pengembang : Python Software Foundation Rilis terbaru : 3.2.3 / 11 April 2012; 46 hari lalu 2.7.3 / 11 April 2012; 46 hari lalu / Sistem pengetikan: duck, dynamic, strong Implementasi : CPython, IronPython, Jython, Python for S60, PyPy Dialek : Cython, RPython, Stackless Python Terpengaruh oleh : ABC,ALGOL 68,C,C++,Dylan Haskell,Icon,Java,Lisp,Modula-3,Perl Mempengaruhi : Boo, Cobra, D, Falcon, Groovy, JavaScript, Ruby Sistem operasi : Cross-platform Lisensi : Python Software Foundation License,GNU GPL Situs web : python.org

Kesimpulan

Kenapa python banyak digunakan sehingga terpopuler di tahun 2016, karena python salah satu pemrograman opensource sehingga dapat dipahami apabila ingin di pahami, apa yang menjadi python melambung tinggi sampai saat ini, diantaranya mempunyai kepustkaan yang luas untuk memduahkan programmer selain itu ada module yang di sediakan oleh python nya langsung. Apabila pembaca tau tentang instagram atau django itu adalah situs asuhan dari mark zuckerburg instagram dan django adalah web yg menyediakan berbagai hal kalian ketahui django

adalah server kapastitas besar dan instagram kalian ketahu sendiri kan itu dibuat dengan pemrograman python menurut disca7x.blogspot.co.id

CHAPTER 2

OVERVIEW

OVERVIEW

- python

Figure 2.1 Logo

- penjelasan overview di python

2.1 Overview

Python adalah bahasa script tingkat tinggi, ditafsirkan, interaktif dan berorientasi objek. Python dirancang agar mudah dibaca. Ini menggunakan kata kunci bahasa Inggris sering di mana bahasa lainnya menggunakan tanda baca, dan memiliki konstruksi sintaksis lebih sedikit daripada bahasa lainnya.

Python is interpreted : diproses pada saat runtime oleh interpreter

Tidak perlu untuk mengkompilasi program anda sebelum mengeksekusi itu. Hal ini merupakan mirip dengan php

Python is Interactive: Anda dapat benar-benar duduk di prompt Python dan berinteraksi dengan penafsir langsung untuk menulis program Anda.

Python is Object-Oriented: Python mendukung gaya Berorientasi Objek atau teknik pemrograman yang merangkum kode di dalam objek.

Python is a Beginner's Language: Python adalah bahasa yang besar untuk programmer tingkat pemula dan mendukung pengembangan berbagai aplikasi dari pengolahan teks sederhana untuk browser WWW untuk game.

2.2 Fitur overview dalam python itu adalah

- Easy-to-learn: Python memiliki beberapa kata kunci, struktur sederhana, dan sintaks yang jelas. Hal ini memungkinkan siswa untuk mengambil bahasa dengan cepat.
- Easy-to-read: kode Python lebih jelas dan terlihat mata.

- Easy-to-maintain: kode sumber Python cukup mudah-untuk-menjaga.

A broad standard library: bulk Python perpustakaan sangat portabel dan cross-platform yang kompatibel pada UNIX, Windows, dan Macintosh.

Interactive Mode: Python memiliki dukungan untuk mode interaktif yang memungkinkan pengujian interaktif dan debugging dari potongan kode.

Portable: Python dapat dijalankan pada berbagai macam platform perangkat keras dan memiliki antarmuka yang sama pada semua platform.

Extendable: Anda dapat menambahkan modul tingkat rendah ke interpreter Databases: Python menyediakan antarmuka untuk semua database komersial utama.

GUI Programming: Python mendukung aplikasi GUI yang dapat dibuat dan porting ke banyak panggilan sistem, perpustakaan dan sistem jendela, seperti Windows Scalable: Python menyediakan struktur dan dukungan yang lebih baik untuk program besar dari shell scripting.

2.3 Fitur overview terbaik adalah

IT mendukung metode pemrograman fungsional dan terstruktur serta OOP.

Hal ini dapat digunakan sebagai bahasa scripting atau dapat dikompilasi untuk byte-kode untuk membangun aplikasi besar.

Ini memberikan tingkat tinggi sangat tipe data dinamis dan mendukung memeriksa jenis dinamis.

IT mendukung pengumpulan sampah otomatis.

Hal ini dapat dengan mudah diintegrasikan dengan C, C ++, COM, ActiveX, CORBA, dan Java.

Hal tersebut menjadi terpopuler karena kemudahan bagi programmer yang menjadikan python pemograman terbaik pada tahun 2016

Beberapa setandarisasi dalam python beriorientasi objek yaitu

2.4 Oprasi interface

Sejumlah fungsi yang terkait dengan system oprasi

```
- import os
- os.getcwd()~~~~~
Python34
- os.chdir('/server/accesslogs')
- os.system('mkdir~today')~
0
```

$$Filewildcard \tag{2.1}$$

Menyediakan fungsi untuk membuat daftar file dari pencarian direktori wildcard

```
- import glob
- glob.glob('*.py')
['primes.py', 'random.py', 'quote.py']
```

Parameter baris perintah Script umum yang sering memang-

gil parameter baris perintah.

- import sys

- print(sys.argv)

[’demo.py’, ’one’, ’two’, ’three’]

Redirection dan program pemutusan

Untuk menampilkan peringatan dan pesan kesalahan

sys.stderr.write(’Warning, log file not found starting a new one \n’)

Warning, log file not found starting a new one

String

Penccokan kompleks dan manipulasi solusi optimal

- import re

- re.findall(r’ \bf[a-z]*’, ’which foot or hand fell fastest’)

[’foot’, ’fell’, ’fastest’]

- re.sub(r’(\b[a-z]+) \1’, r’ \1’, ’cat in the the hat’)

’cat in the hat’

String biasa

- ’tea for too’.replace(’too’, ’two’)

’tea for two’

Matematika

Point penghitungan matematika

- import math

- math.cos(math.pi / 4)

0.70710678118654757

- math.log(1024, 2)

10.0

Code acak matematika

- import random

- random.choice(['apple', 'pear', 'banana'])

'apple'

- random.sample(range(100), 10) # sampling without replacement

[30, 83, 16, 4, 8, 81, 41, 50, 18, 33]

- random.random() # random float

0.17970987693706186

- random.randrange(6) # random integer chosen from range(6)

4

Akses internet

Memproses data yang di terima dari url untuk mengirim email

```
\noindent
- from urllib.request import urlopen
- for line in urlopen('http://tycho.usno.navy.mil/cgi-bin/time
...~~~~ line = line.decode('utf-8')~ Decoding the binary data
...~~~~ if~'EST' in line or 'EDT' in line:look for Eastern Tim
...~~~~~~~~ print(line)
<BR>Nov. 25, 09:43:32 PM EST
- import smtplib
- server = smtplib.SMTP('localhost')
- server.sendmail('soothsayer@example.org', 'jcaesar@example.o
... """To: jcaesar@example.org
... From: soothsayer@example.org
...
... Beware the Ides of March.
... """)
- server.quit()
Tanggal dan waktu
Datetime yang kompleks
- dates are easily constructed and formatted
- from datetime import date
- now = date.today
- now
datetime.date(2003, 12, 2)
- now.strftime
'12-02-03. 02 Dec 2003 is a Tuesday on the 02 day of December.
- dates support calendar arithmetic
- birthday = date(1964, 7, 31)
- age = now  birthday
- age.days
14368
Data kompresi
data umum untuk pengarsipan dan kompresi format
- import zlib
- s = b'witch which has which witches wrist watch'
- len(s)
```

```
41
- t = zlib.compress(s)
 len(t)
37
- zlib.decompress(t)
b'witch which has which witches wrist watch'
- zlib.crc32(s)
226805979
metric kinerja
alat pengukuran yang di sediakan langsung oleh python
- from timeit import Timer
- Timer('t=a; a=b; b=t', 'a=1; b=2').timeit()
0.57535828626024577
- Timer('a,b = b,a', 'a=1; b=2').timeit()
0.54962537085770791
uji module
pengembangan perangkat lunak berkualitas tinggi
def average(values):
~~~ """Computes the arithmetic mean of a list of numbers.
~~~ - print(average([20, 30, 70]))
~~~ 40.0
~~~ """
~~~ return sum(values) / len(values)
import doctest
doctest.testmod()~~
cara code dengang file terpisah
import unittest
class TestStatisticalFunctions(unittest.TestCase):
~~~ def test $  \_  $average(self):
~~~~~~~ self.assertEqual(average([20, 30, 70]), 40.0)
~~~~~~~ self.assertEqual(round(average([1, 5, 7]), 1), 4.3)
~~~~~~~ self.assertRaises(ZeroDivisionError, average, [])
~~~~~~~ self.assertRaises(TypeError, average, 20, 30, 70)
```

unittest.main()

2.4.1 ada juga standar ikhtisar yang sering digunakan oleh progremer

yaitu

- pengenalan python
- mengisntal python
- matematika + numbers
- string
- list
- if elif else
- for loop
- for else range breake
- pass dan continue
- while loop
- function
- function dan arguments
- return velue
- lamda function
- sope global dan local
- more on list
- sticks dan queues

pengenalan python

menginstal python

numbers + matematika

string

list

if elif else

for loop

for else, range, breake

continue dan pass

while loop

function

function dan arguments

return value

lambda function

scope global dan local

more on list

stacks and queues

CHAPTER 3

ENVIRONTMENT SETUP

3.1 Overview Phyton

Python adalah bahasa script tingkat tinggi, ditafsirkan, interaktif dan berorientasi objek. Python dirancang agar mudah dibaca. Ini menggunakan kata kunci bahasa Inggris sering di mana bahasa lainnya menggunakan tanda baca, dan memiliki konstruksi sintaksis lebih sedikit daripada bahasa lainnya. Python diinterpretasikan: Python diproses pada saat runtime oleh interpreter. Anda tidak perlu mengkompilasi program Anda sebelum menjalankannya. Ini mirip dengan PERL dan PHP. Python mendukung multi paradigma pemrograman, utamanya; namun tidak dibatasi; pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang

dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi.

3.1.1 Situs Belajar Python

1. Learn Python Bagi pembaca yang tertarik untuk belajar bahasa pemrograman Python secara gratis, situs ini menjadi salah satu pilihan yang bijak. Dalam situs Learn Python ini pembaca akan dihadapkan pada halaman utama yang berisi penjelasan, tutorial, dan kolom pembelajaran interaktif. Disini, pembaca dapat belajar bahasa pemrograman Python mulai dari dasar hingga tingkat lanjut dengan berbagai penjelasan serta tutorial dasar untuk memahami bahasa pemrograman Python. Developer dapat langsung memasukkan kode-kode latihan pada kolom pembelajaran interaktif yang nantinya dapat dijalankan untuk melihat apakah kode tersebut bisa berjalan atau terjadi kesalahan. Selain itu, pembaca dapat juga mengetahui keinginan dari hasil-hasil kode latihan yang diberikan oleh Learn Python pada kolom pembelajaran interaktif.

2. Codecademy Bisa dibilang ini salah satu situs yang menawarkan pembelajaran dan latihan untuk beberapa bahasa situs yang populer di dunia seperti Python, JavaScript, jQuery, Ruby, PHP, Python, HTML, dan CSS dengan tingkatan level yang disesuaikan. Developer dapat mempelajar bahasa pemrograman Python di situs ini dengan interaktif dan baik. Nantinya developer akan diberikan halaman latihan dua kolom yang terdiri dari pengenalan pada kolom kiri dan latihan pada kolom kanan. Pada kolom kanan ini developer dapat langsung mengetikan baris kode untuk pemrograman dan dapat

langsung dijalankan secara langsung. Nantinya developer dapat melihat persentase mengenai tingkatan bahasa pemrograman yang telah dipelajar.

3. Treehouse Treehouse merupakan salah satu situs yang mengajarkan beragam bahasa pemrograman mulai dari web hingga aplikasi mobile. Beberapa materi kursus yang ditawarkan oleh situs ini di antaranya Learn Python, Android Development, Web Designer, dan masih banyak lagi. Dalam situs ini, pembaca maupun developer akan disuguhkan jalur latihan bahasa pemrograman Python mulai dari dasar hingga tingkat lanjut. Dengan menggunakan situs ini, developer dapat melakukan latihan pemrograman secara interaktif. Nantinya developer akan meraih skor dari kemampuannya dalam menghadapi latihan yang ditawarkan Treehouse.

4. Trinket Situs ini menghadirkan kursus bahasa pemrograman Python yang bisa dibilang lengkap. Pembaca maupun developer akan diberikan berbagai materi dan tutorial yang interaktif. Dalam halaman materi yang disampaikan akan terdapat kolom interaktif yang akan membuat developer dapat mempelajari Python dengan lebih mudah. Untuk materi bahasa pemrograman Python yang disampaikan akan bertahap mulai dari dasar hingga tingkat lanjut. Developer juga dapat belajar bersama visual kura-kura yang akan disajikan pada kolom interaktif setiap developer menjalankan kode-kode yang telah diselesaikan. Kura-kura tersebut akan bergerak yang disertai dengan hasil akhir dari kode-kode yang dijalankan. Bisa dibilang metode pembelajaran yang ditawarkan oleh situs Trinket ini menyenangkan.

3.2 Environtment Setup For Python

Bahasa pemrograman python tersedia diberbagai platform, termasuk juga terdapat pada linux dan max os x. Untuk mengecek apakah python sudah terinstall atau belum, pertama tama bukalah command prompt atau terminal kom-

puter dan ketikan python. Akan muncul info atau status mengenai apakah python sudah terinstall dan berapa versinya atau belum terinstall sama sekali. Pada gambar

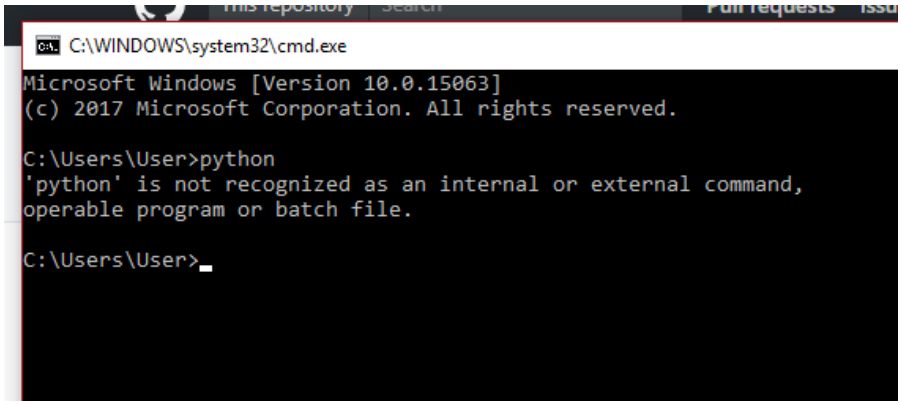


Figure 3.1 Contoh skirpt pada command prompt.

3.1 memberitahukan bahwa pada komputer yang saya pakai sekarang ini belum ada atau belum terinstall python. Ketika python belum terinstall maka sistem akan mendeteksi bahwa comand yang diberikan tidak cocok dengan yang ada disistem. Akan tetapi kalau sudah terinstall maka akan memunculkan nama python begitu pula versi python yang di install. Python menyediakan dukungan yang kuat untuk integrasi dengan bahasa pemrograman lain dan alat-alat bantu lainnya. Python hadir dengan pustakapustaka standar yang dapat diperluas serta dapat dipelajari hanya dalam beberapa hari. Sudah banyak programmer Python yang menyatakan bahwa mereka mendapatkan produktivitas yang lebih tinggi. Mereka juga merasakan bahwa Python meningkatkan kualitas

```
#!/usr/bin/python
import sys
try:
   # open file stream
   file = open(file_name, "w")
except IOError:
   print "There was an error writing to", file_name
```

```
   sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
   file_text = raw_input("Enter text: ")
   if file_text == file_finish:
      # close the file
      file.close
      break
   file.write(file_text)
   file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
   print "Next time please enter something"
   sys.exit()
try:
   file = open(file_name, "r")
except IOError:
   print "There was an error reading file"
   sys.exit()
file_text = file.read()
file.close()
print file_text
```

3.2.1 Memasang Phyton

Distribusi Python tersedia untuk berbagai macam platform. Anda hanya perlu mendownload kode biner yang berlaku untuk platform Anda dan menginstal Python. Jika kode biner untuk platform Anda tidak tersedia, Anda memerlukan kompiler C untuk mengkompilasi kode sumber secara manual. Kompilasi kode sumber menawarkan fleksibilitas lebih dalam hal pilihan fitur yang Anda butuhkan dalam instalasi Anda. Berikut adalah ikhtisar singkat tentang menginstal Python di berbagai platform :

3.2.2 Menyiapkan PATH

Program dan file eksekusi lainnya bisa berada di banyak direktori, jadi sistem operasi menyediakan jalur pencarian yang mencantumkan direktori yang dicari OS untuk executable. Path disimpan dalam variabel lingkungan, yang merupakan string bernama yang dikelola oleh sistem operasi. Variabel ini berisi informasi yang tersedia untuk perintah shell dan program lainnya. The path variabel disebut sebagai PATH di Unix atau jalan pada Windows (Unix adalah kasus sensitif, Windows tidak). Di Mac OS, installer menangani detail jalur. Untuk meminta juru bahasa Python dari direktori tertentu, Anda harus menambahkan direktori Python ke path Anda.

3.2.3 installasi Python pada windows

1. Unduh Python 2.7 di python.org.
2. Jalankan file instalasi Python yang telah diunduh. Secara default Python akan terinstall di folder C:¿Python27.
3. Selanjutnya mengatur variable environment, buka Control Panel-¿System and Security-¿System.
4. Klik Advanced system settings, Environment Variables.
5. Pada System variables, pilih Path, klik Edit.
6. Pada Variable value, tambahkan ;C:¿Python27.

3.2.3.1 Pengujian Python pada windows

1. Cara pertama: ketik python untuk menjalankan Python Interpreter. Jika berhasil akan ditampilkan versi Python yang digunakan beserta interpreternya. Coba ketikkan print hello world untuk menampilkan teks hello world.
2. Cara kedua: buat file hello.py, yang isinya print hello world, simpan di mana saja suka-suka kamu, misalnya di D:¿python-code. Buka command prompt, Run -¿ cmd. Kemudian masuk ke folder D:¿python-code, lalu ketik python hello.py untuk menjalankan file Python yang telah dibuat.

3.2.4 Installasi Python pada Unix/Linux

Berikutini adalah simple steps untuk menginstall python pada Unix/Linux.

1. Buka web browser dan ketik alamat https://www.python.org/downloads/
2. Ikuti link untuk mendownload source code yang bisa digunakan oleh Unix/Linux
3. Download dan kemudian extrak file
4. Edit modules/setup file jika mau customize beberapa pengaturan
5. Jalankan perintah ./configure script
6. Installasi

Lokasi standar penginstallan python ada pada /usr/local/bin dan libraries nya berada /usr/local/lib/pythonXX dimana XX adalah versi python yang di install.

3.2.5 Setting Path pada Unix/Linux

Untuk menambahkan direktori python di path pada Unix

1. In the csh shell ketik setenv PATH PATH:/usr/local/bin/python lalu tekan enter
2. In the bash shell(Linux) ketik export ATH=PATH:/usr/local/bin/python lalu tekan enter
3. In the sh or ksh shell ketik PATH=PATH:/usr/local/bin/python lalu tekan enter
4. Note - /usr/local/bin/python adalah path untuk direktori Python

3.2.6 Fitur Python

Fitur-Fitur Python Python memiliki beberapa fitur yang menjadikan bahasa pemrograman ini berbeda dari bahasa lain antara lain :

1. Memiliki kepustakaan yang luas, dalam distribusi Python telah disediakan modul- modul siap pakai untuk berbagai keperluan.
2. Memiliki tata bahasa yang jernih dan mudah dipelajari.
3. Memiliki aturan layout kode sumber yang memudahkan
4. pengecekan, pembacaan kembali dan penulisan ulang kode sumber.
5. Berorientasi obyek.
6. Memiliki sistem pengelolaan memori otomatis (garbage collection, seperti java) Modular, mudah dikembangkan dengan menciptakan modul-modul baru; modul- modul tersebut dapat dibangun dengan bahasa Python maupun C/C++.
7. Memiliki fasilitas pengumpulan sampah otomatis, seperti halnya pada bahasa pemrograman Java, python memiliki fasilitas pengaturan penggunaan memory komputer sehingga para pemrogram tidak perlu melakukan pengaturan memory komputer secara langsung.
8. Memiliki banyak faslitas pendukung sehingga mudah dalam pengoprasiannya.

3.2.7 Pengujian

Cara pertama: ketik python untuk menjalankan Python Interpreter. Jika berhasil akan ditampilkan versi Python yang digunakan beserta interpreternya. Coba ketikkan print hello world untuk menampilkan teks hello world.

C:¿python Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec 5 2015, 20:40:30) [MSC v.1500 64 bit (AMD64)] on win32 Type ”help”, ”copyright”, ”credits” or ”license” for more information. ¿¿¿ print ”hello world” hello world ¿¿¿

3.2.8 Alasan Menggunakan Phyton

Python memiliki konsep desain yang bagus dan sederhana, yang berfokus pada kemudahan dalam penggunaan. Kode

Python dirancang untuk mudah dibaca, dipelajari, digunakan ulang, dan dirawat. Selain itu, Python juga mendukung pemograman berorientasi objek dan pemograman fungsional.

3.2.9 Tipe Data Phyton

Menggunakan tipe Number

Tipe data Number digunakan untuk menyimpan nilai-nilai numerik. Tipe ini merupakan tipe data immutable, yang artinya jika kita mengubah nilai dari sebuah data, maka kita akan mengalokasikan obyek baru. Sama seperti tipe data lainnya, obyek Number dibuat ketika kita memberikan sebuah nilai padanya. Contoh:

¿¿¿data = 1

Kita juga dapat mengubah nilai yang ada dalam variable data tersebut.

¿¿¿data = data + 1 ¿¿¿data=3.50 ¿¿¿floatdat = 7.5 ¿¿¿data = floatdat

Kita dapat menghapus sebuah obyek ataupun banyak obyek dengan menggunakan pernyataan del. Misalnya:

¿¿¿del data ¿¿¿del data, floatdat

Python mengelompokkan tipe Number dalam 4 macam, yaitu:

Plain Integer Plain integer atau bilangan bulat merupakan tipe data yang sering kita temui pada semua bahasa pemrograman. Integer ini mempunyai range nilai antara -2^32 sampai 2^311. Tipe ini juga dapat ditulis dalam bentuk octal (di tanda awalan 0) maupun hexadesimal (ditandai awalan 0x atau 0X). Contoh: 10 100 6542 -784 083 -042 -0x43 0X61

Long Integer Long integer sangat membantu kita untuk perhitungan di luar range nilai integer. Secara virtual, tidak ada batasan nilai tergantung besar virtual memory yang kita gunakan. Akhiran l atau L disetiap nilai bilangan bulat menandakan bahwa data tersebut bertipe long integer. 562718819L -0x526718L 012L -567299101L

Floating Point Real Number Tipe ini sering disebut sebagai tipe real (atau float). Tipe ini sama dengan tipe double di C. Nilai float mempunyai dua bagian, bagian titik desimal

dan bagian eksponensial. Tanda positif atau negatif diantara e merupakan tanda eksponen. Contoh nilai float: 0.0 14.5 -15.4 32.3+e18 -90.76712 -90. -32.54e100 70.2-E12

Complex Number Sebuah bilangan kompleks biasanya ditunjukkan oleh bentuk a + bj, dimana a adalah bagian real dan b adalah bagian imajiner. Bagian imajiner merupakan bilangan di awal tanda j atau J. Berikut ini contoh bilangan kompleks: 3.14j 45j 54.56+12.1J 3e+36J

Bagian real dan imajiner dari bilangan kompleks dapat kita pisahkan menggunakan data atribut, yaitu menggunakan real dan imag. Sedangkan untuk mendapatkan konjugasi dari bilangan kompleks tersebut, kita dapat menggunakan metode conjugate().

¿¿¿ kompleks = 23.45-1.23J ¿¿¿ kompleks.real 23.45 ¿¿¿ kompleks.imag -1.23 ¿¿¿ kompleks.conjugate() (23.45+1.23j)

String merupakan salah satu tipe data yang sering digunakan dalam pemrograman Python. Sebuah string dapat dinyatakan sebagai kumpulan karakter yang dibatasi oleh satu atau dua tanda petik. Inilah contohnya,

¿¿¿ nama = ”Klinik Python Indonesia” ¿¿¿ nama ’Klinik Python Indonesia’ ¿¿¿ slm = ’Salam Python Dahsyat!’ ¿¿¿ slm ’Salam Python Dahsyat!’ ¿¿¿ print slm Salam Python Dahsyat!

Dari contoh di atas, ketika kita memanggil variabel secara langsung maka akan ditampilkan isi dari variabel tersebut dengan sebuah tanda petik. Namun jika kita menggunakan pernyataan print, maka tanda petik tersebut akan dihilangkan.

Menampilkan Tanda Petik Sebagai String

Di dalam sebuah string tidak dapat berisi tanda petik yang sama dengan tanda petik yang digunakan oleh string tersebut. Misalkan, ketika kita menuliskan ’Py’thon’ maka akan muncul pesan kesalahan (syntax error). Agar tidak muncul pesan kesalahan, kita bisa mengganti tanda petik luarnya dengan tanda petik ganda, misalnya ”Py’thon”. Tanda petik juga dapat ditulis setelah tanda backslash agar dapat ditampilkan sebagai string.

¿¿¿ str = ”Py’thon” ¿¿¿ str ”Py’thon” ¿¿¿ str2 = ’Py”thon’ ¿¿¿ str2 ’Py”thon’ ¿¿¿ ”ÖK, s̈ampai ketemu lagi.” ’”OK, ” sampai ketemu lagi.’

Kode n atau Triple Quotes Juga Boleh!

Jika kita ingin menuliskan string yang panjang dalam beberapa baris, maka kita perlu menambahkan tanda backslash diikuti huruf n (n) sebagai tanda baris baru.

¿¿¿ teks = ”Python adalah bahasa pemrograman yang powerfull.nTerbukti Python bisa dijalankan di segala platform OS.nJadi, saatnya kita menggunakan Python sebagai bahasa permrograman nsehari-hari. Salam Python Dahsyat!”

Tanda n akan memberikan perintah membuat baris baru jika kita memanggil teks dengan pernyataan print.

¿¿¿ print teks Python adalah bahasa pemrograman yang powerfull. Terbukti Python bisa dijalankan di segala platform OS. Jadi, saatnya kita menggunakan Python sebagai bahasa permrograman sehari-hari. Salam Python Dahsyat!

Kita juga dapat menggunakan tanda petik tiga (triple quotes), ”””” atau ”’, untuk menuliskan sebuah string yang panjang dalam beberapa baris.

¿¿¿ teks = ””””Python adalah bahasa pemrograman yang powerfull. ... Terbukti Python bisa dijalankan di segala platform OS. ... Jadi, saatnya kita menggunakan Python sebagai bahasa pemrograman ... sehari-hari. Salam Python Dahsyat!””””

Tampilan teks dengan pernyataan print seperti di bawah ini,

¿¿¿ print teks Python adalah bahasa pemrograman yang powerfull. Terbukti Python bisa dijalankan di segala platform OS. Jadi, saatnya kita menggunakan Python sebagai bahasa pemrograman sehari-hari. Salam Python Dahsyat!

Menggabungkan String

Untuk menggabungkan dua buah string atau lebih, kita dapat menggunakan operator +. Sedangkan untuk menggandakan string, kita gunakan operator *.

¿¿¿ blog = ’Klinik’ + ’Python’ ¿¿¿ blog ’KlinikPython’ ¿¿¿ newblog = blog*5 ¿¿¿ newblog ’KlinikPython-

KlinikPythonKlinikPythonKlinikPythonKlinikPython’ ¿¿¿ blog *= 4 ¿¿¿ print blog KlinikPythonKlinikPython-KlinikPythonKlinikPython

Jika dua string ditulis secara berurutan, maka secara langsung kedua string tersebut akan digabungkan.

¿¿¿ blog = ’Klinik”Python’ ¿¿¿ blog ’KlinikPython’

Menentukan Panjang String

Panjang dari sebuah string dapat kita temukan dengan menggunakan fungsi len().

¿¿¿ len(blog) 12

Memecah String

Tidak seperti bahasa lainnya, Python tidak mendukung tipe Karakter. Untuk mengambil satu karakter atau lebih dari sebuah string, kita dapat memecah string tersebut menggunakan indeks (disebut Metode Irisan). Irisan terdiri dari dua indeks yang dipisahkan tanda koma.

¿¿¿ buah = ’Nanas’ ¿¿¿ buah[0] ’N’ ¿¿¿ buah[0:2] ’Na’ ¿¿¿ buah[0:4] ’Nana’ ¿¿¿ buah[0:5] ’Nanas’

Dari contoh di atas, panjang string buah adalah 5. Ketika kita menghitung maju, indeks bernilai 0 sampai (panjang-string 1) dimulai dari kiri ke kanan. Maka dari itu, kita dapat mengakses setiap karakter dalam range 0 sampai 4.

Sebuah string juga dapat dihitung mundur, dengan indeks -1 sampai (negatif panjang-string) dimulai dari kanan ke kiri. Berikut gambaran lengkapnya, baik itu penghitungan maju atau mundur.

Contoh penggunaan penghitungan mundur,

¿¿¿ buah[-1] s ¿¿¿ buah[-5] N ¿¿¿ buah[-5:-1] Nana ¿¿¿ buah[1:-1] ana

Jika kita lupa berapa nilai indeks awala atau indeks akhir, kita dapat kosongkan indeks tersebut.

¿¿¿ buah[:3] Nan ¿¿¿ buah[2:] nas

Pengosongan indeks akan menyebabkan semua string ditampilkan.

¿¿¿ buah[:] Nanas

String Bersifat Immutable

Tipe data String pada Python bersifat immutable, yang artinya sekali dibuat maka tidak dapat diubah kembali. Ada

pertanyaan bagus, jika string bersifat immutable, mengapa kita bisa mengubah nilai dari variabel string tersebut? Jawabannya sangat sederhana. Ketika kita memberikan nilai yang berbeda pada variabel string, sebuah obyek baru berhasil kita buat. Lihat contoh berikut,

¿¿¿ nama = ”Klinik” ¿¿¿ nama ’Klinik’ ¿¿¿ id(nama) 3076962016L ¿¿¿ nama = ”Python” ¿¿¿ id(nama) 3077289888L

Catat bahwa ketika string nama telah kita buat, maka identitas dari variabel ini dapat kita ketahui dengan menggunakan fungsi id(). Jika kita ubah nilai dari variabel nama tersebut, maka identitasnya juga berubah. Hal ini menandakan bahwa obyek baru telah dibuat. Penggantian nilai pada string pada posisi indeks tertentu akan menghasilkan pesan kesalahan.

¿¿¿ nama[0] = ’K’ Traceback (most recent call last): File ””, line 1, in TypeError: ’str’ object does not support item assignment

Kita juga dapat menambahkan sebuah string baru pada string lama.

¿¿¿ ’Si’ + nama[0] ’SiP’ ¿¿¿ ’J’ + nama[1:] ’Jython’

CHAPTER 4

BASIC SYNTAX

4.1 Basic Syntax

4.1.1 Pengenal Python

Pengenal Python Pengenal Python adalah nama yang digunakan untuk mengidentikasi variabel, fungsi, kelas, modulatauobjeklainnya. PengenaldimulaidenganhurufA sampai Z atau huruf a sampai z atau garis bawah diikuti oleh nol atau lebih huruf, garis bawah dan angka (0 sampai 9). Python tidak mengizinkan karakter tanda baca seperti , $, dan % dalam pengenal. Python adalah bahasa pemrograman yang sensitif. Dengan demikian, Tenaga Kerja dan Tenaga Kerja adalah dua pengidentikasi yang berbeda dengan Python. Berikut adalah konvensi penamaan untuk pengenal Python - Nama kelas dimulai dengan huruf besar. Semua pengenal lainnya mulai dengan huruf kecil. Memulai pengenal dengan

satu garis bawah terkemuka menunjukkan bahwa pengenal bersifat pribadi. Memulai pengenal dengan dua garis bawah terkemuka menunjukkan pengenal yang sangatpribadi. Jika pengenal juga diakhiri dengan dua tanda garis bawah, identier adalah nama khusus yang ditentukan bahasa. Bahasa Python memiliki banyak kesamaan dengan Perl, C, dan Java. Namun, ada beberapa perbedaan yang pasti antara bahasa. Program Python Pertama Mari kita jalankan program dalam mode pemrograman yang berbeda. Pemrograman Mode Interaktif Memohon interpreter tanpa melewatkan le script sebagai parameter menampilkan prompt berikut :

```
Python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2 Type help, c

Ketik teks berikut pada prompt Python dan tekan Enter:
Python 2.4.3 ( #1, Nov 11 2010, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)]
on linux2 Type help, copyright,
credits or license for more information.

Ketik teks berikut pada prompt Python dan
tekan Enter:


print Hello, Python!
```

Jika Anda menjalankan versi baru Python, Anda perlu menggunakan pernyataan cetak dengan tanda kurung seperti pada cetak `(Halo,Python!);`. Namun dengan versi Python 2.4.3, ini menghasilkan hasil sebagai berikut:

Hello, Python!

4.1.2 Pemrograman Mode Script

Memohon interpreter dengan parameter script memulai eksekusi script dan berlanjut sampai script selesai. Saat skrip selesai, juru bahasa tidak lagi aktif. Mari kita tuliskan pro-

gram Python sederhana dalam sebuah naskah. File Python memiliki ekstensi .py. Ketik kode sumber berikut di le. Objek Dengan Python, seperti semua bahasa berorientasi objek, ada kumpulan kode dan data yang disebut objek, yang biasanya mewakili potongan dalam model konseptual suatu sistem. Objek dengan Python dibuat (yaitu, instantiated) dari template yang disebut kelas (yang akan dibahas kemudian, sebanyak bahasa dapat digunakan tanpa memahami kelas). Mereka memiliki atribut, yang mewakili berbagai potongan kode dan data yang membentuk objek. Untuk mengakses atribut, seseorang menuliskan nama objek yang diikuti oleh suatu periode (selanjutnya disebut titik), diikuti dengan nama atribut. Contohnya adalah atribut atas dari string, yang mengacu pada kode yang mengembalikan salinan string di mana semua huruf adalah huruf besar. Untuk mendapatkan ini, perlu untuk memiliki cara untuk merujuk ke objek (dalam contoh berikut, jalan adalah string literal yang membangun objek).

4.1.3 Pengertian Python

Python adalah salah satu pemrograman yang terus berkembang dan bertahan dikarenakan dukungan komunitas yang aktif diseluruh dunia. Banyak forum-forum ataupun blogger-blogger yang sering membagi pengalaman dalam menggunakan python. Hal ini memudahkan bagi pengguna pemula maupun pengembang untuk bertanya dan sharing tentang ilmu pemrograman ini. Gambar berikut 4.1 adalah gambar logo python dengan lambang 2 ular python yang saling membelakangi. ular diatas berwarna biru dan ular dibawah berwarna kuning.

4.1.4 Kelebihan dan Kekurangan Python

Kelebihan yang dimiliki oleh Python :

- Tidak ada tahapan kompilasi dan penyambungan (link) sehingga kecepatan perubahan pada masa pembuatan sistem aplikasi meningkat.

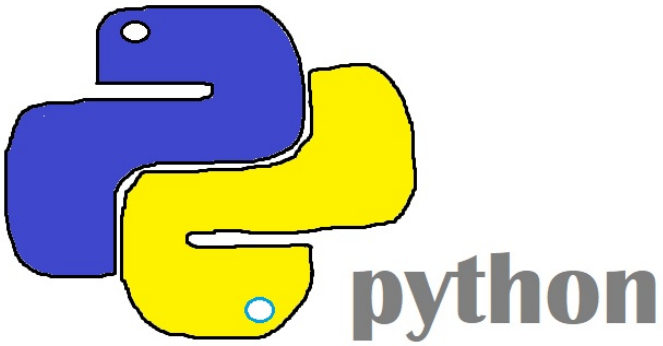


Figure 4.1 Gambar Logo Python

- Tidak ada deklarasi tipe data yang merumitkan sehingga program menjadi lebih sederhana, singkat, dan eksible.
- Manajemen memori otomatis yaitu kumpulan sampah memori sehingga dapat menghindari pencacatan kode.
- Tipe data dan operasi tingkat tinggi yaitu kecepatan pembuatan sistem aplikasi menggunakan tipe objek yang telah ada.
- Pemrograman berorientasi objek.
- Pelekatan dan perluasan dalam C.
- Terdapat kelas, modul, eksepsi sehingga terdapat dukungan pemrograman skala besar secara modular.
- Pemuatan dinamis modul C sehingga ekstensi menjadi sederhana dan berkas biner yang kecil
- Pemuatan kembali secara dinamis modul phyton seperti memodikasi aplikasi tanpa menghentikannya.
- Model objek universal kelas Satu.

- Konstruksi pada saat aplikasi berjalan.
- Interaktif, dinamis dan alamiah.
- Akses hingga informasi interpreter.
- Portabilitas secara luas seperti pemrograman antar platform tanpa ports.
- Kompilasi untuk portable kode byte sehingga kecepatan eksekusi bertambah dan melindungi kode sumber.
- Antarmuka terpasang untuk pelayanan keluar seperti perangkat Bantu system, GUI, persistence, database, dll.

Kekurangan yang dimiliki Python :

- Beberapa penugasan terdapat diluar dari jangkauan python, seperti bahasa pemrograman dinamis lainnya, python tidak secepat atau esien sebagai statis, tidak seperti bahasa pemrograman kompilasi seperti bahasa C.
- Disebabkan python merupakan interpreter, python bukan merupakan perangkat bantu terbaik untuk pengantar komponen performa kritis.
- Python tidak dapat digunakan sebagai dasar bahasa pemrograman implementasi untuk beberapa komponen, tetapi dapat bekerja dengan baik sebagai bagian depan skrip antarmuka untuk mereka.
- Python memberikan esiensi dan eksibilitas tradeoff by dengan tidak memberikannya secara menyeluruh. Python menyediakan bahasa pemrograman optimasi untuk kegunaan, bersama dengan perangkat bantu yang dibutuhkan untuk diintegrasikan dengan bahasa pemrograman lainnya.
- Banyak terdapat referensi lama terutama dari pencarian google, python adalah pemrograman yang sangat lambat. Namun belum lama ini ditemukan bahwa Google, Youtube, DropBox dan beberapa software sistem banyak menggunakan Python.

Dibalik kelebihan dan kekurangan yang dimiliki, Kini Python menjadi salah satu bahasa pemrograman yang populer digunakan oleh pengembangan web, aplikasi web, aplikasi perkantoran, simulasi, dan masih banyak lagi. Hal ini disebabkan karena Python bahasa pemrograman yang dinamis dan mudah dipahami. Python memiliki hak cipta. Seperti Perl, kode sumber Python sekarang tersedia di bawah GNU General Public License (GPL). Python sekarang dikelola oleh tim pengembangan inti di institut tersebut, walaupun Guido van Rossum masih memegangperanpentingdalammengarahkankemajuannya.

4.1.5 Fitur Python

Python memiliki fitur yang meliputi:

1. Mudah dipelajari: Python memiliki beberapa kata kunci, struktur sederhana, dan sintaks yang jelas. Hal ini memungkinkan siswa untuk mengambil bahasa dengan cepat.
2. Mudah dibaca: kode Python lebih jelas dan terlihat oleh mata.
3. Mudahdipelihara: kodesumberPythoncukupmudahuntuk dipelihara.
4. Perpustakaan standar yang luas: sebagian besar perpustakaan Python sangat portabel dan kompatibel dengan platformcross-platformdiUNIX,Windows,danMacintosh.
5. Mode Interaktif: Python memiliki dukungan untuk mode interaktif yang memungkinkan pengujian interaktif dan debugging dari cuplikan kode.
6. Portable: Python dapat berjalan di berbagai platform perangkat keras dan memiliki antarmuka yang sama pada semua platform.
7. Dapat diperpanjang: Anda dapat menambahkan modul tingkat rendah ke penerjemah Python. Modul ini memu-

ngkinkan programmer untuk menambahkan atau menyesuaikan alat mereka agar lebih esien.

8. Database: Python menyediakan antarmuka untuk semua database komersial utama.

4.1.6 Fitur Python

Fitur Python meliputi:

1. Mudah dipelajari: Python memiliki beberapa kata kunci, struktur sederhana, dan sintaks yang jelas. Hal ini memungkinkan siswa untuk mengambil bahasa dengan cepat.

2. Mudah dibaca: kode Python lebih jelas dan terlihat oleh mata.

3. Mudah dipelihara: kode sumber Python cukup mudah untuk dipelihara. Perpustakaan standar yang luas: sebagian besar perpustakaan. Python sangat portabel dan kompatibel dengan platform cross-platform di UNIX,Windows, dan Macintosh.

4. Mode Interaktif: Python memiliki dukungan untuk mode interaktif yang memungkinkan pengujian interaktif dan debugging dari cuplikan kode.

5. Portable: Python dapat berjalan di berbagai platform perangkat keras dan memiliki antarmuka yang sama pada semua platform.

6. Dapat diperpanjang: Anda dapat menambahkan modul tingkat rendah ke penerjemah Python. Modul ini memungkinkan programmer untuk menambahkan atau menyesuaikan alat mereka agar lebih efisien.

7. Database: Python menyediakan antarmuka untuk semua database komersial utama.

4.1.7 Pernyataan Multi-Line

Pernyataan di Python biasanya diakhiri dengan baris baru. Python, bagaimanapun, memungkinkan penggunaan karakter kelanjutan baris (
) untuk menunjukkan bahwa garis tersebut harus dilanjutkan. Misalnya

```
\subsection {Pernyataan Multi-Line}
Pernyataan di Python biasanya diakhiri dengan baris baru. Pyth
\begin {equation}

total = item_one + \
item_two + \
item_three
\end {verbatim}

Pernyataan yang ada di dalam kurung [], {}, atau () tidak perl

\begin{verbatim}
days = ['Monday', 'Tuesday', 'Wednesday',
'Thursday', 'Friday']
```

4.1.8 Kutipan pada Python

Python menerima kutipan tunggal (') ganda (") dan triple (''' atau """) untuk menunjukkan literal string, selama jenis kutipan yang sama dimulai dan mengakhiri string. Tanda kutip triple digunakan untuk membentang string di beberapa baris. Misalnya:

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
\end {verbatim}

\subsection{Komentar pada Python}
Tanda hash (\#)\ yang tidak berada di dalam string literal mem
\begin{verbatim}
```

```
#!/usr/bin/python

# First comment
print \"Hello, Python!\" # second comment
```

Dan hasilnya sebagai berikut :
Hello, Python!
Anda bisa mengetikkan komentar di baris yang sama setelah sebuah pernyataan atau ungkapan

```
name = \"Madisetti\" # This is again comment
```

Anda dapat mengomentari beberapa baris sebagai berikut

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

4.1.8.1 Variabel

$$days = [' Monday', ' Tuesday', ' Wednesday', ' Thursday', ' Friday'] \tag{4.1}$$

4.1.9 Kutipan pada Python

Python menerima kutipan tunggal (’), ganda (”) dan triple (’ ” atau ’” ”) untuk menunjukkan literal string, selama jenis kutipan yang sama dimulai dan mengakhiri string. Tanda kutip triple digunakan untuk membentang string di beberapa baris. Misalnya:

$$word =' word' sentence = "Thisisasentence. \tag{4.2}$$

4.1.10 Komentar pada Python

Tanda hash yang tidak berada di dalam string literal memulai sebuah komentar. Semua karakter setelah dan sampai akhir garis fisik adalah bagian dari komentar dan penafsir Python mengabaikannya.

$$!/usr/bin/pythonprint" Hello, Python!" \tag{4.3}$$

Dan hasilnya sebagai berikut

$$Hello, Python! \tag{4.4}$$

Anda bisa mengetikkan komentar di baris yang sama setelah sebuah pernyataan atau ungkapan

$$name = "Madisetti" This is again comment \tag{4.5}$$

Anda dapat mengomentari beberapa baris sebagai berikut

$$This is a comment. I said that already. \tag{4.6}$$

4.1.11 Variabel

Jadi variable itu untuk menyimpan data sebagai penampung nya dan nilai adalah isi dari variable itu sendiri. Contoh variable katakana saja my int = 7 itu juga bisa di ubah menjadi my int =3 Code merubah nilai variablenya My int = 7 My int = Print my int

4.1.12 Whitespace

Whitespace adalah penyusun kode untuk penulisan oleh karena itu harus berhati hati karena sangat sensitive Contoh code nya:

```
  Def spam()
Eggs + 12
  Return eggs
  Print spam()
```

Itu merupakan kode penulisan yang salah maka akan muncul pesan error. Pesan error yang muncul tentang penulisan dan bisa di edit agar tidak terjadi kesalahan penulisan lagi. Contoh yang benar :

```
  Def spam():
  Eggs = 12
  Return eggs
  Print spam()
```

Kita coba contohkan penulisan oprasi matematika nya. Code nya adalah:

```
Jumlah = 10+10
Print jumlah
Pejumlahan = 72 + 23
Pengurangan 108  204
Perkalian = 108 * 0,5
Pembagian = 108 / 9
```

Maka akan muncul print yang sesuai inputan yang kalian buat. Selain itu bisa juga perpangkatan

Kalkulator pangkatan meggunakan eight

Contohnya `eight = \2** 3\`

Eight itu adalah variable baru yang kita buat untuk menge-set nilai menjadi 8.

Contoh: stopwatch sederhana yang biasa nya di gunakan untuk ujian peraktek. Penjelasan tentang stopwatch yaitu alat untuk mengukur kecepatan suatu benda secara akurat. Sebelum membuat nya kita harus mengetahui cara kerja stopwatch itu sendiri. Memulai angka nya dari 0 karena angka 0 adalah angka pertama dalam waktu. Code nya an-tara lain:

```
Depanjam = 0
Jam = 0
Depanmenit = 0
Menit = 0
Depandetik = 0
Detik = 0
```

Lalu gunakan import time Untuk system loop jadi setelah detik mencapai 9 maka menit akan bertambah 1 seperti itu.

```
While true :
Time.sleep(1)
Detik += 1
If detik == 9:
```

```
Detik = 0
Depandetik += 1
If depandetik == 6:
Menit +=1
Depandetik = 0
Detik = 0
If menit == 9:
Menit = 0
Depanmenit += 1
If depanmenit == 6:
Jam +=1
Depanmenit = 0
Menit = 0
If ja, == 9 :
Depanjam += 1
Jam = 0
Print ( $ " $ $  {  $0 $  }  $ $  {  $1 $  }  $ $  {  $2 $  }
```

Run maka akan jadi program simple tentang stopwatch sederhana.

CHAPTER 5

VARIABEL TYPE

5.1 Python Variabel Type

Satu dari fitur yang paling powerful dari sebuah bahasa pemrograman adalah kemampuan untuk memanipulasi variabel. Sebuah variabel adalah nama yang merujuk ke sebuah nilai. Variabel tidak lain hanyalah lokasi memori yang dipesan untuk menyimpan nilai. Ini berarti bahwa ketika kita membuat variabel, maka kita memesan beberapa ruang di memori. Berdasarkan tipe data sebuah variabel, penafsir mengalokasikan memori dan memutuskan apa yang dapat disimpan dalam memori yang dipesan. Oleh karena itu, dengan menetapkan tipe data yang berbeda ke variabel, kita dapat menyimpan bilangan bulat, desimal atau karakter dalam variabel ini. Pernyataan pemberian nilai(assigment statement) akan memberikan nilai pada variabel:

```
pesan = "Apa kabar, bro ?"
n = 17
pi = 3.14159
```

Contoh diatas melakukan tiga pemberian nilai. Yang pertama memberikan nilai string Äpa kabar, bro ?ïada variabel bernama pesan. Yang Kedua memberikan nilai integer 17 kepada n, dan yang ketiga memberikan nilai bilangan floating-point 3.14159 kepada variabel dengan nama pi. Token pemberian nilai, tanda =, agar tidak bingung jangan disamakan dengan tanda sama dengan, yang mana menggunakan token ==. Pernyataan pemberian nilai mengikat sebuah nama di sebelah kiri dari operator, dan nilainya, di sebelah kanannya. Inilah mengapa kamu akan mendapatkan error jika kamu menulis:

```
17 = n
File "<interactive input>", line 1
SyntaxError: can't assign to literal
```

Ketika membaca atau menulis kode, katakan dalam hati ñ diberikan nilai 17.· Jangan katakan ñ sama dengan 17.· Cara umum untuk merepresentasikan variabel pada kertas adalah dengan menulis namanya dengan tanda panah mengarah ke nilai variabelnya. Gambar jenis ini dinamakan state snapshot karena ia memperlihatkan state atau kondisi dari setiap variabel pada instan waktu tertentu. (Pikirkan ini sebagai variabel keadaan pikiran). Diagram ini memperlihatkan hasil dari pengeksekusian pernyataan pemberian nilai. Jika kamu meminta interpreter untuk menilai sebuah variabel, ia akan menghasilkan nilai dari variabel terkait pada waktu sekarang. 'Apa kabar, bro ?' n 17 pi 3.14159 Kita menggunakan variabel-variabel pada program untuk mengingat hal-hal, misalnya skore terkini ketika sedang ada pertandingan sepak bola. Tapi variabel tetaplah variabel. Ini artinya mereka bisa berganti seiring waktu, sama halnya dengan papan skor pada pertandingan bola. Kamu bisa memberikan nilai pada variabel, dan kemudian memberikan nilai lainnya pada variabel yang sama. (Ini berbeda dari sudut pandang matematika. Pada matematika, jika kamu memberi 'x' ni-

lai 3, itu tidak bisa mengubah link nilai menjadi nilai yang berbeda pada pertengahan perhitungan yang kamu lakukan!)

```
hari = "Kamis"
hari
'Kamis'
hari = "Jumat"
hari
'Jumat'
hari = 21
hari
21
```

Kamu akan menyadari kita mengubah nilai dari hari sebanyak tiga kali, dan ketika pemberian nilai yang ketiga kita bahkan membuatnya merujuk pada nilai yang berbeda tipe dari yang sebelumnya. Pemrograman itu kebanyakan tentang bagaimana komputer mengingat sesuatu, misal, Jumlah panggilan tak terjawab pada handphone mu,dan kemudian mengatur untuk memperbaharui atau merubah variabel ketika kamu melewatkan panggilan lainnya.

5.1.1 Nama Variabel dan Keywords

Nama variabel bisa ditulis panjang. Mereka bisa berisi huruf maupun digit angka, tapi harus diawali dengan huruf atau underscore. Meskipun dibolehkan untuk menggunakan huruf besar, tapi pada umumnya kita tidak menggunakannya. Jika kamu menggunakannya, ingat kalau besar kecilnya huruf itu berpengaruh. Wayandan dan wayan itu merupakan dua variabel yang berbeda.

Karakter underscore bisa ada pada nama. Biasanya digunakan pada nama yang terdiri dari lebih dari satu kata, misalnya seperti `nama atau hargajualproduk.`

Ada beberapa situasi yang mana nama yang diawali dengan underscore memiliki arti yang spesial, jadi aturan yang paling aman untuk pemula adalah memulai sebuah nama hanya dengan menggunakan huruf kecil. Jika kamu memberikan variabel nama yang ilegal, kamu akan mendapatkan syntax error:

```
123doremi = "tiga not awal"
SyntaxError: invalid syntax
gaji  \$   = 1000000
SyntaxError: invalid syntax
class = "Kewirausahaan 121"
SyntaxError: invalide syntax
```

5.1.2 Tipe data standar

Data yang tersimpan dalam memori bisa bermacam-macam. Misalnya, usia seseorang disimpan sebagai nilai numerik dan alamatnya disimpan sebagai karakter alfanumerik. Python memiliki berbagai jenis data standar yang digunakan untuk menentukan operasi yang mungkin dilakukan pada mereka dan metode penyimpanan untuk masing-masingnya.

Python memiliki lima tipe data standar -

- Angka
- Tali
- Daftar
- Tuple
- Kamus

123doremi adalah ilegal karena tidak dimulai dengan huruf. gaji $ juga ilegal karena memakai karakter ilegal, tanda dollar seharusnya tidak boleh. Tapi apa yang salah dengan class ? Ternyata karena class merupakan satu dari keyword (kata kunci) yang dimiliki Python. Keyword mendefinisikan aturan syntax bahasa dan struktur, dan maka dari itu tidak diperkenakan untuk digunakan sebagai nama variabel. Python memiliki tiga puluhan keyword (dan hingga kini Python masih meningkatkannya dengan memperkenalkan atau menghilangkan satu atau dua). Kamu mungkin berniat untuk menyimpan daftar ini untuk mempermudah. Jika si interpreter komplain mengenai satu sari nama variabel mu dan kamu tidak tahu mengapa demikian, lihatlah apakah nama variabel yang kamu buat masuk daftar diatas. Jika iya, ganti dengan nama yang lain. Programer umumnya memilih

nama untuk variabel mereka agar memiliki arti dan bisa dimengerti oleh pembaca manusia −− dengan demikian maka akan membantu si programer untuk mendokumentasikan, atau mengingat, apa gunanya variabel tersebut.

Pemula biasanya bingung dengan maksud dari berguna untuk pembaca manusia dengan berguna untuk mesin atau komputer. Jadi mungkin mereka akan berpikir salah, bahwa ketika mereka memanggil beberapa variabel dengan nama average ataupi, itu akan dengain ajaib menghitung sebuah average / rata-rata, atau dengan ajaib tahu kalau pi memiliki nilai seperti 3.14159. Tidak ! Komputer tidak akan mengerti itu, komputer tidak akan mengerti apa yang kamu inginkan dari variabel hanya karena namanya unik.

Jadi kamu mungkin akan menemui beberapa instruktur yang memang sengaja tidak memilih nama yang berarti ketika mereka mengajar pemula bukan karena kita tidak menganggap itu merupakan kebiasaan yang baik, tapi karena kita mencoba untuk menekankan ulang pesannya kepada kalian seorang programer harus menulis sendiri kode program untuk menghitung average-nya, dan kamu harus menulis pernyataan pemberian nilai untuk memberikan nilai pada variabel pidengan nilai yang kamu inginkan.

5.1.3 Pernyataan

Sebuah pernyataan (statement) adalah perintah/instruksi yang bisa dieksekusi/dijalankan oleh Python interpreter. Hingga kini kita hanya baru melihat pernyataan pemberian nilai. Beberapa jenis lain dari pernyataan yang akan kita lihat dengan singkat adalah pernyataan while, pernyataan for, pernyataan if, dan pernyataan import. (Ada banyak lagi yang lainnya!). Ketika kamu mengetikan pernyataan pada command line, Python akan menjalankannya. Pernyataannya sendiri tidak menghasilkan hasil apapun. Variabel tidak lain hanyalah lokasi memori reserved untuk menyimpan nilai. Ini berarti bahwa ketika Anda membuat variabel Anda memesan beberapa ruang di memori. Berdasarkan tipe data sebuah variabel, penafsir mengalokasikan memori dan memutuskan apa yang dapat disimpan dalam memori yang

dipesan. Oleh karena itu, dengan menetapkan tipe data yang berbeda ke variabel, Anda dapat menyimpan bilangan bulat, desimal atau karakter dalam variabel ini.

5.1.3.1 Variabel Variabel adalah lokasi memori yang dicadangkan untuk menyimpan nilai-nilai. Ini berarti bahwa ketika Anda membuat sebuah variabel Anda memesan beberapa ruang di memori. Variabel menyimpan data yang dilakukan selama program dieksekusi, yang natinya isi dari variabel tersebut dapat diubah oleh operasi - operasi tertentu pada program yang menggunakan variabel. Penulisan variabel Python sendiri juga memiliki aturan tertentu, yaitu :

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

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

5.1.3.2 Menilai Ekspresi Sebuah ekspresi merupakan perpaduan antara nilai, variabel, operator, dan pemanggilan fungsi. Jika kamu mengetik sebuah ekspresi pada Python prompt, maka si interpreter akan menilainya dan menampilkan hasilnya:

```
1 + 1
2
len(hello)
5
```

Pada contoh ini len merupakan fungsi built-in yang ada di Python yang akan menghasilkan jumlah karakter dari sebuah string. Sebelumnya kita sudah melihat fungsi print dan type,

jadi ini adalah contoh fungsi ketiga kita. Proses penilaian dari sebuah ekspresi akan menghasilkan sebuah nilai, itulah mengapa ekspresi bisa ada di sisi sebelah kanan dari pernyataan pemberian nilai. Nilai dengan sendirinya adalah ekspresi sederhana, dan begitu juga variabel.

```
17
17
y = 3.14
x = len(hello)
x
5
Y

3.14
```

5.1.3.3 Menetapkan Nilai ke Variabel Variabel Python tidak memerlukan deklarasi eksplisit untuk memesan ruang memori. Deklarasi terjadi secara otomatis saat Anda menetapkan nilai ke variabel.Tanda sama = digunakan untuk menetapkan nilai pada variabel. Operand di sebelah kiri = operator adalah nama variabel dan operan di sebelah kanan = operator adalah nilai yang tersimpan dalam variabel.Misalnya:

```
counter˜=˜100˜˜˜˜˜˜˜    An integer assignment
miles˜˜˜=˜1000.0˜˜˜˜    A floating point
name˜˜˜˜=˜"John"˜˜˜˜    A string
print counter
print miles
print name
```

Di sini, 100, 1000.0 dan J̈ohnädalah nilai yang diberikan untuk melawan, mil, dan variabel nama masing-masing.Ini menghasilkan hasil sebagai berikut: 100 1000.0 John Beberapa Tugas Python memungkinkan Anda untuk menetapkan nilai tunggal ke beberapa variabel secara bersamaan. Misalnya:

```
a = b = c = 1
```

Di sini, sebuah objek bilangan bulat dibuat dengan nilai 1, dan ketiga variabel ditugaskan ke lokasi memori yang

sama.Anda juga dapat menetapkan beberapa objek ke beberapa variabel. Misalnya: `a,b,c = 1,2,"john"` Di sini, dua objek bilangan bulat dengan nilai 1 dan 2 masing-masing diberikan pada variabel a dan b masing-masing, dan satu objek string dengan nilai ̈johnd̈iberikan ke variabel c.

5.1.3.4 Tipe data standar Data yang tersimpan dalam memori bisa bermacam-macam. Misalnya, usia seseorang disimpan sebagai nilai numerik dan alamatnya disimpan sebagai karakter alfanumerik. Python memiliki berbagai jenis data standar yang digunakan untuk menentukan operasi yang mungkin dilakukan pada mereka dan metode penyimpanan untuk masing-masing metode. Python memiliki lima tipe data standar :

1. Angka
2. Tali
3. Daftar
4. Tuple
5. Kamus

5.1.3.5 Nomor Python Nomor tipe data menyimpan nilai numerik. Nomor objek dibuat saat Anda memberikan nilai pada mereka. Misalnya:

```
var1 = 1
var2 = 10
```

Anda juga dapat menghapus referensi ke objek nomor dengan menggunakan del statement. Sintaks dari pernyataan del adalah `del var1[,var2[,var3[....,varN]]]` Anda dapat menghapus satu objek atau beberapa objek dengan menggunakan pernyataan del. Misalnya:

```
del var
del var a, var b
```

Python mendukung empat jenis numerik yang berbeda:

1. int (bilangan bulat yang ditandatangani)
2. Panjang (bilangan bulat panjang, mereka juga bisa diwakili dalam oktal dan heksadesimal)
3. float (floating point real value)
4. kompleks (bilangan kompleks)

Python memungkinkan Anda untuk menggunakan huruf kecil l dengan panjang, tapi disarankan agar Anda hanya menggunakan huruf besar L untuk menghindari kebingungan dengan nomor 1. Python menampilkan bilangan bulat panjang dengan huruf besar L. Sebuah bilangan kompleks terdiri dari sepasang bilangan floating-point yang diinisialisasi langsung yang dinotasikan dengan x + yj, di mana x dan y adalah bilangan real dan j adalah unit imajiner.

5.1.3.6 String Python String dengan Python diidentifikasi sebagai kumpulan karakter bersebelahan yang ditunjukkan dalam tanda petik.Python memungkinkan untuk kedua pasang tanda kutip tunggal atau ganda.Subset string dapat diambil dengan menggunakan operator slice ([] dan [:]) dengan indeks mulai dari 0 pada awal string dan bekerja dengan cara mereka dari -1 di akhir. Tanda plus + adalah operator concatenation string dan tanda bintang ∗ adalah operator pengulangan. Misalnya:

```
str = 'Hello World!'
print˜str˜˜˜˜˜˜˜˜  Prints complete string
print str[0]˜˜˜˜˜˜  Prints first character of the string
print str[2:5]˜˜˜˜  Prints characters starting from 3rd to 5th
print str[2:]˜˜˜˜˜  Prints string starting from 3rd character
print˜str˜*˜2˜     Prints string two times
print str + "TEST"  \#  Prints concatenated string
```

Ini akan menghasilkan hasil sebagai berikut: Hello World! H llo llo World! Hello World!Hello World! Hello World!TEST

5.1.3.7 Daftar Python Daftar adalah jenis data majemuk Python yang paling serbaguna. Daftar berisi item yang dipisahkan

dengan tanda koma dan dilampirkan dalam tanda kurung siku []. Sampai batas tertentu, daftar serupa dengan array di C. Salah satu perbedaan di antara keduanya adalah bahwa semua item yang termasuk dalam daftar dapat terdiri dari tipe data yang berbeda. Nilai yang tersimpan dalam daftar dapat diakses menggunakan operator slice ([] dan [:]) dengan indeks mulai dari 0 di awal daftar dan bekerja dengan cara mereka untuk mengakhiri -1. Tanda plus + adalah daftar operator concatenation, dan asterisk * adalah operator pengulangan. Misalnya :

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']
print list          Prints complete list
print list[0]       Prints first element of the list
print list[1:3]     Prints elements starting from 2nd till 3r
print list[2:]      Prints elements starting from 3rd element
print tinylist * 2  Prints list two times
print list + tinylist  Prints concatenated lists
```

Ini menghasilkan hasil sebagai berikut:

```
['abcd', 786, 2.23, 'john', 70.200000000000003]
[786, 2.23]
[2.23, 'john', 70.200000000000003]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.200000000000003, 123, 'john']
```

5.1.3.8 Tupel Pytho Sebuah tupel adalah jenis data urutan lain yang serupa dengan daftar.Sebuah tupel terdiri dari sejumlah nilai yang dipisahkan dengan koma. Tidak seperti daftar, bagaimanapun, tupel tertutup dalam tanda kurung. Perbedaan utama antara daftar dan tupel adalah: Daftar tertutup dalam tanda kurung [] dan elemen dan ukurannya dapat diubah, sementara tupel dilampirkan dalam tanda kurung () dan tidak dapat diperbarui.Tupel bisa dianggap sebagai daftar hanya-baca. Misalnya:

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
tinytuple = (123, 'john')
print tuple          Prints complete list
```

```
print tuple[0]~~~~~~~  Prints first element of the list
print tuple[1:3]~~~~~  Prints elements starting from 2nd till
print tuple[2:]~~~~~~  Prints elements starting from 3rd eleme
print~tinytuple~* 2    Prints list two times
print~tuple~+~tinytuple     Prints concatenated lists
```

Ini menghasilkan hasil sebagai berikut:

```
(\'abcd\', 786, 2.23, \'john\', 70.200000000000003)
abcd
(786, 2.23)
(2.23, \'john\', 70.200000000000003)
(123, 'john', 123, 'john')
(\'abcd\', 786, 2.23, 'john', 70.200000000000003, 123, 'john')
```

Kode berikut tidak valid dengan tupel, karena kami mencoba memperbarui tupel, yang tidak diizinkan.Kasus serupa dimungkinkan dengan daftar:

```
tuple = ( 'abcd',~786 , 2.23, 'john', 70.2  )
list = [ 'abcd', 786 ,~2.23, 'john', 70.2  ]
tuple[2]~=~1000~    Invalid syntax with tuple
list[2]~=~1000~~    Valid syntax with list
```

5.1.3.9 Kamus Python Kamus Python adalah jenis tipe tabel hash. Mereka bekerja seperti array asosiatif atau hash yang ditemukan di Perl dan terdiri dari pasangan kunci-nilai. Kunci kamus bisa hampir sama dengan tipe Python, tapi biasanya angka atau string. Nilai, di sisi lain, bisa menjadi objek Python yang sewenang-wenang. Kamus ditutupi oleh kurung kurawal dan nilai dapat diberikan dan diakses menggunakan kawat gigi persegi []. Misalnya:

```
dict =  {  }
dict['one'] = "This is one"
dict[2]~~~~ = "This is two"
tinydict = {  'name': 'john','code':6734, 'dept': 'sales' }
print dict['one']~~ ~~~  Prints value for 'one' key
print dict[2]~~~~~~~~~~  Prints value for 2 key
print~tinydict~~~~~~~~   Prints complete dictionary
print tinydict.keys()~~  Prints all the keys
```

```
print˜tinydict.values()   Prints all the values
```

Ini menghasilkan hasil sebagai berikut:

```
This is one
This is two
{  'dept': 'sales', 'code': 6734, 'name': 'john' }
['dept', 'code', 'name']
['sales', 6734, 'john']
```

Kamus tidak memiliki konsep keteraturan antar elemen. Tidak benar mengatakan bahwa unsur-unsurnyaṙusak̈;.Mereka hanya unordered.

5.1.4 Konversi Tipe Data

Terkadang, Anda mungkin perlu melakukan konversi antara jenis built-in. Untuk mengonversi antar jenis, Anda cukup menggunakan nama jenis sebagai fungsi. Ada beberapa fungsi built-in untuk melakukan konversi dari satu tipe data ke tipe data yang lain. $ Fungsi ini mengembalikan objek baru yang mewakili nilai yang dikonversi.

CHAPTER 6

BASIC OPERATOR

6.1 Python Basic Operator

Operator adalah konstruksi yang dapat memanipulasi nilai operan.

Perhatikan ungkapan 4 + 5 = 9. Di sini, 4 dan 5 disebut operan dan + disebut operator.

6.1.1 Jenis Operator

Bahasa Python mendukung jenis operator berikut.

1. Operator Aritmatika

Figure 6.1 Contoh Operator Aritmatika

2. Operator Perbandingan (Relasional)
3. Operator Penugasan
4. Operator Logis
5. Bitwise Operator
6. Operator keanggotaan
7. Operator Identitas

Mari kita lihat semua operator satu per satu.

Operator Aritmatika Python

Asumsikan variabel a memegang 10 dan variabel b memegang 20, maka

+ Tambahan
Menambahkan nilai di kedua sisi operator.
Contoh A + b = 30

- Pengurangan
Kurangi operan tangan kanan dari operan tangan kiri.
Contoh a - b = -10

* Perkalian
Kalikan nilai di kedua sisi operator
Contoh a * b = 200

/ Divisi
Membagi operan tangan kiri dengan tangan kanan operan
Contoh B / a = 2

% Modulus

Membagi operan tangan kiri dengan operan tangan kanan dan mengembalikan sisa
Contoh B % a = 0

** Eksponen
Melakukan perhitungan eksponensial (daya) pada operator
Contoh A ** b = 10 ke daya 20

//

Divisi Lantai - Pembagian operan dimana hasilnya adalah hasil bagi di mana angka setelah titik desimal dikeluarkan. Tapi jika salah satu operan negatif, hasilnya berlantai, yaitu terbulatkan dari nol (menuju negatif tak terbatas):

9 // 2 = 4 dan 9.0 // 2.0 = 4.0, -11 // 3 = -4, -11.0 // 3 = -4.0

Operator Perbandingan Python

Operator ini membandingkan nilai di kedua sisi dan memutuskan hubungan di antara keduanya. Mereka juga disebut operator relasional.

Asumsikan variabel a memegang 10 dan variabel b memegang 20, maka

==
Jika nilai dua operan sama, maka kondisinya menjadi benar.
Contoh (a == b) tidak benar

! =
Jika nilai dua operan tidak sama, maka kondisinya menjadi benar.

¿
Jika nilai operan kiri lebih besar dari nilai operan kanan, maka kondisinya menjadi benar.
Contoh (a¿ b) tidak benar

¡

Jika nilai operan kiri kurang dari nilai operan kanan, maka kondisinya menjadi benar.
Contoh (A ¡b) adalah benar.

¿ =
Jika nilai operan kiri lebih besar dari atau sama dengan nilai operan kanan, maka kondisinya menjadi benar.
Contoh (a¿ = b) tidak benar

¡=
Jika nilai operan kiri kurang dari atau sama dengan nilai operan kanan, maka kondisinya menjadi benar.
Contoh (a ¡= b) adalah benar.

Operator Penugasan Python

Asumsikan variabel a memegang 10 dan variabel b memegang 20, maka -

=
Menetapkan nilai dari operan sisi kanan ke operan sisi kiri

Contoh : c = a + b memberi nilai a + b ke c
+ = Tambahkan DAN

Ini menambahkan operan kanan ke operan kiri dan menetapkan hasilnya ke operan kiri
Contoh : c + = a setara dengan c = c + a
- = Kurangi DAN

Ini mengurangi opera kanan dari operan kiri dan menetapkan hasilnya ke operan kiri
Contoh : c - = a setara dengan c = c a
* = Multiply DAN

Ini mengalikan operand kanan dengan operan kiri dan menetapkan hasilnya ke operan kiri
Contoh : c * = a setara dengan c = c * a
/ = Bagilah dan

Ini membagi operan kiri dengan operan kanan dan menetapkan hasilnya ke operan kiri
Contoh : C / = a adalah setara dengan c = c / ac / = a adalah setara dengan c = c / a
% = Modulus DAN

Dibutuhkan modulus menggunakan dua operan dan menetapkan hasilnya ke operan kiri
Contoh : C % = a setara dengan c = c % a
** = Eksponen DAN

Melakukan perhitungan eksponensial (daya) pada operator dan memberikan nilai pada operan kiri
Contoh : C ** = a setara dengan c = c ** a
// Divisi Lantai

Ini melakukan pembagian lantai pada operator dan memberikan nilai ke operan kiri
Contoh : C // = a sama dengan c = c // a
Operator Bitwise Python

Operator Bitwise bekerja pada bit dan melakukan operasi bit by bit. Asumsikan jika a = 60; Dan b = 13; Sekarang dalam format biner mereka akan menjadi seperti berikut -

```
a = 0011 1100
b = 0000 1101
a & b = 0000 1100
A | b = 0011 1101
a ^ b = 0011 0001
~ a = 1100 0011
```

Ada beberapa operator Bitwise berikut yang didukung oleh bahasa Python

& Biner DAN
Operator menyalin sedikit ke hasil jika ada di kedua operan
Contoh : (A & b) (berarti 0000 1100)

| Biner ATAU

Ini salinan sedikit jika ada di salah satu operan.
Contoh : (A | b) = 61 (berarti 0011 1101)

ˆ Biner XOR
Ini salinan bit jika diatur dalam satu operand tapi tidak keduanya.
Contoh : (A ˆ b) = 49 (berarti 0011 0001)

∼ Binary Ones Complement
Ini tidak mencolok dan memiliki efek bit 'flipping'.
Contoh : (∼ A) = -61 (berarti bentuk pelengkap 1100 0011 dalam 2 karena nomor biner yang ditandatangani

¡¡ Pergeseran Kiri Biner
Nilai operan kiri dipindahkan ke kiri oleh jumlah bit yang ditentukan oleh operan kanan.
Contoh : Sebuah ¡¡ = 240 (berarti 1111 0000)

¿¿ Binary Right Shift
Nilai operan kiri dipindahkan tepat dengan jumlah bit yang ditentukan oleh operan kanan.
Contoh : a ¿¿ = 15 (berarti 0000 1111)

Operator Logika Python

Ada beberapa operator logis berikut yang didukung oleh bahasa Python. Asumsikan variabel a memegang 10 dan variabel b memegang 20 kemudian

Digunakan untuk membalik keadaan logis operannya.

Keanggotaan Python Operator

Operator keanggotaan Python menguji keanggotaan secara berurutan, seperti senar, daftar, atau tupel. Ada dua operator keanggotaan seperti yang dijelaskan di bawah ini

Di
Mengevaluasi ke true jika menemukan sebuah variabel dalam urutan yang ditentukan dan false sebaliknya.

Contoh : X di y, di sini menghasilkan 1 jika x adalah anggota dari urutan y.

tidak masuk
Mengevaluasi ke true jika tidak menemukan variabel dalam urutan yang ditentukan dan false sebaliknya.
Contoh : X tidak di y, di sini tidak menghasilkan 1 jika x bukan anggota urutan y.

Operator Identitas Python

Operator identitas membandingkan lokasi memori dari dua objek. Ada dua operator Identitas yang dijelaskan di bawah ini:

aku s
Mengevaluasi ke true jika variabel di kedua sisi operator menunjuk ke objek yang sama dan salah sebaliknya.
Contoh: X adalah y, di sini adalah hasil dalam 1 jika id (x) sama dengan id (y).

Tidak
Mengevaluasi false jika variabel di kedua sisi operator menunjuk ke objek yang sama dan benar sebaliknya.
Contoh: X bukan y, ini bukan hasil 1 jika id (x) tidak sama dengan id (y).

Operator Python Diutamakan

berikut mencantumkan semua operator dari preseden tertinggi sampai yang terendah.

**
Eksponensiasi (naik ke tampuk kekuasaan)

~ + -
Pelengkap, unary plus dan minus (nama metode untuk dua yang terakhir adalah + @ dan - @)

* / % //

Kalikan, bagi, modulo dan pembagian lantai

+ -
Penambahan dan pengurangan

¿¿ ¡¡
Pergeseran bitwise kanan dan kiri

&
Bitwise 'DAN'

^ |
Bitwise eksklusif 'OR 'dan reguler' OR'

¡= ¡¿¿ =
Operator perbandingan

¡¿ ==! =
Operator kesetaraan

= % = / = // = - = + = * = ** =
Operator penugasan

Bukan
Operator identitas

Bukan di
Operator keanggotaan

Tidak atau dan
Operator logika

Peran operator dalam proses perhitungan matematika sangatlah penting. Selain operator Aritmatika, Python juga mendukung operator berkondisi yang berfungsi untuk membandingkan suatu nilai dengan nilai yang lain. Operator-operator yang didukung oleh Python yaitu operator Unari (+ dan) dan operator Binari (+, -, *, /, %, dan **). Pada ekspresi Aritmatika berikut: x = y + z

y dan z disebut sebagai operan dari operator +. Tabel di bawah ini menjelaskan tentang berbagai macam operator yang digunakan untuk segala perhitungan di Python.

Jika sebuah ekspresi melibatkan lebih dari satu operator, Python secara otomatis akan memilih operator mana yang akan diutamakan dahulu. Sebagai contoh:

```
>>> x=7+3*6
>>> x
25
>>> y=100/4*5
>>> y
125
```

Operator ** memiliki urutan tertinggi diantara operator lainnya. Operator * mempunyai urutan lebih tinggi daripada operator +, dan operator / mempunyai urutan yang sama dengan operator *. Pada ekspresi x = 7 + 3 * 6, bagian 3 * 6 akan dieksekusi pertama kali menghasilkan 18, yang kemudian ditambahkan dengan 7. Sedangkan ekspresi y = 100/4*5, bagian 100/4dieksekusi terlebih dahulu karena operator / berada disebelah kiri dari operator *.

Kita dapat mengubah urutan prioritas dari operator Aritmatika dengan menggunakan kurung-buka-kurung-tutup (). Operator () memiliki urutan tertinggi diantara tiga tipe lainnya. Operator () mempunyai urutan dari kiri ke kanan pada ekspresi di dalamnya. Berikut ini contohnya:

¿¿¿ x=(7+3)*6 ¿¿¿ x 60 ¿¿¿ y=100/(4*5) ¿¿¿ y 5 ¿¿¿

z=7+(5*(8/2)+(4+6)) ¿¿¿ z 37

Operator modulus % akan memberikan nilai sisa dari pembagian integer. Berikut contoh penggunaan operator modulus:

¿¿¿ 7 % 3 1 ¿¿¿ 0 % 3 0 ¿¿¿ 1.0 % 3.0 1.0

Operator eksponensial akan memberikan nilai pangkat dari suatu bilangan. Contoh penggunaan operator eksponensial:

¿¿¿ 5**2 25 ¿¿¿ 5**-2 0.04 ¿¿¿ -5**2 -25 ¿¿¿ (-5)**2 25

Operator Penugasan Python

Asumsikan variabel a memegang 10 dan variabel b memegang 20, maka -

=
Menetapkan nilai dari operan sisi kanan ke operan sisi kiri

Contoh : c = a + b memberi nilai a + b ke c
+ = Tambahkan DAN

Ini menambahkan operan kanan ke operan kiri dan menetapkan hasilnya ke operan kiri
Contoh : c + = a setara dengan c = c + a
- = Kurangi DAN

Ini mengurangi opera kanan dari operan kiri dan menetapkan hasilnya ke operan kiri
Contoh : c - = a setara dengan c = c a
 * = Multiply DAN

Ini mengalikan operand kanan dengan operan kiri dan menetapkan hasilnya ke operan kiri
Contoh : c * = a setara dengan c = c * a
 / = Bagilah dan

Ini membagi operan kiri dengan operan kanan dan menetapkan hasilnya ke operan kiri
Contoh : C / = a adalah setara dengan c = c / ac / = a adalah setara dengan c = c / a
 % = Modulus DAN

Dibutuhkan modulus menggunakan dua operan dan menetapkan hasilnya ke operan kiri
Contoh : C % = a setara dengan c = c % a
 ** = Eksponen DAN

Melakukan perhitungan eksponensial (daya) pada operator dan memberikan nilai pada operan kiri
Contoh : C ** = a setara dengan c = c ** a
 // Divisi Lantai

Ini melakukan pembagian lantai pada operator dan memberikan nilai ke operan kiri
Contoh : C // = a sama dengan c = c // a
 Operator Bitwise Python

Operator Bitwise bekerja pada bit dan melakukan operasi bit by bit. Asumsikan jika a = 60; Dan b = 13; Sekarang dalam format biner mereka akan menjadi seperti berikut -

a = 0011 1100

b = 0000 1101

a & b = 0000 1100

A | b = 0011 1101

a ^ b = 0011 0001

~ a = 1100 0011

Ada beberapa operator Bitwise berikut yang didukung oleh bahasa Python

& Biner DAN
Operator menyalin sedikit ke hasil jika ada di kedua operan
Contoh : (A & b) (berarti 0000 1100)

| Biner ATAU
Ini salinan sedikit jika ada di salah satu operan.
Contoh : (A | b) = 61 (berarti 0011 1101)

^ Biner XOR
Ini salinan bit jika diatur dalam satu operand tapi tidak keduanya.
Contoh : (A ^ b) = 49 (berarti 0011 0001)

~ Binary Ones Complement
Ini tidak mencolok dan memiliki efek bit 'flipping'.
Contoh : (~ A) = -61 (berarti bentuk pelengkap 1100 0011 dalam 2 karena nomor biner yang ditandatangani

¡¡ Pergeseran Kiri Biner
Nilai operan kiri dipindahkan ke kiri oleh jumlah bit yang ditentukan oleh operan kanan.
Contoh : Sebuah ¡¡ = 240 (berarti 1111 0000)

¿¿ Binary Right Shift
Nilai operan kiri dipindahkan tepat dengan jumlah bit yang ditentukan oleh operan kanan.
Contoh : a ¿¿ = 15 (berarti 0000 1111)

Operator Perbandingan Python

Operator ini membandingkan nilai di kedua sisi dan memutuskan hubungan di antara keduanya. Mereka juga disebut operator relasional.

Asumsikan variabel a memegang 10 dan variabel b memegang 20, maka

==
Jika nilai dua operan sama, maka kondisinya menjadi benar.
Contoh (a == b) tidak benar

! =
Jika nilai dua operan tidak sama, maka kondisinya menjadi benar.

¿
Jika nilai operan kiri lebih besar dari nilai operan kanan, maka kondisinya menjadi benar.
Contoh (a¿ b) tidak benar

¡
Jika nilai operan kiri kurang dari nilai operan kanan, maka kondisinya menjadi benar.
Contoh (A ¡b) adalah benar.

¿ =
Jika nilai operan kiri lebih besar dari atau sama dengan nilai operan kanan, maka kondisinya menjadi benar.
Contoh (a¿ = b) tidak benar

¡=
Jika nilai operan kiri kurang dari atau sama dengan nilai operan kanan, maka kondisinya menjadi benar.
Contoh (a ¡= b) adalah benar.
Operator Python Diutamakan

berikut mencantumkan semua operator dari preseden tertinggi sampai yang terendah.

**

Eksponensiasi (naik ke tampuk kekuasaan)

∼ + -
Pelengkap, unary plus dan minus (nama metode untuk dua yang terakhir adalah + @ dan - @)

* / % //
Kalikan, bagi, modulo dan pembagian lantai

+ -
Penambahan dan pengurangan

¿¿ ¡¡
Pergeseran bitwise kanan dan kiri

&
Bitwise 'DAN'

ˆ |
Bitwise eksklusif 'OR 'dan reguler' OR'

¡= ¡¿¿ =
Operator perbandingan

¡¿ ==! =
Operator kesetaraan

= % = / = // = - = + = * = ** =
Operator penugasan

Bukan
Operator identitas

Bukan di
Operator keanggotaan

Tidak atau dan
Operator logika

CHAPTER 7

DESICION MAKING

7.1 Python Decision Making

Pengambilan keputusan adalah antisipasi kondisi yang terjadi saat pelaksanaan program dan menentukan tindakan yang dilakukan sesuai kondisi. Struktur keputusan mengevaluasi banyak ekspresi yang menghasilkan TRUE atau FALSE sebagai hasil. Anda perlu menentukan tindakan mana yang harus diambil dan pernyataan mana yang akan dijalankan jika hasilnya BENAR atau SALAH sebaliknya. Berikut adalah bentuk umum dari struktur pengambilan keputusan yang khas atau khusus yang ditemukan di sebagian besar bahasa pemrograman 7.1. Bahasa pemrograman Python menyediakan jenis dan juga laporan pengambilan keputusan. Berikut ini adalah penjelasan tentang pernyataan dan deskripsinya :

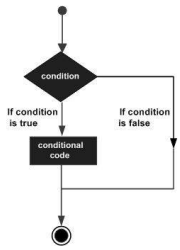


Figure 7.1 struktur pada python

1. if statements : sebuah if statement terdiri dari ekspresi boolean diikuti oleh satu atau lebih pernyataan.
2. if...else statements : sebuah if statement dapat diikuti oleh opsinal else statement, yang mengeksekusi ketika ekspresi boolean adalah palsu.
3. nested if statements : anda dapat menggunakan satu if atau else if pernyataan di dalam lain if atau else if statements.

Decision making atau Pemilihan keputusan pada python sangat penting untuk pemrograman komputer. Akan ada banyak situasi saat Anda diberi dua pilihan atau lebih dan Anda harus memilih opsi berdasarkan kondisi yang diberikan. Misalnya,

1. Seorang murid dengan nilai lebih dari 90 disebut siswa pintar
2. Seorang murid dengan nilai dibawah 90 dan diatas 30 disebut Siswa Standard
3. Seorang murid dengan nilai dibawah 30 disebut siswa bodoh

Umumnya juga, decision making dalam bahasa pemrograman yang sering digunakan adalah :

1. if...else statement : berguna saat kita harus mengambil keputusan dari dua pilihan. Misalnya, jika seorang siswa mendapatkan nilai lebih dari angka 95, maka siswa itu pintar, jika tidak, situasi seperti itu dapat dikodekan.

2. if...elseif...else statement : merupakan optional dari if...else statement, yang sangat berguna untuk menentukan berbagai kondisi yang lebih dari 2.
3. switch statement : merupakan alternative dari if statement. Setiap nilai disebut case, dan variabel yang dicek untuk setiap switch case.

F...ELIF...ELSE statement sama seperti IF...ELSEIF...ELSE pada bahasa pemrograman Java, yaitu digunakan menyeleksi beberapa ekspresi (lebih dari satu), apabila eskpresi1 pertama bernilai true, maka akan dijalankan statement1, jika ekspresi2 kedua bernilai true, maka akan dijalankan statement2, dan seterusnya.

Dari contoh diatas, pertanyaannya adalah bagaimana cara menulis kode pemrograman untuk menangani situasi seperti itu. Hampir semua bahasa pemrograman memberikan pernyataan kondisional diatas berdasarkan diagram decision di bawah.

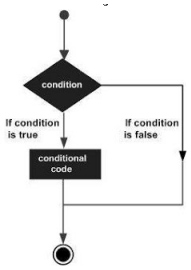


Figure 7.2 diagram decision

Mari kita membuat sebuah program C dengan bantuan jika pernyataan kondisional untuk mengubah situasi yang diberikan di atas menjadi kode pemrograman:

```
#include <stdio.h>

main() {
   int x = 45;
   if( x > 95) {
      printf( "siswa pintar");
   }
```

```
    if( x < 30) {
       printf( "siswa bodoh");
    }
    if( x < 95 && x > 30 ) {
       printf( "Siswa Standard");
    }
}
```

Outputnya adalah Siswa Standard

Bahasa pemrograman Python mengasumsikan nilai non-nol dan non-nullsebagai TRUE, dan jika itu adalah nol atau nol , maka diasumsikan sebagai nilai FALSE. Bahasa pemrograman Python menyediakan jenis pernyataan pengambilan keputusan berikut.

Mari kita membahas setiap keputusan secara singkat

Setelah tutorial mengenai

7.1.1 variable dan operator

pada bahasa pemrograman python akan membahas mengenai percabangan/pengambilan keputusan. Percabangan atau pengambilan keputusan adalah pengkondisian yang terjadi ketika aplikasi berjalan, kemudian ada aksi-aksi tertentu atau kondisi tertentu sehingga aplikasi harus bereaksi terhadap hal itu. Atau dalam bahasa pemrograman umum dikenal dengan IF, THEN, ELSE sebagai contoh pengaplikasian dari pengambilan keputusan ini.

if statements Sebuah if statement terdiri dari ekspresi boolean diikuti oleh satu atau lebih pernyataan.

Pada pembahasan python decisiom makaing ini, kami menggunakan perangkat raspberry pi 2 dengan sistem operasi rasbian jessie. Sangat ringan dan tentunya python secara default ada di dalamnya. Kebetulan dalam tulisan ini masih menggunakan python versi 2, meskipun ada python versi 3 juga.

Python core tidak menyediakan s̈witch ätau ” case s̈eperti bahasa pemrograman lain. Tapi kita bisa menggunakan statemen if, elif yang bisa menggantikan s̈witch ätau c̈ase

Di bawah ini merupakan tipe-tipe percabangan yang disediakan oleh python.

IF : Mengandung ekspresi boolean dan diikuti oleh satu atau banyak statemen

IF ELSE : IF bisa diikuti oleh optional statemen yaitu ELSE, yang akan dieksekusi ketika ekspresi boolean bernilai FALSE

NESTED IF atau IF bersarang : Kita bisa menggunakan IF, ELSE IF di dalam IF, ELSE IF lainnya

Contoh dalam python untuk IF :

```
varAngka1 = 123
varAngka2 = 0
if varAngka1:
print ”Nilai : TRUE”
print varAngka1
if varAngka2:
print ”Nilai : TRUE”
print varAngka2
```

```
#/usr/bin/python
a = raw_input("Masukkan Angka = ")
b = int(a)
if (b%2==0):
        print "Genap"
else:
        print "Ganjil"
```

Contoh untuk IF ELIF ELSE, di python sintak ini bisa ditulis dengan lebih singkat yaitu elif :

```
varAngka = 123
if varAngka==200:
print ”Nilai : TRUE”
print varAngka
elif varAngka==123:
print ”Nilai : TRUE”
print varAngka
else:
print ”Nilai : FALSE”
print varAngka
```

Contoh untuk NESTED IF :

```
varAngka = 89


if varAngka<100:

                        print \"Nilai : TRUE\"

                        print varAngka

                        if varAngka > 80:

                                                print \"Nilai

                        elif varAngka > 60:

                                                print \"Nilai

                        elif varAngka > 40:

                                                 print \"Nilai :

                        elif varAngka > 20:

                                                print \"Nilai : D
```

```
                        $else:

                                                \ print \"Nilai :

else:

                        print \"Nilai : FALSE\"

                        print varAngka
  end{verbatim}
```

Contoh lain pada NESTED IF dengan kondisi pilihan lebih dari s
\vspace{12}

```
Begin{verbatim}
#/usr/bin/python
pilihan=2
if pilihan==1:
        print "DOTA 2"
else:
        if pilihan==2:
print "GTA V Online"
else:
print "Semua Game"
end{verbatim}
```

Perintah seperti diatas itu adalah NESTED IF yaitu terdapat IF

Statemen IF juga bisa ditulis dalam 1 baris saja, misalnya sep

```
if varAngka1: print \"Nilai : TRUE\"
```

Dari sintak percabangan sudah bisa kita lihat perbedaan di ant

Suite pernyataan tunggal

Jika rangkaian klausa jika hanya terdiri dari satu baris

Berikut adalah contoh klausa satu baris jika -

```
var = 100
if ( var~ == 100 ) : print "Value of expression is 100"
print "Good bye!"
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai b

```
Value of expression is 100
Good bye! \hspace*{1.31in}
```

Pengambilan keputusan (kondisi if) digunakan untuk mengantisip

Pada python ada beberapa statement atau kondisi diantaranya ad

Jika kondisi bernilai salah maka statement atau kondisi if tid

Dibawah ini adalah contoh penggunaan kondisi if pada Python

Dari contoh diatas, jika program dijalankan maka akan mencetak

Selanjutnya Anda bisa mempelajari kondisi if else

Pengambilan keputusan (kondisi if else) tidak hanya digunakan

Pada python ada beberapa statement atau kondisi diantaranya ad

Kondisi if else adalah kondisi dimana jika pernyataan benar (t

Dibawah ini adalah contoh penggunaan kondisi if else pada Pyth

Kondisi if else adalah jika kondisi bernilai TRUE maka akan di

nilai = 3

Jika pernyataan pada if bernilai TRUE maka if akan dieksekusi,

```
if(nilai > 7):

 print(\"Selamat Anda Lulus\")

else:

        print(\"Maaf Anda Tidak Lulus\")
```

Pada contoh diatas, jika program dijalankan maka akan mencetak

Selanjutnya kita akan mempelajari per-kondisi an pada python y

Pengambilan keputusan (kondisi if elif) merupakan lanjutan ata
Dibawah ini adalah contoh penggunaan kondisi elif pada Python

Contoh penggunaan kondisi elif

```
hari   \_  ini = \"Minggu\"

if(hari   \_  ini == \"Senin\"):

        print(\"Saya akan kuliah\")
```

```
elif(hari   \_  ini == \"Selasa\"):

        print(\"Saya akan kuliah\")

elif(hari   \_  ini == \"Rabu\"):

        print(\"Saya akan kuliah\")

elif(hari   \_  ini == \"Kamis\"):

        print(\"Saya akan kuliah\")

elif(hari   \_  ini == \"Jumat\"):

 $  $  $  $ print("Saya akan kuliah")

elif(hari $  \_  $ini == "Sabtu"):

 $  $  $  $ print("Saya akan kuliah")

elif(hari $  \_  $ini == "Minggu"):

 $  $  $  $ print("Saya akan libur")
```

Pada contoh diatas, jika program dijalankan maka akan mencetak

```
\"Saya akan libur\".

    Pernyataan tersebut digunakan untuk pengambilan keputusan


\begin{enumerate}
\item
if kondisi   \_  1:


~~~~~~ pernyataan   \_  pernyataan   \_  1


\item
elif kondisi   \_  2:


~~~~~~ pernyataan   \_  pernyataan   \_  2


\item
elif kondisi   \_  3:


~~~~~~ pernyataan   \_  pernyataan   \_  3


\item
else kondisi   \_  n:


~~~~~~ pernyataan   \_  pernyataan-n


~~~~~~


\item
if kondisi   \_  1,elif kondisi   \_  2,elif kondisi   \_  3,e
```

```
~~~~berupa~suatu ekspresi yang menghasilkan nilai logika (bena
~~~
\end{enumerated}
```

Contoh Code yang dijalankan pada modus interaktif

```
\begin{verbatim}
  x = 5


~ y = 100


~~~ terbesar = x


~~ if terbesar < y:


terbesar = y

~~~ terbesar = 100
```

Python tidak menggunakan { } untuk menyertakan blok kode untuk penggunaan if or loop or fungsi yang lainnya. Sebaliknya, python menggunakan titik dua (:) dan indentasi atau spasi untuk pernyataan kelompok. Tes boolean untuk if tidak perlu dalam tanda kurung (perbedaan besar dari C++ atau Java), dan dapat memiliki *elif* dan *else*.

Nilai apapun dapat digunakan sebagai if-test. pada ñol ñilai-nilai semua dihitung sebagai false:Tidak ada,0,string kosong,list kosong, dictionary kosong. Ada juga tipe Boolean dengan dua nilai: True dan False (jika dikonversi ke

int, ini adalah 1 dan 0). Python memiliki operasi perbandingan yang biasa: ==, =, <, <=,>,> = Tidak seperti Java dan C.Operator boolean bisa juga di eja seperti * and *, * or *, * not * (Python tidak menggunakan gaya C).

```
matakuliah = 'matematika' matematika,fisika

nilai = 70 100,80,50
if nilai ¿=100 or nilai¿=80 :
  if matakuliah == 'matematika':
      print 'anda mendapat nilai A dalam mata kuliah matematika'
elif matakuliah == 'fisika':
  print 'anda mendapat nilai A dalam mata kuliah Fisika'
elif nilai ¿=70 and matakuliah=='matematika':
      print 'anda mendapat nilai B dalam mata kuliah matematika'
elif nilai ¿=70 and matakuliah=='fisika':
  print 'anda mendapat nilai B dalam mata kuliah fisika'
else:
  print 'nilai dan matakuliah tidak ada'
```

Seperti halnya bahasa pemrograman yang lain, tentu python juga mempunyai perintah untuk pengambilan suatu keputusan terhadap kondisi tertentu, yang disebut percabangan. Percabangan pada bahasa pemrograman python menggunakan perintah if, ya sama dengan bahasa pemrograman yang lain. Bagaimana cara menggunakan perintah if ini dalam bahasa pemrograman python? berikut adaah cara penggunaan percabangan if yang tepat

Cara penulisan dari perintah if secara garis besar adalah seperti berikut:

```
if ¡kondisi 1¿:
¡perintah yang dijalankan 1¿
elif ¡kondisi 2¿:
¡perintah yang dijalankan 2¿
else: ¡perintah yang dijalankan 3¿
```

Perintah-perintah yang dipergunakan antara lain

If dan If bersarang

Elif (singkatan dari: else if) dan

else.

IF Bersarang

Adapun tanda titik dua diletakan setelah kondisi, sedangkan untuk perintah yang dijalankan jika kondisi if terpenuhi diberi tab atau 4 spasi pada depannya untuk menandakan bahwa perintah tersebut berada didalam if, contoh dalam source code. Misal kita ingin menentukan angka genap atau ganjil:

```
angka = 7
if angka % 2 == 0:
print 'genap'
else: print 'ganjil'
```

Dari perintah diatas akan menghasilkan nilai yang diprint adalah 'ganjil'. Tanda % (persen) disini merupakan operator untuk modulus, yaitu sisa bagi. Adapun jalannya dari program diatas adalah, jika angka dalam hal ini nilainya 7 jika di modulus dengan 2, menyisahkan nilai nol maka data yang diprint adalah genap, jika tidak menyisahkan nilai nol maka data yang diprint adalah ganjil.

Bagaimana halnya dengan kondisi yang lebih dari satu. Misal kita ingin menentukan game yang kita sukai:

```
pilihan = 2
if pilihan == 1:
print 'DOTA2'
else: if pilihan == 2:
print 'GTA V Online'
else:
print 'Semua Game'
```

Perintah diatas merupakan if bersarang yaitu terdapat if didalam if, dapat juga dituliskan dengan perintah dibawah ini dengan menggunakan elif:

Elif

Merupakan suatu pemilihan kondisi dimana dalam kondisi tersebut, terdapat lagi kondisi lain Contoh kodingnya :

```
Program Kategori Berat hewan qurban
pilihan = 300
if pilihan ¿ 300:
print 'Sapi boleh diqurban'
```

elif pilihan ¡ 300 :
print 'Sapi belum boleh diqurban'
else:
print Rawat dulu sapinya yang benar'

Mana yang terbaik dari kedua cara penulisan kondisi if yang lebih dari satu diatas itu tentunya sesuai dengan kebutuhan kita masing-masing dalam membuat suatu aplikasi. Dalam if pun kita bisa membuat dua atau lebih persyaratan dalam kondisi if

Else
contohnya: angka = 2
if angka ¡= 10 and angka ¿= 1 :
print 'angka diantara 1 dan 10'
else:
print 'angka diluar jangkauan'

Percabangan Pada Bahasa Pemrograman Python.

Seperti halnya bahasa pemrograman yang lain, tentu python juga mempunyai perintah untuk pengambilan suatu keputusan terhadap kondisi tertentu, yang disebut percabangan. Percabangan pada bahasa pemrograman python menggunakan perintah if, ya sama dengan bahasa pemrograman yang lain. Bagaimana cara menggunakan perintah if ini dalam bahasa pemrograman python? Yuk mari kita sama-sama melihat cara penggunaan perintah if ini.

Cara penulisan dari perintah if secara garis besar adalah seperti berikut:

if ¡kondisi 1¿:
    ¡perintah yang dijalankan 1¿
elif ¡kondisi 2¿:
    ¡perintah yang dijalankan 2¿
else:
    ¡perintah yang dijalankan 3¿

Perintah-perintah yang dipergunakan antara lain if, elif (singkatan dari: else if) dan else. Adapun tanda titik dua diletakan setelah kondisi, sedangkan untuk perintah yang dijalankan jika kondisi if terpenuhi diberi tab atau 4 spasi pada depannya untuk menandakan bahwa perintah tersebut

berada didalam if, contoh dalam source code. Misal kita ingin menentukan angka genap atau ganjil:

```
angka = 7
if angka % 2 == 0:
    print 'genap'
else:
    print 'ganjil'
```

Dari perintah diatas akan menghasilkan nilai yang diprint adalah 'ganjil'. Tanda % (persen) disini merupakan operator untuk modulus, yaitu sisa bagi. Adapun jalannya dari program diatas adalah, jika angka dalam hal ini nilainya 7 jika di modulus dengan 2, menyisahkan nilai nol maka data yang diprint adalah genap, jika tidak menyisahkan nilai nol maka data yang diprint adalah ganjil.

Bagaimana halnya dengan kondisi yang lebih dari satu. Misal kita ingin menentukan buah yang kita sukai:

```
pilihan = 2
if pilihan == 1:
    print 'buah durian'
else:
    if pilihan == 2:
        print 'buah mangga'
    else:
        print 'semua buah'
```

Perintah diatas merupakan if bersarang yaitu terdapat if didalam if, dapat juga dituliskan dengan perintah dibawah ini dengan menggunakan elif:

```
pilihan = 2
if pilihan == 1:
    print 'buah durian'
elif pilihan == 2:
    print 'buah mangga'
else:
    print 'semua buah'
```

Mana yang terbaik dari kedua cara penulisan kondisi if yang lebih dari satu diatas itu tentunya sesuai dengan kebutuhan kita masing-masing dalam membuat suatu aplikasi, seperti kata orang, banyak jalan menuju roma begitu juga

dengan pemrograman, banyak jalan untuk menuliskan suatu perintah untuk menghasilkan hasil tertentu... :)

Dalam if pun kita bisa membuat dua atau lebih persyaratan dalam kondisi if contohnya:

```
angka = 2
if angka <= 10 and angka >= 1 :
~~~ print 'angka diantara 1 dan 10'
else:
~~~ print 'angka diluar jangkauan'
```

CHAPTER 8

LOOP

8.1 Python Loops

Figure 8.1 Perulangan

Python dikenal sebagai bahasa pemograman interpreter, karena Python dieksekusi dengan sebuah interpreter. Satu hal yang telah kita ketahui bahwa bahasa pemograman Python adalah bahasa pemograman yang mudah dibaca dan terstruktur. Python sangat mementingkan indentasi, sehingga kita perlu melakukan indentasi secara konsisten. In-

dentasi tersebut dipermudah dengan penggunaan tombol Tab dan dimulai dari kolom pertama untuk setiap blok baru[1]. Python memiliki kelebihan lain yang sangat penting dibanding bahasa pemrograman yang terdahulu:

- Python merupakan open-source software, yang artinya ia dapat diperoleh secara gratis. Bahkan python sudah otomatis terinstall di Linux.
- Python tersedia pada semua operating systems (OS) terkenal seperti Linux, Unix, Windows, dan MacOS. Suatu script python yang ditulis pada OS tertentu, dapat dijalankan di OS lain tapa ada modifikasi sedikitpun.
- Python lebih mudah dipelajari sekaligus lebih mudah ”dibaca” dibandingkan dengan bahasa pemrograman lainnya.
- Python dan program ekstensinya mudah diinstall.

Python berdiri di atas landasan pondasi Java and C++. Halhal seperti classes, methods, inheritance, yang kerapkali diimplementasikan pada bahasa yang bersifat object-oriented, juga dapat diimplementasikan di python.[2]

Secara umum, perulangan adalah blok kode yang dieksekusi berulang kali. Semua bahasa pemrograman menyediakan berbagai model struktur perulangan, seperti contohnya pada PHP ada while, for, dan foreach. Python juga menyediakan berbagai model tipe untuk menghandel perulangan.

Perintah perulangan di gunakan untuk mengulang pengeksekusian statemen-statemen hingga berkali-kali sesuai dengan iterasi yang diinginkan. Dalam python, perintah untuk perulangan (loop) adalah while dan for.

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

bahkan beribu-ribu kode. Untuk itu Anda perlu menggunakan pengulangan di dalam bahasa pemrograman Python.

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

- While Loop
- For Loop
- Nested Loop

8.1.1 While Loop

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

Dibawah ini adalah contoh penggunaan pengulangan While Loop.

Contoh penggunaan While Loop

```
count = 0 \par
while (count < 9): \par
 \$  \$  \$  \$ print ('The count is:', count) \par
 \$  \$  \$  \$ count = count + 1 \par
print ("Good bye!") \par
```

8.1.2 For Loop

Perulangan for disebut juga counted loop $perulanganyangterhitung$ Pengulangan for digunakan untuk pengulangan dengan muatan yang banyak[3]. keistimewaan perulanga pada for adalah , perulangan dapat di hentikan pada saat kondisi tertentu. pada python, statemen for bekerja mengulang berbagai macam tipe data sekuensial seperti pada list, string dan tuple Contohnya Seperti :

```
for a in range(0, 10):
print a
```

Hasil Outputnya :

```
python for.py
0
1
2
3
4
5
6
7
8
9
```

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

Dibawah ini adalah contoh penggunaan pengulangan While Loop.

Contoh pengulangan for sederhana

angka = [1,2,3,4,5]

for x in angka:

$ $ $ $ print(x)

Contoh pengulangan for

buah = [”nanas”, ”apel”, ”jeruk”]

for makanan in buah:

$ $ $ $ print(S̈aya suka makan, makanan)

Looping artinya adalah pengulangan. Misalnya anda mendapat tugas untuk menghitung akar bilangan-bilangan dari 1 sampai 10. Ada 2 cara untuk menyelesaikan tugas tersebut, pertama, salinlah source-code berikut pada python editor lalu diberi nama looping01.py

1. from numpy import sqrt # hanya function sqrt yang dipanggil
2. print sqrt(1)
3. print sqrt(2)
4. print sqrt(3)
5. print sqrt(4)
6. print sqrt(5)
7. print sqrt(6)
8. print sqrt(7)
9. print sqrt(8)
10. print sqrt(9)
11. print sqrt(10)

Jalankan source-code di atas dengan menekan tombol F5, maka akan muncul hasil sebagai berikut :

1. 0
2. 41421356237
3. 73205080757
4. 0
5. 2360679775
6. 44948974278
7. 64575131106
8. 82842712475
9. 0
10. 16227766017

Cara kedua dengan teknik looping, yaitu : from numpy import sqrt for i in range(1,10+1): print sqrt(i) Simpanlah source-code ini dengan nama looping02.py, lalu jalankan dengan F5, akan nampak hasil yang sama yaitu

1. 0
2. 41421356237
3. 73205080757
4. 0
5. 2360679775
6. 44948974278
7. 64575131106
8. 82842712475
9. 0
10. 16227766017

Mari sejenak kita bandingkan antara looping01.py dan looping02.py. Kedua source-code itu memiliki tujuan yang sama yaitu menghitung akar bilangan dari 1 sampai 10.

Perbedaannya, looping01.py berisi 11 baris statemen, sedangkan looping02.py hanya 3 baris statemen. Coba cek ukuran file-nya! Ukuran file looping01.py (di laptop saya) adalah 179 byte, sementara ukuran looping02.py adalah 72 byte. Dengan demikian dapat disimpulkan bahwa looping02.py lebih efisien dibanding looping01.py.[2]

8.1.3 Nested Loop

Nested Loop (Perulangan Bertingkat) adalah semua tipe perulangan yang dapat dipakai di dalam perulangan yang lain. Jadi Perulangan for bisa dipakai di dalam for yang lain, perulangan for bisa berada didalam perulangan while, perulangan while bisa dipakai di dalam perulangan while yang lain, dan perulangan while bisa di dalam perulangan for.

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

Loop.

Contoh penggunaan Nested Loop :

```
i = 2

while(i ¡ 100):

$ $ $ $ j = 2

$ $ $ $ while(j ¡= (i/j)):

$ $ $ $ $ $ $ $ if not(i $ % $j): break

$ $ $ $ $ $ $ $ j = j + 1

$ $ $ $ if (j ¿ i/j) : print i, ” is prime”
```

$ $ $ $ i = i + 1

print ”Good bye!”

Perhatikan contoh berikut ini:

```
print (”1”)
print (”2”)
print (”3”)
print (”4”)
print (”5”)
print (”6”)
print (”7”)
print (”8”)
print (”9”)
print (”10”)
```

Contoh program diatas adalah program untuk menampilkan angka 1 sampai dengan 10 tanpa perulangan. Tanpa menggunakan perulangan, programmer harus menuliskan semua statement diatas sehingga source code menjadi lebih banyak dan tidak efisien. Bayangkan kalau programmer disuruh menampilkan angka 1 sampai dengan 1000000 tanpa menggunakan perulangan

Dengan menggunakan perulangan, source code lebih pendek dan efisien. Perhatikan contoh program untuk mencetak angka 1 sampai dengan 10 dengan menggunakan konsep perulangan di bawah ini.

beginverbatim i = 1

```
while(i ¡ 11):
    print(i)
    i = i+1
endverbatim
```

Bandingkan kedua program diatas, Mana yang lebih efisien? Mana yang lebih simple?

Ada 3 macam bentuk perulangan pada Python, yaitu:
FOR Loop
WHILE Loop
dan Loop bersarang (Nested Loop)

Selain membahas 3 bentuk perulangan diatas, tutorial ini juga membahas control perulangan, meliputi:
Break Statement
Continue Statement
dan Pass Statement

8.1.3.1 Contoh Penggunaan Nested Loop

Format nested loop $fordidalamfor$

```
For iterasi_var_1 in urutan_1:
Statements_untuk_perulangan_for_yang_di_luar
...
For iterasi_var_1 in urutan_2:
Statements_untuk_perulangan_for_yang_di_dalam
...
Statements_untuk_perulangan_for_yang_di_luar
...
```

Format nested loop $whiledidalamwhile$

```
While expressions:
Statements_untuk_perulangan_while_yang_di_dalam
...
Statements_untuk_perulangan_whle_yang_di_luar
...
```

Contoh :

```
X = int(input(Masukkan jumlah bariss: ))
For i in range (x) :
For j in range(i+1):
Print(*, end=)
Print()
```

Saat di Run Module maka : Masukkan jumlah bariss: 5 $inputkan5$ * ** *** **** ***** Muncul 5 baris isi bintang

8.1.3.2 Nested Loop for Nested Data Disini kita memiliki list data dari murid-murid. Jadi, setiap murid memiliki nama yang dipasangkan dengan list subyek(mata pelajaran) yang mereka ambil. Dan akan mencetak setiap nama murid, dan jumlah dari subyek (mata pelajaran) yang mereka ambil

```
students = [
    ("John", ["TIK", "IPS"]),
    ("Vusi", ["Matematika", "TIK", "IPA"]),
    ("Jess", ["TIK", "Bahasa Indonesia", "Ekonomi", "Pendidika
    ("Sarah", ["Biologi", "Matematika", "Ekonomi", "Kimia"]),
    ("Zuki", ["Sosiologi", "Ekonomi", "Biologi", "Matematika",

for (name, subjects) in students:
    print(name, "takes", len(subjects), "courses")
```

Lalu, setelah dijalankan (run) maka akan tampil seperti ini: John takes 2 courses Vusi takes 3 courses Jess takes 4 courses Sarah takes 4 courses Zuki takes 5 course

8.1.4 FOR Loop

FOR Loop digunakan untuk melakukan perulangan atau iterasi sampai batas atau range yang telah ditentukan.

Dibawah ini adalah sintak dasar FOR Loop di Python.

for iterating $ _ $var in range:

statements(s)

Contoh Program Perhatikan contoh program For Loop pada Python: Contoh 1

$$Programmencetakangka1s/d10 \quad (8.1)$$

```
i = 10 \par
for i in range(10): \par
~~ print(i+1) \par
~~ i = i+1 \par
```

Fungsi $ $range() $ $biasanya digunakan sebagai counter pada perulangan bentuk For. range(10) artinya menampikan perulangan sebanyak 10 elemen. Apabila program diatas Anda jalankan, maka akan menampilkan angka 1 sampai dengan 10 seperti output di bawah ini:

1
2
3
4
5
6
7
8
9

10

Contoh 2

Program mencetak angka -1 s/d 8

beginverbatim i = 10
for i in range(-10, 10, 2): $ # $ range(range awal, range akhir, selisih)
print(i)
endverbatim Perhatikan pada range(-10, 10, 2) artinya

perulangan akan dimulai dari batas awal -10 sampai dengan batas akhir 10 dengan selisih 2. Apabila program diatas

Anda jalankan, maka akan menampilkan output berikut ini:

-10
-8
-6
-4
-2
0
2
4
6
8

Contoh 3

Program menampilkan huruf Belajar Python
for huruf in 'Belajar Python':
print (huruf)

Apabila program diatas Anda jalankan, maka akan menghasilkan output berikut ini:

B
e
l
a
j
a
r
P
y
t
h
o
n

Contoh 4 Program berikut akan menampilkan perulangan

dari list atau tupple.

Program menampilkan huruf Belajar Python

beginverbatim makanan = [’Pizza’, ’Nasi Bebek’, ’Rujak Buah’]
for makan in makanan:
print (M̈akanan Favorit :̈, makan)
endverbatim Apabila program diatas Anda jalankan, maka

akan menghasilkan output berikut ini:

Makanan Favorit : Pizza
Makanan Favorit : Nasi Bebek
Makanan Favorit : Rujak Buah

8.1.5 While Loop

While Loop akan menjalankan statemet selama kondisi terpenuhi (atau bernilai true). Di bawah ini adalah sintak dasar

dari While Loop pada Python Contoh Program Coba Anda

ketik program di bawah ini:

Program mencetak angka 1 s/d 10

```
beginverbatim i = 1
while(i ¡ 11):
print(i)
i = i+1
endverbatim
```

Apabila program diatas Anda jalankan, maka akan menghasilkan output seperti di bawah ini:

```
1
2
3
4
5
6
7
8
9
10
```

8.1.6 Infinite Loop

Infinite Loop adalah kondisi perulangan, dimana statement akan dijalankan terus menerus tanpa berhenti. Akan berhenti kalau Anda menekan tombol CTRL+C. Di bawah ini contoh

program Infinite Loop

```
program menampilkan tulisan Python tanpa henti

flag = 1

while (flag): print ('Python')
print ('Good bye!')
```

8.1.7 Nested Loop

Nested Loop secara sederhana adalah perulangan di dalam perulangan. Di bawah ini adalah sintak dasar Nested Loop

pada Python:

```
for iterating $ _ $var in sequence:
   for iterating $ _ $var in sequence:
      statements(s)
   statements(s)
```

atau yang menggunakan while loop

```
while expression:
```

```
while expression:
  statement(s)
statement(s)
```

Contoh Program Di bawah ini adalah contoh program implementasi Nested Loop untuk mencetak bilangan prima dari 2 sampai 30.

Program menampilkan bilangan prima dari 2 s/d 30

```
i = 2 \par
while(i < 30): \par
~~ j = 2 \par
~~ while(j <= (i/j)): \par
~~~~~ if not(i \$  \%  \$j): break \par
~~~~~ j = j + 1 \par
~~ if (j > i/j) : print (i, \" adalah bilangan prima\") \par
~~ i = i + 1 \par

print (G̈ood bye!)̈
```

Apabila program diatas Anda jalankan, maka akan menampilkan output seperti di bawah ini.

```
2  adalah bilangan prima
3  adalah bilangan prima
5  adalah bilangan prima
7  adalah bilangan prima
11  adalah bilangan prima
13  adalah bilangan prima
17  adalah bilangan prima
19  adalah bilangan prima
23  adalah bilangan prima
```

29 adalah bilangan prima

Pengulangan adalah salah satu hal penting yang ada di bahasa pemrograman. Pengulangan digunakan misalnya untuk meng-update $ $nama $ $file $ $yang cukup banyak jumlahnya, atau mengakses piksel satu persatu pada gambar.

Python memiliki tiga jenis pengulangan yang wajib Anda cermati untuk membuat sebuah aplikasi dengan Python. Pengulangan yang pertama adalah $ $while. Dengan menggunakan $ $while, Anda dapat membuat kondisi tertentu untuk menghentikan $ $while. Biasanya $ $while $ $digunakan untuk melakukan $ $loopingyang tidak pasti. Coba lihat contoh berikut (Anda dapat menulisnya dalam sebuah $ $file, kemudian eksekusi $ $file $ $tersebut di konsol):

```
beginverbatim i = 0
while True:
    if i ¡ 10:
        print ”Saat ini i bernilai: ”, i
        i = i + 1
    elif i ¿= 10:
        break
endverbatim Pada potongan kode diatas, $ $while $ $akan
```

terus berputar selama i masih kurang dari 10. Jika sudah lebih dari 10 maka $ $while $ $akan berhenti. Pengulangan $ $whilejuga biasa digunakan di aplikasi konsol, untuk menahan $ $user $ $mengisikan semua input yang diperlukan dan baru akan berhenti setelah semua input dan proses interaksi berakhir. Jika kode diatas kita jalankan, maka $ $output-nya akan seperti ini:

```
Saat ini i bernilai: 0
Saat ini i bernilai: 1
Saat ini i bernilai: 2
Saat ini i bernilai: 3
Saat ini i bernilai: 4
Saat ini i bernilai: 5
Saat ini i bernilai: 6
Saat ini i bernilai: 7
```

```
Saat ini i bernilai: 8
Saat ini i bernilai: 9
```

Sekarang kita coba gunakan $ $for. Pengulangan $ $for $ $biasa digunakan untuk pengulangan yang sudah jelas banyaknya. Misal, Anda ingin mengulang sebuah pengulangan sampai 10 kali atau mengeluarkan semua hasil $ $query $ $dari $ $databasedi halaman HTML. Berikut ini adalah contoh kode untuk pengulangan $ $for:

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

Jika dijalankan maka kode diatas akan mengeluarkan $ $output $ $seperti ini:

```
0
1
2
3
4
5
6
7
8
9
```

Tidak hanya mengiterasi deretan angka, pengulangan $ $for $ $pun dapat Anda gunakan untuk mengulang sesuatu yang $ $iterable $ $seperti $ $list, $ $tuple, $ $dictionary, dan $ $iterable object $ $lainnya. Berikut ini kita ambil contoh dengan mengulang sebuah $ $list $ $yang berisi karakter anime Dragonball Super:

```
dragonball $ _ $super $ _ $character = [S̈on Goku, V̈egeta, B̈eerus, T̈runks, Ẅhiz, C̈hampa]
for character in dragonball $ _ $super $ _ $character:
    print character
```

Jika kita jalankan potongan kode tadi, maka $ $output-nya akan seperti berikut:

- Son Goku
- Vegeta
- Beerus
- Trunks
- Whiz
- Champa
- For Loop

Seperti pada bahasa pemrograman lainnya, for loop sudah menjadi standar namun berbeda-beda tata cara penulisan nya di setiap pemrograman.

Sekarang kita langsung buat contoh di Python. $ $

Contoh iterasi pada String

```
for˜n in 'Python':   \par
˜˜˜ print 'Huruf :', n \par
```

iterasi pada List biasa

```
mobil = ['sedan', 'truk', 'angkot']
for p in mobil:
   print 'Mobil :', mobil
```

iterasi pada list melalui index

```
for i in range(len(mobil)):
   print 'Mobil :', mobil[i]
```

iterasi angka / range

```
for a in range(1,10):
    print "Angka :", a
    if(a == 5): $ # $ditambah conditional
```

```
        print ”Saya dapat angka : ”,a

iterasi loop nested
for a in range(1,10):
    for x in range(11,20):
        b = a * x
        print ”Angka :”, b

loop dgn break
for letter in ’Python’:
   if letter == ’h’:
      break
   print ’Current Letter :’, letter

print ”Good job !!!”
```

8.1.8 While Loop

WHile dipakai untuk looping dimana iterasi akan dilakukan selama kondisi yang diberikan benar. While ini juga bisa di pakai untuk Infinite loop.

```
Contoh While
count = 0
while count ¡ 100:
$ $ $ $ $ $print ”Count ke : ”, count
$ $ $ $ $ $count = count + 1

infinite loop
”’
Set loop ini untuk kondisi dimana suatu syarat tidak pernah TRUE
”’

setvar =1
while setvar == 1
$ $ $ $ input = input $ _ $raw(”Masukan angka :”)
$ $ $ $ print ”Angka anda : ”, input
```

loop diatas akan berhenti jika anda stop manual misal dgn CTRL+C di terminal
”’

ELSE statement di while loop. di Python kita bisa set WHile loop lalu dikasih kondisi
”’

```
count = 0 \par
while count < 5: \par
 \$  \$  \$  \$  \$  \$print "count : ",count \par
 \$  \$  \$  \$  \$  \$count = count + 1 \par
else: \par
 \$  \$  \$  \$ print "Lihat yang masuk sini apa : ",count \pa
```

while dgn break

```
angka = 10
while angka ¿ 0:

  print ’Angka :’, angka
  angka = angka -1
  if angka == 7:
    break
```

8.1.9 Penggunaan loop dengan else statement

Python mendukung untuk memiliki pernyataan lain yang terkait dengan pernyataan lingkaran Jika else statement digunakan dengan for loop, pernyataan yang lain dijalankan saat loop telah habis mengulangi daftar. Jika else statement digunakan dengan loop sementara, pernyataan yang lain dijalankan saat kondisinya menjadi salah.

8.1.10 Middle-test loop

Middle-test loop adalah sebuah perulangan yang akan mengeksekusi pada beberapa bagian body, kemudian akan

melakukan pengujian exit berarti menguji dalam kondisi exit, lalu kemungkinan akan mengeksekusi beberapa bagian body lainnya. Disini dapat menggunakan while dan break secara bersama-sama. Terkadang kita membutuhkan looping dengan pengujian di tengah daripada pengujian di atas maupun di akhir.

8.1.11 Penjelasan Penggunaan For Loop

For loop secara tradisional digunakan saat Anda memiliki blok kode yang ingin Anda ulangi beberapa kali. Python untuk pernyataan iterates atas anggota urutan dalam urutan, mengeksekusi blok setiap waktu. Kontras untuk pernyataan dengan loop ” while ”, digunakan bila suatu kondisi perlu diperiksa setiap iterasi, atau untuk mengulang blok kode selamanya.

CHAPTER 9

LISTS

9.1 LISTS

Struktur data yang paling dasar dengan Python adalah urutannya. Setiap elemen berurutan diberi nomor - posisinya atau indeksnya. Indeks pertama adalah nol, indeks kedua adalah satu, dan seterusnya. Python memiliki enam jenis

urutan built-in, namun yang paling umum adalah daftar dan tupel, yang akan kami lihat di tutorial ini. Ada beberapa hal yang dapat Anda lakukan dengan semua tipe urutan. Operasi ini meliputi pengindeksan, pengiris, penambahan, perbanyak, dan pengecekan keanggotaan. Selain itu, Python memiliki fungsi built-in untuk menemukan panjang urutan dan untuk menemukan elemen terbesar dan terkecilnya.

9.1.1 Daftar Python

Daftar ini adalah datatype paling serbaguna yang tersedia dengan Python yang dapat ditulis sebagai daftar nilai yang dipisahkan koma (item) antara tanda kurung siku. Hal penting tentang daftar adalah item dalam daftar tidak perlu jenis yang sama. Membuat daftar sederhana memasukkan berbagai nilai yang dipisahkan koma di antara tanda kurung siku.

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Serupa dengan indeks string, daftar indeks mulai dari 0, dan daftar dapat diiris, digabungkan dan seterusnya. Mengakses Nilai dalam Daftar Untuk mengakses nilai dalam daftar, gunakan tanda kurung siku untuk mengiris beserta indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut. $ # $! / Usr / bin / python

List1 = ['fisika', 'kimia', 1997, 2000];
List2 = [1, 2, 3, 4, 5, 6, 7];

Cetak "list1 [0]:", list1 [0]
Cetak "list2 [1: 5]:", list2 [1: 5]

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut -

List1 [0]: fisika
List2 [1: 5]: [2, 3, 4, 5]

9.1.1.1 Memperbarui Daftar Anda dapat memperbarui satu atau beberapa elemen daftar dengan memberikan potongan di sisi kiri operator penugasan, dan Anda dapat menambahkan ke elemen dalam daftar dengan metode append (). Misalnya -

$ # $! / Usr / bin / python

list = [’physics’, ’chemistry’, 1997, 2000];

```
print "Value available at index 2 : " \par
print list[2] \par
list[2] = 2001; \par
print "New value available at index 2 : " \par
print list[2] \par
```

Catatan: append () metode dibahas di bagian selanjutnya.

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut -

Nilai tersedia di indeks 2:

1997

Nilai baru tersedia di indeks 2:

2001

9.1.1.2 Hapus Daftar Elemen Untuk menghapus elemen daftar, Anda dapat menggunakan salah satu pernyataan del jika Anda tahu persis elemen yang Anda hapus atau metode hapus () jika Anda tidak mengetahuinya. Misalnya -

$ # $! / Usr / bin / python

```
List1 = [’fisika’, ’kimia’, 1997, 2000]; \par
\vspace{12pt}
Daftar cetak1 \par
\vspace{12pt}
Del list1 [2]; \par
\vspace{12pt}
Cetak \"Setelah menghapus nilai pada indeks 2:\" \par
\vspace{12pt}
```

```
Daftar cetak1 \par
Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai b
['Fisika', 'kimia', 1997, 2000] \par
Setelah menghapus nilai pada indeks 2: \par
['Fisika', 'kimia', 2000] \par
```

Catatan: hapus () metode dibahas di bagian selanjutnya.

9.1.1.3 Operasi Daftar Dasar Daftar merespons operator + dan * seperti string; Mereka berarti penggabungan dan pengulangan di sini juga, kecuali hasilnya adalah daftar baru, bukan string.

Sebenarnya, daftar merespons semua operasi urutan umum yang kami gunakan pada senar di bab sebelumnya.

Python Expression Results Description

len([1, 2, 3]) 3 Length

[1, 2, 3] + [4, 5, 6] [1, 2, 3, 4, 5, 6] Concatenation

['Hi!'] * 4 ['Hi!', 'Hi!', 'Hi!', 'Hi!'] Repetition

3 in [1, 2, 3] True Membership

for x in [1, 2, 3]: print x, 1 2 3 Iteration

Indexing, Slicing, dan Matrixes

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

Dengan asumsi masukan berikut -

L = ['spam', 'Spam', 'SPAM!']

Python Expression

1. Results Description
2. L[2] 'SPAM!' Offsets start at zero
3. L[-2] 'Spam' Negative: count from the right
4. L[1:] ['Spam', 'SPAM!'] Slicing fetches sections

Built-in List Functions $ & $ Methods:

Python includes the following list functions

SN	Function with Description
1.	cmp(list1, list2) Compares elements of both lists.
2.	len(list) Gives the total length of the list.
3.	max(list) Returns item from the list with max value.
4.	min(list) Returns item from the list with min value.
5.	list(seq) Converts a tuple into list.

Python includes following list methods

SN	Methods with Description
1.	list.append(obj) Appends object obj to list
2.	list.count(obj) Returns count of how many times obj occurs in list
3.	list.extend(seq) Appends the contents of seq to list
4.	list.index(obj) Returns the lowest index in list that obj appears

5. list.insert(index, obj)

   Inserts object obj into list at offset index

6. list.pop(obj=list[-1])

   Removes and returns last object or obj from list

7. list.remove(obj)

   Removes object obj from list

8. list.reverse()

   Reverses objects of list in place

9. list.sort([func])

   Sorts objects of list, use compare func if given

Python memiliki tipe daftar built-in yang hebat dengan nama daftar. Daftar literal ditulis dalam tanda kurung siku []. Daftar bekerja sama dengan senar - gunakan fungsi len () dan tanda kurung siku [] untuk mengakses data, dengan elemen pertama di indeks 0. (Lihat daftar dokumen python.org resmi).

```
~ Warna = ['merah', 'biru', 'hijau'] \par
~ cetak warna [0]  \$  \#  \$ \$  \#  \$ merah \par
~ cetak warna [2]  \$  \#  \$ \$  \#  \$ hijau \par
~ Cetak len (warna)  \$  \#  \$ \$  \#  \$ 3 \par
```

Tugas dengan = daftar tidak membuat salinan. Sebagai gantinya, tugas membuat kedua variabel menunjuk ke satu daftar di memori.

```
~ B = warna  \$  \#  \$ \$  \#  \$ Tidak menyalin daftar \par
```

```
\"Daftar kosong\" hanyalah sepasang kurung kosong []. '+' Beke
FOR dan IN \par
```

Python's * untuk * dan * in * constructs sangat berguna, dan penggunaan pertama dari yang akan kita lihat adalah dengan daftar. * Untuk * membangun - untuk daftar var - adalah cara mudah untuk melihat setiap elemen dalam daftar (atau koleksi lainnya). Jangan menambah atau menghapus dari daftar selama iterasi.

```
~ kotak = [1, 4, 9, 16] \par
~ Jumlah = 0 \par
~ Untuk num dalam kotak: \par
~~~ Jumlah + = num \par
~ Jumlah cetak \par
```

Jika Anda tahu hal macam apa yang ada dalam daftar, gunakan nama variabel dalam lingkaran yang menangkap informasi seperti ñum;, atau ñame;, atau ürl;. Karena kode python tidak memiliki sintaks lain untuk mengingatkan Anda tentang tipe, nama variabel Anda adalah cara kunci bagi Anda untuk tetap mempertahankan apa yang sedang terjadi.

* Dalam * membangun sendiri adalah cara mudah untuk menguji apakah sebuah elemen muncul dalam daftar (atau koleksi lainnya) - nilai dalam koleksi - tes jika nilainya ada dalam koleksi, mengembalikan True / False.

```
~ Daftar = ['larry', 'curly', 'moe'] \par
~ jika 'keriting' dalam daftar: \par
~~~ Cetak 'yay' \par
```

The for / in constructs sangat umum digunakan pada kode Python dan bekerja pada tipe data selain list, jadi sebaiknya hafalkan sintaksnya. Anda mungkin memiliki kebiasaan dari

bahasa lain di mana Anda memulai pengulangan manual melalui koleksi, dengan Python yang seharusnya Anda gunakan untuk / in.

Anda juga dapat menggunakannya untuk / dalam mengerjakan sebuah string. String bertindak seperti daftar karakternya, jadi untuk ch di s: print ch mencetak semua karakter dalam sebuah string.

Jarak

Fungsi range (n) menghasilkan angka 0, 1, ... n-1, dan range (a, b) mengembalikan a, a + 1, ... b-1 - sampai tapi tidak termasuk angka terakhir . Kombinasi fungsi for-loop dan range () memungkinkan Anda membuat numerik tradisional untuk loop:

```
 \$  \#  \$ \$  \#  \$ print the numbers from 0 through 99 \pa
~ for i in range(100): \par
~~~ print i \par
```

Ada varian xrange () yang menghindari biaya membangun keseluruhan daftar untuk kasus sensitif kinerja (dalam Python 3000, range () akan memiliki perilaku kinerja yang baik dan Anda dapat melupakan xrange ()).

Sementara Loop

Python juga memiliki standar while-loop, dan break dan continue statements bekerja seperti di C ++ dan Java, mengubah jalannya loop terdalam. Di atas untuk / dalam loop memecahkan kasus umum iterasi pada setiap elemen dalam daftar, namun loop sementara memberi Anda kontrol penuh atas angka indeks. Berikut adalah loop sementara yang mengakses setiap elemen ke-3 dalam daftar:

```
~  \$  \#  \$ \$  \#  \$ Mengakses setiap elemen ke-3 dalam da
~ I = 0 \par
~ sementara i <len (a): \par
```

```
~~~ cetak sebuah [i] \par
~~~ i = i + 3 \par
```

Daftar metode

Berikut adalah beberapa metode daftar umum lainnya.

1. List.append (elem) - menambahkan satu elemen ke akhir daftar. Kesalahan umum: tidak mengembalikan daftar baru, cukup modifikasi yang asli.
2. List.insert (indeks, elem) - memasukkan elemen pada indeks yang diberikan, menggeser elemen ke kanan.
3. List.extend (list2) menambahkan elemen dalam list2 ke akhir daftar. Menggunakan + atau + = pada daftar sama dengan menggunakan extend ().
4. List.index (elem) - mencari elemen yang diberikan dari awal daftar dan mengembalikan indeksnya. Melempar ValueError jika elemen tidak muncul (gunakan ”in” untuk memeriksa tanpa ValueError).
5. List.remove (elem) - mencari instance pertama dari elemen yang diberikan dan menghapusnya (melempar ValueError jika tidak ada)
6. List.sort () - menyusun daftar di tempat (tidak mengembalikannya). (Fungsi yang diurutkan () yang ditunjukkan di bawah ini lebih diutamakan.)
7. List.reverse () - membalik daftar di tempat (tidak mengembalikannya)
8. List.pop (index) - menghapus dan mengembalikan elemen pada indeks yang diberikan. Mengembalikan elemen paling kanan jika indeks dihilangkan (kira-kira kebalikan dari append ()).

Perhatikan bahwa ini adalah metode pada daftar objek, sedangkan len () adalah fungsi yang mengambil daftar (atau string atau apapun) sebagai argumen.

1. Daftar = [’larry’, ’curly’, ’moe’]
2. List.append (’shemp’) $ # $ $ # $ append elem di akhir
3. List.insert (0, ’xxx’) $ # $ $ # $ masukkan elem pada indeks 0
4. list.extend ([’yyy’, ’zzz’]) $ # $ $ # $ tambahkan daftar elems at end
5. daftar cetak $ # $ $ # $ [’xxx’, ’larry’, ’curly’, ’moe’, ’shemp’, ’yyy’, ’zzz’]
6. Print list.index (’keriting’) $ # $ $ # $ 2
7. List.remove (’curly’) $ # $ $ # $ cari dan hapus elemen itu
8. List.pop (1) $ # $ $ # $ menghapus dan mengembalikan ’larry’
9. daftar cetak $ # $ $ # $ [’xxx’, ’moe’, ’shemp’, ’yyy’, ’zzz’]

Kesalahan umum: perhatikan bahwa metode di atas tidak mengembalikan daftar yang dimodifikasi, mereka hanya memodifikasi daftar aslinya.

1. Daftar = [1, 2, 3]
2. Print list.append (4) $ # $ $ # $ TIDAK, tidak bekerja, append () return Tidak ada
3. $ # $ $ # $ Pola yang benar:
4. List.append (4)
5. Daftar cetak $ # $ $ # $ [1, 2, 3, 4]
   st Build Up

Salah satu pola yang umum adalah dengan memulai daftar daftar kosong [], lalu gunakan append () atau extend () untuk menambahkan elemen ke dalamnya:

```
~ List = []  \$  \#  \$ \$  \#  \$ Mulai sebagai daftar kosong
~ List.append ('a')  \$  \#  \$ \$  \#  \$ Gunakan append () u
~ List.append ('b') \par
```

Daftar irisan

Slice bekerja pada daftar seperti halnya senar, dan juga dapat digunakan untuk mengubah sub-bagian daftar.

```
Daftar = ['a', 'b', 'c', 'd'] \par
Daftar cetak [1: -1]  \$  \#  \$ \$  \#  \$ ['b', 'c'] \par
Daftar [0: 2] = 'z'  \$  \#  \$ \$  \#  \$ ganti ['a', 'b'] de
Daftar cetak  \$  \#  \$ \$  \#  \$ ['z', 'c', 'd'] \par
```

Tipe data daftar memiliki beberapa metode lagi. Berikut adalah semua metode daftar objek:

1. List.append (x)

   Tambahkan item ke bagian akhir daftar. Setara dengan [len (a):] = [x].

2. list.extend (iterable)

   Perluas daftar dengan menambahkan semua item dari iterable. Setara dengan [len (a):] = iterable.

3. list.insert (i, x)

   Masukkan item pada posisi tertentu. Argumen pertama adalah indeks dari elemen yang sebelum dimasukkan, jadi a.insert (0, x) memasukkan di bagian depan daftar, dan a.insert (len (a), x) setara dengan a.append (x).

4. List.remove (x)

Hapus item pertama dari daftar yang nilainya x. Ini adalah kesalahan jika tidak ada item seperti itu.

5. List.pop ([i])

Hapus item pada posisi yang diberikan dalam daftar, dan kembalikan. Jika tidak ada indeks yang ditentukan, a.pop () menghapus dan mengembalikan item terakhir dalam daftar. (Tanda kurung siku di sekitar i pada tanda tangan metode menunjukkan bahwa parameternya adalah opsional, bukankah Anda harus mengetikkan tanda kurung siku pada posisi itu. Anda akan sering melihat notasi ini di Referensi Perpustakaan Python.)

6. List.clear ()

Hapus semua item dari daftar. Setara dengan del a [:].

7. List.index (x [, start [, end]])

Kembalikan indeks berbasis nol dalam daftar item pertama yang nilainya x. Meningkatkan ValueError jika tidak ada item seperti itu.

Argumen dan argumen opsional dimulai dengan interpretasi seperti notasi irisan dan digunakan untuk membatasi pencarian ke urutan berikutnya dari daftar. Indeks yang dikembalikan dihitung relatif terhadap awal urutan penuh daripada argumen awal.

8. List.count (x)

Kembalikan berapa kali x muncul dalam daftar.

9. List.sort (key = None, reverse = False)

Urutkan item daftar di tempat (argumen dapat digunakan untuk kustomisasi sortir, lihat diurutkan () untuk penjelasan mereka).

10. List.reverse ()

    Membalikkan unsur daftar di tempat.

11. List.copy ()

    Kembalikan salinan daftar yang dangkal. Setara dengan [:].

Contoh yang menggunakan sebagian besar metode daftar:

```
>>> fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'ap
>>> fruits.count('apple') \par
2 \par
>>> fruits.count('tangerine') \par
0 \par
>>> fruits.index('banana') \par
3 \par
>>>~fruits.index('banana', 4)   \$  \#  \$ Find next banana st
6 \par
>>> fruits.reverse() \par
>>> fruits \par
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange
>>> fruits.append('grape') \par
>>> fruits \par
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange
>>> fruits.sort() \par
>>> fruits \par
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orang
>>> fruits.pop() \par
'pear' \par
```

mungkin telah memperhatikan bahwa metode seperti insert, remove atau sortir yang hanya memodifikasi daftar tidak memiliki nilai pengembalian tercetak - mereka mengembalikan default None. [1] Ini adalah prinsip desain untuk semua struktur data yang bisa berubah dengan Python.

```
>>> stack = [3, 4, 5] \par
>>> stack.append (6) \par
>>> stack.append (7) \par
>>> susun \par
[3, 4, 5, 6, 7] \par
>>> stack.pop () \par
7 \par
>>> susun \par
[3, 4, 5, 6] \par
>>> stack.pop () \par
6 \par
>>> stack.pop () \par
5 \par
>>> susun \par
[3, 4] \par
```

9.2 Menggunakan Daftar sebagai Antrian

Hal ini juga memungkinkan untuk menggunakan daftar sebagai antrian, di mana elemen pertama yang ditambahkan adalah elemen pertama yang diambil (f̈irst-in, first-ouẗ); Namun, daftar tidak efisien untuk tujuan ini. Sementara menambahkan dan muncul dari akhir daftar dengan cepat, melakukan sisipan atau muncul dari awal daftar lambat (karena semua elemen lainnya harus digeser oleh satu).

Untuk menerapkan antrean, gunakan collections.deque yang dirancang agar cepat ditambahkan dan muncul dari kedua ujungnya. Sebagai contoh:

¿¿¿

```
>>> dari koleksi import deque \par
>>> antrian = deque (["Eric", "John", "Michael"]) \par
>>> queue.append ("Terry")  \$  \#  \$ Terry tiba \par
>>> queue.append ("Graham")  \$  \#  \$ Graham tiba \par
>>> queue.popleft ()  \$  \#  \$ Yang pertama tiba sekarang pe
'Eric' \par
>>> queue.popleft ()  \$  \#  \$ Yang kedua tiba sekarang perg
'John' \par
>>> antrian  \$  \#  \$ Sisa antrian sesuai urutan kedatangan
deque (['Michael', 'Terry', 'Graham']) \par
```

List adalah sejumlah object yang dipisahkan oleh tanda koma (,) dan diapit oleh kurung siku ([ ]). Begini contohnya:

```
>>> a = [1.0, 2.0, 3.0] # cara membuat list
>>> a.append(4.0) # tambahkan 4.0 kedalam list
>>> print a
[1.0, 2.0, 3.0, 4.0]
>>> a.insert(0,0.0) # sisipkan 0.0 pada posisi 0
>>> print a
[0.0, 1.0, 2.0, 3.0, 4.0]
>>> print len(a) # menentukan panjang list
5
```

Jika kita memberikan statemen b = a, maka itu tidak berarti bahwa variabel b terpisah dengan variabel a. Di python, statemen seperti itu diartikan hanya sebagai pemberian nama lain (alias) kepada variabel a. Artinya, perubahan yang terjadi baik itu di a ataupun di b, maka hasil akhir mereka berdua akan sama saja. Setiap perubahan yang terjadi di b akan berdampak di a. Untuk meng-copy a secara independen, gunakan statemen c = a[:], sebagaimana dicontohkan berikut ini :

```
>>> a = [1.0, 2.0, 3.0]
>>> b = a # b adalah alias dari a
>>> b[0] = 5.0 # isi elemen b diubah
>>> print a
[5.0, 2.0, 3.0] # perubahan di b tercermin di a
>>> c = a[:] # c kopian dari a
>>> c[0] = 1.0 # isi elemen c diubah
```

```
>>> print a
[5.0, 2.0, 3.0] # a tidak dipengaruhi c
```

Matrik dapat direpresentasikan sebagai list-list yang disusun berbaris. Berikut adalah matrik./citesuparno2013komputasi 3 3 dalam bentuk list:

```
>>> a = [[1, 2, 3], \
[4, 5, 6], \
[7, 8, 9]]
>>> print a[1] # Print baris kedua (elemen 1)
```

CHAPTER 10

TUPLES

10.1 TUPLES

Sebuah tupel adalah urutan objek Python yang tidak berubah. Tupel adalah urutan, seperti daftar. Perbedaan antara tupel dan daftar adalah, tupel tidak dapat diubah tidak seperti daftar dan tupel menggunakan tanda kurung, sedangkan daftar menggunakan tanda kurung siku. Tuple juga merupakan tipe data yang berurut (sequence data type) yang fungsinya hampir sama List. Namun Tuple juga berbeda sifatnya, yaitu Tuple bersifat immutable maksudnya data di dalam Tuple tidak dapat diubah atau dihapuskan. Sebuah Tuple terdiri dari beberapa nilai yang dipisah oleh tanda koma , . Tidak seperti List, tipe data Tuple ditandai dengan tanda kurung ().. Ini adalah contohnya,

10.2 Perbedaan antara list dengan tuple

1. Nilai dalam Tuple tidak bisa diganti.
2. Kalau dalam List diawali dan diakhiri dengan tanda kurung siku [], Tuple dimulai dan ditutup dengan tanda kurung .

Nilai-nilai pada Tuple yang sudah tentukan pada awal program tidak akan bisa diganti sampai akhir programnya. Dan tentunya index pada tuple selalu dimulai dari angka 0

Membuat Tuple sangatlah mudah, nilai-nilai dapat dipisahkan dengan tanda koma. Seperti contoh di bawah ini :

```
Tup1 = ("awal","tengah","akhir", 1,2);

months = ('January','February','March','April','May','June','J
```

Berikut ini contoh membuat Tuple, kemudian menampilkan nilai, update $dalamhalinibukanmeng-updatenilaidalamTuplemelainkanmembuatTuplebarudankemudiandigabungkanden$ dan delete Tuple $menghapusTuple, bukanmenghapusnilaidalamTuple$.

```
>>> NamaSiswa = ("Indra Riksa", "Agien Farhan", "Saryoni", "Be
>>> NamaSiswa
('Indra Riksa', 'Agien Farhan', 'Saryoni, 'Berlin', 'Kindi')
```

Kita dapat mengisi sebuah Tuple tanpa memakai tanda kurung, tapi hal ini tidak dianjurkan jika Tuple tersebut berisi data yang besar. Contoh di bawah ini jika kita ingin membuat Tuple bersarang, ada Tuple di dalam Tuple.

```
>>> NamaKota = "Surabaya", "Jakarta"
>>> NamaKota
('Surabaya', 'Jakarta')
>>> KotaBesar = NamaKota, ("Bandung", "Yogyakarta", "Medan")
>>> KotaBesar
(('Surabaya', 'Jakarta'), ('Bandung', 'Yogyakarta', 'Medan'))
```

Sesuai dengan yang sudah dibahas di awal tadi, perbedaan utama dari Tuple dan List yaitu : Tuple bersifat immutable

tetap, kita tidak diizinkan untuk mengganti nilai yang ada atau menghapus data yang ada dalam Tuple tersebut. Jika kita menghapus atau mengubah data yang sudah ada sebelumnya, maka pesan kesalahan akan di tampilkan oleh interpreter Python.

```
>>> NamaKota[1] = \"Medan\"
Traceback (most recent call last):
File "", line 1, in
NamaKota[1] = \"Medan\"

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

Kita dapat menggunakan indeks atau irisan untuk mengakses nilai yang ada di dalam Tuple. Berikut contohnya,

```
>>> NamaSiswa[1]
'Indra Riksa'
>>> NamaSiswa[0:2]
('Indra Riksa', 'Agien Farhan')
>>> KotaBesar[:2]
(('Surabaya', 'Jakarta'), ('Bandung', 'Yogyakarta', 'Medan'))
>>> KotaBesar[1][0]
'Bandung'

>>> TupleAku = ("a", 2, 3, 4)
>>> TupleAku
('a', 2, 3, 4)
>>> TupleDia = ("b", 5, 6, 7)
>>> TupleDia
('b', 5, 6, 7)
>>> TupleGab = TupleAku + TupleDia
>>> TupleGab
('a', 2, 3, 4, 'b', 5, 6, 7)
```

Contoh di atas, dua Tuple dibuat secara terpisah, TupleAku dan TupleDia. Dua Tuple ini dicocokkan pada Tuple lainnya TupleGab menggunakan operator +. Perlu dicatat bahwa TupleGab berisi nilai dari TupleAku dan TupleDia. Metode ini dapat digunakan untuk menambahkan elemen data lain pada sebuah Tuple.

```
>>> IniTuple = ("x", "y", "z")
>>> IniTuple = IniTuple + ("a", "b")
>>> IniTuple
('x', 'y', 'z', 'a', 'b')
```

Dan kita juga dapat membuat Tuple dengan objek-objek yang immutable yaitu seperti List. Sedemikian sehingga, kita dapat mengubah nilai yang ada dalam List tersebut. Berikut contohnya:

```
>>> TupleData = (222, "ayam", [555, "telur", "sapi"])
>>> TupleData
(222, 'ayam', [555, 'telur', 'sapi'])
>>> TupleData[2][1] = 777
>>> TupleData
(222, 'ayam', [555, 777, 'sapi'])
```

Pada contoh tersebut, pertama kita membuat sebuah Tuple yang berisi sebuah List. Setelah itu, kita ubah sebuah nilai yang ada dalam List tersebut. Dapat disimpulkan bahwa objek multitable dalam Tuple dapat diubah, meskipun Tuple sendiri bersifat immutable.

Kita juga bisa menggabungkan beberapa tuples. Untuk menggabungkan beberapa tuples, Anda dapat menggunakan operator concatenation +.

Berikut adalah contohnya:

```
#Nama File : tuples_concat.py
tup1 = (1998, 1997);
tup2 = ('Agien Farhan', 'Kindi');

# Buat variable tuples baru untuk menampung hasilnya
tup3 = tup1 + tup2;
print (tup3)
```

Jika kalian jalankan program diatas, maka akan menghasilkan output sebagai berikut: (1998, 1997, 'Agien Farhan', 'Kindi')

Jika kita ingin membuat sebuah sebuah variable dengan Tuple kosong, kita hanya cukup memberikan tanda kurung pada variabel tersebut. Panjang Tuple kosong tersebut adalah 0. Berikut adalah contohnya,

```
>>> TupleBebas = ()
>>> TupleBebas
()
>>> len(TupleBebas)
0
```

Jika kita membuat sebuah Tuple yang hanya isi berisi satu data, maka harus ditambahkan sebuah tanda yaitu koma. Jika tidak menggunakan sebuah tanda koma, maka tipe data tersebut akan dianggap sebagai tipe variabel dari sebuah Tuple. Berikut adalah contohnya,

```
>>> SatuData = ("Kymco")
>>> len(SatuData)
5
```

Berikut contoh jika kita memberikan tanda koma,

```
>>> TupleSatu = ("Kymco",)
>>> TupleSatu
('Kymco',)
>>> len(TupleSatu)
1
```

Membuat tuple semudah memasukkan nilai-nilai yang dipisahkan koma. Opsional Anda dapat memasukkan nilai-nilai yang dipisahkan koma ini di antara tanda kurung juga. Misalnya :

```
Tup1 = ('fisika', 'kimia', 1997, 2000);
Tup2 = (1, 2, 3, 4, 5);
Tup3 = "a", "b", "c", "d";
```

Tuple kosong ditulis sebagai dua tanda kurung yang tidak berisi apa : tup1 = (); Untuk menulis tupel yang berisi satu nilai, Anda harus menyertakan koma, meskipun hanya ada satu nilai : Tup1 = (50,); Seperti indeks string, indeks tuple mulai dari 0, dan mereka dapat diiris, digabungkan, dan seterusnya.

10.3 Mengakses Nilai pada Tuples

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

```
 \$  \#  \$! / Usr / bin / python

Tup1 = ('fisika', 'kimia', 1997, 2000);
Tup2 = (1, 2, 3, 4, 5, 6, 7);

Cetak "tup1 [0]:", tup1 [0]
Cetak "tup2 [1: 5]:", tup2 [1: 5]
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut - tup1 [0]: fisika Tup2 [1: 5]: [2, 3, 4, 5]

10.4 Memperbarui Tupel

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

```
 \$  \#  \$\! / Usr / bin / python \

Tup1 = (12, 34.56);
Tup2 = ('abc', 'xyz');

 \$  \# \$ Tindakan berikut tidak berlaku untuk tupel
 \$  \#  \$ Tup1 [0] = 100;

 \$  \#  \$ Jadi mari kita buat tupel baru sebagai berikut
Tup3 = tup1 + tup2;
Cetak tup3
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut : $12, 34.56,' abc',' xyz'$

10.5 Hapus Elemen Tuple

Menghapus elemen tuple individual tidak mungkin dilakukan. Tentu saja, tidak ada yang salah dengan menggabungkan tuple lain dengan unsur-unsur yang tidak diinginkan dibuang. Untuk secara eksplisit menghapus keseluruhan tuple, cukup gunakan del statement. Sebagai contoh:

```
\$  \#  \$! / Usr / bin / python

Tup = ('fisika', 'kimia', 1997, 2000);

Cetak tup
Del tup;
Cetak "Setelah menghapus tup:"
Cetak tup
```

Ini menghasilkan hasil berikut. Perhatikan pengecualian yang diangkat, ini karena setelah del tup tupel tidak ada lagi : $'Fisika', 'kimia', 1997, 2000$ Setelah menghapus tup: Traceback $panggilanterakhir$: Ñile "test.py", baris 9, di ¡module¿ Cetak tup; NameError: nama 'tup' tidak didefinisikan

10.6 Operasi Tuple Dasar

Tupel merespons operator + dan * seperti string; Mereka berarti penggabungan dan pengulangan di sini juga, kecuali hasilnya adalah tupel baru, bukan string. Sebenarnya, tupel menanggapi semua operasi urutan umum yang kami gunakan pada senar di bab sebelumnya : Python Expression, Results, dan Description

```
len((1, 2, 3) 3 Length
(1, 2, 3) + (4, 5, 6) (1, 2, 3, 4, 5, 6) Concatenation
('Hi!',) * 4 ('Hi!', 'Hi!', 'Hi!', 'Hi!') Repetition
3 in (1, 2, 3) True Membership
for x in (1, 2, 3): print x, 1 2 3 Iteration
```

10.6.1 Indexing, Slicing, dan Matrixes

Karena tupel adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk tupel seperti yang mereka lakukan untuk string. Dengan asumsi masukan berikut : L = ('spam', 'Spam', 'SPAM!')

Python Expression Results Description L[2] 'SPAM!' Offsets start at zero L[-2] 'Spam' Negative: count from the right L[1:] ['Spam', 'SPAM!'] Slicing fetches sections

10.7 Tidak melampirkan delimiters

Setiap kumpulan beberapa objek, yang dipisahkan koma, ditulis tanpa mengidentifikasi simbol, yaitu tanda kurung untuk daftar, tanda kurung untuk tupel, dll., Default tupel, seperti yang ditunjukkan dalam contoh singkat ini :

```
\$  \#  \$! / Usr / bin / python
cetak 'abc', -4.24e93, 18 + 6.6j, 'xyz'
x, y = 1, 2;
Cetak "Nilai x, y:", x, y
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut :

```
abc -4.24e + 93 (18 + 6.6j) xyz
Nilai x, y: 1 2
Built-in Fungsi Tuple
```

Python mencakup fungsi tupel berikut : SN Function with Description 1 cmp(tuple1, tuple2)

Compares elements of both tuples. 2 len(tuple)

Gives the total length of the tuple. 3 max(tuple)

Returns item from the tuple with max value. 4 min(tuple)

Returns item from the tuple with min value. 5 tuple(seq)

10.8 Converts a list into tuple

Dalam pemrograman Python, tuple mirip dengan daftar. Perbedaan antara keduanya adalah kita tidak bisa mengubah unsur tuple begitu diberikan sedangkan dalam daftar, elemen bisa diubah.

10.9 Keuntungan Tuple over List

Karena, tupel sangat mirip dengan daftar, keduanya juga digunakan dalam situasi yang sama. Namun, ada beberapa keuntungan dari penerapan tupel dari daftar. Di bawah ini tercantum beberapa keuntungan utama:

1. Kami umumnya menggunakan tuple untuk tipe data heterogen dan berbeda untuk tipe data homogen (sejenis).

2. Karena tupel tidak dapat diubah, iterasi melalui tupel lebih cepat daripada daftar. Jadi ada sedikit peningkatan kinerja.

3. Tupel yang mengandung unsur yang tidak berubah dapat digunakan sebagai kunci untuk kamus. Dengan daftar, ini tidak mungkin.

4. Jika Anda memiliki data yang tidak berubah, menerapkannya sebagai tupel akan menjamin bahwa itu tetap dilindungi penulisan.

Dalam pemrograman Python, tuple mirip dengan daftar. Perbedaan antara keduanya adalah kita tidak bisa mengubah unsur tuple begitu diberikan sedangkan dalam daftar, elemen bisa diubah.

10.10 Membuat Tuple

Sebuah tuple dibuat dengan menempatkan semua item (elemen) di dalam tanda kurung (), dipisahkan dengan koma. Tanda kurung bersifat opsional namun merupakan praktik yang baik untuk menuliskannya. Sebuah tuple dapat memiliki sejumlah item dan mereka mungkin memiliki tipe yang berbeda (integer, float, list, string etc.). 10.1 10.2

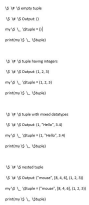


Figure 10.1

Figure 10.2

10.11 Membuat tuple dengan satu elemen

Memiliki satu elemen dalam kurung saja tidak cukup. Kita membutuhkan koma trailing untuk menunjukkan bahwa sebenarnya ada tupel.

```
 $  \#  $ only parentheses is not enough
 $  \#  $ Output: <class 'str'>
my $  \_  $tuple = ("hello")
print(type(my $  \_  $tuple))

 $  \#  $ need a comma at the end
 $  \#  $ Output: <class 'tuple'>
my $  \_  $tuple~= ("hello",)
print(type(my $  \_  $tuple))

 $  \#  $ parentheses is optional
 $  \#  $ Output: <class 'tuple'>
my $  \_  $tuple = "hello",
print(type(my $  \_  $tuple))
```

10.12 Mengakses Elemen dalam Tuple

Ada berbagai cara untuk mengakses elemen tuple.

10.12.1 Pengindeksan

Kita bisa menggunakan operator indeks [] untuk mengakses item di tupel dimana indeks dimulai dari 0. Jadi, tupel yang memiliki 6 elemen akan memiliki indeks dari 0 sampai 5. Mencoba mengakses elemen lain yang (6, 7, ...) akan menghasilkan IndexError. Indeks harus berupa bilangan bulat, jadi kita tidak bisa menggunakan float atau jenis lainnya. Ini akan menghasilkan TypeError. Demikian juga, tuple bersarang diakses menggunakan pengindeksan nested, seperti yang ditunjukkan pada contoh di bawah ini.

```
my $  \_  $tuple = ('p','e','r','m','i','t')

 $  \#  $ Output: 'p'
print(my $  \_  $tuple[0]) \

 $  \#  $ Output: 't'
print(my $  \_  $tuple[5])

 $  \#  $ index must be in range
 $  \#  $ If you uncomment line 14,
 $  \#  $ you will get an error.
 $  \#  $ IndexError: list index out of range

 $  \#  $print(my $  \_  $tuple[6])

 $  \#  $ index must be an integer
 $  \#  $ If you uncomment line 21,
 $  \#  $ you will get an error.
 $  \#  $ TypeError: list indices must be integers, not float

 $  \#  $my $  \_  $tuple[2.0]

 $  \#  $ nested tuple
n $  \_  $tuple = ("mouse", [8, 4, 6], (1, 2, 3))

 $  \#  $ nested index
 $  \#  $ Output: 's'
print(n $  \_  $tuple[0][3])
```

```
 $  \#  $ nested index
 $  \#  $ Output: 4
print(n $  \_  $tuple[1][1])
```

10.12.2 Slicing

Kita bisa mengakses berbagai item dalam tupel dengan menggunakan operator pengiris - titik dua ":".

```
my $  \_  $tuple = ('p','r','o','g','r','a','m','i','z')

 $  \#  $ elements 2nd to 4th
 $  \#  $ Output: ('r', 'o', 'g')
print(my $  \_  $tuple[1:4])

 $  \#  $ elements beginning to 2nd
 $  \#  $ Output: ('p', 'r')
print(my $  \_  $tuple[:-7])

 $  \#  $ elements 8th to end
 $  \#  $ Output: ('i', 'z')
print(my $  \_  $tuple[7:])

 $  \#  $ elements beginning to end
 $  \#  $ Output: ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z'
print(my $  \_  $tuple[:])
```

10.13 Mengubah Tuple

Tidak seperti daftar, tupel tidak dapat diubah. Ini berarti elemen tupel tidak dapat diubah begitu telah ditetapkan. Tapi, jika elemen itu sendiri adalah datatype yang bisa berubah seperti daftar, item nested-nya bisa diubah. Kita juga bisa menugaskan tuple ke nilai yang berbeda $reassignment$.

```
my $  \_  $tuple = (4, 2, 3, [6, 5])

 $  \#  $ we cannot change an element
 $  \#  $ If you uncomment line 8
```

```
 $  \#  $ you will get an error:
 $  \#  $ TypeError: 'tuple' object does not support item assi

 $  \#  $my $  \_  $tuple[1] = 9

 $  \#  $ but item of mutable element can be changed
 $  \#  $ Output: (4, 2, 3, [9, 5])
my $  \_  $tuple[3][0] = 9
print(my $  \_  $tuple)

 $  \#  $ tuples can be reassigned
 $  \#  $ Output: ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z'
my $  \_  $tuple = ('p','r','o','g','r','a','m','i','z')
print(my $  \_  $tuple)
```

10.14 Python Tuples

Tutorial Tuple Python menjelaskan tupel dan bagaimana menggunakannya dengan Python. Dengan Python, tupel hampir sama dengan daftar. Jadi, mengapa kita harus menggunakannya? Satu perbedaan utama antara tupel dan daftar adalah bahwa tupel tidak dapat diubah. Artinya, Anda tidak dapat menambahkan, mengubah, atau menghapus elemen dari tuple. Tupel mungkin tampak aneh pada awalnya, tapi ada alasan bagus mengapa mereka tidak bisa berubah. Sebagai pemrogram, kita mengacaukan sesekali. Kami mengubah variabel yang tidak ingin kami ubah, dan terkadang, kami hanya ingin hal-hal menjadi konstan sehingga kami tidak sengaja mengubahnya nanti. Namun, jika kita mengubah pikiran kita, kita juga bisa mengubah tupel menjadi daftar atau daftar menjadi tupel. Faktanya adalah kita perlu membuat usaha sadar untuk mengatakan Python, saya ingin mengubah tupel ini menjadi sebuah daftar sehingga saya bisa memodifikasinya. Cukup mengoceh, mari kita lihat sebuah tuple beraksi! Otak Anda masih sakit dari pelajaran terakhir? Jangan khawatir, yang satu ini akan membutuhkan sedikit pemikiran. Kita akan kembali ke sesuatu yang sederhana - variabel - tapi sedikit lebih mendalam. Pikirkanlah

- variabel menyimpan satu bit informasi. Mereka mungkin muntah-muntah (tidak di karpet ...) informasi itu kapan saja, dan sedikit informasi mereka dapat berubah sewaktu-waktu. Variabel sangat bagus dengan apa yang mereka lakukan - menyimpan informasi yang mungkin berubah seiring berjalannya waktu.

Tapi bagaimana jika Anda perlu menyimpan daftar panjang informasi, yang tidak berubah dari waktu ke waktu? Katakanlah, misalnya, nama bulan dalam setahun. Atau mungkin daftar panjang informasi, itu memang berubah seiring berjalannya waktu? Katakanlah, misalnya, nama semua kucing Anda. Anda mungkin mendapatkan kucing baru, beberapa mungkin mati, beberapa mungkin menjadi makan malam Anda (kami harus menukar resep!). Bagaimana dengan buku telepon? Untuk itu Anda perlu melakukan sedikit referensi - Anda akan memiliki daftar nama, dan dilampirkan pada masing-masing nama tersebut, nomor teleponnya. Bagaimana Anda melakukannya?

Untuk ketiga masalah ini, Python menggunakan tiga solusi berbeda - daftar, tupel, dan kamus:

1. Daftar adalah apa yang mereka tampaknya - daftar nilai. Masing-masing diberi nomor, mulai dari nol - yang pertama diberi nomor nol, yang kedua 1, yang ketiga 2, dll. Anda dapat menghapus nilai dari daftar, dan menambahkan nilai baru sampai akhir. Contoh: nama kucing Anda banyak.

2. Tupel sama seperti daftar, tapi Anda tidak dapat mengubah nilainya. Nilai yang Anda berikan terlebih dahulu, adalah nilai yang Anda pakai untuk sisa program. Sekali lagi, setiap nilai diberi nomor mulai dari nol, untuk referensi mudah. Contoh: nama bulan dalam setahun.

3. Kamus serupa dengan apa yang namanya namanya - kamus. Dalam kamus, Anda memiliki 'indeks' kata-kata, dan untuk masing-masing definisi. Dengan kata Python, kata itu disebut 'kunci', dan definisi sebuah 'nilai'. Nilai dalam kamus tidak diberi nomor - keduanya tidak sesuai urutan tertentu, kuncinya adalah hal yang sama. (Se-

tiap tombol harus unik, meskipun!) Anda dapat menambahkan, menghapus, dan memodifikasi nilai-nilai di kamus. Contoh: buku telepon jadi ada yang lebih hidup dari pada nama kucing Anda. Anda perlu menghubungi saudara perempuan, ibu, anak laki-laki, pria buah, dan orang lain yang perlu tahu bahwa kucing favorit mereka sudah meninggal. Untuk itu Anda membutuhkan buku telepon.

Sekarang, daftar yang telah kami gunakan di atas tidak sesuai untuk buku telepon. Anda perlu mengetahui nomor berdasarkan nama seseorang - bukan sebaliknya, seperti yang kami lakukan pada kucing. Dalam contoh bulan dan kucing, kami memberi nomor komputer, dan itu memberi kami sebuah nama. Kali ini kami ingin memberi nama komputer, dan ini memberi kami nomor. Untuk ini kita butuh kamus.

Jadi bagaimana kita membuat kamus? Letakkan peralatan pengikat Anda, bukan itu yang maju. ingat, kamus memiliki kunci, dan nilai. Dalam buku telepon, Anda punya nama orang, lalu nomor mereka. Melihat kesamaan? Saat pertama kali membuat kamus, sangat mirip membuat tupel atau daftar. Tupel memiliki (dan) benda, daftar memiliki [dan] benda. Tebak apa! kamus memiliki $ { $dan $ } $ hal - kurung kurawal. Berikut adalah contoh di bawah ini, menampilkan kamus dengan empat nomor telepon di dalamnya:

Selain List, Tuple juga merupakan tipe data urutan $sequencedatatype$ yang secara fungsi sama dengan List. Namun Tuple berbeda sifatnya, yaitu Tuple bersifat immutable yang mana data di dalam Tuple tidak dapat kita ubah atau dihapuskan. Sebuah Tuple terdiri dari beberapa nilai yang dipisahkan oleh tanda koma (,). Tidak seperti List, tipe data Tuple ditandai dengan tanda kurung ().

Tupel berguna untuk mewakili bahasa lain yang sering disebut catatan beberapa informasi terkait yang dimiliki bersama, seperti catatan siswa Anda. Tidak ada deskripsi tentang apa yang masing-masing bidang ini berarti, tapi kita bisa menebaknya. Sebuah tupel memungkinkan kita

m̈engumpulkanïnformasi yang terkait dan menggunakannya sebagai satu hal.

CHAPTER 11

DICTIONARY

11.1 Python Dictionary

Dictionary Python adalah kumpulan pasangan kunci:nilai (selanjutnya disebut: key-value) yang tak berurutan. Dictionary Python ini sama halnya dengan hash-table atau array-asosiatif di pemrograman Perl.

Suatu kunci (key) pada Dictionary bersifat Unique (unik), yang artinya adalah satu kunci hanya memiliki satu nilai. Aturan penulisannya berupa key:value. Sebuah Dictionary ditandai dengan adanya kurung kurawal . Setiap pasangan key:value dipisah dengan tanda koma.

Setiap kunci dipisahkan dari nilainya oleh titik dua (:), item dipisahkan oleh koma, dan semuanya tertutup dalam kurung kurawal. Kamus kosong tanpa barang ditulis hanya dengan dua kurung kurawal, seperti ini: $ { $ $ } $.

Kunci unik dalam kamus sementara nilai mungkin tidak. Nilai kamus bisa berupa tipe apa pun, namun kunci harus berupa tipe data yang tidak berubah seperti string, angka, atau tupel.

Dictionary menyediakan wadah yang sangat berguna yang memungkinkan kita mencari nilai menggunakan kunci[4]. Dalam sebuah Dictionary tidak boleh ada dua key yang sama, maka memberikan nilai ke key yang sudah ada akan menghapus nilai yang lama. Untuk menambah key baru, Python memiliki syntax yang sama. Ini bisa menimbulkan kerancuan pada saat akan melakukan penambahan key baru, tapi ternyata hanya mengubah key yang sudah ada. Sebagai catatan, key yang baru terletak di bagian tengah bukan di belakang. Hal tersebut kebetulan saja, karena sebenarnya pada Dictionary memang tidak ada standar pengurutan yang berlaku.

11.1.1 Accessing Values in Dictionary

Untuk mengakses elemen kamus, Anda dapat menggunakan tanda kurung siku yang sudah dikenal bersama dengan kunci untuk mendapatkan nilainya. Berikut adalah contoh sederhana :

```
  \$  \#  \$!/usr/bin/python
  dict =  \$  \{  \$'Name': 'Zara', 'Age': 7, 'Class': 'First'
 print "dict['Name']: ", dict['Name']
 print "dict['Age']: ", dict['Age']
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut : dict['Name']: Zara dict['Age']: 7 Jika kita mencoba mengakses item data dengan sebuah kunci, yang bukan bagian dari kamus, kita mendapatkan error sebagai berikut :

```
\$  \#  \$!/usr/bin/python
dict =  \$  \{  \$'Name': 'Zara', 'Age': 7, 'Class': 'First' \
print "dict['Alice']: ", dict['Alice']
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut :

```
dict['Alice']:
Traceback (most recent call last):
~~ File "test.py", line 4, in <module>
~~~~~  \print "dict['Alice']: ", dict['Alice'];
KeyError: 'Alice'
```

11.1.2 Updating Dictionary

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

```
\$  \#  \$!/usr/bin/python
  dict =  \$  \{  \$'Name': 'Zara', 'Age': 7, 'Class': 'First'
  dict['Age'] = 8;  \$  \#  \$ update existing entry
  dict['School'] = "DPS School";  \$  \#  \$ Add new entry
  print "dict['Age']: ", dict['Age']
  print "dict['School']: ", dict['School']
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut : dict['Age']: 8 dict['School']: DPS School

11.1.3 Delete Dictionary Elements

Anda dapat menghapus elemen kamus individual atau menghapus keseluruhan isi kamus. Anda juga dapat menghapus seluruh kamus dalam satu operasi. Untuk menghapus seluruh kamus secara eksplisit, cukup gunakan del statement. Berikut adalah contoh sederhana :

```
   \$  \#  \$!/usr/bin/python
  dict =  \$  \{  \$'Name': 'Zara', 'Age': 7, 'Class': 'First'
  del dict['Name'];  \$  \#  \$ remove entry with key 'Name'
  dict.clear();~~~~  \$  \#  \$ remove all entries in dict
  del dict~;~~~~~~   \$  \#  \$ delete entire dictionary
  print "dict['Age']: ", dict['Age']
  print "dict['School']: ", dict['School']
```

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

lagi - dict[’Age’]: Traceback (most recent call last): File ”test.py”, line 8, in ¡module¿ print ”dict[’Age’]: ”, dict[’Age’]; TypeError: ’type’ object is unsubscriptable Note: del () metode dibahas di bagian selanjutnya.

11.1.4 Properties of Dictionary Keys

Nilai kamus tidak memiliki batasan. Mereka bisa menjadi objek Python yang sewenang-wenang, baik objek standar atau objek yang ditentukan pengguna. Namun, hal yang sama tidak berlaku untuk kunci. Ada dua hal penting yang perlu diingat tentang kunci kamus Lebih dari satu entri per kunci tidak diperbolehkan. Yang berarti tidak ada kunci duplikat yang diperbolehkan. Ketika kunci duplikat ditemui selama penugasan, tugas terakhir akan menang. Sebagai contoh

```
 \$  \#  \$!/usr/bin/python
dict =  \$  \{  \$'Name': 'Zara', 'Age': 7, 'Name': 'Manni'
print "dict['Name']: ", dict['Name']
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut : dict[’Name’]: Manni (b) Tombol harus tidak berubah. Yang berarti Anda bisa menggunakan string, angka atau tupel sebagai tombol kamus tapi sesuatu seperti [’key’] tidak diperbolehkan. Berikut adalah contoh sederhana:

```
 \$  \#  \$!/usr/bin/python
dict =  \$  \{  \$['Name']: 'Zara', 'Age': 7 \$  \}  \$
print "dict['Name']: ", dict['Name']
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut : Traceback (most recent call last): File ”test.py”, line 3, in ¡module¿ dict = $ { $[’Name’]: ’Zara’, ’Age’: 7 $ } $; TypeError: list objects are unhashable Built-in Dictionary Functions $ & $ Methods $ - $ Python includes the following dictionary functions $ - $ Python includes following dictionary methods $ - $

11.1.5 Dictionaries Introductions

Kami sudah mengenal daftar di bab sebelumnya. Di bab kusus Python online kami akan mempresentasikan kamus dan operator dan metode pada kamus. Program atau skrip Python tanpa daftar dan kamus hampir tidak dapat dibayangkan. Seperti daftar kamus yang bisa dengan mudah diubah, bisa menyusut dan berkembang ad libitum pada saat run time. Mereka menyusut dan tumbuh tanpa perlu membuat salinan. Kamus dapat dimuat dalam daftar dan sebaliknya. Tapi apa perbedaan antara daftar dan kamus? Daftar diurutkan dari objek, sedangkan kamus tidak berurutan. Tapi perbedaan utamanya adalah item dalam kamus diakses melalui kunci dan tidak melalui posisinya. Kamus adalah array asosiatif (juga dikenal sebagai hash). Kunci kamus mana pun dikaitkan (atau dipetakan) ke sebuah nilai. Nilai kamus bisa berupa tipe data Python. Jadi kamus adalah pasangan kunci-nilai tak berurutan. Kamus tidak mendukung urutan operasi dari jenis data urutan seperti string, tupel dan daftar. Kamus termasuk tipe pemetaan built-in. Mereka adalah satu-satunya wakil semacam ini! Di akhir bab ini, kami akan menunjukkan bagaimana kamus dapat diubah menjadi satu daftar, berisi (kunci, nilai) -tupel atau dua daftar, yaitu satu dengan kunci dan satu dengan nilainya. Transformasi ini bisa dilakukan secara terbalik juga.

11.1.6 How to create a dictionary?

Membuat kamus sama mudahnya dengan menempatkan item dalam kurung kurawal $ { $ $ } $ dipisahkan dengan koma. Item memiliki kunci dan nilai yang sesuai dinyatakan sebagai pasangan, kunci: nilai. Sementara nilai dapat berupa tipe data apa pun dan dapat diulang, kunci harus terdiri dari tipe yang tidak dapat diubah (string, number atau tupel dengan elemen yang tidak berubah) dan harus unik.

```
  \$  \#  \$ empty dictionary
my \$  \_  \$dict =  \$  \{  \$ \$  \}  \$
  \$  \#  \$ dictionary with integer keys
my \$  \_  \$dict =  \$  \{  \$1: 'apple', 2: 'ball' \$  \}
```

```
 \$  \#  \$ dictionary with mixed keys
my \$  \_  \$dict =  \$  \{  \$'name': 'John', 1: [2, 4, 3]
 \$  \#  \$ using dict()
my \$  \_  \$dict = dict( \$  \{  \$1:'apple', 2:'ball' \$
 \$  \#  \$ from sequence having each item as a pair
my \$  \_  \$dict = dict([(1,'apple'), (2,'ball')])
```

Seperti yang bisa Anda lihat di atas, kita juga bisa membuat kamus menggunakan fungsi built-in dict ().

11.1.7 How to access elements from a dictionary?

Sementara pengindeksan digunakan dengan jenis wadah lain untuk mengakses nilai, kamus menggunakan tombol. Kunci dapat digunakan baik di dalam tanda kurung siku atau dengan metode get (). Perbedaan saat menggunakan get () adalah mengembalikan Elemen alih-alih KeyError, jika kuncinya tidak ditemukan.

```
my \$  \_  \$dict =  \$  \{  \$'name':'Jack', 'age': 26 \$
 \$  \#  \$ Output: Jack
print(my \$  \_  \$dict['name'])
 \$  \#  \$ Output: 26
print(my \$  \_  \$dict.get('age'))
 \$  \#  \$ Trying to access keys which doesn't exist throws
 \$  \#  \$ my \$  \_  \$dict.get('address')
 \$  \#  \$ my \$  \_  \$dict['address']
```

Saat menjalankan program, hasilnya adalah: Jack 26

11.1.8 How to change or add elements in a dictionary?

Kamus bisa berubah-ubah. Kita bisa menambahkan item baru atau mengubah nilai barang yang ada menggunakan operator penugasan. Jika kuncinya sudah ada, nilai akan diperbarui, jika ada kunci baru: pasangan nilai ditambahkan ke kamus. Script.py

```
my \$  \_  \$dict =  \$  \{  \$'name':'Jack', 'age': 26 \$
 \$  \#  \$ update value
my \$  \_  \$dict['age'] = 27
```

```
 \$  \#  \$Output:  \$  \{  \$'age': 27, 'name': 'Jack' \$
print(my \$  \_  \$dict)
 \$  \#  \$ add item
my \$  \_  \$dict['address']~= 'Downtown'
 \$  \#  \$ Output:  \$  \{  \$'address': 'Downtown', 'age':
print(my \$  \_  \$dict).
```

Saat menjalankan program, hasilnya adalah:
$ { $'name': 'Jack', 'age': 27 $ } $ $ { $'name': 'Jack', 'age': 27, 'address': 'Downtown' $ } $

11.1.9 How to delete or remove elements from a dictionary?

Kita bisa menghapus item tertentu dalam kamus dengan menggunakan metode pop (). Metode ini menghilangkan item dengan tombol yang disediakan dan mengembalikan nilainya. Metodenya, popitem () dapat digunakan untuk menghapus dan mengembalikan item yang sewenang-wenang (key, value) membentuk kamus. Semua item dapat dihapus sekaligus dengan menggunakan metode clear (). Kita juga bisa menggunakan kata kunci del untuk menghapus setiap item atau keseluruhan kamus itu sendiri. Scrip.py

```
 \$  \#  \$ create a dictionary
squares~=  \$  \{  \$1:1, 2:4, 3:9, 4:16, 5:25 \$  \}  \$
 \$  \#  \$ remove a particular item
 \$  \#  \$ Output: 16
print(squares.pop(4))~
 \$  \#  \$ Output:  \$  \{  \$1: 1, 2: 4, 3: 9, 5: 25 \$  \
print(squares)
 \$  \#  \$ remove an arbitrary item
 \$  \#  \$ Output: (1, 1)
print(squares.popitem())
 \$  \#  \$ Output:  \$  \{  \$2: 4, 3: 9, 5: 25 \$  \}  \$
print(squares)
 \$  \#  \$ delete a particular item
del~squares[5]
 \$  \#  \$ Output:  \$  \{  \$2: 4, 3: 9 \$  \}  \$
print(squares)
 \$  \#  \$ remove all items
```

```
squares.clear()
 \$  \#  \$ Output:  \$  \{  \$ \$  \}  \$
print(squares)
 \$  \#  \$ delete the dictionary itself
del squares
\$  \#  \$ Throws Error
\$  \#  \$ print(squares)
```

When you run the program, the output will be: 16 $ { $1: 1, 2: 4, 3: 9, 5: 25 $ } $ (1, 1) $ { $2: 4, 3: 9, 5: 25 $ } $ $ { $2: 4, 3: 9 $ } $ $ { $ $ } $ Tipe data daftar memiliki beberapa metode lagi. Berikut adalah semua metode daftar objek: list.append (x) Tambahkan item ke bagian akhir daftar; setara dengan [len (a):] = [x]. list.extend (L) Perluas daftar dengan menambahkan semua item dalam daftar yang diberikan; setara dengan [len (a):] = L. list.insert (i, x) Masukkan item pada posisi tertentu. Argumen pertama adalah indeks dari elemen yang sebelum dimasukkan, jadi a.insert (0, x) memasukkan di bagian depan daftar, dan a.insert (len (a), x) setara dengan a.append (x). list.remove (x) Hapus item pertama dari daftar yang nilainya x. Ini adalah kesalahan jika tidak ada item seperti itu. list.pop ([i]) Hapus item pada posisi yang diberikan dalam daftar, dan kembalikan. Jika tidak ada indeks yang ditentukan, a.pop () menghapus dan mengembalikan item terakhir dalam daftar. (Tanda kurung siku di sekitar i pada tanda tangan metode menunjukkan bahwa parameternya adalah opsional, bukankah Anda harus mengetikkan tanda kurung siku pada posisi itu. Anda akan sering melihat notasi ini di Referensi Perpustakaan Python.) list.index (x) Kembalikan indeks di daftar item pertama yang nilainya x. Ini adalah kesalahan jika tidak ada item seperti itu. list.count (x) Kembalikan berapa kali x muncul dalam daftar. list.sort (cmp = None, key = None, reverse = False) Urutkan item daftar di tempat (argumen dapat digunakan untuk kustomisasi sortir, lihat diurutkan () untuk penjelasan mereka). list.reverse () Membalik unsur daftar, di tempat. Nilai Dictionary tidak memiliki batasan. Mereka bisa menjadi objek Python yang sewenang-wenang, baik objek standar atau objek yang diten-

tukan pengguna. Namun, hal yang sama tidak berlaku untuk key. Ada dua hal penting yang harus diingat tentang Dictionary Key. Pertama lebih dari satu entri per key tidak diperbolehkan. Yang berarti tidak ada key duplikat yang diperbolehkan. Saat key duplikat ditemui selama penugasan, tugas terakhir akan menang. Kedua key harus tidak berubah. Yang berarti Anda bisa menggunakan string, angka atau tuples sebagai tombol kamus tapi sesuatu seperti [’key’] tidak diperbolehkan.

CHAPTER 12

FUNCTIONS

12.1 Python Functions

Fungsi adalah blok kode terorganisir dan dapat digunakan kembali yang digunakan untuk melakukan tindakan tunggal dan terkait. Fungsi menyediakan modularitas yang lebih baik untuk aplikasi Anda dan tingkat penggunaan kode yang tinggi. Seperti yang sudah Anda ketahui, Python memberi Anda banyak fungsi built-in seperti cetak (), dll. Tetapi Anda juga dapat membuat fungsi Anda sendiri. Fungsi ini disebut fungsi yang ditentukan pengguna.

Ada tiga jenis fungsi dengan Python:

1. Built-in Functions, seperRekursif ti help () untuk meminta bantuan, min () untuk mendapatkan nilai minimum, print () untuk mencetak objek ke terminal.

2. User-Defined Functions (UDFs), yang merupakan fungsi yang dibuat untuk membantu pengguna
3. Anonymous Functions, yang juga disebut fungsi lambda karena tidak dinyatakan dengan kata kunci def standar.

Keuntungan-keuntungan menggunakan Function, yaitu:

1. Program besar dapat di bagi menjadi program-program kecil menggunakan function.
2. Kemudahan mencari error atau kesalahan karena alur logika jelas dan kesalahan dapat ditempatkan pada suatu modul tertentu.
3. Memodifikasi dan memperbaiki program dapat digunakan pada suatu modul tertentu saja tanpa mempengaruhi keseluruhan program.
4. Dapat digunakan lagi (Reusability) oleh program atau fungsi lain.
5. Meminimalkan penulisan perintah yang sama.

function di Python biasanya mempunyai sebuah parameter dan return statement. Function di Python mempunyai pola sebagai berikut:

```
def nama function yang ingin anda buat (param1, param2, ... pa

   # kode Anda diisi
    return sesuatu
```

Tipe data yang dikembalikan bisa bermacam jenis tipe data yang didukung Python. Meskipun parameter yang akan diterima oleh function tersebut.

12.2 Kategori Fungsi

1. Standard Library Function ialah fungsi-fungsi yang telah disediakan oleh Interpreter Python dalam file-file atau librarynya. Misalnya: raw_input(), input(), print(), open(), len(), max(), min(), abs() dan lain-lain.

2. Programme-Defined Function ialah function yang dibuat oleh programmer itu sendiri. Function ini memiliki nama tertentu yang unik dalam programnya, letaknya terpisah dari program utama, dan bisa dijadikan satu ke dalam suatu library yang di buat oleh programmer itu sendiri.

Dalam python terdapat dua buah perintah yang dapat digunakan untuk membuat sebuah fungsi, yaitu def dan lambda. def adalah perintah standar di dalam python untuk mendefinisikan sebuah fungsi. Tidak seperti function dalam bahasa pemrograman compiler seperti C/C++, def dalam bahasa python merupakan perintah yang executable, artinya function yang tidak akan aktif sampai python merunning perintah def tersebut. Sedangkan lambda, dalam python lebih dikenal dengan nama Anonymous Function (Fungsi yang tidak disebutkan namanya). Lambda bukanlah sebuah perintah (statemen) namun lebih kepada ekspresi (expression).

12.2.1 Defining a Function

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

1. Blok fungsi dimulai dengan defensi kata kunci diikuti oleh nama fungsi dan tanda kurung (()).

2. Setiap parameter masukan atau argumen harus ditempatkan di dalam tanda kurung ini. Anda juga dapat menentukan parameter di dalam tanda kurung ini.

3. Pernyataan fungsi pertama dapat berupa pernyataan opsional - string dokumentasi fungsi atau docstring.

4. Blok kode dalam setiap fungsi dimulai dengan titik dua (:) dan indentasi.

5. Pernyataan kembali [ekspresi] keluar dari sebuah fungsi, secara opsional menyampaikan kembali ekspresi ke pemanggil. Pernyataan pengembalian tanpa argumen sama dengan return None.

12.2.2 Parameter Fungsi

Parameter Fungsi adalah sebuah fungsi yang dapat mempunyai daftar argumen (parameter) atau tidak.

12.2.2.1 Parameter Posisi Parameter Posisi merupakan parameter fungsi pada urutan posisi yang valid. Saat pemanggilan fungsi, parameter itu diharuskan sesuai dengan jumlah parameter yang sudah didefinisikan.

Contoh :

```
def perhitunan (k,l) :
            m=k+l
            return m
perhitungan (4,5)
Output = 9
```

12.2.2.2 Parameter Nama Parameter Nama adalah Dipanggilnya fungsi nilai pada parameternya dijelaskan langsung dari nama parameternya.

Contoh :

```
def totalTransfer(hari, uangTransfer):
            gaji= hari*uangTransfer
            return gaji
totalTransfer(hari=20,uangTransfer=20000)
Output = 400000
```

12.2.2.3 Parameter Default Parameter default adalah nilai yang diambil ke parameter pada saat pendefinisian fungsi. Contoh :

```
def koin (ratusan, ribuan=5) :
Print koin ratusannya adalah ; , ratusan;
Print koin ribuannya adalah ; , ribuan;
Uang(7)
Output : 7 dan 5 , nilai default yang digunakan.
```

12.2.2.4 Parameter Bertipe List Parameter Bertipe List adalah satu parameter yang diawali oleh tanda *. Contoh :

```
def angka(*bilangan) ;
```

```
            Print angka
angka (2,7,9)
Output : (2,7,9)
```

12.3 Syntax

Berikut adalah contoh program kalkulator menggunakan Function pada Python :

```
def menu()
    print "Hallo Guys :) Selamat Datang di Program Kalkulator"
    print "Pilih Operasi Bilangan :"
    print " "
    print "1) Penjumlahan"
    print "2) Pengurangan"
    print "3) Perkalian"
    print "4) Pembagian"
    print "5) Keluar Program"
    print " "
    return input ("Pilih Operasi Matematikanya : ")

# Function untuk Penjumlahan
def tbh(a,b):
    print a, "+", b, "=", a + b

# Function untuk Pengurangan
def kur(a,b):
    print b, "+", a, "=", b + a

# Function untuk Perkalian
def kal(a,b):
    print a, "*", b, "=", a * b

# Function untuk Pembagian
def bag(a,b):
    print a, "/", b, "=", a / b

loop = 1
choice = 0
```

```
while loop == 1:
    choice = menu()
    if choice == 1:
        tbh(input("Tambahkan Ini: "),input("Dengan ini: "))
    elif choice == 2:
        kur(input("Kurangkan Ini: "),input("Dengan ini: "))
    elif choice == 3:
        kur(input("Kalikan Ini: "),input("Dengan ini: "))
    elif choice == 4:
        kur(input("Bagikan Ini: "),input("Dengan ini: "))
    elif choice == 5:
        loop = 0

print "GoodBye :( Terimakasih Telah Menggunakan calc.py :)"
```

Berikut contoh Program Mencetak Menu dan Menghitung Luas menggunakan Function pada Python :

```
def menu():
print "Menu Options"
print
print "1. Segitiga"
print "2. Persegi Panjang"
print "3. Lingkaran"
print "4. Keluar"

def segitiga():
print "Menghitung Luas Segitiga"
a = input("Tambahkan Alas : ")
t = input("Tambahkan Tinggi : ")
luas = (a*t)/2
print "Luas Segitiga yaitu ",luas
print
print "Coba lagi [Y/N]? "
back = raw_input().upper()
if back == "Y":
menu()
else:
exit()
```

```
def persegipanjang():
print "Menghitung Luas Persegi Panjang"
p = input("Masukkan Panjang : ")
l = input("Masukkan Lebar : ")
luas = p*l
print "Luas Persegi Panjang yaitu ",luas
print
print "Coba lagi [Y/N]? "
back = raw_input().upper()
if back == "Y":
menu()
else:
exit()

def lingkaran():
print "Menghitung Luas Lingkaran"
r = input("Tambahkan Jari-Jari : ")
luas = 3.14*(r**2)
print "Luas Lingkaran yaitu ",luas
print
print "Coba lagi [Y/N]? "
back = raw_input().upper()
if back == "Y":
menu()
else:
exit()

#Program Perhitungan Luas
print "Selamat Bahagia di Program Menghitung Luas"
print "-----------------------------------------------"
print
menu()
while 1:
#input
pilih = input("Masukkan options : ")
if pilih == 1:
Segitiga()
elif pilih == 2:
```

```
Persegi Panjang()
elif pilih == 3:
Lingkaran()
elif pilih == 4:
print "\n"*100
break
else:
print "Sorry Option yang di masukkan tidak ada"
print "Coba lagi [Y/N] ? "
coba = raw_input().upper()
if coba == "Y":
menu()
else:
print "\n"*100
break

def functionname( parameters ):
"function \$  \_  \$docstring"
function \$  \_  \$suite
return [expression]
```

Secara default, parameter memiliki perilaku posisi dan Anda perlu memberi tahu mereka dengan urutan yang sama seperti yang ditetapkan. Example: Fungsi berikut mengambil string sebagai parameter masukan dan mencetaknya di layar standar.

```
def printme( str ):
"This prints a passed string into this function"
print str
return \par
```

12.4 Calling a Function

Mendefinisikan sebuah fungsi hanya memberinya sebuah nama, menentukan parameter yang akan disertakan dalam fungsi dan menyusun blok kode. Setelah struktur dasar fungsi selesai, Anda dapat menjalankannya dengan memanggilnya dari fungsi lain atau langsung dari prompt

Python. Berikut adalah contoh untuk memanggil fungsi printme ().

```
\$  \#  \$!/usr/bin/python

\$  \#  \$ Function definition is here
def printme( str ):
"This prints a passed string into this function"
print str
return;
\$  \#  \$ Now you can call printme function

printme("I'm first call to user defined function!")
printme("Again second call to the same function")
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut

I'm first call to user defined function! Again second call to the same function

12.5 Pass by reference vs value

Semua parameter (argumen) dalam bahasa Python dilewatkan dengan referensi. Ini berarti jika Anda mengubah parameter yang mengacu pada suatu fungsi, perubahan tersebut juga mencerminkan kembali fungsi pemanggilan. Sebagai contoh:

```
\$  \#  \$!/usr/bin/python

\$  \#  \$ Function definition is here
def changeme( mylist ):
"This changes a passed list into this function"
mylist.append([1,2,3,4]);
"Values inside the function: ", mylist

return


\$  \#  \$ Now you can call changeme function
```

```
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist
```

Di sini, kita mempertahankan referensi objek yang dilewati dan menambahkan nilai pada objek yang sama. Jadi, ini akan menghasilkan hasil sebagai berikut:

```
Values˜inside the function:  [10, 20, 30, [1, 2, 3, 4]]
Values˜outside the function:  [10, 20, 30, [1, 2, 3, 4]]
```

Ada satu contoh lagi di mana argumen dilewatkan melalui referensi dan rujukannya ditimpa di dalam fungsi yang disebut.

```
\$  \# \$!/usr/bin/python
\$  \#  \$ Function definition is here
def changeme( mylist ):
"This changes a passed list into this function"
˜˜ mylist = [1,2,3,4];  \$  \#  \$ This would assig new refere
˜˜ print "Values inside the function: ", mylist

˜˜ return

\$  \#  \$ Now you can call changeme function

mylist = [10,20,30];

changeme( mylist );

print "Values outside the function: ", mylist
```

Parameter mylist adalah local ke fungsi changeme. Mengubah mylist dalam fungsi tidak mempengaruhi mylist. Fungsi ini tidak menghasilkan apa-apa dan akhirnya ini akan menghasilkan hasil sebagai berikut:

```
Values˜inside the function:  [1, 2, 3, 4]

Values˜outside the function:  [10, 20, 30]
```

Function Arguments

Anda dapat memanggil fungsi dengan menggunakan jenis argumen formal berikut:

```
\$ \bullet \$ Argumen yang dibutuhkan
\$ \bullet \$ Argumen kata kunci

\$ \bullet \$ Argumen baku

\$ \bullet \$ Argumen panjang variable
```

12.6 Required arguments

Argumen yang diperlukan adalah argumen yang diberikan ke sebuah fungsi dalam urutan posisi yang benar. Di sini, jumlah argumen dalam pemanggilan fungsi harus sesuai persis dengan definisi fungsi. Untuk memanggil fungsi printme (), Anda pasti perlu melewati satu argumen, jika tidak maka akan memberikan kesalahan sintaks sebagai berikut :

```
\$  \#  \$!/usr/bin/python
\$  \#  \$ Function definition is here
def printme( str ):

~~  "This prints a passed string into this function"

~~  print str

~~  return;

\$  \#  \$ Now you can call printme function

printme()
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut:

```
Traceback (most recent call last):

 ~ File "test.py", line 11, in <module>
~~~ printme();
```

```
 TypeError: printme() takes exactly 1 argument (0 given)
```

12.7 Keyword arguments

Argumen kata kunci terkait dengan pemanggilan fungsi. Bila Anda menggunakan argumen kata kunci dalam pemanggilan fungsi, penelepon mengidentifikasi argumen berdasarkan nama parameter. Hal ini memungkinkan Anda melewatkan argumen atau menempatkannya agar tidak bermasalah karena penerjemah Python dapat menggunakan kata kunci yang diberikan agar sesuai dengan nilai parameter. Anda juga dapat membuat panggilan kata kunci ke fungsi printme () dengan cara berikut -

```
\$  \#  \$!/usr/bin/python

\$  \#  \$ Function definition is here

 def printme( str ):

~~ "This prints a passed string into this function"

~~ print str

~~ return;

 \$  \#  \$ Now you can call printme function

printme( str = "My string")
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut:

My string

Contoh berikut memberikan gambaran yang lebih jelas. Perhatikan bahwa urutan parameter tidak masalah.

```
\$  \#  \$!/usr/bin/python

\$  \#  \$ Function definition is here
```

```
def printinfo( name, age ):

~~ "This prints a passed info into this function"

~~ print "Name: ", name

~~ print "Age ", age

~~ return;


\$  \#  \$ Now you can call printinfo function
printinfo( age=50, name="miki" )
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut: Name:ñiki
Age5̃0

12.8 Default arguments

Argumen default adalah argumen yang mengasumsikan nilai default jika nilai tidak diberikan dalam pemanggilan fungsi untuk argumen itu. Contoh berikut memberi ide pada argumen default, ini mencetak usia default jika tidak lulus.

```
\$  \#  \$!/usr/bin/python

\$  \#  \$ Function definition is here
def printinfo( name, age = 35 ):
~~ "This prints a passed info into this function"
~~ print "Name: ", name
~~ print "Age ", age

~~ return;

\$  \#  \$ Now you can call printinfo function

printinfo( age=50, name="miki" )
```

```
printinfo( name="miki" )
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut
Name:m̃iki
Age5̃0
Name:m̃iki
Age3̃5

12.9 Variable-length arguments

Anda mungkin perlu memproses sebuah fungsi untuk argumen lebih banyak daripada yang Anda tentukan saat menentukan fungsinya. Argumen ini disebut variable-lengtharguments dan tidak disebutkan dalam definisi fungsi, tidak seperti argumen yang dibutuhkan dan standar.

Sintaks untuk fungsi dengan argumen variabel non-kata kunci adalah ini :

```
def functionname([formal \$  \_  \$args,] *var \$  \_  \$args

~~ "function \$  \_  \$docstring"

~~ function \$  \_  \$suite

~~ return [expression]
```

Tanda asterisk (*) ditempatkan sebelum nama variabel yang menyimpan nilai dari semua argumen variabel nonkeyword. Tuple ini tetap kosong jika tidak ada argumen tambahan yang ditentukan selama pemanggilan fungsi. Berikut adalah contoh sederhana :

```
\$  \#  \$!/usr/bin/python

\$  \#  \$ Function definition is here

def printinfo( arg1, *vartuple ):
```

```
~~ "This prints a variable passed arguments"

~~ print "Output is: "
~~ print arg1
~~ for var in vartuple:
~~~~~ print var
~~ return;

\$  \#  \$ Now you can call printinfo function
printinfo( 10 )
printinfo( 70, 60, 50 )
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut - Output is: 0 Output is: 70 60 50

12.10 The Anonymous Functions

Fungsi ini disebut anonim karena tidak dinyatakan secara standar dengan menggunakan kata kunci def. Anda bisa menggunakan kata kunci lambda untuk membuat fungsi anonim yang kecil.

1. Bentuk lambda bisa mengambil sejumlah argumen tapi hanya mengembalikan satu nilai dalam bentuk ekspresi. Mereka tidak dapat berisi perintah atau beberapa ekspresi.

2. Fungsi anonim tidak bisa menjadi panggilan langsung untuk dicetak karena lambda membutuhkan ekspresi

3. Fungsi Lambda memiliki namespace lokal mereka sendiri dan tidak dapat mengakses variabel selain yang ada dalam daftar parameter dan yang ada di namespace global.

4. Meskipun tampak bahwa lambda adalah versi satu baris dari sebuah fungsi, mereka tidak setara dengan pernyataan inline di C atau C ++, yang tujuannya adalah dengan melewatkan alokasi stack fungsi selama pemanggilan untuk alasan kinerja.

12.11 Syntax

Sintaks fungsi lambda hanya berisi satu pernyataan, yaitu sebagai berikut : lambda [arg1 [,arg2,.....argn]]:expression Berikut adalah contoh untuk menunjukkan bagaimana lambda bentuk fungsi bekerja :

```
   \$  \#  \$!/usr/bin/python
   \$  \#  \$ Function definition is here
  sum = lambda arg1, arg2: arg1 + arg2;


   \$  \#  \$ Now you can call sum as a function
 print "Value of total : ", sum( 10, 20 )
  print "Value of total : ", sum( 20, 20 )
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut - Value of total : 30 Value of total : 40

12.12 The return Statement

Pernyataan kembali [ekspresi] keluar dari sebuah fungsi, secara opsional menyampaikan kembali ekspresi ke pemanggil. Pernyataan pengembalian tanpa argumen sama dengan return None. Semua contoh di atas tidak mengembalikan nilai apapun. Anda bisa mengembalikan nilai dari sebuah fungsi sebagai berikut :

```
  \$  \#  \$!/usr/bin/python
  \$  \#  \$ Function definition is here
 def sum( arg1, arg2 ):
  ~~  \$  \#  \$ Add both the parameters and return them."
  ~~ total = arg1 + arg2
  ~~ print "Inside the function : ", total
  ~~ return total;

   \$  \#  \$ Now you can call sum function
  total = sum( 10, 20 );
 print "Outside the function : ", total
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut - Insidẽthe function : 30 Outsidẽthe function : 30

12.13 Scope of Variables

Semua variabel dalam sebuah program mungkin tidak dapat diakses di semua lokasi dalam program tersebut. Ini tergantung di mana Anda telah menyatakan sebuah variabel. Contoh penggunaan scope variabel :

```
 def bljrScope(A):
 A = 10
 print "Nilai A di dalam fungsi, a = ", A
 # program utama
 A = 30
 print "Nilai a di luar fungsi, a = ", A
 bljrScope(A)
\end{verbatim
 Output :
 Nilai a di luar fungsi, a = 30
 Nilai A di dalam fungsi, a = 10
Pada contoh diatas, variabel A didefinisikan di dua tempat yai
Ruang lingkup variabel menentukan bagian dari program di mana
   \$ \bullet \$ Variabel global
   \$ \bullet \$ Variabel local

\section{Global vs. Local variables}
Variabel yang didefinisikan di dalam badan fungsi memiliki lin
Ini berarti bahwa variabel lokal dapat diakses hanya di dalam
\begin{verbatim}
 \$ \# \$!/usr/bin/python

  total = 0;  \$  \#  \$ This is global variable.
  \$  \#  \$ Function definition is here
  def sum( arg1, arg2 ):
  ~~  \$  \#  \$ Add both the parameters and return them."
```

```
  ~~ total = arg1 + arg2;  \$  \#  \$ Here total is local vari
  ~~ print "Inside the function local total : ", total
  ~~ return total;

   \$  \#  \$ Now you can call sum function
  sum( 10, 20 );
 \ print "Outside the function global total : ", total
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut : Inside the function local total : 30 Outside the function global total : 0

12.14 Pembelajaran Function Python

Function dalam Python didefinisikan menggunakan kata kunci def. Setelah def ada nama pengenal function diikut dengan parameter yang diapit oleh tanda kurung dan diakhir dingan tanda titik dua :. Baris berikutnya berupa blok fungsi yang akan dijalankan jika fungsi dipanggil.

CHAPTER 13

FILES I/O

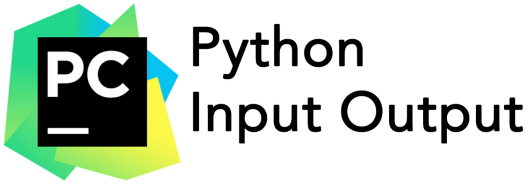


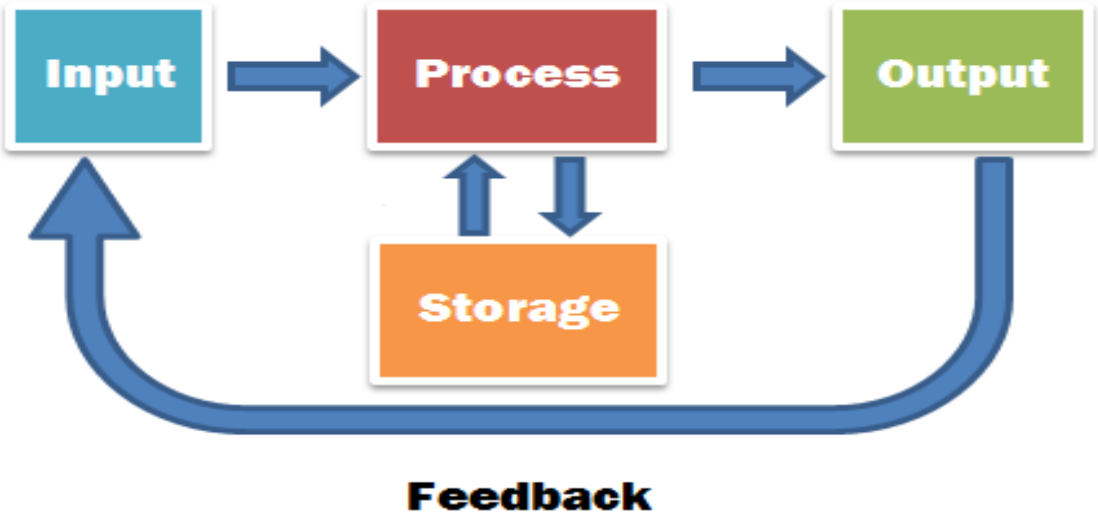
Figure 13.1 File I/O

13.1 Pengertian

13.1.1 Python File I/O

Unit input adalah (masukan) suatu data yang berbentuk documet bait itu data data foto, data data hurup ataupun data tanggal ke dalam sebuah system yang terkomputerisasi. unit output (keluaran) biasanya digunakan untuk menampilkan data, atau dengan kata lain untuk menangkap data yang telah diinputkan terlebih dahulu dalam sebuah penyimpanan media elektronic , contohnya data yang akan ditampilkan pada layar monitor atau printer.

I/O Input/Ouput Read [IOR] dan untuk tulis I/O Input/Output Write [IOW]. File adalah lokasi bernama pada disk untuk menyimpan informasi terkait. Ini digunakan untuk menyimpan data secara permanen dalam memori nonvolatile (misalnya hard disk). Karena, random access memory (RAM) bersifat volatile sehingga kehilangan datanya saat komputer dimatikan, kita menggunakan file untuk penggunaan data masa depan. Bila kita ingin membaca dari atau menulis ke file kita perlu membukanya terlebih dahulu. Bila sudah selesai, perlu ditutup, agar sumber yang diikat dengan file tersebut dibebaskan.Oleh karena itu, dengan Python, sebuah operasi file berlangsung dengan urutan sebagai berikut.



$$PythonFileI/O \tag{13.1}$$

1. Buka file
2. Membaca atau menulis (melakukan operasi)
3. Tutup file tersebu

Membuka sebuah file
Python memiliki built-in function open () untuk membuka file. Fungsi ini mengembalikan objek file, juga disebut handle, karena digunakan untuk membaca atau memodifikasi file yang sesuai.

```
>>>~f~=~open("test.txt")
 $  \#  $ open file in current directory \par
\noindent
>>>~f = open("C:/Python33/README.txt")
 $  \#  $ specifying full path \par
\vspace{12pt}
```

Kita bisa menentukan mode saat membuka file. Dalam mode, kami menentukan apakah kita ingin membaca 'r', menulis 'w' atau menambahkan 'a' ke file. Kita juga menentukan apakah kita ingin membuka file dalam mode teks atau mode biner. Defaultnya adalah membaca dalam mode teks. Dalam mode ini, kita mendapatkan string saat membaca dari file. Di sisi lain, mode biner mengembalikan byte dan ini adalah mode yang akan digunakan saat berhadapan dengan file non-teks seperti file gambar atau exe.

1. 'r'
   Buka file untuk dibaca. (default)

2. 'w'
   Buka file untuk menulis. Membuat file baru jika tidak ada atau memotong file jika file tersebut ada.

3. 'x'
   Buka file untuk pembuatan eksklusif. Jika file sudah ada, operasi gagal

4. 'a'
   Buka untuk menambahkan di akhir file tanpa memotongnya. Membuat file baru jika tidak ada.

5. 't'
   Buka dalam mode teks. (default)

6. 'b'
   Buka dalam mode biner.

7. '+'
   Buka file untuk mengupdate (membaca dan menulis)

```
f~=~open("test.txt")~~~
 $  \#  $ equivalent to 'r' or 'rt' \par
\noindent
f~= open("test.txt",'w')
 $  \#  $ write in text mode \par
\noindent
f = open("img.bmp",'r+b')
 $  \#  $ read and write in binary mode \par
\vspace{12pt}
\noindent
```

Tidak seperti bahasa lain, karakter 'a' tidak menyiratkan angka 97 sampai dikodekan menggunakan ASCII (atau pengkodean setara lainnya). Apalagi, pengkodean default bergantung pada platform. Di jendela, itu adalah 'cp1252' tapi 'utf-8' di Linux. Jadi, kita juga tidak harus bergantung pada pengkodean default atau kode kita akan berperilaku berbeda di berbagai platform. Oleh karena itu, ketika bekerja dengan file dalam mode teks, sangat disarankan untuk menentukan jenis pengkodean.

f = open("test.txt",mode = 'r',encoding = 'utf-8')

13.2 Menutup sebuah File

Ketika kita selesai dengan operasi ke file, kita perlu menutupinya dengan benar. Menutup file akan membebaskan sumber daya yang terkait dengan file dan dilakukan dengan menggunakan metode close (). Python memiliki pengumpul sampah untuk membersihkan benda yang tidak difermentasi tapi, kita tidak boleh bergantung padanya untuk menutup file.

f = open("test.txt",encoding = 'utf-8')
# perform file operations
f.close()

Metode ini tidak sepenuhnya aman. Jika pengecualian terjadi saat kita melakukan operasi dengan file, kode keluar tanpa menutup file. Cara yang lebih aman adalah dengan menggunakan try ... akhirnya blok.

```
try:
  f = open("test.txt",encoding = 'utf-8')
  # perform file operations
finally:
  f.close()
```

Dengan cara ini, kita dijamin bahwa file tersebut benar tertutup bahkan jika pengecualian dinaikkan, menyebabkan aliran program berhenti. Cara terbaik untuk melakukannya adalah dengan menggunakan pernyataan. Ini memastikan file ditutup saat blok di dalam dengan keluar. Kita tidak perlu secara eksplisit memanggil metode close (). Hal itu dilakukan secara internal.

```
with open("test.txt",encoding = 'utf-8') as f:
  # perform file operations
```

13.3 Menulis ke File

Untuk menulis ke file kita perlu membukanya dalam mode write 'w', tambahkan 'a' atau exclusive creation 'x'. Kita harus berhati-hati dengan mode 'w' karena akan menimpa file jika sudah ada. Semua data sebelumnya terhapus. Menulis string atau urutan byte (untuk file biner) dilakukan dengan menggunakan metode write (). Metode ini mengembalikan jumlah karakter yang ditulis ke file.

```
with open("test.txt",'w',encoding = 'utf-8') as f:
  f.write("my first file \n")
  f.write("This file \n \n")
  f.write("contains three lines \n")
```

Program ini akan membuat file baru bernama 'test.txt' jika tidak ada. Jika memang ada, itu akan ditimpa. Kita harus menyertakan karakter newline sendiri untuk membedakan garis yang berbeda.

Membaca Dari File

Untuk membaca isi sebuah file, kita harus membuka file dalam mode baca. Ada berbagai metode yang tersedia untuk tujuan ini. Kita bisa menggunakan metode read (size) untuk membaca dalam jumlah ukuran data. Jika parameter ukuran tidak ditentukan, bunyinya dan kembali ke akhir file.

```
¿¿¿ f = open("test.txt",'r',encoding = 'utf-8')
¿¿¿ f.read(4)   # read the first 4 data
'This'

¿¿¿ f.read(4)   # read the next 4 data
' is '

¿¿¿ f.read()    # read in the rest till end of file
'my first file \nThis file \ncontains three lines \n'

¿¿¿ f.read() # further reading returns empty sting
```

Kita dapat melihat, metode read () mengembalikan baris baru sebagai ' \ n'. Begitu akhir file tercapai, kita mendapatkan string kosong untuk dibaca lebih lanjut. Kita bisa mengubah kursor file kita saat ini (posisi) dengan menggunakan metode seek (). Demikian pula metode tell () mengembalikan posisi kita saat ini (dalam jumlah byte).

```
¿¿¿ f.tell()   # get the current file position
56

¿¿¿ f.seek(0)  # bring file cursor to initial position
0
```

¿¿¿ print(f.read()) # read the entire file
This is my first file
This file
contains three lines

Kita bisa membaca file line-by-line menggunakan for loop. Ini efisien dan cepat.

¿¿¿ for line in f:
... print(line, end = ”)
...
This is my first file
This file
contains three lines

Baris dalam file itu sendiri memiliki karakter baris baru ’ \ n’. Terlebih lagi, print () parameter akhir untuk menghindari dua baris baru saat mencetak. Sebagai alternatif, kita dapat menggunakan metode readline () untuk membaca setiap baris file. Metode ini membaca sebuah file sampai newline, termasuk newline character.

¿¿¿ f.readline()
’This is my first file \n’

¿¿¿ f.readline()
’This file \n’

¿¿¿ f.readline()
’contains three lines \n’

¿¿¿ f.readline()
”

Terakhir, metode readlines () mengembalikan daftar baris yang tersisa dari keseluruhan file. Semua metode membaca ini mengembalikan nilai kosong saat akhir file (EOF) tercapai.

¿¿¿ f.readlines()

['This is my first file \n', 'This file \n', 'contains three lines \n']

Metode Berkas Python

Ada berbagai metode yang tersedia dengan objek file. Beberapa di antaranya telah digunakan pada contoh di atas. Berikut adalah daftar lengkap metode dalam mode teks dengan deskripsi singkat. Atribut Objek file Setelah file dibuka dan Anda memiliki satu file objek, Anda bisa mendapatkan berbagai informasi yang berkaitan dengan file tersebut. Berikut adalah daftar semua atribut yang terkait dengan objek file:

```
#!/usr/bin/python

# Open a file
fo = open("foo.txt", "wb")
print "Name of the file: ", fo.name
print "Closed or not : ", fo.closed
print "Opening mode : ", fo.mode
print "Softspace flag : ", fo.softspace
```

Membaca dan Menulis File

Objek file menyediakan seperangkat metode akses untuk membuat hidup kita lebih mudah. Kita akan melihat bagaimana menggunakan metode read () dan write () untuk membaca dan menulis file. Metode tulis () Metode write () menulis string apapun ke file yang terbuka. Penting untuk dicatat bahwa string Python dapat memiliki data biner dan bukan hanya teks. Metode write () tidak menambahkan karakter baris baru (' \ n') ke akhir string -
Sintaksis

```
#!/usr/bin/python
```

```
# Open a file
fo = open(”foo.txt”, ”wb”)
fo.write( ”Python is a great language. \nYeah its great!!
\n”);

# Close opend file
fo.close()
#!/usr/bin/python

# Open a file
fo = open(”foo.txt”, ”r+”)
str = fo.read(10);
print ”Read String is : ”, str

# Close opend file
fo.close()
```

```
Read String is : Python is
```

```
#!/usr/bin/python

# Open a file
fo = open(”foo.txt”, ”r+”)
str = fo.read(10);
print ”Read String is : ”, str

# Check current position
position = fo.tell();
print ”Current file position : ”, position

# Reposition pointer at the beginning once again
position = fo.seek(0, 0);
str = fo.read(10);
print ”Again read String is : ”, str
# Close opend file
fo.close()
```

```
Read String is : Python is
Current file position : 10
Again read String is : Python is
```

```
os.rename(current_file_name, new_file_name)

#!/usr/bin/python
import os

# Rename a file from test1.txt to test2.txt
os.rename( "test1.txt", "test2.txt" )

#!/usr/bin/python
import os

# Delete file test2.txt
os.remove("text2.txt")

os.mkdir("newdir")

#!/usr/bin/python
import os

# Create a directory "test"
os.mkdir("test")

#!/usr/bin/python
import os

# Changing a directory to "/home/newdir"
os.chdir("/home/newdir")

#!/usr/bin/python
import os

# This would give location of the current directory
os.getcwd()

#!/usr/bin/python
import os

# This would remove "/tmp/test" directory.
os.rmdir( "/tmp/test" )
```

Membuka sebuah file

Untuk membuka file teks yang Anda gunakan, well, open () function. Sepertinya masuk akal. Anda melewatkan parameter tertentu untuk membuka () untuk memberitahukannya di mana file harus dibuka - 'r' untuk dibaca saja, 'w' untuk tulisan saja (jika ada file lama, akan dituliskan), 'a 'Untuk menambahkan (menambahkan sesuatu ke akhir file) dan' r + 'untuk membaca dan menulis. Tapi kurang bicara, mari kita buka file untuk membaca (Anda bisa melakukan ini dengan mode idle python Anda). Buka file teks biasa. Kami kemudian akan mencetak apa yang kita baca di dalam file:
Contoh Kode 1 - Membuka sebuah file

```
Openfile = open ('pathtofile', 'r')
Openfile.read ()
```

Itu menarik. Anda akan melihat banyak simbol ' \ n'. Ini merupakan baris baru (di mana Anda menekan enter untuk memulai baris baru). Teksnya benar-benar tidak diformat, tapi jika Anda melewati keluaran openfile.read () untuk mencetak (dengan mengetikkan print openfile.read ()) akan diformat dengan baik. Carilah dan Anda Temukan Apakah Anda mencoba mengetik di print openfile.read ()? Apakah itu gagal? Kemungkinan besar, dan alasannya adalah karena 'kursor' telah mengubah tempatnya. Kursor Kursor apa Nah, kursor yang sebenarnya tidak bisa kamu lihat, tapi tetap kursor. Kursor tak terlihat ini memberitahukan fungsi baca (dan banyak fungsi I / O lainnya) dari mana mulai. Untuk mengatur di mana kursor berada, Anda menggunakan fungsi seek (). Ini digunakan dalam bentuk seek (offset, dari mana). Mana yang opsional, dan menentukan mana yang harus dicari. Jika darimana 0, byte / huruf dihitung dari awal. Jika 1, byte dihitung dari posisi kursor saat ini. Jika 2, maka byte dihitung dari akhir file. Jika tidak ada yang diletakkan di sana, 0 diasumsikan.

Offset menggambarkan seberapa jauh dari mana kursor bergerak. sebagai contoh:

openfile.seek (45,0) akan memindahkan kursor ke 45 byte / huruf setelah permulaan file.
Openfile.seek (10,1) akan memindahkan kursor ke 10 byte / huruf setelah posisi kursor saat ini.
openfile.seek (-77,2) akan memindahkan kursor ke 77 byte / huruf sebelum akhir file (perhatikan - sebelum angka 77)

Cobalah sekarang juga. Gunakan openfile.seek () untuk pergi ke tempat manapun di file dan kemudian mencoba mengetik print openfile.read (). Ini akan dicetak dari tempat yang Anda inginkan. Tapi sadari bahwa openfile.read () memindahkan kursor ke akhir file - Anda harus mencari lagi.

Fungsi I / O Lainnya

Ada banyak fungsi lain yang membantu Anda dalam berurusan dengan file. Mereka memiliki banyak kegunaan yang memberdayakan Anda untuk berbuat lebih banyak, dan membuat hal-hal yang dapat Anda lakukan lebih mudah. Mari kita lihat kirim (), readline (), readlines (), tulis () dan close (). Kirim () mengembalikan tempat kursor berada dalam file. Tidak memiliki parameter, ketik saja (seperti contoh di bawah ini yang akan ditampilkan). Ini sangat berguna, untuk mengetahui apa yang Anda maksud, di mana letaknya, dan kontrol kursor yang sederhana. Untuk menggunakannya, ketik fileobjectname.tell () - dimana fileobjectname adalah nama dari file objek yang Anda buat saat Anda membuka file (di openfile = open ('pathtofile', 'r') nama objek file adalah openfile). Readline () membaca dari mana kursor sampai akhir baris. Ingat bahwa akhir baris bukan tepi layar Anda - garis berakhir saat Anda menekan enter untuk membuat baris baru. Ini berguna untuk hal-hal seperti membaca log peristiwa, atau mengalami sesuatu yang progresif untuk mengolahnya. Tidak ada parameter yang harus Anda lewati ke readline (), meskipun secara opsional Anda

dapat memberi tahu jumlah maksimal byte / huruf untuk dibaca dengan meletakkan nomor di tanda kurung. Gunakan dengan fileobjectname.readline ().

Readlines () sama seperti readline (), namun readlines () membaca semua baris dari kursor dan seterusnya, dan mengembalikan sebuah daftar, dengan setiap elemen daftar memegang satu baris kode. Gunakan dengan fileobjectname.readlines (). Misalnya, jika Anda memiliki file teks:
Contoh Kode 2 - contoh file teks

Baris 1

Baris 3
Baris 4

Baris 6

Fungsi write (), menulis ke file. Bagaimana kamu menebak nya??? Ini menulis dari mana kursor berada, dan menimpa teks di depannya - seperti di MS Word, di mana Anda menekan 'insert' dan menulis di atas teks lama. Untuk memanfaatkan fungsi yang paling penting ini, letakkan string di antara tanda kurung untuk ditulis mis. fileobjectname.write ('ini adalah string'). Dekat, Anda bisa mencari, menutup file sehingga Anda tidak dapat lagi membaca atau menulis sampai Anda membuka kembali lagi. Cukup sederhana Untuk menggunakan, Anda akan menulis fileobjectname.close (). Sederhana! Dengan mode siaga Python, buka file uji (atau buat yang baru ...) dan mainkan fungsi ini. Anda bisa melakukan penyuntingan teks yang sederhana (dan sangat merepotkan).

Mmm, acar Pickles, dengan Python, harus dimakan. Rasa mereka hanya untuk membiarkan pemrogram meninggalkan mereka di lemari es.Ok, hanya bercanda disana. Pickles, dengan Python, adalah objek yang disimpan ke sebuah file. Objek dalam kasus ini bisa berupa variabel, instance dari kelas, atau daftar, kamus, atau tupel. Hal lain juga bisa acar, tapi dengan batas. Objek kemudian dapat dipulihkan, atau tidak dicemari, nanti. Dengan kata lain, Anda 'menyimpan'

benda Anda. Jadi bagaimana kita acar? Dengan fungsi dump (), yang ada di dalam modul acar - jadi pada awal program Anda, Anda harus menulis acar impor. Cukup sederhana Kemudian buka file kosong, dan gunakan pickle.dump () untuk menjatuhkan objek ke file itu. Mari kita coba itu:
Contoh Kode 3 - pickletest.py

```
# # # pickletest.py
# # # PICKLE AN OBJECT

# mengimpor modul acar
Acar impor

# Mari membuat sesuatu untuk menjadi acar
# Bagaimana dengan daftar?
picklelist = ['one', 2, 'three', 'four', 5, 'dapatkah kamu menghitung?']

# Sekarang buat file
# ganti nama file dengan file yang ingin Anda buat
File = open ('filename', 'w')

# Sekarang mari kita pilih picklelist
pickle.dump (picklelist, file)

# Tutup file, dan pengawetan Anda sudah selesai
file.close ()
```

Kode untuk melakukan ini diletakkan seperti pickle.load (object _to _pickle, file _object) di mana:

object _to _pickle adalah objek yang ingin Anda acar (yaitu simpan ke file)
file _object adalah objek file yang ingin Anda tulis (dalam kasus ini, objek file adalah 'file')

Setelah Anda menutup file tersebut, buka di notepad dan lihat apa yang Anda lihat. Seiring dengan beberapa kue gibblygook lainnya, Anda akan melihat potongan daftar yang kami buat.

Sekarang untuk membuka kembali, atau unpickle, file Anda. Untuk menggunakan ini, kita akan menggunakan pickle.load ():

Contoh kode 4 - unpickletest.py

```
# # # unpickletest.py
# # # unpickle file

# mengimpor modul acar
Acar impor

# sekarang buka file untuk dibaca
# ganti nama file dengan path ke file yang Anda buat di pickletest.py
unpicklefile = open ('filename', 'r')

# sekarang muat daftar yang kita acar ke objek baru
unpickledlist = pickle.load (unpicklefile)

# Tutup file, hanya untuk keamanan
unpicklefile.close ()

# Mencoba menggunakan daftar
untuk item dalam unpickledlist:
   Item cetak
```

CHAPTER 14

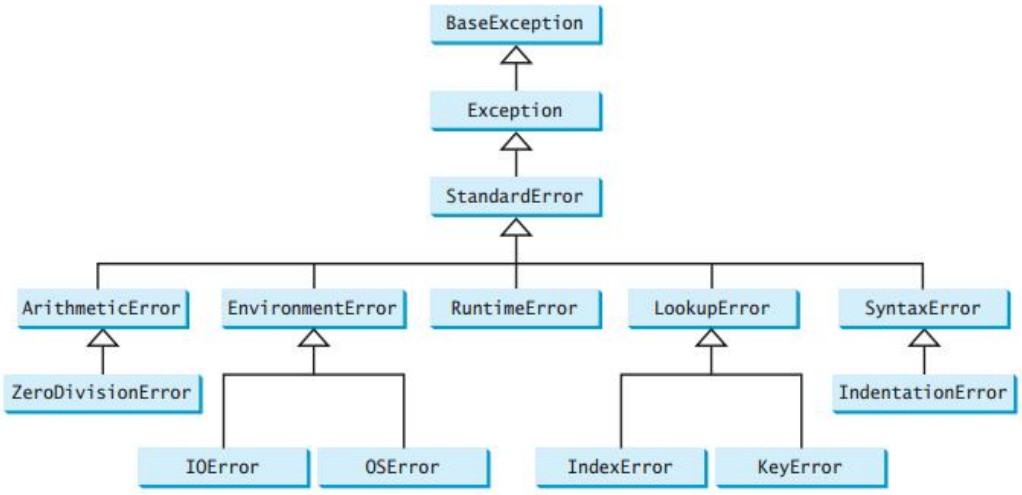
EXCEPTIONS

Figure 14.1 Python Exceptions Handling

14.1 Pengertian

14.1.1 Python Exceptions Handling

Python menyediakan dua fitur yang sangat penting untuk menangani kesalahan tak terduga dalam program Python Anda dan menambahkan kemampuan debugging di dalamnya Exception Handling: Ini akan dibahas dalam tutorial ini. Berikut adalah daftar standar Pengecualian yang tersedia dengan Python: Pengecualian Standar. Penegasan: Ini akan dibahas dalam Asertions dengan tutorial Python. Daftar Pengecualian Standar Penegasan dengan Python Penegasan adalah pemeriksaan kewarasan yang dapat Anda aktifkan atau matikan saat Anda selesai dengan pengujian program Anda. Cara termudah untuk memikirkan sebuah pernyataan adalah menyamakannya dengan pernyataan kenaikan gaji-jika (atau lebih akurat, pernyataan kenaikan-jika-tidak). Sebuah ekspresi diuji, dan jika hasilnya muncul salah, pengecualian akan meningkat. Penegasan dilakukan dengan pernyataan tegas, kata kunci terbaru untuk Python, diperkenalkan di versi 1.5. Pemrogram sering menempatkan asersi pada awal fungsi untuk memeriksa masukan yang valid, dan setelah pemanggilan fungsi untuk memeriksa keluaran yang valid. Pernyataan tegas Ketika menemukan pernyataan tegas, Python mengevaluasi ekspresi yang menyertainya, yang semoga benar. Jika ungkapannya salah, Python menimbulkan pengecualian AssertionError. Sintaks untuk menegaskan adalah -menegaskan Ekspresi [, Argumen].



PythonExceptionsHandling (14.1)

14.2 Kelas dasar untuk semua pengecualian

14.2.1 EXCEPTION NAME

1. StopIteration

   Dibesarkan ketika metode (iterator) berikutnya dari iterator tidak mengarah ke objek apa pun.

2. SystemExit Dibesarkan oleh fungsi sys.exit ()

3. StandardError

   Kelas dasar untuk semua pengecualian built-in kecuali StopIteration dan SystemExit

4. ArithmeticError

   Kelas dasar untuk semua kesalahan yang terjadi untuk perhitungan numerik.

5. OverflowError
   Dibesarkan saat perhitungan melebihi batas maksimum untuk tipe numerik.

6. FloatingPointError
   Dibesarkan saat perhitungan floating point gagal.

7. ZeroDivisionError
   Dibesarkan saat pembagian atau modulo nol dilakukan untuk semua tipe numerik.

8. AssertionError
   Dibesarkan jika terjadi kegagalan pernyataan Assert.

9. AttributeError
   Dibesarkan jika terjadi kegagalan referensi atribut atau penugasan.

10. EOFError
    Dibesarkan bila tidak ada input dari fungsi raw_input () atau input () dan akhir file tercapai.

11. ImportError
    Dibesarkan saat sebuah pernyataan impor gagal.

12. KeyboardInterrupt
    Dibesarkan saat pengguna menyela eksekusi program, biasanya dengan menekan Ctrl + c.

13. LookupError
    Kelas dasar untuk semua kesalahan pencarian.

14. KeyError

    Dibesarkan saat sebuah indeks tidak ditemukan secara berurutan. Dibesarkan saat kunci yang ditentukan tidak ditemukan dalam kamus.

15. NameError

    Dibesarkan saat pengenal tidak ditemukan di namespace lokal atau global.

16. EnvironmentError

    Dibesarkan saat mencoba mengakses variabel lokal dalam suatu fungsi atau metode namun tidak nilai yang ditugaskan padanya. Kelas dasar untuk semua pengecualian yang terjadi di luar lingkungan Python.

17. IOError

    IOError Dibesarkan saat operasi input / output gagal, seperti pernyataan cetak atau fungsi open () saat mencoba membuka file yang tidak ada. Dibangkitkan untuk kesalahan terkait sistem operasi.

14.3 SyntaxError

1. IndentationError

   Dibesarkan saat ada kesalahan dengan sintaks Python. Dibesarkan saat indentasi tidak ditentukan dengan benar.

2. SystemError

   Dibesarkan saat penafsir menemukan masalah internal, namun bila kesalahan ini ditemui juru bahasa Python tidak keluar.

3. SystemExit

Dibesarkan saat juru bahasa Python berhenti dengan menggunakan fungsi sys.exit (). Jika tidak ditangani dalam kode, menyebabkan penafsir untuk keluar.

4. TypeError

   Dibesarkan saat operasi atau fungsi dicoba yang tidak valid untuk tipe data yang ditentukan.

5. ValueError

   Dibesarkan saat fungsi bawaan untuk tipe data memiliki jenis argumen yang valid, namun argumen tersebut memiliki nilai yang tidak valid yang ditentukan.

6. RuntimeError

   Dibesarkan saat kesalahan yang dihasilkan tidak termasuk dalam kategori apa pun.

Jika asersi gagal, Python menggunakan ArgumentExpression sebagai argumen untuk AssertionError. Penegasan Pengecualian pengecualian dapat ditangkap dan ditangani seperti pengecualian lainnya dengan menggunakan perintah try-except, namun jika tidak ditangani, mereka akan menghentikan program dan menghasilkan traceback.

Contoh Berikut adalah fungsi yang mengubah suhu dari derajat Kelvin sampai derajat Fahrenheit. Karena nol derajat Kelvin sedingin yang didapatnya, fungsi itu mundur jika melihat suhu negatif

```
 $  \#  $!/usr/bin/python

def KelvinToFahrenheit(Temperature):

~~ assert (Temperature >= 0),"Colder than absolute zero!"

~~ return ((Temperature-273)*1.8)+32
```

```
print KelvinToFahrenheit(273)

print int(KelvinToFahrenheit(505.78))

print KelvinToFahrenheit(-5)
\vspace{14pt}

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai b
\vspace{12pt}

32.0

451

Traceback (most recent call last):

File "test.py", line 9, in

print KelvinToFahrenheit(-5)

File "test.py", line 4, in KelvinToFahrenheit

assert (Temperature >= 0),"Colder than absolute zero!"

AssertionError: Colder than absolute zero!
\vspace{12pt}
```

Apa itu Exception?

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

Ketika skrip Python menimbulkan pengecualian, ia harus menangani pengecualian begitu saja sehingga berhenti dan berhenti. Menangani pengecualian Jika Anda memiliki beberapa kode yang mencurigakan yang mungkin menim-

bulkan pengecualian, Anda dapat mempertahankan program Anda dengan menempatkan kode yang mencurigakan di coba: blokir. Setelah dicoba: blokir, sertakan sebuah pernyataan kecuali:, diikuti oleh blok kode yang menangani masalah ini seaman mungkin. Sintaksis Berikut adalah sintaks sederhana coba .... kecuali ... blok lain

```
try:

~~ You do your operations here;

~~ ......................

except \textit{ExceptionI}:

~~ If there is ExceptionI, then execute this block.

except \textit{ExceptionII}:

~~ If there is ExceptionII, then execute this block.

~~ ......................

else:

~~ If there is no exception then execute this block
\vspace{12pt}
\vspace{12pt}
```

Berikut adalah beberapa poin penting tentang sintaks yang disebutkan di atas -

Pernyataan percobaan tunggal dapat memiliki banyak kecuali pernyataan. Ini berguna saat blok coba berisi pernyataan yang mungkin membuang berbagai jenis pengecualian. Anda juga bisa memberikan klausa umum ke-

cuali klausul, yang menangani pengecualian apapun.Setelah klausa kecuali, Anda bisa memasukkan klausul lain. Kode di blok yang lain dijalankan jika kode di coba: blok tidak menimbulkan pengecualian.Blok yang lain adalah tempat yang baik untuk kode yang tidak perlu dicoba: perlindungan blokir. Contoh Contoh ini membuka file, menulis konten di file, dan keluar dengan anggun karena tidak ada masalah sama sekali -

```
 $  \#  $!/usr/bin/python
\vspace{12pt}

try:

~~ fh = open("testfile", "w")

~~ fh.write("This is my test file for exception handling!!")

except IOError:

~~ print "Error: can $  \setminus  $'t find file or read data"

else:

~~ print "Written content in the file successfully"

~~ fh.close()
\vspace{16pt}

Ini menghasilkan hasil sebagai berikut -
\vspace{12pt}

Written content in the file successfully
\vspace{12pt}
```

Klausul kecuali tanpa pengecualian anda juga dapat menggunakan pernyataan kecuali tanpa pengecualian yang didefinisikan sebagai berikut -

```
try:

~~ You do your operations here;

~~ ......................

except:

~~ If there is any exception, then execute this block.

~~ ......................

else:

~~ If there is no exception then execute this block.
\vspace{12pt}
\vspace{16pt}
```

Pernyataan try-except semacam ini menangkap semua pengecualian yang terjadi. Dengan menggunakan jenis try-except statement ini tidak dianggap sebagai praktik pemrograman yang bagus, karena menangkap semua pengecualian namun tidak membuat programmer mengenali akar permasalahan yang mungkin terjadi. Klausul Kecuali dengan Beberapa Pengecualian Anda juga dapat menggunakan pernyataan kecuali yang sama untuk menangani beberapa pengecualian sebagai berikut -

```
try:
   You do your operations here;
   ......................
except(Exception1[, Exception2[,...ExceptionN]]]):
   If there is any exception from the given exception list,
```

```
  then execute this block.
  .....................
else:
  If there is no exception then execute this block.


#!/usr/bin/python

try:
  fh = open("testfile", "w")
  fh.write("This is my test file for exception handling!!")
finally:
  print "Error: can \'t find file or read data"


#!/usr/bin/python

try:
  fh = open("testfile", "w")
  try:
    fh.write("This is my test file for exception handling!!")
  finally:
    print "Going to close the file"
    fh.close()
except IOError:
  print "Error: can \'t find file or read data"
```

Bila dikecualikan dilempar di blok coba, eksekusi langsung lolos ke blok akhirnya. Setelah semua pernyataan di blok akhirnya dieksekusi, pengecualian dinaikkan lagi dan ditangani dalam pernyataan kecuali jika ada di lapisan yang lebih tinggi dari pernyataan try-except. Argumen Eksepsi Pengecualian dapat memiliki argumen, yang merupakan nilai yang memberi informasi tambahan tentang masalah tersebut. Isi argumen bervariasi menurut pengecualian. Anda menangkap argumen pengecualian dengan menyediakan sebuah variabel dalam klausa kecuali sebagai berikut -

try:
  You do your operations here;
  .....
except *ExceptionType, Argument*:
  You can print value of Argument here...

Jika Anda menulis kode untuk menangani satu pengecualian, Anda dapat memiliki variabel mengikuti nama pengecualian dalam pernyataan kecuali. Jika Anda menjebak beberapa pengecualian, Anda dapat memiliki variabel mengikuti tuple pengecualian. Variabel ini menerima nilai pengecualian yang sebagian besar mengandung penyebab pengecualian. Variabel tersebut dapat menerima satu nilai atau beberapa nilai dalam bentuk tuple. Tuple ini biasanya berisi error string, error number, dan error location. Pengecualian yang Ditentukan Pengguna Python juga memungkinkan Anda membuat pengecualian sendiri dengan menurunkan kelas dari pengecualian standar built-in. Berikut adalah contoh yang berkaitan dengan RuntimeError. Di sini, sebuah kelas dibuat yang dikelompokkan dari RuntimeError. Ini berguna saat Anda perlu menampilkan informasi yang lebih spesifik saat pengecualian tertangkap. Di blok percobaan, pengecualian yang ditentukan pengguna dinaikkan dan ditangkap di blok kecuali. Variabel e digunakan untuk membuat sebuah instance dari class Networkerror.

```
(x,y) = (5,0)
try:
  z = x/y
except ZeroDivisionError:
  print ”divide by zero”
```

Jika Anda ingin memeriksa pengecualian dari kode, Anda bisa memiliki:

```
(x,y) = (5,0)
try:
```

```
  z = x/y
except ZeroDivisionError as e:
    z = e # representation: "¡exceptions.ZeroDivisionError instance at
0x817426c¿"
print z # output: "integer division or modulo by zero"
```

General Error Catching

Terkadang, Anda ingin menangkap semua kesalahan yang mungkin dihasilkan, tapi biasanya Anda tidak melakukannya. Dalam kebanyakan kasus, Anda ingin menjadi sespesifik mungkin (CatchWhatYouCanHandle). Pada contoh pertama di atas, jika Anda menggunakan klausul pengecualian catch-all dan pengguna menekan Ctrl-C, menghasilkan KeyboardInterrupt, Anda tidak ingin program mencetak "bagi dengan nol". Namun, ada beberapa situasi di mana yang terbaik untuk menangkap semua kesalahan. Misalnya, Anda menulis modul ekstensi ke layanan web. Anda ingin informasi kesalahan untuk output output halaman web, dan server untuk terus berjalan, jika mungkin. Tapi Anda tidak tahu kesalahan apa yang mungkin Anda masukkan ke dalam kode Anda. Dalam situasi seperti ini, Anda mungkin ingin mengode sesuatu seperti ini:

```
import sys
try:
  untrusted.execute()
except: # catch *all* exceptions
  e = sys.exc_info()[0]
  write_to_page( "¡p¿Error: %s¡/p¿" % e )
```

Menemukan Nama Pengecualian Spesifik Pengecualian standar yang dapat diajukan dijelaskan secara rinci pada: Lihatlah dokumentasi kelas untuk mengetahui pengecualian apa yang bisa diberikan oleh kelas tertentu. Lihat juga: Di wiki ini: WritingExceptionClasses, TracebackModule. Untuk gagasan umum (non-Python specific) tentang pengecualian, berkonsultasilah dengan ExceptionPatterns. Untuk menulis tentang ...

- Berikan contoh IOError, dan interpretasikan kode IOError.
- Berikan contoh beberapa pengecualian. Penanganan beberapa kecuali dalam satu baris.

Pertanyaan Penanganan Kesalahan Umum, Di bagian ”penanganan kesalahan umum” di atas, tertulis untuk menangkap semua pengecualian, Anda menggunakan kode berikut:

```
import sys
try:

  untrusted.execute()

except: # catch *all* exceptions

  e = sys.exc_info()[0]

  write_to_page( ”¡p¿Error: %s¡/p¿” % e )


try:

  untrusted.execute()

except Exception as e:

  write_to_page( ”¡p¿Error: %s¡/p¿” % str(e) )
```

Seseorang menunjukkan bahwa ”kecuali” menangkap lebih dari sekedar ”kecuali Pengecualian sebagai e.” Mengapa demikian? Apa bedanya? LionKimbro Untuk saat ini (versi ¡= 2.4) pengecualian tidak harus diwarisi dari Exception. Jadi polos ’kecuali:’nangkap semua pengecualian, tidak hanya sistem. Pengecualian string adalah salah satu contoh pengecualian yang tidak mewarisi dari Exception. MikeRovner Saya percaya bahwa pada 2,7, pengecualian masih tidak harus diwariskan dari Exception atau bahkan BaseException. Namun, seperti Python 3,

pengecualian harus subclass BaseException. - gajah jim Mendapatkan Informasi Berguna dari Pengecualian Jadi, saya punya sesuatu seperti:

```
(a,b,c) = d
```

dan Python kembali:

```
ValueError: unpack list of wrong size
```

... dan begitulah, Anda tentu bertanya-tanya, ”Nah, apa yang ada di d?”

Anda tahu - Anda bisa mencetak di sana, dan itu berhasil. Tapi adakah cara yang lebih baik dan lebih menarik untuk mendapatkan informasi yang diketahui orang? Anda bisa melakukan sesuatu seperti:

```
try:
  a, b, c = d
except Exception as e:
  e.args += (d,)
  raise
```

Atribut .args pengecualian adalah tuple dari semua argumen yang dilewatkan (biasanya argumen satu dan satu-satunya adalah pesan kesalahannya). Dengan cara ini Anda dapat mengubah argumen dan menaikkan kembali, dan informasi tambahan akan ditampilkan. Anda juga bisa membuat pernyataan cetak atau login di blok kecuali. Perhatikan bahwa tidak semua pengecualian subclass Exception (meski hampir semua dilakukan), jadi ini mungkin tidak menangkap beberapa pengecualian; Selain itu, pengecualian tidak diperlukan untuk memiliki atribut .args (meskipun jika pengecualian subclass Exception dan tidak mengesampingkan _ _init _ _ tanpa memanggil superclass-nya), maka kode yang ditulis mungkin gagal Namun dalam prakteknya hampir tidak pernah (dan Jika ya, Anda harus memper-

baiki pengecualian yang tidak sesuai!) Bukankah lebih baik mencegahnya untuk melakukan remediasi? ¿ Joel Spolsky mungkin programmer hebat C ++, dan sarannya untuk desain antarmuka pengguna sangat berharga, tapi Python bukan C ++ atau Java, dan argumennya tentang pengecualian tidak berlaku dengan Python. Joel berpendapat: ”Mereka tidak terlihat dalam kode sumber Melihat kumpulan kode, termasuk fungsi yang mungkin atau mungkin tidak membuang pengecualian, tidak ada cara untuk melihat pengecualian mana yang mungkin dilempar dan dari mana.Ini berarti bahwa pemeriksaan kode yang hati-hati pun tidak. Saya bisa mengungkapkan potensi bug. ”

(Perhatikan bahwa ini juga merupakan argumen di balik pengecualian yang diperiksa oleh Java - sekarang eksplisit bahwa pengecualian dapat dilemparkan - kecuali bahwa RuntimeException masih dapat dibuang ke mana saja. -jJ) Saya tidak mengerti argumen ini. Dalam kode sumber acak, tidak ada cara untuk mengetahui apakah akan gagal hanya dengan inspeksi. Jika Anda melihat:

```
x = 1
result = myfunction (x)
```

Anda tidak dapat mengetahui apakah fungsi saya gagal pada saat runtime hanya dengan inspeksi, jadi mengapa harus itu penting apakah gagal menabrak pada saat runtime atau gagal dengan meningkatkan pengecualian?

(Crashing itu buruk Dengan secara eksplisit menyatakan pengecualian, Anda memperingatkan orang-orang bahwa mereka mungkin ingin mengatasinya Jawa melakukannya dengan canggung C tidak memiliki cara yang baik untuk melakukannya sama sekali, karena kesalahan kembali masih di band Untuk pengembalian reguler Di python, pengecualian passthrough tidak ditandai, namun kondisi kesalahan menonjol di tempat mereka diciptakan, dan biasanya tidak meniru hasil yang benar. -jJ)

Argumen Joel yang mengemukakan pengecualian hanyalah sebuah goto yang menyamar sebagian benar. Tapi begitu juga untuk loop, sementara loop, fungsi dan metode! Seperti konstruksi lainnya, pengecualian adalah gotos yang dijinakkan dan dipekerjakan untuk Anda, bukan yang liar dan berbahaya. Anda tidak bisa melompat * di mana saja *, hanya tempat yang sangat terbatas.

Joel juga menulis:

”Mereka membuat terlalu banyak titik keluar yang mungkin untuk sebuah fungsi.Untuk menulis kode yang benar, Anda benar-benar harus memikirkan setiap jalur kode yang mungkin melalui fungsi Anda. Setiap kali Anda memanggil fungsi yang dapat meningkatkan pengecualian dan tidak menangkapnya di Spot, Anda menciptakan peluang untuk kejutan bug yang disebabkan oleh fungsi yang dihentikan tiba-tiba, meninggalkan data dalam keadaan tidak konsisten, atau jalur kode lainnya yang tidak Anda pikirkan. ”

```
try:
   You do your operations here;
   ......................
except(Exception1[, Exception2[,...ExceptionN]]]):
   If there is any exception from the given exception list,
   then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

```
#!/usr/bin/python

try:
   fh = open(”testfile”, ”w”)
   fh.write(”This is my test file for exception handling!!”)
finally:
```

```
    print ”Error: can \’t find file or read data”


  #!/usr/bin/python

  try:
    fh = open(”testfile”, ”w”)
    try:
      fh.write(”This is my test file for exception handling!!”)
    finally:
      print ”Going to close the file”
      fh.close()
  except IOError:
      print ”Error: can \’t find file or read data”
```

Bila dikecualikan dilempar di blok coba, eksekusi langsung lolos ke blok akhirnya. Setelah semua pernyataan di blok akhirnya dieksekusi, pengecualian dinaikkan lagi dan ditangani dalam pernyataan kecuali jika ada di lapisan yang lebih tinggi dari pernyataan try-except. Argumen Eksepsi. Python juga memungkinkan Anda membuat pengecualian sendiri dengan menurunkan kelas dari pengecualian standar built-in. Berikut adalah contoh yang berkaitan dengan RuntimeError. Di sini, sebuah kelas dibuat yang dikelompokkan dari RuntimeError. Ini berguna saat Anda perlu menampilkan informasi yang lebih spesifik saat pengecualian tertangkap. Di blok percobaan, pengecualian yang ditentukan pengguna dinaikkan dan ditangkap di blok kecuali. Variabel e digunakan untuk membuat sebuah instance dari class Networkerror.

CHAPTER 15

CLASESS/OBJECT

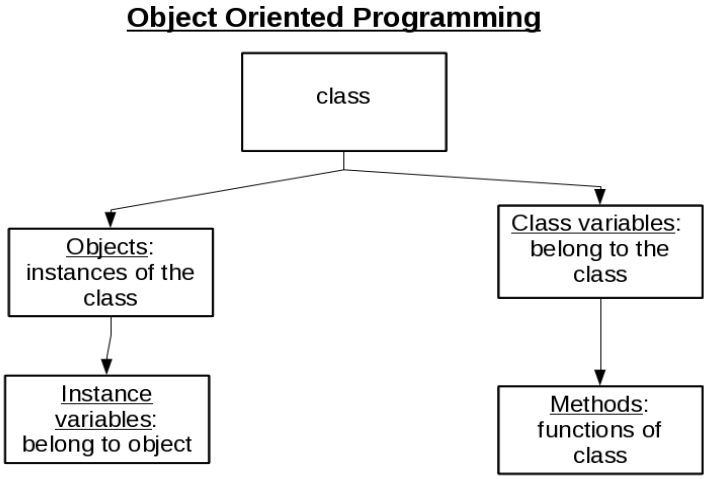
Figure 15.1 Python Object Oriented

15.1 Pengertian

15.1.1 Python Object Oriented

Phyton merupakan bahasa pemrograman yang berorientasi obyek dinamis, dapat digunakan untuk bermacam-macam pengembangan perangkat lunak. Phyton menyediakan dukungan yang kuat untuk integrasi dengan bahasa pemrograman lain dan alat-alat bantu lainnya. Object Oriented Programming (OOP) adalah sebuah pendekatan pemrograman dimana objek didefinisikan dengan metode (fungsi, action, atau events) dan sifat (nilai serta karakteristik), sehingga mudah dibaca, lebih banyak kode yang dapat digunakan kembali. Python telah menjadi bahasa berorientasi objek sejak itu ada. Karena itu, menciptakan dan menggunakan kelas dan objek sangat mudah. Bab ini membantu Anda menjadi ahli dalam menggunakan dukungan pemrograman berorientasi objek Objek adalah sesuatu yang menamping nilai atau data dan dapat dikenakan operasi tertentu:

1. Class atau Kelas: adalah struktur data yang bisa digunakan untuk mendefinisikan objek yang menyimpan data bersama-sama nilai-nilai dan perilaku. Kelas adalah suatu entitas yang merupakan bentuk program dari suatu abstraksi untuk permasalahan dunia nyata, dan instans dari class merupakan realisasi dari beberapa objek.
2. Inheritace atau pewarisan merupakan konsep dalam pemrograman berbasis objek yang memungkinkan untuk membuat suatu kelas dengan didasarkan pada kelas yang sudah ada sehingga mewarisi semua method dan atributnya. Dengan cara seperti ini, semua method dan atribut yang terdapat pada kelas induk diturunkan ke kelas turunannya. Namun kelas turunannya dapat menambah method baru atau atribut baru tersendiri.
3. Method Constructor merupakan sebuah method yang akan otomatis dipanggil ketika objek diinstantiasi. Construktor umumnya digunakan untuk melakukan inisialisasi terhadap suatu variabel atau method.



$$PythonObjectOriented \tag{15.1}$$

Jika Anda tidak memiliki pengalaman sebelumnya dengan pemrograman berorientasi objek (OO), Anda mungkin ingin berkonsultasi dengan kursus perkenalan atau setidaknya tutorial semacam itu sehingga Anda dapat memahami konsep dasarnya. Namun, di sini adalah pengenalan kecil Object-Oriented Programming (OOP) untuk membawa Anda pada kecepatan - Ikhtisar Terminologi OOP.

1. Kelas: Prototipe yang ditentukan pengguna untuk objek yang mendefinisikan seperangkat atribut yang menjadi ciri objek kelas apa pun. Atribut adalah data anggota (variabel kelas dan variabel contoh) dan metode, diakses melalui notasi titik.

2. Variabel kelas: Variabel yang dimiliki oleh semua instance kelas. Variabel kelas didefinisikan dalam kelas tapi di luar metode kelas manapun. Variabel kelas tidak digunakan sesering variabel contoh.

3. Anggota data: Variabel kelas atau variabel contoh yang menyimpan data yang terkait dengan kelas dan objeknya.

4. Fungsi overloading: Penugasan lebih dari satu perilaku ke fungsi tertentu. Operasi yang dilakukan bervariasi menurut jenis objek atau argumen yang terlibat.
5. Contoh variabel: Variabel yang didefinisikan di dalam metode dan hanya dimiliki oleh instance kelas saat ini.
6. Warisan: Pengalihan karakteristik kelas ke kelas lain yang berasal darinya. Contoh: Objek individual dari kelas tertentu. Obyek obj yang termasuk dalam Lingkaran kelas, misalnya, adalah turunan dari Lingkaran kelas.
7. Instansiasi: Pembuatan sebuah instance dari sebuah kelas.
8. Metode: Jenis fungsi khusus yang didefinisikan dalam definisi kelas.
9. Objek: Contoh unik dari struktur data yang didefinisikan oleh kelasnya. Objek terdiri dari kedua anggota data (variabel kelas dan variabel contoh) dan metode.
10. Operator overloading: Penugasan lebih dari satu fungsi ke operator tertentu.

Salah satu alasan yang paling penting untuk mempertimbangkan bekerja di OOP adalah bahwa ia menyediakan pendekatan pemodelan langsung dan memecahkan masalah didunia nyata.

15.1.2 Membuat Kelas

Pernyataan kelas membuat definisi kelas baru. Nama kelas segera mengikuti kelas kata kunci diikuti oleh titik dua sebagai berikut -

class ClassName: 'Optional class documentation string' class _suite

Kelas memiliki kumpulan dokumentasi, yang bisa diakses melalui ClassName . _ _ doc _ _. Class _suite terdiri dari semua pernyataan komponen yang mendefinisikan anggota kelas, atribut dan fungsi data.

Berikut adalah contoh kelas Python sederhana -

```
class Employee:
~~ 'Common base class for all employees'
~~ empCount = 0

~~ def  $  \_  $ $  \_  $init $  \_  $ $  \_  $(self, name, sa
~~~~~ self.name = name ~~~~~ self.salary = salary
~~~~~ Employee.empCount += 1
~~~~~ def displayCount(self):
~~~~~ print "Total Employee  $  \%  $d"  $  \%  $ Employee.emp
~~ def displayEmployee(self):

~~~~~~print "Name : ", self.name,  ", Salary: ", self.salary
```

15.1.3 Variable empCount

Variabel empCount adalah variabel kelas yang nilainya dibagi di antara semua contoh kelas ini. Ini bisa diakses sebagai Employee.empCount dari dalam kelas atau di luar kelas. Metode pertama _ _init _ _ () adalah metode khusus, yang disebut metode konstruktor kelas atau inisialisasi yang Python panggil saat Anda membuat instance baru dari kelas ini. Anda menyatakan metode kelas lain seperti fungsi normal dengan pengecualian bahwa argumen pertama untuk setiap metode adalah self. Python menambahkan argumen diri ke daftar untuk Anda; Anda tidak perlu memasukkannya saat Anda memanggil metode.

15.1.4 Membuat Instance Objects

Untuk membuat contoh kelas, Anda memanggil kelas menggunakan nama kelas dan meneruskan argumen apa pun yang diterima metode _ _init _ _-nya.

”Ini akan menciptakan objek pertama kelas Karyawan” Emp1 = Karyawan (”Zara”, 2000) ”Ini akan menciptakan objek kedua dari kelas Karyawan” Emp2 = Karyawan (”Manni”, 5000)

15.1.5 Mengakses Atribut

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

Emp1.displayEmployee () Emp2.displayEmployee (Cetak ”Jumlah Karyawan % d” % Employee.empCount

Sekarang, meletakkan semua konsep bersama -

```
 $  \#  $!/usr/bin/python
class Employee:
~~ 'Common base class for all employees'
~~ empCount = 0
~~ def  $  \_  $ $  \_  $init $  \_  $ $  \_  $(self, name, sa
~~~~~ self.name = name
~~~~~ self.salary = salary
~~~~~ Employee.empCount += 1
~~ def displayCount(self):
~~~~ print "Total Employee  $  \%  $d"  $  \%  $ Employee.empC
~~ def displayEmployee(self):
~~~~~~print "Name : ", self.name,  ", Salary: ", self.salary
"This would create first object of Employee class"
emp1 = Employee("Zara", 2000)
"This would create second object of Employee class"
emp2 = Employee("Manni", 5000)
emp1.displayEmployee()
emp2.displayEmployee()
print "Total Employee  $  \%  $d"  $  \%  $ Employee.empCount
```

When the above code is executed, it produces the following result $ - $

Name : Zara ,Salary: 2000 Name : Manni ,Salary: 5000 Total Employee 2 You can add, remove, or modify attributes of classes and objects at any time $ - $ emp1.age = 7 # Add an 'age' attribute. emp1.age = 8 $ # $Modify'age'attribute.del\ emp1.age$\# Delete 'age' attribute.

Alih-alih menggunakan pernyataan normal untuk mengakses atribut, Anda dapat menggunakan fungsi berikut -

Getattr (obj, name [, default]): untuk mengakses atribut objek.

Hasattr (obj, name): untuk memeriksa apakah ada atribut atau tidak.

Setattr (obj, name, value): untuk mengatur atribut. Jika atribut tidak ada, maka akan dibuat.

The delattr (obj, name): untuk menghapus sebuah atribut.

Hasattr (emp1, 'age') $ # $ Mengembalikan true jika atribut 'age' ada

Getattr (emp1, 'age') $ # $ Mengembalikan nilai atribut 'age'

Setattr (emp1, 'age', 8) $ # $ Set attribute 'age' di 8 Delattr (empl, 'age') $ # $ Hapus atribut 'umur'

Atribut Atribut Built-In

Setiap kelas Python terus mengikuti atribut bawaan dan mereka dapat diakses menggunakan operator dot seperti atribut lainnya -

~~~ $ _ $ $ _ $dict $ _ $ $ _ $:
Kamus yang berisi namespace kelas.

~~~ $ _ $ $ _ $doc $ _ $ $ _ $:
String dokumentasi kelas atau tidak, jika tidak terdefinisi.

~~~ $ _ $ $ _ $name $ _ $ $ _ $:
nama kelas ~~~ $ _ $ $ _ $module $ _ $ $ _ $:
Nama modul dimana kelas didefinisikan. Atribut ini " _ _main _ _" dalam mode interaktif.
~~~ $ _ $ $ _ $bases $ _ $ $ _ $:
Tupel yang mungkin kosong yang berisi kelas dasar, sesuai urutan kejadiannya dalam daftar kelas dasar.

Untuk kelas di atas mari kita coba untuk mengakses semua atribut ini

```
$ \# $!/usr/bin/python
class Employee:
~~ 'Common base class for all employees'
~~ empCount = 0
~~ def $ \_ $ $ \_ $init $ \_ $ $ \_ $(self, name, sa
~~~~~ self.name = name
~~~~~ self.salary = salary
```

```
~~~~~ Employee.empCount += 1
~~ def displayCount(self):
~~~~ print "Total Employee  $  \%  $d"  $  \%  $ Employee.empC
~~ def displayEmployee(self): \par
~~~~~~print "Name : ", self.name,  ", Salary: ", self.salary
print "Employee. $  \_  $ $  \_  $doc $  \_  $ $  \_  $:", Emp
print "Employee. $  \_  $ $  \_  $name $  \_  $ $  \_  $:", Em
print "Employee. $  \_  $ $  \_  $module $  \_  $ $  \_  $:",
print "Employee. $  \_  $ $  \_  $bases $  \_  $ $  \_  $:", E
print "Employee. $  \_  $ $  \_  $dict $  \_  $ $  \_  $:", Em
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut - Karyawan .
$ _ $ $ _ $ doc $ _ $ $ _ $:
kelas dasar umum untuk semua karyawan Karyawan .
$ _ $ $ _ $ name $ _ $ $ _ $:
Karyawan Karyawan . $ _ $ $ _ $ modul $ _ $ $ _ $:
_ _main _ _ Karyawan .
$ _ $ $ _ $ bases $ _ $ $ _ $:
() Karyawan . $ _ $ $ _ $ dict $ _ $ $ _ $:
{’ _ _module _ _’: ’ _ _main _ _’, ’displayCount’: ¡function displayCount at 0xb7c84994¿, ’empCount’: 2, ’DisplayEmployee’: ¡function displayEmployee at 0xb7c8441c¿,
’ $ _ $ $ _ $doc $ _ $ $ _ $’:
’Kelas dasar umum untuk semua karyawan’,
’ $ _ $ $ _ $init $ _ $ $ _ $’:
¡function _ _init _ _ di 0xb7c846bc¿ }

15.1.6 Menghancurkan Objek (Pengumpulan Sampah)

Python menghapus objek yang tidak dibutuhkan (tipe built-in atau instance kelas) secara otomatis untuk membebaskan ruang memori. Proses dimana Python secara berkala mengumpulkan kembali blok memori yang tidak lagi digunakan disebut Koleksi Sampah. Pengumpul sampah Python berjalan selama eksekusi program dan dipicu saat penghitungan referensi objek mencapai nol. Jumlah referensi referensi berubah karena jumlah alias yang menunjukkannya berubah. Jumlah referensi objek meningkat saat diberi nama baru atau ditempatkan dalam wadah (daftar, tupel, atau kamus). Jum-

lah referensi objek berkurang saat dihapus dengan del, rujukannya ditugaskan kembali, atau rujukannya tidak sesuai. Ketika penghitungan referensi objek mencapai nol, Python mengumpulkannya secara otomatis. a = 40 $ # $ Buat objek ¡40¿ B = a $ # $ Tingkatkan ref. Hitung ¡40¿ c = [b] $ # $ Tingkatkan ref. Hitung ¡40¿ Del # Penurunan ref. Hitung ¡40¿ b = 100 # Kurangi ref. Hitung ¡40¿ C [0] = -1 # Kurangi ref. Hitung ¡40¿

Anda biasanya tidak akan memperhatikan kapan pengumpul sampah menghancurkan contoh yatim piatu dan mengembalikan ruangnya. Tapi kelas bisa menerapkan metode khusus _ _del _ _ (), yang disebut destructor, yang dipanggil saat instance tersebut hendak dimusnahkan. Metode ini bisa digunakan untuk membersihkan sumber daya non memori yang digunakan oleh sebuah instance.

Contoh Penghancur `$ \_ $ $ \_ $del $ \_ $ $ \_ $` () ini mencetak nama kelas sebuah instance yang akan dihancurkan -

```
$ \# $!/usr/bin/python
class Point:
~~ def $ \_ $ $ \_ $init $ \_ $ $ \_ $( self, x=0, y=
~~~~~ self.x = x
~~~~~ self.y = y
~~ def $ \_ $ $ \_ $del $ \_ $ $ \_ $(self):
~~~~~ class $ \_ $name = self. $ \_ $ $ \_ $class $ \_
~~~~~ print class $ \_ $name, "destroyed"
pt1 = Point()
pt2 = pt1
pt3 = pt1
print id(pt1), id(pt2), id(pt3) $ \# $ prints the ids of th
del pt1
del pt2
del pt3
```

15.1.7 Kelas Warisan

Alih-alih mulai dari nol, Anda dapat membuat kelas dengan menurunkannya dari kelas yang sudah ada sebelumnya dengan mencantumkan kelas induk dalam tanda kurung sete-

lah nama kelas yang baru. Kelas anak mewarisi atribut kelas induknya, dan Anda dapat menggunakan atribut tersebut seolah-olah mereka didefinisikan di kelas anak. Kelas anak juga dapat mengesampingkan data anggota dan metode dari orang tua. Sintaksis Kelas turunan dinyatakan seperti kelas orang tua mereka; Namun, daftar kelas dasa yang diwarisi dari diberikan setelah nama kelas -

Kelas SubClassName (ParentClass1 [, ParentClass2, ...]):

```
~~ 'String dokumentasi kelas opsional'
~~ Class $  \_  $suite
 $  \#  $!/usr/bin/python

class~Parent:~~~~~~   $  \#  $ define parent class
~~ parentAttr = 100
~~ def  $  \_  $ $  \_  $init $  \_  $ $  \_  $(self):
~~~~~ print "Calling parent constructor"
~~ def parentMethod(self):
~~~~~ print 'Calling parent method'
~~ def setAttr(self, attr):
~~~~~ Parent.parentAttr = attr
~~ def getAttr(self):
~~~~~ print "Parent attribute :", Parent.parentAttr
class Child(Parent):  $  \#  $ define child class
~~ def  $  \_  $ $  \_  $init $  \_  $ $  \_  $(self):
~~~~~ print "Calling child constructor"
~~ def childMethod(self):
~~~~~ print 'Calling child method'
c~=~Child()~~~~~~~    $  \#  $ instance of child
c.childMethod()~~~~~  $  \#  $ child calls its method
c.parentMethod()~~~~  $  \#  $ calls parent's method
c.setAttr(200)~~~~~~  $  \#  $ again call parent's method
c.getAttr()~~~~~~~~~  $  \#  $ again call parent's method
Calling child constructor
Calling child method
Calling parent method
Parent attribute : 200
class~A:~~~~~~   $  \#  $ define your class A
.....
```

```
class~B:~~~~~~~   $  \#  $ define your class B
.....
class~C(A,~B):    $  \#  $ subclass of A and B
```

Anda dapat menggunakan fungsi issubclass () atau isinstance () untuk memeriksa hubungan dua kelas dan contoh. Fungsi boolean issubclass (sub, sup) mengembalikan true jika sub subclass yang diberikan memang merupakan subclass dari superclass sup. The isinstance (obj, Class) fungsi boolean mengembalikan true jika obj adalah turunan dari Class Class atau merupakan instance dari subclass of Class.

15.1.8 Metode utama

Anda selalu dapat mengganti metode kelas induk Anda. Salah satu alasan untuk mengesampingkan metode orang tua adalah karena Anda mungkin menginginkan fungsi khusus atau berbeda di subkelas Anda.

Contoh

```
 $  \#  $! / Usr / bin / python
class parent:  $  \#  $ define parent class
~~ Def myMethod (diri):
~~~~~ Cetak 'metode induk panggilan'

Kelas anak (orang tua):  $  \#  $ define child class
~~ Def myMethod (diri):
~~~~~ Cetak 'metode memanggil anak'
C = Anak ()  $  \#  $ contoh anak
C.myMethod ()  $  \#  $ metode panggilan balik anak
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut - Memanggil metode anak Metode Base Overloading Berikut daftar tabel beberapa fungsionalitas generik yang dapat Anda timpa di kelas Anda sendiri -

15.1.9 Operator overloading

Misalkan Anda telah membuat kelas Vektor untuk mewakili vektor dua dimensi, apa yang terjadi bila Anda menggunakan operator plus untuk menambahkannya? Kemungkinan besar Python akan berteriak pada Anda. Anda bisa,

bagaimanapun, menentukan metode _ _add _ _ di kelas Anda untuk melakukan penambahan vektor dan operator plus akan berperilaku sesuai harapan -

Contoh

```
 $ \# $! / Usr / bin / python
Kelas vektor:
~~ def $ \_ $ $ \_ $init $ \_ $ $ \_  $ (diri, a, b):
~~~~~ Self.a = a
~~~~~ Self.b = b
~~ def  $  \_  $ $  \_  $str  $  \_  $ $  \_  $ (diri):
~~~~~ Return 'Vector ( $  \%  $ d, $  \%  $ d)' $  \%  $ (self
~~
~~ Def  $  \_  $ $  \_  $add  $  \_  $ $  \_  $ (diri sendiri,
~~~~~ return Vector (self.a + other.a, self.b + other.b)
v1 = vektor (2,10)
v2 = vektor (5, -2)
cetak v1 + v2
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut - Vektor (7,8)

15.1.10 Persembunyian data

Atribut objek mungkin atau mungkin tidak terlihat di luar definisi kelas. Anda perlu memberi nama atribut dengan awalan ganda ganda, dan atribut tersebut kemudian tidak langsung terlihat oleh orang luar. Contoh

```
 $ \# $! / Usr / bin / python

Kelas JustCounter:
~~  $  \_  $ $  \_  $secretCount = 0
~
~~ def menghitung (diri):
~~~~~ self . $  \_  $ $  \_  $ secretCount + = 1
~~~~~ cetak diri . $  \_  $ $  \_  $ secretCount
counter = JustCounter ()
Counter.count ()
Counter.count ()
```

```
print counter . $  \_  $ $  \_  $ secretCount
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut - 1 2 Traceback (panggilan terakhir): File ”test.py”, baris 12, di ¡module¿ print counter . $ _ $ $ _ $ secretCount AttributeError: instance JustCounter tidak memiliki atribut ’ $ _ $ $ _ $secretCount’

Python melindungi anggota tersebut dengan mengganti namanya secara internal untuk memasukkan nama kelas. Anda dapat mengakses atribut seperti object. _className _ _attrName. Jika Anda akan mengganti baris terakhir Anda sebagai berikut, maka akan berhasil untuk Anda - print counter. _JustCounter _ _secretCount Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut - 1 2 2 Contoh

Kelas bisa mewarisi kelas lainnya. Kelas dapat mewarisi atribut dan perilaku (metode) dari kelas lain, yang disebut kelas super. Sebuah kelas yang mewarisi dari kelas super disebut Sub-kelas. Kelas super kadang disebut nenek moyang juga. Ada hubungan hierarki antar kelas. Jika kita melihat lebih dekat contoh sebelumnya tentang akun kelas, kita dapat melihat bahwa model ini dapat memenuhi kebutuhan bank sebenarnya. Bank biasanya memiliki jenis akun yang berbeda, mis. Rekening Tabungan, Rekening Giro dan lain-lain. Meskipun jenis akun yang berbeda ini sangat berbeda, namun tetap memiliki banyak sifat dan metode yang sama. Misalnya. Setiap akun memiliki dan membutuhkan nomor rekening, pemegang dan saldo. Selanjutnya mungkin bagi masing-masing untuk menyetor atau menarik uang. Jadi, ada sesuatu seperti akun ”mendasar” darimana mereka mewarisi. Warisan digunakan untuk membuat kelas baru dengan menggunakan kelas yang ada. Yang baru dapat diciptakan dengan memperluas dan dengan membatasi kelas yang ada. Sekarang saatnya untuk kembali ke Python dan melihat bagaimana kelas diimplementasikan dengan Python. Kita mulai dengan kelas yang paling sederhana, yang bisa didefinisikan. Kami hanya memberikan nama tapi menghi-

langkan semua spesifikasi lebih lanjut dengan menggunakan kata kunci n. Class Account (objek):

Kami belum mendefinisikan atribut atau metode apa pun di kelas akun sederhana kami. Sekarang kita akan membuat sebuah instance dari kelas kosong ini:

¿¿¿ dari Account import Account ¿¿¿ x = Akun () ¿¿¿ cetak x ¡Account.Account objek di 0x7f364120ab90¿ Sebuah metode berbeda dari satu fungsi saja dalam dua aspek: Itu milik kelas dan itu didefinisikan dalam kelas Parameter pertama dalam definisi suatu metode harus menjadi referensi ”diri” pada instance kelas Sebuah metode disebut tanpa parameter ini ”diri” Kami memperluas kelas kami dengan mendefinisikan beberapa metode. Tubuh dari metode ini masih belum ditentukan: kelas Account (objek): Transfer def (self, target, amount): lulus Def deposit (self, amount): lulus Def withdraw (self, amount): lulus def keseimbangan (diri): lulus

Python tidak memiliki konstruktor eksplisit seperti C ++ atau Java, tapi metode `$ \_ $ $ \_ $init $ \_ $ $ \_ $ ()` dengan Python ada Tapi secara tegas, akan salah jika menyebutnya sebagai konstruktor, karena contoh baru sudah ”dibangun” pada saat metode `$ \_ $ $ \_ $init $ \_ $ $ \_ $` dipanggil. Tapi bagaim digunakan - seperti konstruktor pada bahasa pemrograman berorientasi objek lainnya - untuk menginisialisasi variabel instance dari sebuah objek. Definisi metode init terlihat seperti definisi metode lainnya:

Def `$ \_ $ $ \_ $init $ \_ $ $ \_ $` (self, holder, number, balance, credit $ _ $line = 1500): self.Holder = pemegang Nomor self.Number = self.Balance = keseimbangan self.CreditLine = credit $ _ $line

Apa yang kami katakan tentang konstruktor berlaku bagi penghancur juga. Tidak ada destruktor ”nyata”, tapi ada yang serupa, yaitu metode _ _del _ _. Hal ini disebut ketika contoh ini akan hancur. Jika kelas dasar memiliki metode _ _del _ _ (), metode _ _del _ _ () kelas turunan, jika ada, harus

secara eksplisit memanggilnya untuk memastikan penghapusan komponen kelas dasar contoh yang tepat. Contoh berikut menunjukkan kelas dengan konstruktor dan destruktor:

```
Kelas Salam:
~~~ Def  $  \_  $ $  \_  $init  $  \_  $ $  \_  $ (diri, nama)
~~~~~~~ self.name = nama
~~~ Def  $  \_  $ $  \_  $del  $  \_  $ $  \_  $ (diri):
~~~~~~~ Cetak "Destructor dimulai"
~~~ Def SayHello (diri):
~~~~~~~ Cetak "Halo", self.name
```

CHAPTER 16

NETWORKING

16.1 Pengertian Jaringan

Jaringan yaitu sekumpulan komputer yang dihubungkan dengan kabel sehingga komputer yang satu dengan komputer yang lainnya dapat saling komunikasi, bertukar informasi sharing file, printer, dan sebagainya. Networking merupakan salah satu cabang ilmu dunia Teknik Informatika yang membahas tentang komunikasi antar komputer. Materi networking yang di berikan di sekolah atau di perkuliahan saat ini sepertinya belum cukup memadai dari yang diharapkan. Bagi mereka yang sangat ingin mendalami tentang ilmu networking bisa mempelajarinya dari artikel-artikel di internet, dan biasanya ketika kita menemukan artikel tentang materi networking yang ingin dipelajari sering sekali ditemukan kata-kata atau istilah-istilah

yang belum dimengerti, biasanya kita akan mencari kata-kata tersebut dengan mengetikkan keywordnya di mesin pencari Google. lalu kita akan belajar memahami kata tersebut, setelah kita mengerti kita akan kembali mempelajari materi yang tadi. cara ini tentu tidak efektif. maka dari sebaiknya sebelum kita mempelajari mengenai networking kita pelajari dulu dari yang paling dasar, yaitu istilah-istilah dalam networking. Networking sangat dibutuhkan, terutama pada zaman yang semakin lama semakin canggih seperti ini, karena jaringan itu tentu sangat penting untuk berlangsungnya hubungan atau komunikasi antar komputer. Misalnya saja untuk berbagi atau sharing printer, tidak mungkin setiap komputer memiliki printer satu-satu makannya dibuatlah jaringan komputer itu untuk berbagi penggunaan printer secara bersama-sama dan juga berfungsi untuk sharing internet, satu komputer (server) dapat ip address dari isp, lalu si server itu membagikan koneksi internet ke client-client dikantornya.

16.2 Jenis-Jenis Jaringan berdasarkan jangkuan

16.2.1 Local Area Networking (LAN)

Yaitu Jaringan yang dibatasi oleh area yang relative kecil, umumnya dibatasi oleh area lingkungan seperti sebuah perkantoran di sebuah gedung, atau sebuah sekolah, dan biasanya tidak jauh dari sekitar 1 km persegi.

16.2.2 Metropolitan Area Networking (MAN)

Yaitu Jaringan yang lebih luas dari LAN, MAN biasanya meliputi area yang lebih besar seperti area propinsi, antar gedung. Mengapa MAN itu dikatakan lebih luas dari LAN?, Yah, karena jaringan MAN itu terhubung dari beberapa jaringan LAN yang dihubungkan melalui switch lagi.

16.2.3 Wide Area Networking (WAN)

Yaitu Jaringan yang lingkupnya biasanya sudah menggunakan sarana Satelit ataupun kabel bawah laut sebagai con-

toh keseluruhan jaringan BANK BNI yang ada di Indonesia ataupun yang ada di Negara-negara lain. Menggunakan sarana WAN, Sebuah Bank yang ada di Bandung bisa menghubungi kantor cabangnya yang ada di Hongkong, hanya dalam beberapa menit. Biasanya WAN agak rumit dan sangat kompleks, menggunakan banyak sarana untuk menghubungkan antara LAN dan WAN ke dalam Komunikasi Global seperti Internet.

16.3 Manfaat Jaringan Komputer

Berbicara mengenai manfaat dari jaringan komputer. Terdapat banyak sekali manfaat jaringan komputer, antara lain :

1. Dengan jaringan komputer, kita bisa mengakses file yang kita miliki sekaligus file orang lain yang telah diseberluaskan melalui suatu jaringan, semisal jaringan internet.
2. Melalui jaringan komputer, kita bisa melakukan proses pengiriman data secara cepat dan efisien.
3. Jaringan komputer membantu seseorang berhubungan dengan orang lain dari berbagai negara dengan mudah.
4. Selain itu, pengguna juga dapat mengirim teks, gambar, audio, maupun video secara real time dengan bantuan jaringan komputer.
5. Kita dapat mengakses berita atau informasi dengan sangat mudah melalui internet dikarenakan internet merupakan salah satu contoh jaringan komputer.

16.4 Macam-Macam Jaringan Komputer

Umumnya jaringan komputer di kelompokkan menjadi 5 kategori, yaitu berdasarkan jangkauan geografis, distribusi sumber informasi/ data, media transmisi data, peranan dan hubungan tiap komputer dalam memproses data, dan berdasarkan jenis topologi yang digunakan. Berikut penjabaran lengkapnya :

16.4.1 A. Berdasarkan Jangkauan Geografis

16.4.1.1 LAN Local Area Network atau yang sering disingkat dengan LAN merupakan jaringan yang hanya mencakup wilayah kecil saja, semisal warnet, kantor, atau sekolah. Umumnya jaringan LAN luas areanya tidak jauh dari 1 km persegi. Biasanya jaringan LAN menggunakan teknologi IEEE 802.3 Ethernet yang mempunyai kecepatan transfer data sekitar 10, 100, bahkan 1000 MB/s. Selain menggunakan teknologi Ethernet, tak sedikit juga yang menggunakan teknologi nirkabel seperti Wi-fi untuk jaringan LAN. Keuntungan dari penggunaan Jenis Jaringan Komputer LAN seperti lebih irit dalam pengeluaran biaya operasional, lebih irit dalam penggunaan kabel, transfer data antar node dan komputer labih cepat karena mencakup wilayah yang sempit atau lokal, dan tidak memerlukan operator telekomunikasi untuk membuat sebuah jaringan LAN. Kerugian dari penggunaan Jenis Jaringan LAN adalah cakupan wilayah jaringan lebih sempit sehingga untuk berkomunikasi ke luar jaringan menjadi lebih sulit dan area cakupan transfer data tidak begitu luas. Berbeda dengan Jaringan Area Luas atau Wide Area Network(WAN), maka LAN mempunyai karakteristik sebagai berikut:

- Mempunyai pesat data yang lebih tinggi.
- Meliputi wilayah geografi yang lebih sempit.
- Tidak membutuhkan jalur telekomunikasi yang disewa dari dari operator telekomunikasi.

16.4.1.2 MAN Metropolitan Area Network atau MAN merupakan jaringan yang mencakup suatu kota dengan dibekali kecepatan transfer data yang tinggi. Bisa dibilang, jaringan MAN merupakan gabungan dari beberapa jaringan LAN. Jangakauan dari jaringan MAN berkisar 10-50 km. MAN hanya memiliki satu atau dua kabel dan tidak dilengkapi dengan elemen switching yang berfungsi membuat rancangan menjadi lebih simple. Keuntungan dari Jenis Jaringan

Komputer MAN ini diantaranya adalah cakupan wilayah jaringan lebih luas sehingga untuk berkomunikasi menjadi lebih efisien, mempermudah dalam hal berbisnis, dan juga keamanan dalam jaringan menjadi lebih baik. Kerugian dari Jenis Jaringan Komputer MAN seperti lebih banyak menggunakan biaya operasional, dapat menjadi target operasi oleh para Cracker untuk mengambil keuntungan pribadi, dan untuk memperbaiki jaringan MAN diperlukan waktu yang cukup lama.

16.4.1.3 WAN Wide Area Network atau WAN merupakan jaringan yang jangkauannya mencakup daerah geografis yang luas, semisal sebuah negara bahkan benua. WAN umumnya digunakan untuk menghubungkan dua atau lebih jaringan lokal sehingga pengguna dapat berkomunikasi dengan pengguna lain meskipun berada di lokasi yang berbebeda. Keuntungan Jenis Jaringan Komputer WAN seperti cakupan wilayah jaringannya lebih luas dari Jenis Jaringan Komputer LAN dan MAN, tukar-menukar informasi menjadi lebih rahasia dan terarah karena untuk berkomunikasi dari suatu negara dengan negara yang lainnya memerlukan keamanan yang lebih, dan juga lebih mudah dalam mengembangkan serta mempermudah dalam hal bisnis. Kerugian dari Jenis Jaringan WAN seperti biaya operasional yang dibutuhkan menjadi lebih banyak, sangat rentan terhadap bahaya pencurian data-data penting, perawatan untuk jaringan WAN menjadi lebih berat.

16.4.2 B. Berdasarkan Distribusi Sumber Informasi/Data

16.4.2.1 Jaringan Terpusat Yang dimaksud jaringan terpusat adalah jaringan yang terdiri dari komputer client dan komputer server dimana komputer client bertugas sebagai perantara dalam mengakses sumber informasi/ data yang berasal dari komputer server. Dalam jaringan terpusat, terdapat istilah dumb terminal (terminal bisu), dimana terminal ini tidak memiliki alat pemroses data.

16.4.2.2 Jaringan Terdistribusi Jaringan ini merupakan hasil perpaduan dari beberapa jaringan terpusat sehingga memu-

ngkinkan beberapa komputer server dan client yang saling terhubung membentuk suatu sistem jaringan tertentu.

16.4.3 C. Berdasarkan Media Transmisi Data yang Digunakan

16.4.3.1 Jaringan Berkabel (Wired Network) Media transmisi data yang digunakan dalam jaringan ini berupa kabel. Kabel tersebut digunakan untuk menghubungkan satu komputer dengan komputer lainnya agar bisa saling bertukar informasi/ data atau terhubung dengan internet. Salah satu media transmisi yang digunakan dalam wired network adalah kabel UTP.

16.4.3.2 Jaringan Nirkabel (Wireless Network) Dalam jaringan ini diperlukan gelombang elektromagnetik sebagai media transmisi datanya. Berbeda dengan jaringan berkabel (wired network), jaringan ini tidak menggunakan kabel untuk bertukar informasi/ data dengan komputer lain melainkan menggunakan gelombang elektromagnetik untuk mengirimkan sinyal informasi/ data antar komputer satu dengan komputer lainnya. Wireless adapter, salah satu media transmisi yang digunakan dalam wireless network.

16.4.4 D. Berdasarkan Peranan dan Hubungan Tiap Komputer dalam Memproses Data

16.4.4.1 Jaringan Client-Server Jaringan ini terdiri dari satu atau lebih komputer server dan komputer client. Biasanya terdiri dari satu komputer server dan beberapa komputer client. Komputer server bertugas menyediakan sumber daya data, sedangkan komputer client hanya dapat menggunakan sumber daya data tersebut.

16.4.4.2 Jaringan Peer to Peer Dalam jaringan ini, masing-masing komputer, baik itu komputer server maupun komputer client mempunyai kedudukan yang sama. Jadi, komputer server dapat menjadi komputer client, dan sebaliknya komputer client juga dapat menjadi komputer server.

16.4.5 E. Berdasarkan Topologi Jaringan yang Digunakan

Topologi jaringan adalah bentuk perancangan baik secara fisik maupun secara logik yang digunakan untuk membangun sebuah jaringan komputer. rancangan ini sangat erat kaitannya dengan metode access dan media pengiriman yang digunakan. Topologi yang ada sangatlah tergantung dengan letak geofrapis dari masing-masing terminal, kualitas kontrol yang dibutuhkan dalam komunikasi ataupun penyampaian pesan, serta kecepatan dari pengiriman data.

16.4.5.1 Topologi Bus 16.1 Gambar ini 16.1 adalah topologi

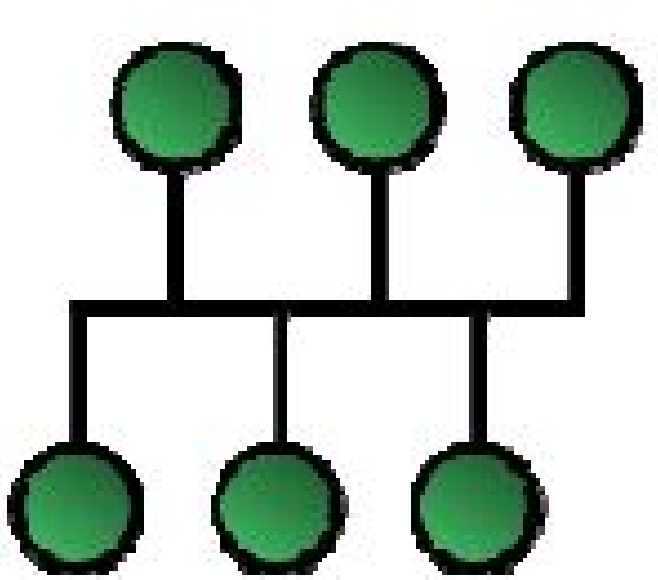


Figure 16.1 Topologi bus.

bus,berdasarkan artikel yudianto Topologi bus merupakan topologi yang banyak digunakan pada masa penggunaan kabel sepaksi menjamur[5]. Dengan menggunakan T-Connector(dengan terminator 50ohm pada ujung network), maka komputer atau perangkat jaringan lainnya bisa den-

gan mudah dihubungkan satu sama lain. Kesulitan utama dari penggunaan kabel sepaksi adalah sulit uuntuk mengukur apakah kabel sepakksi yang di gunakan benar-benar matching atau tidak. Karena kalau tidak sungguh-sungguh diukur secara benar akan merusak NIC yang digunakan dan kinerja jaringan menjadi terhambat, tidak mencapai kemampuan maksimalnya. Topologi ini juga sering digunakan pada jaringan dengan basis FO.

16.4.5.2 Topologi Star 16.2 Gambar ini 16.2 adalah topologi

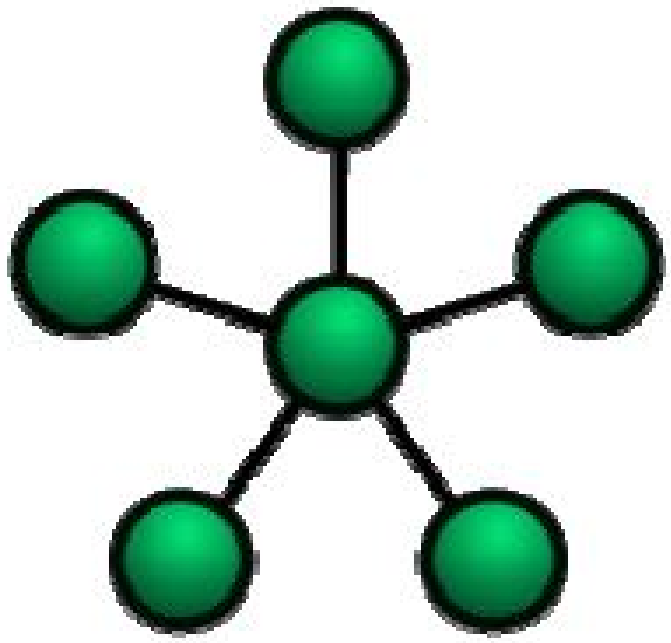


Figure 16.2 Topologi star.

star,berdasarkan artikel tudianto Topologi bintang merupakan bentuk topologi jaringan yang berupa konvergensi dari node tengah ke setiap node atau pengguna[5]. Topologi jaringan bintang termasuk topologi jaringan dengan biaya menengah.

16.4.5.3 Topologi Ring 16.3 Gambar ini 16.3 adalah topologi

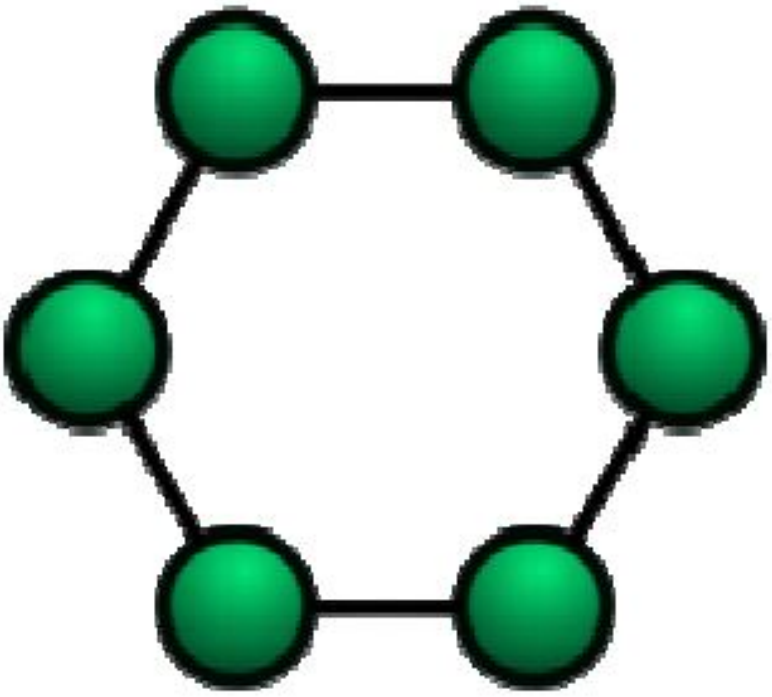


Figure 16.3 Topologi ring.

ring/cincin,berdasarkan artikel yudianto Topologi cincin adalah topologi jaringan berbentuk rangkaian titik yang masing-masing terhubung ke dua titik lainnya, sedemikian sehingga membentuk jalur melingkar membentuk cincin[5]. Pada Topologi cincin,masing - masing titik/node berfungsi sebagai repeater yang akan memperkuat sinyal disepanjang sirkulasinya, artinya masing - masing perangkat saling bekerjasama untuk menerima sinyal dari perangkat sebelumnya kemudian meneruskannya pada perangkat sesudahnya, proses menerima dan meneruskan sinyal data ini dibantu oleh TOKEN. TOKEN berisi informasi bersamaan dengan data yang berasal dari komputer sumber, token kemudian akan melewati titik/node dan akan memeriksa

apakah informasi data tersebut digunakan oleh titik/node yang bersangkutan, jika ya maka token akan memberikan data yang diminta oleh node untuk kemudian kembali berjalan ke titik/node berikutnya dalam jaringan. Jika tidak maka token akan melewati titik/node sambil membawa data menuju ke titik/node berikutnya. proses ini akan terus berlangsung hingga sinyal data mencapai tujuannya.

16.4.5.4 Topologi Mesh 16.4 Gambar ini 16.4 adalah topologi

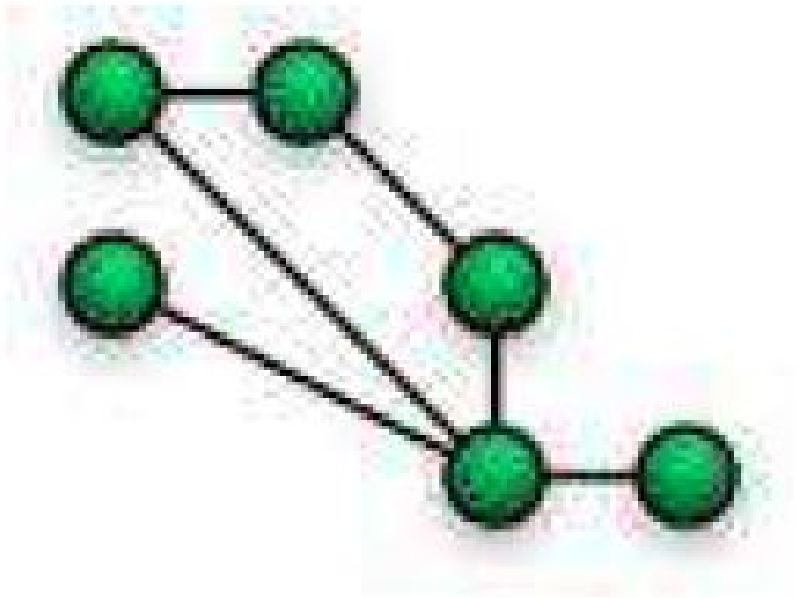


Figure 16.4 Topologi mesh.

mesh, berdasarkan artikel dari yudianto Topologi mesh adalah suatu bentuk hubungan antar perangkat dimana setiap perangkat terhubung secara langsung ke perangkat lainnya yang ada di dalam jaringan[5]. Akibatnya, dalam topologi mesh setiap perangkat dapat berkomunikasi langsung dengan perangkat yang dituju (dedicated links). Dengan demikian maksimal banyaknya koneksi antar perangkat pada jaringan bertopologi mesh ini dapat dihitung yaitu sebanyak

$$n(n-1)/2 \tag{16.1}$$

. Selain itu karena setiap perangkat dapat terhubung dengan perangkat lainnya yang ada di dalam jaringan maka setiap perangkat harus memiliki sebanyak n-1 Port Input/Output (I/O ports).Berdasarkan pemahaman di atas, dapat dicontohkan bahwa apabila sebanyak 5(lima) komputer akan dihubungkan dalam bentuk topologi mesh maka agar seluruh koneksi antar komputer dapat berfungsi optimal, diperlukan kabel koneksi sebanyak

```
5(5-1)/2 = 10
```

kabel koneksi, dan masing-masing komputer harus memiliki port I/O sebanyak 5-1 = 4 port.

16.4.5.5 Topologi Tree 16.5 Gambar ini 16.5 adalah topologi

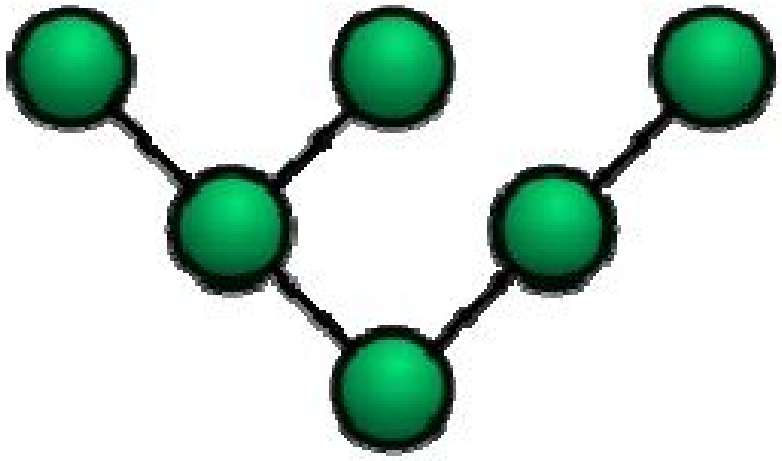


Figure 16.5 Topologi tree.

tree, berdasarkan artikel dari yudianto Topologi Pohon adalah kombinasi karakteristik antara topologi bintang dan topologi bus[5]. Topologi ini terdiri atas kumpulan topologi bintang yang dihubungkan dalam satu topologi bus sebagai

jalur tulang punggung atau backbone. Komputer-komputer dihubungkan ke hub, sedangkan hub lain di hubungkan sebagai jalur tulang punggung.Topologi jaringan ini disebut juga sebagai topologi jaringan bertingkat. Topologi ini biasanya digunakan untuk interkoneksi antar sentral dengan hirarki yang berbeda. Untuk hirarki yang lebih rendah digambarkan pada lokasi yang rendah dan semakin keatas mempunyai hirarki semakin tinggi. Topologi jaringan jenis ini cocok digunakan pada sistem jaringan komputer.Pada jaringan pohon, terdapat beberapa tingkatan simpul atau node.Pusat atau simpul yang lebih tinggi tingkatannya, dapat mengatur simpul lain yang lebih rendah tingkatannya. Data yang dikirim perlu melalui simpul pusat terlebih dahulu. Misalnya untuk bergerak dari komputer dengan node-3 kekomputer node-7 seperti halnya pada gambar, data yang ada harus melewati node-3, 5 dan node-6 sebelum berakhir pada node-7.

16.4.5.6 Topologi Linier

16.6 Gambar ini 16.6 adalah topologi

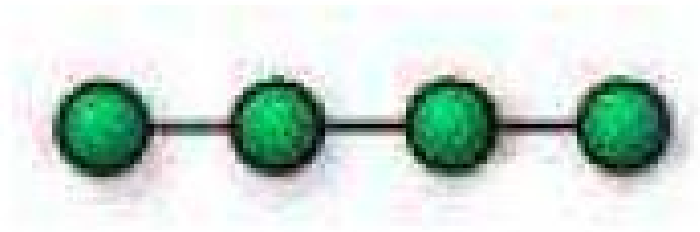


Figure 16.6 Topologi linier.

linier, Topologi ini biasa disebut dengan topologi bus beruntut, tata letak ini termasuk tata letak umum.

16.5 Alat-alat jaringan

Macam-macam alat jaringan antara lain :

Table 16.1 Kekurangan dan Kelebihan

No	Kelebihan	Kekurangan
.1	Hemat kabel	Deteksi dan isolasi kesalahan sangat kecil
.2	Tata letak kabel sederhana	Kepadatan lalu lintas tinggi
.3	Mudah dikembangkan	Keamanan dta kurang terjamin

16.5.1 ROUTER

Router adalah sebuah alat yang mengirimkan paket data melalui sebuah jaringan atau Internet menuju tujuannya, alat ini sangatlah penting untuk meneruskan jaringan satu ke jaringan lainnya yang berbeda kelas/subnet/ip. melalui sebuah proses yang dikenal sebagai routing. Proses routing terjadi pada lapisan 3 (Lapisan jaringan seperti Internet Protocol) dari stack protokol tujuh-lapis OSI. Router berfungsi sebagai penghubung antar dua atau lebih jaringan untuk meneruskan data dari satu jaringan ke jaringan lainnya. Router berbeda dengan switch. Switch merupakan penghubung beberapa alat untuk membentuk suatu Local Area Network (LAN).

16.5.2 SWITCH

Switch adalah perangkat jaringan komputer yang bekerja di OSI Layer 2, Data Link Layer. Switch kerjanya sebagai penyambung atau concentrator dalam Jaringan komputer. Switch mengenal MAC Adressing shingga dia bisa memilah paket data mana yang akan di teruskan/dilanjutkan ke mana.

16.5.3 ACCESS POINT

Access point adalah perangkat yang digunakan sebagai pembuat koneksi wireless pada jaringan komputer. Fungsi Access point diantaranya: Sebagai perangkat jaringan yang berfungsi membuat jaringan komputer tanpa kabel, atau biasa disebut WI-FI (Wireless Fidelity) Belajar Network Programming pada python, melalui fungsi-fungsi TCP/IP, SOCKET, dll.

16.6 Latihan Jaringan pada python

Pada latihan ini, kita akan mencoba mengirim data dari server menuju klien dengan menggunakan Socket pada python.

16.6.1 server.py

Penjelasan fungsi-fungsi tsb akan dijelaskan dibawah ini:

1. socket.socket(): Membuat socket baru menggunakan alamat yang sudah ada, tipe socket, dan nomor protocol.
2. socket.bind(address): Menyalin/mengikat socket ke alamat yang ada.
3. socket.listen(backlog): Menunggu koneksi yang sudah dibuat dari socket tersebut. backlog merupakan sebuah argumen yang menyatakan batas maximal nomor antrian koneksi dan paling tidak sampai dengan 0; nilai maximum tergantung dari sistem(biasanya 5), dan nilai minimumnya harus mencapai 0.
4. socket.accept(): Nilai yang dikembalikan atau diberikan adalah sepasang(conn, address) dimana conn adalah socket baru yaitu sebuah objek yang biasa digunakan untuk mengirim dan menerima data dari koneksi tersebut dan address adalah alamat yang terikat ke socket pada akhir koneksi.
5. socket.send(bytes[, flags]): Mengiri data ke socket. Socket harus terkoneksi oleh remote Socket. mengembalikan angkat dari bytes yang terkirim. Aplikasi yang bertugas untuk mengecek semua data harus terkirim; hanya jika data ditransimisikan, aplikasi membutuhkan usaha untuk mengirimkan data yang tersisa.
6. socket.close(): Menandakan bahwa socket telah ditutup. semua dari operasi-operasi pada objek socket akan gagal. Remote End tidak akan menerima data lagi (sampai data telah dibersihkan). Socket-socket secara otomatis

tertutup ketika dilakukan garbage-collected, tetapi lebih baik untuk close() mereka secara eksplisit.

Sebelumnya pesan diatas tidak akan muncul sebelum kita menjalankan script client.py pada tab terminal lain. maka setiap kali kita menjalankan script client.py akan terus mengirimkan pesan kepada server maupun client.

contoh networking:

```
192.168.0.0/24 subnet mask 255.255.255.0, 256 IP rumus-nya

(jumlah IP network diatas / 2) + (subnetmask diatasnya)= subne

(256/2) + n = 128, jadi subnet mask buat 128 ip adalah 255.255
(128/2) + 128 = 192, jadi subnetmask buat 64 ip adalah 255.255
(64/2) + 192 = 224, jadi subnet mask buat 32 ip adalah 255.255
(32/2) + 224 = 240, jadi subnet mask buat 16 ip adalah 255.255
(16/2) + 240 = 248, jadi subnet mask buat 8 ip adalah 255.255.
(8/2) + 248 = 252, jadi subnet mask buat 4 ip adalah 255.255.2
(4/2) + 252 = 254, jadi subnet mask buat 2 ip adalah 255.255.2
(2/2) + 254 = 255, jadi subnet mask buat 1 ip adalah 255.255.2
```

CHAPTER 17

DATABASES ACCESS

17.1 Database Access

17.1.1 Pengertian Database

Basis data adalah sekumpulan dari data yang telah disusun sesuai dengan aturan tertentu yang saling berhubungan sehingga memudahkan pengguna dalam mengelolanya juga memudahkan pengguna untuk memperoleh informasi. Selain itu ada juga yang menyebutkan bahwa database sebagai kumpulan le,tabel,atau arsip yang saling terhubung yang disimpan dalam media elektronik. Istilah ”basis data” berawal dari ilmu komputer. Meskipun kemudian artinya semakin luas , memasukkan hal-hal di luar bidang elektronika, artikel ini mengenai basis data komputer. Catatan yang mirip dengan basis data sebenarnya sudah ada sebelum revolusi industri yaitu dalam bentuk buku besar, kuitansi dan kumpulan

data yang berhubungan dengan bisnis. Jadi Database dapat disimpulkan adalah kumpulan informasi yang berkaitan dengan subjek atau tujuan tertentu , seperti pelacakan pesanan pelanggan atau menyimpan koleksi musik. Jika database tidak disimpan di komputer, atau hanya bagian dari dokumen tersebut, Anda dapat melacak semua jenis informasi dari berbagai sumber yang harus diseimbangkan dan ditata .

17.1.2 Manfaat Penggunaan Database

1. Kecepatan dan Kemudahan Database memiliki kemampuan dalam menyeleksi data sehingga menjadi suatu kelompok yang tersusun dengan cepat. Hal inilah yang akhirnya dapat menghasilkan informasi yang dibutuhkan secara cepat pula. Seberapa cepat pemrosesan data oleh database tergantung pada perancangan databasenya.

2. Pemakaian Bersama-sama Suatu database bisa digunakan oleh siapa saja dalam suatu perusahaan. Sebagai contoh database mahasiswa dalam suatu perguruan tinggi dibutuhkan oleh beberapa bagian, seperti bagian admin, bagian keuangan, bagian akademik. Kesemua bidang tersebut membutuhkan database mahasiswa namun tidak perlu masing-masing bagian membuat databasenya sendiri, cukup database mahasiswa satu saja yang disimpan di server pusat. Nanti aplikasi dari masing-masing bagian bisa terhubung ke database mahasiswa tersebut.

3. Kontrol data terpusat Masih berkaitan dengan point ke dua, meskipun pada suatu perusahaan memiliki banyak bagian atau divisi tapi database yang diperlukan tetap satu saja. Hal ini mempermudah pengontrolan data seperti ketika ingin mengupdate data mahasiswa, maka kita perlu mengupdate semua data di masing-masing bagian atau divisi, tetapi cukup di satu database saja yang ada di server pusat.

4. Menghemat biaya perangkat Dengan memiliki database secara terpusat maka di masing-masing divisi tidak

memerlukan perangkat untuk menyimpan database berhubung database yang dibutuhkan hanya satu yaitu yang disimpan di server pusat, ini tentunya memangkas biaya pembelian perangkat.

5. Keamanan Data Hampir semua Aplikasi manajemen database sekarang memiliki fasilitas manajemen pengguna. Manajemen pengguna ini mampu membuat hak akses yang berbeda-beda disesuaikan dengan kepentingan maupun posisi pengguna. Selain itu data yang tersimpan di database diperlukan password untuk mengaksesnya.

6. Memudahkan dalam pembuatan Aplikasi baru Dalam poin ini database yang dirancang dengan sangat baik, sehingga si perusahaan memerlukan aplikasi baru tidak perlu membuat database yang baru juga, atau tidak perlu mengubah kembali struktur database yang sudah ada. Sehingga Si pembuat aplikasi atau programmer hanya cukup membuat atau pengatur antarmuka aplikasinya saja.

Dengan segudang manfaat dan kegunaan yang dimiliki oleh database maka sudah seharusnya semua perusahaan baik itu perusahaan skala kecil apalagi perusahaan besar memiliki database yang dibangun dengan rancangan yang baik. Salah satu manfaat database adalah untuk memudahkan dalam mengakses data.Kemudahan pengaksesan data ini adalah sebagai implikasi dari keteraturan data yang merupakan syarat mutlak dari suatu database yang baik. Ditambah dengan pemanfaatan teknologi jaringan komputer maka manfaat database ini akan semakin besar. Penggunaan database sekaligus teknologi jaringan komputer telah banyak digunakan oleh berbagai macam perusahaan,contohnya saja perbankan yang memiliki cabang di setiap kotanya. Perusahaan Bank tersebut hanya memiliki satu database yang disimpan di server pusat, sedangkan cabang-cabangnya terhubung melalui jaringan komputer untuk mengakses database yang terletak di sever pusat tersebut.

17.1.3 Pengertian Character

Charakter merupakan bagian data yang terkecil, dapat berupa karakter numerik, huruf ataupun karakter-karakter khusus (special characters) yang membentuk suatu item data atau field.

17.1.4 Apa yang dimaksud dengan Field, Record, Table, File, Data dan Basisdata

Field merupakan kumpulan dari karakter yang membentuk satu arti, maka jika terdapat field misalnya seperti KeteranganBarang atau JumlahBarang, maka yang dimunculkan dalam field tersebut harus yang berkaitan dengan keterangan barang dan jumlah barang. Atau field juga bisa disebut sebagai tempat atau kolom yang terdapat dalam suatu tabel untuk mengisikan nama-nama (data) field yang akan di isikan. merepresentasikan suatu atribut dari record yang menunjukkan suatu item dari data, seperti misalnya nama, alamat dan lain sebagainya. Kumpulan dari field membentuk suatu record.

Record merupakan kumpulan field yang sangat lengkap, dan biasanya dihitung dalam satuan baris. Tabel merupakan kumpulan dari beberapa record dan juga field.

File terdiri dari kumpulan field yang menunjukan dari satu kesatuan data yang sejenis. Misalnya seperti file nama siswa berisikan data tentang semua nama siswa yang ada. Data adalah kumpulan fakta atau kejadian yang digunakan sebagai penyelesaian masalah dalam bentuk informasi. Basis data (database) terdiri dari dua kata, yaitu kata basis dan data. Basis merupakan tempat ataupun gudang, ataupun wadah.

Data dapat disebut sebagai kumpulan dari fakta yang mewakili objek, misalnya seperti benda, manusia,barang dan sebagainya yang ditulis ke dalam bentuk angka, huruf, simbol, bunyi, teks, gambar ataupun gabungannya. Jadi dapat disimpulkan bahwa basis data merupakan kumpulan dari data-datayang terorganisasi yang saling berhubungan sedemikian rupa sehingga dapat dengan mudah disimpan,

dimanipulasi, dan dipanggil oleh pemakainya. Karakter atau character yang ada didalam database merupakan bagian data yang terkecil, karakter tersebutdapat berupa karakter numerik, huruf ataupun karakter khusus (special characters) yang membentuk suatu item data atau field.

17.1.5 Sifat-sifat database atau basis data

Data dalam basis data bersifat integrated dan shared Terpadu (integrated), berkas-berkas data yang ada pada basis data saling terkait (terjadi dependensi data); Berbagi data (shared), data yang sama dapat dipakai oleh sejumlah pengguna dalam waktu yang bersamaan. Sering dinamakan sebagi sistem multiuser.

- Internal : Kesatuan (integritas) dari file-file yang terlibat
- Terbagi atau share : Elemen-elemen database dapat dibagikan pada para user baik secara sendiri-sendiri maupun secara serentak dan pada waktu yang sama (concurrent sharing).

17.1.6 Tipe Database

Tipe Database Terdapat 12 tipe database, antara lain:

1. Operational database: Database ini menyimpan data rinci yang diperlukan untuk mendukung operasi dari seluruh organisasi. Mereka juga disebut subject-area databases (SADB), transaksi database, dan produksi database. Contoh: database pelanggan, database pribadi, database inventaris, akuntansi database.
2. Analytical database: Database ini menyimpan data dan informasi yang diambil dari operasional yang dipilih dan eksternal database. Mereka terdiri dari data dan informasi yang dirangkum paling dibutuhkan oleh sebuah organisasi manajemen dan End-user lainnya. Beberapa orang menyebut analitis multidimensi database sebagai database, manajemen database, atau informasi database.
3. Data warehouse: Sebuah data warehouse menyimpan data dari saat ini dan tahun- tahun sebelumnya - data

yang diambil dari berbagai database operasional dari sebuah organisasi.

4. Distributed database: Ini adalah database-kelompok kerja lokal dan departemen di kantor regional, kantor cabang, pabrik-pabrik dan lokasi kerja lainnya. Database ini dapat mencakup kedua segmen yaitu operasional dan user database, serta data yang dihasilkan dan digunakan hanya pada pengguna situs sendiri.
5. End-user database: Database ini terdiri dari berbagai file data yang dikembangkan oleh end-user di workstation mereka. Contoh dari ini adalah koleksi dokumen dalam spreadsheet, word processing dan bahkan download file.
6. External database: Database ini menyediakan akses ke eksternal, data milik pribadi online - tersedia untuk biaya kepada pengguna akhir dan organisasi dari layanan komersial. Akses ke kekayaan informasi dari database eksternal yang tersedia untuk biaya dari layanan online komersial dan dengan atau tanpa biaya dari banyak sumber di Internet.
7. Hypermedia databases on the web: Ini adalah kumpulan dari halaman-halaman multimedia yang saling berhubungan di sebuah situs web. Mereka terdiri dari home page dan halaman hyperlink lain dari multimedia atau campuran media seperti teks, grafik, gambar foto, klip video, audio dll.
8. Navigational database: Dalam navigasi database, queries menemukan benda terutama dengan mengikuti referensi dari objek lain.
9. In-memory databases: Database di memori terutama bergantung pada memori utama untuk penyimpanan data komputer. Ini berbeda dengan sistem manajemen database yang menggunakan disk berbasis mekanisme penyimpanan. Database memori utama lebih cepat daripada dioptimalkan disk database sejak Optimasi algoritma internal menjadi lebih sederhana dan lebih sedikit CPU mengeksekusi instruksi.

10. Document-oriented databases: Merupakan program komputer yang dirancang untuk aplikasi berorientasi dokumen. Sistem ini bisa diimplementasikan sebagai lapisan di atas sebuah database relasional atau objek database. Sebagai lawan dari database relasional, dokumen berbasis database tidak menyimpan data dalam tabel dengan ukuran seragam kolom untuk setiap record. Sebaliknya, mereka menyimpan setiap catatan sebagai dokumen yang memiliki karakteristik tertentu. Sejumlah bidang panjang apapun dapat ditambahkan ke dokumen. Bidang yang dapat juga berisi beberapa bagian data.
11. Real-time databases Real-time: Database adalah sistem pengolahan dirancang untuk menangani beban kerja negara yang dapat berubah terus- menerus. Ini berbeda dari database tradisional yang mengandung data yang terus-menerus, sebagian besar tidak terpengaruh oleh waktu.
12. Relational Database: Database yang paling umum digunakan saat ini. Menggunakan meja untuk informasi struktur sehingga mudah untuk mencari.

17.1.7 Modul python untuk mengakses database MySQL

Untuk mengakses database MySQL dari Python, berikut adalah beberapa langkah sederhana. Yang pertama, server database MySQL harus siap dulu. Karenanya lakukan tahapan berikut ini.

1. Instal server mysql dengan menjalankan perintah sudo apt-get install mysqlserver. Jangan lupa memasukkan password akun root untuk server MySQL.
2. Siapkan database dan tabel. Jalankan perintah mysql -u root -p dari terminal. Selanjutnya kita akan buat tabel yang skemanya seperti berikut ini (hanya ilustrasi saja).

$$mysql > createdatabaseteman; \tag{17.1}$$

mysql¿ use teman; mysql¿ create table alamat(id int not null auto increment primary key, nama varchar(35), alamat text, telepon varchar(15), surat text);

3. ingin membuat user baru yang punya akses penuh ke database yang baru saja dibuat, lakukan tahapan berikut ini. mysql¿ create user 'andri'@'localhost' identified by '123456'; mysql¿ grant all on teman.* to 'andri'@'localhost' with grant option;
4. Langkah selanjutnya adalah membuat modul python untuk mengakses database tersebut. Untuk kasus ini, modul hanya diberi kemampuan untuk melihat seluruh isi tabel, sehingga tabel sebaiknya diisi dulu. Sedangkan kemampuan untuk melakukan operasi update dan delete dapat dibangun menggunakan pola yang sama. Modul tersebut seperti code berikut.

Dalam modul python ini terdiri dari dua fungsi, masing-masing sambung dan selectall. Fungsi pertama membutuhkan parameter terkait nama server, dan akun user serta mengembalikan variabel koneksi. Sedangkan fungsi selectall membutuhkan parameter koneksi yang diperoleh dari fungsi sambung, serta nama database dan nama tabel. Fungsi ini mengembalikan list yang berisi setiap row dalam tabel untuk kebutuhan lain yang belum terdefinisi dalam modul ini. Berikut Contoh selectall dalam python ;

```
def sambung (host,user,passwd):
mycon=MySQLdb.connect(host,user,passwd)
return mycon

def selectall(mycon, dbname, table):
mycur=mycon.cursor()
mycur.execute(use  + dbname)
mycur.execute(select * from  + table)
rows=mycur.fetchall()
a=[]
for i in rows:
nama=i[1]
alamat=i[2]
telepon=i[3]
surat=i[4]
print(nama +   + alamat +   + telepon +   + surat)
```

```
a.append(i)
return a
```

Lalu, bagaimana menggunakannya? Untuk sementara, modul python ini hanya dapat diakses melalui shell python karena belum ada fungsi main yang terdefinsi. Hal ini disebabkan karena rancangan input/ouput masih seadanya. Karenanya, mari masuk ke shell python dengan menjalankan perintah python di terminal. Yang perlu diperhatikan, penggunaan shell python harus dilakukan dari directory di mana modul ini disimpan. Berikut adalah gambaran ketika berada dalam shell python dan menggunakan modul ini.

```
>>> from mymodul import *
>>> mycon=sambung(localhost,andri,123456)
>>> a=selectall(mycon,teman,alamat)
Andri Jl. Sariasih, Sarijadi, Bandung 12450 08123456789 andri@
>>>
```

Di baris pertama, kita import modul dari nama file, untuk selanjutnya import semua fungsi yang ada di dalamnya. Selanjutnya, kita membuat variabel bernama mycon bertipe koneksi ke MySQL (dapat dilihat dengan cara menjalankan perintah type(mycon) dari shell python) dan meng-assigned nilainya dari memanggil fungsi sambung. Selanjutnya, variabel a di-assinged nilainya dari memanggil fungsi selectall. Terlihat bahwa fungsi selectall mencetak nilai yang diperoleh dari operasi select tabel.

Dengan ilustrasi ini, diharapkan dapat memberi inspirasi dalam membuat modul python yang digunakan untuk sebuah aplikasi, misalnya dengan menjadikannya sebagai bagian dari hubungan SIGNAL-SLOT pada QT4. Semoga bermanfaat.

Dalam era informasi dimana kita hidup sekarang, kita dapat melihat seberapa banyak data dunia berubah. Kita pada dasarnya membuat, menyimpan, dan menarik data, secara ekstensif. Harusnya ada sebuah cara untuk menangani semua itu. Semua itu tidak dapat disebarkan kemana-mana tanpa adanya manajemen bukan? Di sini hadir Database Management System (DBMS). DBMS adalah sebuah sistem

software yang memungkinkanmu untuk membuat, menyimpan, memodifikasi, menarik, dan penanganan lainnya terhadap sebuah data dari database. Sistem ini juga bervariasi dalam ukuran, mulai dari sistem kecil yang cukup berjalan pada komputer personal hingga yang lebih besar yang berjalan dalam mainframe.

17.1.8 Python Database API

Python dapat berinteraksi dengan database. Namun, bagaimana cara melakukannya? Python menggunakan apa yang disebut Python Database API dengan tujuan untuk menjadi antarmuka dengan database. API ini mengijinkan kita untuk memprogram database management system (DBMS) yang berbeda. Untuk DBMS yang berbeda itu, bagaimana pun juga, proses yang diikuti pada tingkatan kode tetap sama, yaitu sebagai berikut:

1. Membangun sebuah koneksi ke database pilihanmu. Kita bisa membuat function koneksi, kemudian dipanggil dari file atau script atau halaman lain. Hanya menyediakan fungsi koneksi saja.

2. Kita membuat class koneksi, kemudian dipanggil dari file atau script atau halaman lain. Menyediakan fungsi-fungsi dasar eksekusi perintah ke database seperti execute, mengambil data dan lainnya.

3. Memanipulasi data menggunakan SQL (berinteraksi).

4. Memberitahu koneksi untuk entah menerapkan manipulasi SQL ke data dan membuatnya permanen (commit), atau memberitahunya untuk meninggalkan manipulasi itu (rollback), sehingga mengembalikan data ke keadaan sebelum interaksi terjadi.

5. Kita membuat library php yang isinya koneksi kemudian tinggal pakai dalam aplikasi. Menyediakan fitur-fitur lengkap tentang koneksi ke database, bahkan menyediakan berbagai pilihan jenis database.

Bagaimana caranya menampilkan data dari mysql menggunakan python. Saya asumsikan teman-teman sudah menginstall web server (XAMPP/yang lainnya) dan python di komputer masing-masing. Setelah itu semua siap sekarang silahkan download mysql connector untuk python disini (Sesuaikan dengan versi python yang kalian punya). Setelah itu tinggal install. Setelah selesai installasi, buka phpmyadmin dan buat database dengan nama terserah. Contohnya yaitu trial Lalu import sql berikut trial.sql Setelah di impor, lalu buka teks editor dan copas-kan code berikut showdata.py lalu simpan dengan nama terserah kalian (punya saya show-data.py) dan tinggal kalian run dari command prompt.

17.1.9 Koneksi Database MySQL dengan Python

1. Pastikan kita sudah menginstal package libmysqlclient-dev dengan sebagai berikut: apt-get install libmysqlclient-dev
2. Download modul MySQLdb.
3. Instal modul tersebut ini dengan cara (Pastikan saat login sebagai root ya):

```
$ gunzip MySQL-python-1.2.2.tar.gz
$ tar -xvf MySQL-python-1.2.2.tar
$ cd MySQL-python-1.2.2
$ python setup.py build
$ python setup.py install
```

4. Buat sebuah script seperti berikut (dengan contoh: test.py):

```
import MySQLdb
# Open database connection
db = MySQLdb.connect(localhost,root,password,nama database )
# prepare a cursor object using cursor() method
cursor = db.cursor()
# execute SQL query using execute() method.
```

```
cursor.execute(SELECT VERSION())
# Fetch a single row using fetchone() method.
data = cursor.fetchone()
print Database version : %s  % data
# disconnect from server
db.close()
```

5. Test dengan code berikut: python test.py

17.1.10 Connection Class

Metode ini menggunakan kelas untuk menginisialisasi koneksi pada database. Pada contoh ini hanya menggunakan kelas sebagai inisiator untuk menghubungkan ke database, untuk querynya tetap menggunakan metode biasa.

1. pertama kita membuat kelas koneksi, buatlah file koneksi dengan nama connection.py dan masukan coding berikut :

```
import MySQLdb as mysql
import hashlib
import sys
import warnings
class MysqlUserDB:
     # initialization Connection Database
     # Init Start
     warnings.filterwarnings('error')
     def __init__(self, DBrootHost, DBrootUser, DBrootPass,
         self.DBrootHost = DBrootHost
         self.DBrootUser = DBrootUser
         self.DBrootPass = DBrootPass
         self.DBrootDatabase = DBrootDatabase
         try:
             print("Checking connection of MYSQL ...")
             self.con = mysql.connect(DBrootHost, DBrootUser
             self.cursor = self.con.cursor()
             self.cursor.execute('Select version()')
             print("Connected to Mysql Database\n")
         # except mysql.Error as error:
```

```
            #    print("Error %s\n Stop.\n" % error)
            #    sys.exit()
            except Warning as warn:
                print("Warning", warn)
        def CreateDB(self, DBrootDatabase):
            print("Creating database...")
            try:
                self.cursor.execute('CREATE database if NOT exi
                self.cursor.execute("SHOW DATABASES LIKE %s", (
                dbs = self.cursor.fetchone()
                print("Database created: ", dbs[0])
            except Warning as warn:
                print("Warning: %s \nStopping Process.\n" % war
                sys.exit()
        def GrantsAccess(self, DBrootDatabase):
            print("Accessing Account ...")
            try:
                self.cursor.execute("SHOW DATABASES LIKE %s", (
                result = self.cursor.fetchone()
                print("Access Granted for Database", result[0])
            except Warning as warn:
                print("Warningg %s" % warn)
        def getDB(self):
            return self.cursor
        def delCon(self):
            print("Finishing operation ...")
            self.cursor.close()
            self.con.close()
            print("Finished")
# Init End
```

Dalam kelas fungsi diatas menginisialisasi paramater untuk koneksi yang dinisialisasi pada method init, Sedangkan untuk me-return hasil koneksi menggunakan method getDB hal ini untuk mereturn cursor untuk dipanggil saat akan melakukan eksekusi, dan untuk memutuskan koneksi menggunakan method delcon, sedangkan sisa method lainnya hanya sebagai method pendukung.

2. Setelah itu, buatlah file baru untuk mengetes apakah koneksi berhasil dan melakukan query. buatlah file menggunakan nama connectionTes.py

```
import connection as dbs
import warnings
warnings.filterwarnings('error')
mysqli = dbs.MysqlUserDB(DBrootHost='127.0.0.1', DBrootUser=
                         DBrootPass='', DBrootDatabase='pos
db = mysqli.getDB()
try:
     db.execute("select * from product")
     results = db.fetchall()
     for result in results:
         print(result)
except db.Error as error:
    print(error)
mysqli.delCon()
```

Dalam hal ini mengimport kelas connection sebagai db. Setelah itu di inisialisasi dengan variabel baru bernama mysqli. dan membuat object cursor pada db dengan memanggil object kelas mysql.getDB()

Table 17.1 Arsitektur Database

No	Keterangani
.1	Database Connection
.2	SQL Statements
.3	Result Set
.4	Database Metadata
.5	Prepared Statement

17.1:

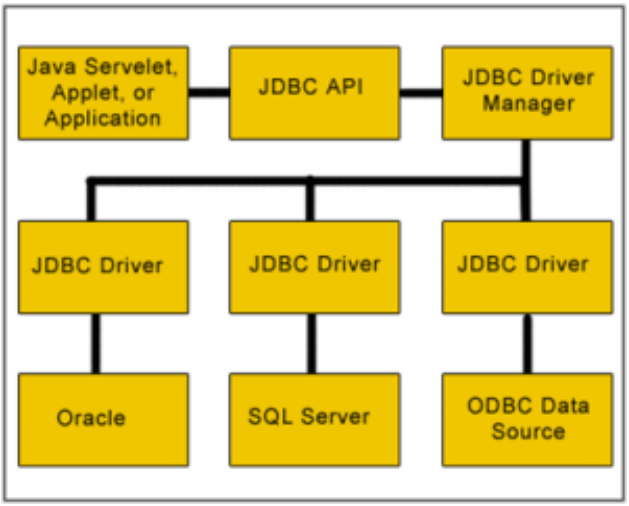


Figure 17.1 Arsitektur

CHAPTER 18

SENDING EMAIL

18.1 Server Pada Mail Server dan Penjelasannya

Pada mail server terdapat 2 server yang berbeda yaitu :

1. Outgoing Server (Sending email) : Protocol server yang menangani adalah SMTP (Simple Mail Transfer Protocol) pada port 25.
2. Incoming Server (Receiving email) : Protocol server yang menangani adalah POP3 (Post Office Protocol) pada port 110 atau IMAP (Internet Message Access Protocol) pada port 143.

Penjelasan dari Server yang menangani outgoing email dan incoming email sebagai berikut

1. SMTP Server : Saat anda mengirimkan email maka email anda akan ditangani SMTP Server dan akan

dikirim ke SMTP Server tujuan, baik secara langsung maupun melalui beberapa SMTP Server dijalurnya. Apabila server tujuan terkoneksi maka email akan dikirim, namun apabila tidak terjadi koneksi maka akan dimasukan ke dalam queue dan di resend setiap 15 menit, apabila dalam 5 hari tidak ada perubahan maka akan diberikan undeliver notice ke inbox pengirim.

2. POP3 Server : Jika menggunakan POP3 Server, apabila kita akan membaca email maka email pada server di download sehingga email hanya akan ada pada mesin yang mendownload email tersebut (kita hanya bisa membaca email tersebut pada device yang mendownload email tersebut).
3. IMAP Server : Jika menggunakan IMAP Server, email dapat dibuka kembali lewat device yang berbeda. Fungsinya adalah mengelola email yang disimpan di server, kemudian email tersebut di ambil oleh client, selain itu IMAP juga meneruskan packet data. Kemampuan ini jauh lebih baik daripada POP (Post Office Protocol) yang hanya memperbolehkan kita mengambil/download semua pesan yang ada tanpa kecuali. IMAP adalah suatu protokol yang umum digunakan untuk pengiriman surat elektronik atau email di Internet. Protokol ini gunakan untuk mengirimkan data dari komputer pengirim surat elektronik ke server surat elektronik penerima. Untuk menggunakan SMTP bisa dari Microsoft Outlook. biasanya untuk menggunakan SMTP di perlukan settingan :

- Email Address : contoh;

  1. anda@domainanda.com
  2. Incoming Mail (POP3, IMAP or HTTP) server : mail.domainanda.com
  3. Outgoing (SMTP) server : mail.domainanda.com
  4. Account Name : anda@domainanda.com
  5. Password : password yang telah anda buat sebelumnya

Pada ilustrasi diatas Siti memiliki alamat email siti@a.id menulis email nya di komputer menggunakan Thunderbird atau Evolution. Pada kolom To: dia ketikkan alamat tujuan yaitu hendra@b.id. Setelah siti menekan tombol send, email yang dikirim langsung menuju ke mesin SMTP server milik ISP 1 yang bernama smtp.a.id Pada server smtp.a.id menerima email dari siti (siti@a.id) yang ditujukan kepada hendra (hendra@b.id). Server mengecek smtp.a.id mencek alamat email tujuan yaitu hendra.@b.id. Mesin server smtp.a.id membutuhkan informasi ke server mana email untuk mesin.b.id harus ditujukan. Untuk memperoleh informasi tersebut tentang domain b.id. Kemudian pada mesin Name Server ns.b.id memberitahukan mesin smtp.a.id bahwa semua email yang ditujukan kepada b.id harus dikirim kepada mesin smtp.b.id.Setelah memperoleh jawaban dari ns.c.id bahwa email harus dikirm ke mesin smtp.b.id maka mesin smtp.a.id berusaha untuk menghubungi mesin smtp.b.id.Setelah mesin smtp.b.id berhasil dihubungi, mesin smtp.a.id mengirimkan teks email dari Siti (siti@a.id) yang ditujukan kepada Hendra(henra@b.id) ke mesin smtp.b.id Hendra (hendra@b.id)yang sedang menjalankan perangkat lunak pembaca email dan mengambil email tersebut dari eail server smtp.b.id barulah email dari Siti (siti@a.id) dapat diunduh melalui PC hendra dan di tampilkan isi emailnya. E-mail disampaikan oleh mail client (MUA, mail user agent) ke mail server (MSA, mail submission agent) menggunakan SMTP pada port 587 atau menggunakan traditional port 25. Dari sini, MSA mengirim mail tersebut ke mail transfer agent miliknya (MTA, mail transfer agent). MTA batas harus menemukan host target, dengan menggunakan DNS untuk mencari mail exchange record (MX record) untuk domain penerima. MX record yang kembali berisi nama dari host target. MTA selanjutnya menghubungkan ke exchange server sebagai SMTP client. Ketika MX target menerima pesan yang masuk, akan ditangani oleh mail delivery agent (MDA) untuk pengiriman pesan secara local. Analisis: Saat PC siti diberi perintah mengirim email ke PC Hendra, kemudian email tersebut terlebih dahulu masuk ke server net-

work dimana dia berada server 1(smtp.a.id), disini server dapat melakukan kegiatan sniffing, Pada server sebelumnya sudah saling terkoneksi dan mendapat authentifikasi dari antar server untuk meneruskan paket email yang akan dikirim protokol yang bekerja pada tahap ini adalah SMTP, kemudian email masuk pada server2 (smtp.b.id).Untuk selanjutnya email dikirim ke PC Hendra (PC Destination) pada tahap ini protokol yang bekerja adalah protokol IMAP. Sehingga dari ilustrasi yang diberikan dapat menggambarkan proses pengiriman email,dan apa saja yang terjadidalam prosesnya. Pada proses pengiriman email terjadi kegiatan sniffing yang dilakukan oleh server. Sniffing adalah kegiatan pengendusan traffic data packet pada suatu jaringan. Selain itu Prinsip kerja dan Porses Pengiriman Email, email juga dibedakan berdasarkan format isinya, yakni sebagai berikut

- Plain Text Email adalah jenis email yang sisanya diformat menggunakan sistem America Standart Code for Information Interchange (ASCII). Tulisan yang dibuat dengan format ini tidak dapat dimodifikasi seperti warna, ukuran jenis font dan lain sebagainya, Tidak ada pengolahan atau penambahan aksesoris.
- HTML Email adalah bahasa standar yang digunakan untuk mengatur tampilan informasi di Internet. Email yang menggunakan format ini umumnya dapat disesuaikan dengan selera pengirimnya, Dengan begitu email tersebut dapat ditambahkan macam-macam aksesoris seperti; penggantian jenis font, warna font dan juga besaran font pada tiap bagian surat.

18.1.1 Apa Itu Port ?

Port adalah socket atau jack koneksi yang terletak di luar unit sistem sebagai tempat kabel - kabel yang berbeda ditancapkan. Port berfungsi untuk mentransmisikan data. Berikut macam - macam port :

1. Port Serial adalah port seri merupakan sebuah port pada personal computer yang berfungsi untuk mentransmisikan satu bit informasi pada satu satuan waktu.

2. Port Pararel adalah sebuah port pada personal computer yang berfungsi sebagai alat komunikasi komputer (motherboard) dengan perangkat luar yang bersifat paralel.
3. Port SCSI (Scuzzy) adalah port berkinerja tinggi yang didefinisikan oleh American National Standart Institute yang digunakan untuk menangani perangkat input/output atau perangkat media penyimpanan.
4. Port USB adalah suatu teknologi yang memungkinkan untuk menghubungkan alat eksternal (peripheral) seperti scenner, printer, mouse, dan perangkat lainnya ke komputer.

18.1.2 Cara Kerja Mail Server (singkat)

Cara kerja mail server mempunyai berbagai macam versi penjelasan mengenai cara kerjanya, dalam artikel ini saya akan menjelaskan 2 versi cara kerja mail server yang sudah saya rangkum dari berbagai sumber. Sebenarnya cara kerja antara versi 1 dan 2 mempunyai inti yang sama, hanya saja penjelasannya yang beda, silahkan anda pilih yang mana.

- Cara Kerja Mail Server Versi 1 : Proses pengiriman e-mail malalui tahapan yang sedikit panjang. Saat e-mail di kirim, maka e-mail tersebut disimpan pada mail server menjadi satu file berdasarkan tujuan e-mail. File ini berisi informasi sumber dan tujuan, serta dilengkapi tanggal dan waktu pengiriman. Pada saat user membaca e-mail berarti user telah mengakses server e-mail dan membaca file yang tersimpan dalam server yang di tampilkan melalui browser user
- Cara Kerja Mail Server Versi 2: Cara kerja ini saya ambil dari Xmodulo, sebelum memahami proses cara kerja mail server sebaiknya anda mengenal terlebih dahulu singkatan - singkatan dari MUA, MTA, MDA dll. Berikut penjelasannya :

  1. Mail User Agent (MUA) : MUA adalah komponen yang berinteraksi dengan pengguna akhir secara lang-

sung. Contoh dari MUA yaitu Thunderbird, MS Outlook, Zimbra Desktop. Interface webmail seperti Gmail.

2. Mail Transfer Agent (MTA) : MTA bertanggung jawab untuk mentransfer email dari mail server mengirimkan sampai ke server penerima email. Contoh MTA yaitu sendmail dan postfix
3. Mail Delivery Agent (MDA) : Dalam surat server tujuan, MTA lokal menerima email masuk dari MTA yang paling jauh. Email tersebut kemudian dikirimkan ke kotak surat pengguna dengan MDA.
4. POP / IMAP : POP dan IMAP adalah protokol yang digunakan untuk mengambil email dari kotak surat penerima server untuk penerima MUA.
5. Mail Exchanger Record (MX) : Record MX adalah entri DNS untuk mail server. Catatan ini menunjuk ke alamat IP ke arah mana email harus ditembak. MX record terendah selalu menang, yaitu mendapat prioritas tertinggi. Sebagai contoh, MX 10 adalah lebih baik daripada MX 20, alamat IP dari MX record dapat bervariasi berdasarkan desain dan konfigurasi persyaratan. seperti yang akan dibahas nanti dalam artikel. Ketika pengirim mengklik tombol kirim, SMTP (MTA) memastikan ujung ke ujung pengiriman email dari pengirim-sisi server ke server tujuan. Setelah mencapai server tujuan, MTA lokal ke server tujuan menerima email, dan di pindahkan ke MDA setempat. MDA kemudian menulis email ke kotak pesan penerima. Ketika penerima memeriksa email, mereka diambil oleh MUA dengan menggunakan protokol seperti POP atau IMAP.

18.2 Sending Email

Mail Server adalah perangkat lunak program yang mendistribusikan le atau informasi sebagai respons atas permintaan yang dikirim via email, mail server juga digunakan-

padabitnetuntukmenyediakanlayananserupaftp. Selainitu-mailserverjuga dapat dikatakan sebagai aplikasi yang digunakan untuk penginstalan email.

Tak hanya sebagai sebuah program mail server juga bisa berupa sebuah komputer yang memang dikhususkan untuk menjalankan sebuah aplikasi perangkat lunak program ini. nah komputer ini di ibaratkan sebagai jantung dari system sebuah email. Program ini biasanya dikelola oleh programer yang disebut dengan post master.

Mail server ini dikelola oleh seorang post master yang memiliki beberapa tugas pokok yaitu mengelola kaun, memonitor bagaimana kinerja server dan melaksanakan tugas administrative lainnya. Biasanya program ini menggunakan protocol antara lain smtp, pop3 dan imap.

Cara Kerja Mail Server

Setelah kamu tahu apa itu mail server, kini saatnya kamu tahu bagaimana mail server bekerja. Pada dasarnya ada dua cara kerja program ini. pertama, proses pengiriman email akan melewati tahapan yang agak panjang. saat email dikirim karena email akan disimpan pada server utama atau email server itu sendiri berdasarkan tujuan email akan dikirimkan kemana. Umumnya file ini berisi informasi yang dimana sumber tujuan, serta adanya waktu pengiriman. Nah saat kamu sebagai user membaca email berarti user telah mengakses server email tersebut dan membaca email yang tersimpan pada server yang di tampilkan pada browser pengguna.

Untuk memahami cara kerja mail server yang kedua ini,kamu harus memahami ada beberapa istilah penting yaitu MUA atau mail user agent yaitu sebuah komponen yang berinteraksi secara langsung, misalnya adalah thunderbird, ms outlook, zimbra atau interface webmail seperti gmail ataupun yahoo.

Selain itu istilah penting mail server lainnya adalah MTA atau mail transfer agent yang bertanggung jawab mentransfer email dari server mail kemudian sampai server mail penerima, contohnya adalah karena sendmail dan postfix. Selain itu MDA atau mail delivery agent, jika mta lokasi menerima email masuk dari mta terpencil maka email akan dikirim kek otak pengguna dengan mda.

Istilah lain dalam mail server ada POP atau IMAP kedua singkatan ini merupakan sesuatu protocol yang digunakan untuk mengunduh email dari kotak penerima server untuk penerima MUA. Kemudian ada mail exchange record atau MX. Istilah ini merujuk pada entri dns untuk server mail. Record mx ini akan menunjuk pada alamt ip dimana email harus ditembakkan. Mx record yang rendah akan selalu menang karena mendapat priritas tertinggi. Contohnya misal mx 10 akan lebih baik dibandingkan dengan mx 20.

Mail Server juga bisa disebut sebagai sebuah komputer yang didedikasikan untuk menjalankan jenis aplikasi perangkat lunak komputer, hal ini dianggap sebagai bagian terpenting dari setiap email sistem. Mail Server biasanya dikelola oleh seorang yang biasanya dipanggil post master.
Tugas Post Master:

- Mengelola Account
- Memonitor Kinerja Server
- Tugas Administratif Lainnya

18.2.1 Protokol Pada Mail Server

Protokol yang umum digunakan antara lain protokol SMTP, POP3 dan IMAP.

1. SMTP (Simple Mail Transfer Protocol) Protokol ini digunakan untuk mengirimkan data dari komputer pengirim surat elektronik ke server surat elektronik penerima. Protokol ini timbul karena desain sistem surat elektronik yang mengharuskan adanya server surat elektronik

yang menampung sementara, sampai surat elektronik diambil oleh penerima yang berhak.

SMTP bisa kita katakan sebagai Sebuah Kantor pos, yang pada dasarnya jika kita mengirim sebuah surat pastinya Surat itu akan dibawa Ke Gudang kantor pos untuk di lakukan penyortiran, Gudang inilah yang dimaksud dengan SMTP, Setelah dilakukan penyortiran maka surat siap untuk diantarkan ketujuan, tapi tidak proses tidak berhenti disini, Jadi surat ini akan dibawa oleh si kurir lalu si Kurir Meletakkanya di Kotak Pos yang biasa kita katakan sebagai PO BOX (PO BOX inilah yang dimaksud dengan POP3) itulah penjelasan singkat tentang SMTP.

SMTP adalah protokol yang cukup sederhana, berbasis teks dimana protokol ini menyebutkan satu atau lebih penerima email untuk kemudian diverifikasi. Jika penerima email valid, maka email akan segera dikirim. SMTP menggunakan port 25 dan dapat dihubungi melalui program telnet. Agar dapat menggunakan SMTP server lewat nama domain, maka record DNS (Domain Name Server) pada bagian MX (Mail Exchange) digunakan.

SMTP (Simple Mail Transfer Protocol) digunakan sebagai standar untuk menampung dan mendistribusikan email. Simple Mail Transfer Protocol atau SMTP digunakan untuk berkomunikasi dengan server guna mengirimkan email dari lokal email ke server, sebelum akhirnya dikirimkan ke server email penerima. Proses ini dikontrol dengan Mail Transfer Agent (MTA) yang ada dalam server email Anda. Port SMTP Default:

- Port 25 Port tanpa dienkripsi
- Port 426 Port SSL/TLS, nama lainnya SMTPS

2. POP3 Post Office Protocol v3 POP3 (Post Office Protocol v3) dan IMAP (Internet Mail Application Protocol) digunakan agar user dapat mengambil dan membaca

email secara remote yaitu tidak perlu login ke dalam sistem shelll mesin mail server tetapi cukup menguhubungi port tertentu dengan mail client yang mengimplementasikan protocol POP3 dan IMAP. POP3 (Post Office Protocol 3) adalah versi terbaru dari protokol standar untuk menerima email. POP3 merupakan protokol client/server dimana email dikirimkan dari server ke email lokal. Digunakan untuk berkomunikasi dengan email server dan mengunduh semua email ke email lokal (seperti Outlook, Thunderbird, Windows Mail, Mac Mail, dan sebagainya), tanpa menyimpan salinannya di server. Biasanya, dalam aplikasi email terdapat pilihan untuk tetap menyimpan salinan email yang diunduh pada server atau tidak.

Apabila kita mengakses akun email yang sama dari perangkat berbeda, akan sangat direkomendasikan untuk menyimpan backup. Hal ini perlu dilakukan sebagai langkah antisipasi apabila perangkat kedua tidak bisa mengunduh email, sementara perangkat pertama sudah menghapusnya.

POP3 adalah sebuah protocol internet yang digunakan untuk mengakses email atau surat elektronik yang masuk ke dalam email client. Fungsi utama dari POP3 ini adalah untuk menyimpan sementara email yang terkirim di dalam sebuah email server, dan kemudian meneruskannya ke dalam email client, dimana baru akan terespon ketika email tersebut sudah dibuka oleh user yang berhak (dalam hal ni adalah mereka yang memegang username dan juga password dari alamat email).

POP3 adalah protokol komunikasi satu arah, yang artinya data diambil dari server dan dikirimkan ke email lokal di perangkat komputer Anda. Port POP3 Default:

- Port 110 Port tanpa dienkripsi
- Port 995 Port SSL/TLS, nama lainnya POP3S

18.2.1.1 Kelebihan Menggunakan POP3

1. Ketika email sudah diunduh melalui aplikasi local mail di komputer, Anda tidak perlu terhubung ke internet apabila Anda ingin membukanya kembali.
2. Kebanyakan tidak ada ukuran limit untuk email yang dikirim dan diterima.
3. Dapat membuka le attachment dengan cepat.
4. Tidak ada ukuran maksimal untuk mailbox, kecuali harddisk komputer Anda penuh.

18.2.2 Kekurangan Menggunakan POP3

1. Jika JavaScript pada email reader diaktifkan, email phishing dengan embed JavaScript dapat terbaca di email.
2. Semua pesan akan disimpan di komputer. Hal ini dapat mengurangi space pada harddisk komputer.
3. Semua file attachment diunduh dan disimpan dalam komputer. Karenanya, potensi komputer terinfeksi virus dari email lebih besar.
4. Folder email terkadang hilang. Jika ini yang terjadi, upaya restore cukup sulit dilakukan.

18.2.3 IMAP (Internet Message Access Protocol)

IMAP (Internet Message Access Protocol) adalah protokol standar untuk mengakses/mengambil e-mail dari server. IMAP memungkinkan pengguna memilih pesan e-mail yang akan ia ambil, membuat folder di server, mencari pesan e-mail tertentu, bahkan menghapus pesan e-mail yang ada. Kemampuan ini jauh lebih baik daripada POP (Post Office Protocol) yang hanya memperbolehkan kita mengambil/download semua pesan yang ada tanpa kecuali.

Internet Message Access Protocol merupakan salah satu dari dua protokol penerimaan email (email retrieval protocol). Juga dikenal dengan singkatan IMAP, Internet Message

Access Protocol merupakan Internet protocol yang beroperasi pada Application layer. Dengan IMAP, mailbox dapat dibaca dan dikelola secara simultan (bersamaan) oleh sejumlah email client berbeda. IMAP seringkali digunakan oleh sebagian besar pengguna Internet untuk mendownload email dari web mail server.

Awalnya disebut sebagai Interim Mail Access Protocol, versi IMAP pertama telah menjalani beberapa revisi sejak dibuat pada tahun 1986. Saat ini disebut sebagai Internet Message Access Protocol, versi IMAP ini merupakan versi IMAP keempat yang telah menjadi standar pada tahun 1994, dan dipublikasikan pada RFC 1730. Pop Office Protocol (POP) merupakan Internet protocol umum lainnya untuk email retrieval. Sebagian besar email server dan email client mendukung baik IMAP dan POP sebagai pilihan lain terhadap protokol unik mereka sendiri. Dibandingkan dengan POP, IMAP memiliki beberapa keunggulan termasuk kemampuan untuk memuat bagian dari email ketimbang menunggu semua attachment di dalamnya. IMAP juga dapat juga menerima konten pesan menggunakan mekanisme MIME. IMAP client juga cenderung tetap dapat terhubung dengan mail server dalam periode waktu yang lebih lama, yang dapat meningkatkan response time secara keseluruhan.

Cara kerja IMAP adalah email client melakukan koneksi ke server email, lalu melakukan sinkronisasi folder. Apabila kita mengklik/ mengakses sebuah folder, maka daftar email berikut isinya (?) juga didownload. Apabila kita menghapus sebuah email, maka email pada server juga dihapus. Dengan kata lain, protokol IMAP seakan-akan memindahkan semua isi mailbox kita ke e-mail clent kita sendiri.

Pada dasranya Protokol IMAP ini dirancang agar user dapat mengakses e-mail pada milbox serta dapat berinteraksi dengan server. PORT yang digunakan untuk protocol ini dalam bentuk TCP/IP yaitu pada PORT nomer 143. Protocol ini menggunakan koneksi yang terus menerus ke server. Ketika e-mail masuk maka anda akan meli-

hat langsung di e-mail kmputer Client (dengan posisi online). Karena e-mail yang masuk ke server maka akan cepat masuk dan dapat segera dilihat juga di Client. Seringkali lebih cepat prosesnya dibandingkan jika menggunakan web interface sendiri yang mirip seperti Blackberry. Namun untuk menggunakan IMAP anda harus menggunakan Koneksi Internet yang cukup baik atau dengan bandwidth yang lumayan besar. Bahkan dengan IMAP jika anda menggunakan 10 client interface web, missal menggunakan Netbook,Notbook,Dekstop, Ponsel danlain sebagainya maka semua akan memperlihatkan e-mail yang sama. Jika anda menggunakan banyak device untuk mengakses e-mail, maka pilihan yang tepat adalah menggunakan IMAP. Karena IMAP lebih baik dengan POP. Tapi IMAP biasanya digunakan untuk dalam jaringan LAN saja karena untuk kapasitas jaringan kecil akan lebih maksimal, jika untuk kapasitas yang lebih besar lagi pilihan yang tepat adalah menggunakan Protokol POP3.

IMAP (Internet Message Access Protocol), seperti halnya POP3, juga digunakan untuk mengirimkan email ke local mail, hanya saja terdapat sedikit perbedaan cara kerja. IMAP merupakan protokol komunikasi dua arah sebagai perubahan yang dibuat pada local mail yang dikirimkan ke server. Pada dasarnya, isi email tetap berada di server. Protokol IMAP lebih direkomendasikan oleh penyedia email seperti Gmail dibandingkan menggunakan POP3. Dalam IMAP, email disimpan di server. ketika Anda akan mengecek email, local mail akan menghubungi server untuk menampilkan pesan email. Sehingga untuk file pesan email tetap berada di server dan tidak didownload ke email lokal. Port IMAP Default:

1. Port 143 Port tanpa dienkripsi
2. Port 993 Port SSL/TLS, nama lainnya IMAPS

18.2.4 Menggunakan SMTP pada python

Python memiliki sebuah modul bernama SMTP Server yang dapat digunakan untuk menerima e-mail dari client dan

menampilkan isinya ke stdout alias konsol. contoh client yang mengirim e-mail ke SMTP server ini:

```
import smtplib
sender = 'mikail@website.com'
recipient = ['sonanjaya@website.com']

message = """From: From Person <%s>
To: To Person <%s>
Subject: Testing SMTP E-Mail

Pesan ini dikirim melalui smtplib dan diterima oleh modul SMTP

""" % (sender, recipient)

try:

    smtpObj = smtplib.SMTP('localhost', 1025)
    smtpObj.sendmail(sender, recipient, message)

    print "Mengirim e-mail berhasil :D..."

except Exception, e:

print str(e)
    print "Error: e-mail gagal terkirim :(..."
```

Pada kode diatas dengan menggunakan smtplib cukup dengan mengakses URL dan port yang dituju, lalu kirim e-mail dengan menggunakan method sendmail() dengan melewatkan pengirim, penerima, dan pesan.

Table 18.1 Contoh email yahoo yang memiliki smtp

No	Keterangani
.1	Yahoo POP Server - pop.mail.yahoo.com port-995
.2	Requires SSL-Yes

18.1: **Skema Singkat**

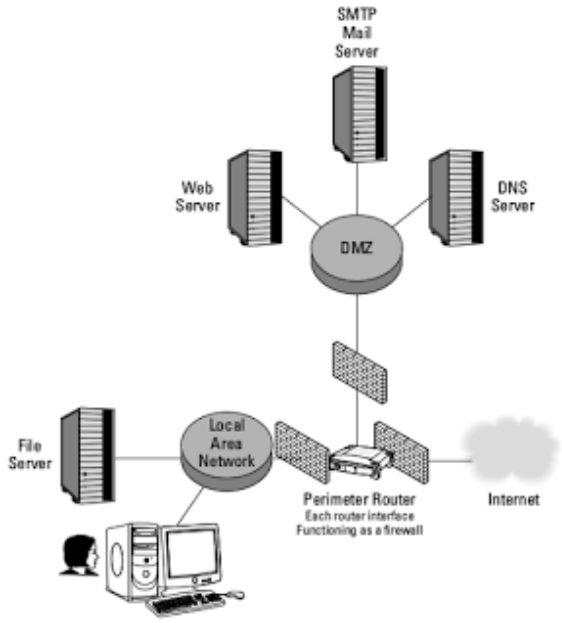


Figure 18.1 SMTP

Mail Server atau yang sering disebut juga E-Mail server, digunakan untuk mengirim surat melalui Internet. Dengan begitu, dapat mempermudah dalam penggunanya, karena lebih cepat dan efisien. Untuk membuat Mail Server, harus terdapat SMTP dan POP3 server, yang digunakan untuk mengirim dan menerima email. Proses pengiriman email bisa terjadi karena adanya SMTP Server (Simple Mail Transfer Protocol). Setelah dikirim, email tersebut akan ditampung sementara di POP3 Server (Post Office Protocol ver. 3). Dan ketika user yang mempunyai email account tersebut online, mail client akan secara otomatis melakukan sinkronisasi dari POP3 Server.

Mungkin kita dulu pernah bahkan sering menggunakan kantor pos sebagai jasa untuk mengirim surat kepada kerabat atau teman kita yang jauh dari tempat kita tinggal, di Kantor pos itu juga terdapat Kurir sebagai tukang mengantarkan surat dari Kantor Pos kepada penerima.

SMTP ini singkatan dengan kepanjangan Simple Mail Transfer Protocol. Dengan pengertian singkat SMTP adalah protokol yang mengatur pengiriman email dari pengirim (Ongoing). Cara kerja SMTP ini sama dengan Kantor Pos, tempat dimana kita menitipkan surat kita agar dikirimkan kepada penerima yang tertera di surat tersebut. Terdapat pada port berapa SMTP? SMTP terdapat pada port 25.

POP3 ini singkatan dari kepanjangan Post Office Protocol ver.3 dan IMAP ini singkatan dari kepanjangan Internet Message Access Protocol. Dengan pengertian singkat POP3/IMAP adalah protokol yang mengatur penerimaan email kepada penerima (Ingoing).

Cara kerja POP3 dan IMAP sama dengan kurir yang mengantarkan surat, tugasnya menyampaikan surat yang sebelumnya sudah dititipkan di Kantor Pos (SMTP) kepada penerima surat. Terdapat pada port berapa POP3? POP3 terdapat pada port 110. Terdapat pada port berapa IMAP? IMAP terdapat pada port 143.

Ada perbedaan antara POP3 dan IMAP, mudahnya. kalau IMAP itu protokol yang biasa kita pakai kalau kita pakai Web Based Mail, contohnya seperti (gmail.com, yahoo.com) dan menggunakan Browser sebagai antarmukanya, kalau POP3 menggunakan Aplikasi Email Client, sebut saja Outlook, Thunderbird, Outlook Express dll.

Bedanya jika kita pakai IMAP, kita hanya bisa melihat isi dari email yang ada pada mailbox kita secara online dan tidak bisa di download ke localdisk kita, berbeda dengan POP3 yang bisa membuka email secara offline dengan syarat ketika komputer online dia mendownload semua email ke local disk sehingga bisa dibaca offline.

CHAPTER 19

MULTITHREADING

19.1 Multithreading

Menjalankan beberapa *thread* mirip dengan menjalankan beberapa program yang berbeda secara bersamaan, namun dengan manfaat berikut :

- Beberapa *thread* dalam proses berbagi ruang data yang sama dengan benang induk dan karena dapat saling berbagi informasi atau berkomunikasi satu sama lain dengan lebih muda daripada jika prosesnya terpisah

- *thread* terkadang disebut proses ringan dan tidak membutuhkan banyak memori atas, mereka lebih murah daripada proses.

Sebuah *thread* memiliki permulaan, urutan eksekusi dan sebuah kesimpulan. Ini memiliki pointer perintah yang melacak dari mana dalam konteksnya saat ini berjalan.

- Hal ini dapat dilakukan sebelum pre-*empted* (*interrepted*)
- Untuk sementara dapat ditunda sementara *thread* lainnya yang sedang berjalan ini disebut unggul.

19.1.1 Memulai Thread Baru

Untuk melakukan *thread* lain, perlu memanggil metode berikut yang tersedia dimodul *thread* :

Thread.start _new _thread (function, args [, kwargs])

Pemanggilan metode ini memungkinkan cara cepat dan tepat untuk membuat *thread* baru di linux dan window.

Pemanggilan metode segera kembali dan anak *thread* dimulai dan fungsi pemanggilan dengan daftar *args* telah berlalu. Saat fungsi kembali ujung *thread* akan berakhir. Disini, *args* adalah tupel argumen. Gunakan tupel kosong untuk memanggil fungsi tanpa melewati argumen. *Kwargs* adalah kamus opsional argumen kata kunci.

Contoh :

```
\#  $!/usr/bin/python}

Import thread
Import time

#  $ Define a function for the thread

Def print $  \_  $time (threadNamw, delay):
Count = 0
While count <5:
Time.sleep(delay)
Count +=1
```

```
Print  $ " $ $  \%  $s :  $  \%  $s $ " $  $  \%  $
(threadName, time.ctime(time.time()))

#  $ Create two thread as follows
try:
thread.start $  \_  $new $  \_  $thread(print $  \_
$time, ( $ " $Thread-1 $ " $, 2, ))
thread.start $  \_  $new $  \_  $thread(print $  \_
$time, ( $ " $Thread-2 $ " $, 4,))
except:
print " $Error: unable to start thread

while 1:
pass
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut :

```
Thread-1 : Thu Jan 22 15:42:17 2009

Thread-1 : Thu Jan 22 15:42:19 2009

Thread-2 : Thu Jan 22 15:42:19 2009

Thread-1 : Thu Jan 22 15:42:21 2009

Thread-2 : Thu Jan 22 15:42:23 2009

Thread-1 : Thu Jan 22 15:42:23 2009

Thread-1 : Thu Jan 22 15:42:23 2009

Thread-1 : Thu Jan 22 15:42:25 2009
```

Thread-2 : Thu Jan 22 15:42:27 2009

Thread-2 : Thu Jan 22 15:42:31 2009

Thread-2 : Thu Jan 22 15:42:35 2009

Meskipun sangat efektif untuk benang tingkat rendah, namun modul *thread* sangat terbatas dibandingkan dengan modul yang baru.

19.1.2 Modul Threading

Modul threading yang lebih baru disertakan dengan Python 2.4 memberikan jauh lebih kuat, dukungan tingkat tinggi untuk *thread* dari modul *thread* dibahas pada bagian sebelumnya.

The *threading* modul mengekpos semua metode dari *thread* dan menyediakan beberapa metode tambahan :

- **threading.activeCount()**
  Mengembalikan jumlah objek *thread* yang aktif
- **threading.currentThread()**
  Mengembalikan jumlah objek *thread* dalam kontrol benang pemanggil
- **threading.enumerate()**
  Mengembalikan daftar semua benda *thread* yang sedang aktif

Selain metode, modul *threading* memiliki *thread* kelas yang mengimplementasikan *threading.* Metode yang disediakan oleh *thread* kelas adalah sebagai berikut :

- **run()**
  Metode adalah titik masuk untuk *thread*
- **start()**

Metode dimulai *thread* dengan memanggil metode run

- **join([time])**
  Menunggu benang untuk mengakhiri
- **isAlive()**
  Metode memeriksa apakah *thread* masih mengeksekusi
- **getName()**
  Metode mengambalikan nama *thread*
- **setName()**
  Metode menetapkan nama *thread*

19.1.3 Membuat Thread Menggunakan Modul

Untuk melaksanakan *thread* baru menggunakan *threading* harus melakukan hal berikut :

- Mendefinisikan subclass dari *thread* kelas
- Menimpa _init _ (self [args]) metode untuk menambahkan argumen tambahan
- Menimpa run(self[args]) metode untuk menerapkan apa *thread* harus dilakukan ketika mulai

Setelah membuat baru *thread* subclass, dapat membuah seuah instance dari itu dan kemudian memulai *thread* baru dengan menerapkan *start(),* yang ada gilirinnya panggilan *run()* metode.

Contoh :

```
#  $!/usr/bin/python

import threading
```

```
import time

exitFlag = 0

class myThread (threading.Thread):
def $  \_  $init $  \_  $(self, threadID, name,
counter) :
threading.Thread. $  \_  $init $  \_  $(self)}
self.threadID = threadID
self.name = name
self.counter = counter
def run (self) :
print  $ " $Starting  $ " $ + self.name
print $  \_  $time(self.name, self.counter, 5)
print  $ " $Exiting  $ " $+ self.name

def print $  \_  $time(threadName, delay, counter):}

while counter:

if exitFlag:
threadName.exit()
time.sleep(delay)
print  $ " $ $  \%  $s:  $  \%  $s $ " $  $  \%
$ (threadName, time.ctime(time.time()))
counter -= 1}

#  $ Create new threads
thread1 = myThread(1,  $ " $Thread-1 $ " $, 1)
thread2 = myThread(2,  $ " $Thread-2 $ " $, 2)
#  $ Start new threads
thread1.start()
thread2.start()
print  $ " $Exiting Main Thread $ " $
```

Ketika kode diatas dijalankan, menghasilkan hasil sebagai berikut:

```
Starting Thread-1
```

```
Starting Thread-2
Exiting Main Thread
Thread-1 : Thu Mar 21 09:10:03 2013
Thread-1 : Thu Mar 21 09:10:04 2013
Thread-2 : Thu Mar 21 09:10:04 2013
Thread-1 : Thu Mar 21 09:10:05 2013
Thread-2 : Thu Mar 21 09:10:06 2013
Thread-1 : Thu Mar 21 09:10:07 2013
Exiting Thread-1
Thread-2 : Thu Mar 21 09:10:08 2013
Thread-2 : Thu Mar 21 09:10:10 2013
Thread-2 : Thu Mar 21 09:10:12 2013
Exiting Thread=2
```

19.1.4 Sinkronisasi Thread

Threading modul disediakan dengan Python termasuk sederhana untuk menerapkan mekanisme bahwa memungkinkan untuk menyinkronkan *thread* penguncian. Sebuah kunci baru dibuat dengan memanggil *lock()* metode yang mengembalikan kunci baru.

The *acquire (blocking)* metode objek kunci baru digunakan untuk memaksa *thread* untuk menjalankan serempak. Opsional *blocking* parameter memungkikan untuk mengontrol apakah *thread* menunggu untuk mendapatkan kunci.

Jika *blocking* diatur ke 0, *thread* segera kembali dengan nilai 0 jika kunci tidak dapat diperoleh dan dengan 1 jika kunci dikuisisi. Jika pemblokiran diatur ke 1, blok dan menunggu kunci yang akan dirilis.

The *release()* metode objek kunci baru digunakan untuk melepaskan kunci ketika tidak lagi diperlukan.

Contoh:

```
#  $!/usr/bin/python

import threading
import time
```

```
class myThread (threading.Thread):
def $  \_  $init $  \_  $(self, threadID, name,
counter):
threading.Thread. $  \_  $init $  \_  $(self)
self.threadID = threadID
self.name = name
self.counter = counter
def run(self)
print  $ " $Starting  $ " $+ self.name
$  \#  $ Get lock to synchronize threads
ThreadLock.acquire()
print $  \_  $time(self.name, self.counter, 3)
#  $ Free lock to realease next thread
ThreadLock.release()
Def print $  \_  $time(threadName, delay, counter):
while counter:
time.sleep(delay)
print  $ " $ $  \%  $s:  $  \%  $s $ " $  $  \%  $
(threadName, time.ctime(time.time()))
counter -= 1
threadLock = threading.Lock()
threads = []
#  $ Create new threads
thread1 = myThread(1,  $ " $Thread-1,1 )
thread2 = myThread(2,  $ " $Thread-2,2 )
#  $ Start new Threads}
thread1.start()
thread2.start()

#Add threads to thread list}
threads.append(thread1)}
thread2.append(thread2)}
\vspace{10pt}

#  $ Wait for all threads to complete}
Fort t in threads:
t.join()
```

```
print  $ " $Exiting Main thread $ " $
```

Bila kode diatas dieksekusi, maka menghasilkan sebagai berikut :

```
   Starting Thread-1
Starting Thread-2
Thread-1: Thu Mar 21 09:11:28 2013
Thread-1: Thu Mar 21 09:11:29 2013
Thread-1: Thu Mar 21 09:11:30 2013
Thread-2: Thu Mar 21 09:11:32 2013
Thread-2: Thu Mar 21 09:11:34 2013
Thread-2: Thu Mar 21 09:11:36 2013
Exiting Main Thread
```

19.1.5 Multithreaded Antrian Prioritas

The queue modul memungkinkan untuk membuat objek antrian baru yang dapat menampung jumlah tertentu item. Ada metode berikut untuk mengontrol antrian :

- **get()**
  Menghapus dan mengembalikan item dari antrian
- **put()**
  Menambahkan item ke antrian
- **qsize()**
  Mengembalikan jumlah item yang saat ini dalam antrian
- **empty()**
  Mengembalikan benar jika antrian kosong jika tidak, salah
- **full()**

Mengembalikan benar jika antrian penuh jika tidak, salah

Contoh:

```
#  $!/usr/bin/python}

import Queue
import threading
import time

exitFlag = 0

class myThread (threading.Thread):
def   $  \_  $init $  \_  $(self, threadID, name, q):
threading.Thread. $  \_  $init $  \_  $(self)
self.name = name
self.q = q
def run(self):
print  $ " $Starting  $ " $+ self.name
process $  \_  $data(self.name, self.q)
print  $ " $Exiting  $ " $+ self.name
def process $  \_  $data(threadName, q):
while not exitFlag:
queuLock.acquire()
if not workQueu.empty():
data = q.get()
queueLock.release()
print  $ " $ $  \%  $s processing  $  \%  $s $ " $
$  \%  $ (threadName, data)
else:
queueLock.release()
time.sleep(1)

threadList = [ $ " $Thread-1 $ " $,  $ " $Thread-2 $
" $,  $ " $Thread-3 $ " $]
nameList = [ $ " $One $ " $,  $ " $Two $ " $,  $ "
$Three $ " $,  $ " $Four $ " $,  $ " $Five $ " $]
queueLock = threading.Lock()
workLock = Queue.Queue(10)
threads = []
threadID = 1
```

```
#  $ Create new threads
For tName in threadList:
thread = myThread(threadID, tName, workQueue)
thread.start()
thread.append(thread)
threadID +=1

#  $ Fill the queue
queueLock.acquire()
for word in nameList:
workQueue.put(word)
queueLock.release()

#  $ Wait for queue to empty
while not workQueue.empty():
pass

#  $ Notify threads it?s time to exit
exitFlag = 1

#  $ Wait for all threads to complete
For t in threads:
t.join()
print  $ " $Exiting Main Thread $ " $
```

Bila kode diatas dieksekusi, maka menghasilkan hasil sebagai berikut:

```
Starting Thread-1
Starting Thread-2
Starting Thread-3
Thread-1 processing One
Thread-2 processing Two
Thread-3 processing Three
Thread-1 processing Four
Thread-2 processing Five
Exiting Thread-3
Exiting Thread-1
Exiting Thread-2
```

Exiting Main Thread

19.2 Cara menggunakan Threading untuk Membuat Benang

Threading menggabungkan semua metode thread dan menampilkan beberapa metode tambahan. Terlepas dari metode di atas, threading juga menyajikan kelas Thread yang dapat dicoba untuk mengimplementasikan thread. Ini adalah varian object-oriented dari multithreading Python. Kelas ¡Thread¿ menerbitkan metode berikut :

Table 19.1 Ukuran

Kelas	Penjelasan Metode
run():	Ini adalah fungsi entry point untuk thread manapun
start():	Metode start () memicu thread saat metode dijalankan dipanggil
join([time]):	Metode join () memungkinkan sebuah program untuk menunggu thread untuk diakhiri

19.3 Mengimplementasikan Thread menggunakan Threading

Buatlah subkelas dari kelas Thread. Timpa metode (self [, args]) untuk memberi argumen sesuai persyaratan. Selanjutnya, timpa metode run (self [args]) untuk mengkodekan logika bisnis benang.

Setelah mendefinisikan subclass Thread baru, harus memberi instantiate untuk memulai thread baru. Berikut 19.1 Mengimplementasikan Thread menggunakan Threading :

```
#Python multithreading example to print current date.
#1. Define a subclass using Thread class.
#2. Instantiate the subclass and trigger the thread.

import threading
import datetime

class myThread (threading.Thread):
    def __init__(self, name, counter):
        threading.Thread.__init__(self)
        self.threadID = counter
        self.name = name
        self.counter = counter
    def run(self):
        print "Starting " + self.name
        print_date(self.name, self.counter)
        print "Exiting " + self.name

def print_date(threadName, counter):
    datefields = []
    today = datetime.date.today()
    datefields.append(today)
    print "%s[%d]: %s" % ( threadName, counter, datefields[0] )

# Create new threads
thread1 = myThread("Thread", 1)
thread2 = myThread("Thread", 2)

# Start new Threads
thread1.start()
thread2.start()

thread1.join()
thread2.join()
print "Exiting the Program!!!"
```

Figure 19.1 Mengimplementasikan Thread menggunakan Threading

CHAPTER 20

XML PROCESSING

20.1 XML Processing

XML adalah bahasa open source portable yang mungkinkan pemrogram mengemangkan aplikasi yang dapat dibaca oleh aplikasi lain, terlepas dari sistem operasi dan bahasa pengembangnya.

Extensible Markup Languange (XML) adalah bahasa markup seperti HTML atau SGML. Ini direkomendasikan oleh World Wide Web Consortium dan tersedia sebagai standar terbuka. XML sangat berguna untuk mencatat data berukuran kecil dan menengah tanpa memerlukan tulang punggung berbasis SQL.

20.1.1 Arsitektur Parsing XML dan API

Perpustakaan standar Python menyediakan seperangkat antarmuka minimal tapi berguna untuk bekerja dengan XML.

Dua API yang paling dasar dan umum digunakan untuk data XML adalah antarmuka SAX dan DOM.

API sederhana untuk XML (SAX): mendaftarkan panggilan kemali untuk acara yang diminati dan kemudian membiarkan parser berjalan melalui dokumen. Ini berguna bila dokumen berukuran besar atau memiliki keterbatasan memori, ini memparsing file tidak pernah tersimpan dalam memori.

API Document Objek Model (DOM): ini adalah rekomendasi World Wide Web Consortium dimana keseluruhan file dibaca ke memori dan disimpan dalam bentuk hierarkies (tree-based) untuk mewakili semua fitur dokumen XML.

SAX jelas tidak bisa memproses informasi secepat DOM saat bisa bekerjadengan file besar. Di sisi lain, menggunakan DOM secara eklusifenar-benar dapat membunuh sumber daya, terutama jika digunakan pada banyak file kecil.

SAX hanya bisa dibaca sementara DOM mengizinkan perubahan pada file XML. Kedua API yang berbeda ini saling melengkapi satu sama lain, tidak ada alasan mengapa tidak dapat menggunakannya untuk proyek besar.

Contoh:

```
<collection shelf="New Arrivals">

<movie title="Enemy Behind">

    <type>War, Thriller</type>

    <format>DVD</format>

    <year>2003</year>
```

```
    <rating>PG</rating>
    <stars>10</stars>
    <description>Talk about a US-Japan war</description>
</movie>
<movie title="Transformers">
    <type>Anime, Science Fiction</type>
    <format>DVD</format>
    <year>1989</year>
    <rating>R</rating>
    <stars>8</stars>
    <description>A schientific fiction</description>
</movie>
    <movie title="Trigun">
    <type>Anime, Action</type>
    <format>DVD</format>
    <episodes>4</episodes>
    <rating>PG</rating>
    <stars>10</stars>
    <description>Vash the Stampede!</description>
```

```
</movie>

<movie title="Ishtar">

    <type>Comedy</type>

    <format>VHS</format>

    <rating>PG</rating>

    <stars>2</stars>

    <description>Viewable boredom</description>

</movie>
</collection>
```

20.1.2 Parsing XML dengan API SAX

SAX adalah antarmuka standar untuk parsing XML berbasis event. Parsing XML dengan SAX umumnya mengharuskan untuk membuat C*ontrolHandler* dengan subclassing xml.sax *controlhandler*.

ControlHandler menangani tag dan atribut tertentu dari XML. Objek *ControlHandler* menyediakan metode untuk menangani berbagai aktivitas parsing. Parsing memanggil metode *ControlHandler* saat memparsing file XML.

Metode *startDocument* dan *endDocument* disebut awal dan akhir setiap elemen. Jika parsing tidak dalam mode namespace, metode *startElement* (tag attribute) dan *endElement* (tag) dipanggil. Jika tidak, metode yang sesuai *startElemenNS* dan *endElemenNS* dipanggil. Disini, tah adalah tag elemen dan atribut adalah atribut.

Metode-metode berikut membuat objek parsing baru dan mengembalikannya. Objek parsing diuat akan menjadi tipe parsing pertama yang ditemukan sistem. xml.sax.make _parser([parser _list]) .Parameter Parser _list pilihan ar-

gumen yang terdiri dari daftar parsing untuk digunakan yang semuanya harus menerapkan metode *make_parse*

Metode-metode berikut membuat parsing SAX dan menggunakannya untuk mengurai dokumen xml.sax.parser(xmlfile, contenthandler[, errorhandler]). Berikut adalah detail dari parameternya:

- *Xmlfile*
  Ini adalah nama file XML yang bisa dibaca.
- *ContentHandler*
  Ini harus menjadi objek *ContenHandler*
- *ErrorHandler*
  Jika ditentukan, e*rrorhandler* harus menjadi objek *ErrorHandler* SAX
- Metode *parseString*

Membuat parsing SAX dan mengurai string XML yang ditentukan : xml.sax.parsertring(xmlstring,contenthandler[, errorhandler]).

Brikut ini adalah detail nama dar parameter :

- XMLstring
  Nama dari string yang bisa dibaca
- ContentHandler
  Menjadi objek ContenHandler
- ErrorHandler
  Menjadi objek ErorHandler SAX

Contoh :

```
$  \#  $!/usr/bin/python

import xml.sax

class MovieHandler( xml.sax.ContentHandler ):
```

```
    def  $  \_  $ $  \_  $init $  \_  $ $  \_  $(self):
      self.CurrentData = ""
      self.type = ""
      self.format = ""
      self.year = ""
      self.rating = ""
      self.stars = ""
     self.description = ""
  $  \#  $ Call when an element starts
  def startElement(self, tag, attributes):
     self.CurrentData = tag
     if tag == "movie":
     print "*****Movie*****"
     title = attributes["title"]
     print "Title:", title
  $  \#  $ Call when an elements ends
 def endElement(self, tag):
     if self.CurrentData == "type":
     print "Type:", self.type
```

```
 elif self.CurrentData == "format":
 print "Format:", self.format
 elif self.CurrentData == "year":
 print "Year:", self.year
 elif self.CurrentData == "rating":
 print "Rating:", self.rating
elif self.CurrentData == "stars":
print "Stars:", self.stars
elif self.CurrentData == "description":
print "Description:", self.description
self.CurrentData = ""
$  \#  $ Call when a character is read
def characters(self, content):
if self.CurrentData == "type":
self.type = content
elif self.CurrentData == "format":
self.format = content
elif self.CurrentData == "year":
self.year = content
```

```
    elif self.CurrentData == "rating":

    self.rating = content

    elif self.CurrentData == "stars":

    self.stars = content

    elif self.CurrentData == "description":

    self.description = content


if (  $  \_  $ $  \_  $name $  \_  $ $  \_  $ == " $
\_  $ $  \_  $main $  \_  $ $  \_  $"):

    $  \#  $ create an XMLReader

    parser = xml.sax.make $  \_  $parser()

    $  \#  $ turn off namepsaces

    parser.setFeature(xml.sax.handler.feature $  \_
    $namespaces, 0)

   $  \#  $ override the default ContextHandler

   Handler = MovieHandler()

   parser.setContentHandler( Handler )

   parser.parse("movies.xml")
```

Ini akan menghasilkan hasil sebagai berikut:

```
*****Movie******
*****Movie*****
Title: Enemy Behind
```

Type: War, Thriller
Format: DVD
Year: 2003
Rating: PG
Stars: 10
Description: Talk about a US-Japan war
*****Movie*****
Title: Transformers
Type: Anime, Science Fiction
Format: DVD
Year: 1989
Rating: R
Stars: 8
Description: A schientific fiction
*****Movie*****
Title: Trigun
Type: Anime, Action
Format: DVD
Rating: PG
Stars: 10
Description: Vash the Stampede!
*****Movie*****
Title: Ishtar
Type: Comedy
Format: VHS
Rating: PG
Stars: 2

20.1.3 Parsing XML dengan API DOM

Document Ovject Model (DOM) adalah API lintas bahasa dari World Wide Web Consortium (W3C) untuk mengakses dan memodifikasi dokumen XML.

DOM sangat berguna untuk aplikasi akses acak. SAX hanya memungkinkan melihat satu bit dokumen sekaligus. Jika melihat satu elemen SAX, tidak memiliki akses ke yang lain.

Berikut adalah cara termudah untuk memuat dokumen XML dengan cepat dan membuat objek minidom menggunakan modul xml.dom. Objek minidom menyediakan metode parsing sederhana yang dengan cepat memuat pohon DOM dari file XML.

Contoh frase memanggil fungsi parsing (file [,parsing]) dari objek minidokumen untuk mengurai file XML yang ditunjuk oleh file ke objek pohon DOM.

contoh:

```
 $  \#  $!/usr/bin/python

from xml.dom.minidom import parse

import xml.dom.minidom

 $  \#  $ Open XML document using minidom parser

DOMTree = xml.dom.minidom.parse("movies.xml")

collection = DOMTree.documentElement

if collection.hasAttribute("shelf"):

   print "Root element :  $  \%  $s"  $  \%  $ collection.
   getAttribute("shelf")

 $  \#  $ Get all the movies in the collection

movies = collection.getElementsByTagName("movie")

 $  \#  $ Print detail of each movie.

for movie in movies:

   print "*****Movie*****"

   if movie.hasAttribute("title"):
```

```
print "Title: $ \% $s" $ \% $ movie.getAttribute
("title")

type = movie.getElementsByTagName('type')[0]

print "Type: $ \% $s" $ \% $ type.childNodes
[0].data

format = movie.getElementsByTagName('format')[0]

print "Format: $ \% $s" $ \% $ format.childNodes
[0].data

rating = movie.getElementsByTagName('rating')[0]

print "Rating: $ \% $s" $ \% $ rating.childNodes
[0].data

description = movie.getElementsByTagName('description')
[0]

print "Description: $ \% $s" $ \% $ description.
childNodes[0].data
```

Ini akan menghasilkan hasil sebagai berikut :
Root element : New Arrivals
*****Movie*****
Title: Enemy Behind
Type: War, Thriller
Format: DVD
Rating: PG
Description: Talk about a US-Japan war
*****Movie*****
Title: Transformers
Type: Anime, Science Fiction
Format: DVD
Rating: R
Description: A schientific fiction

*****Movie*****
Title: Trigun
Type: Anime, Action
Format: DVD
Rating: PG
Description: Vash the Stampede!
*****Movie*****
Title: Ishtar
Type: Comedy
Format: VHS
Rating: PG
Description: Viewable boredom

20.1.4 Membangun Parsing Document XML menggunakan Python

Python mendukung untuk bekerja dengan berbagai bentuk markup data terstruktur. Selain mengurai xml.etree. *ElementTree* mendukung pembuatan dokumen XML yang terbentuk dengan baik dari objek elemen yang dibangun dalam aplikasi. Kelas elemen digunakakan saat sebuah dokumen diurai untuk mengetahui bagaimana menghasilkan bentuk serial dari isinya kemudian dapat ditulis ke sebuah file.

Untuk membuat instance elemeb gunakan fungsi elemen contructor dan *SubElemen()* pabrik. contoh :

```
Import xml.etree.ElementTree as xml

     filename =  $ " $/home/abc/Desktop/test $  \_  $xml
     .xml $ " $}

     toot = xml.Element( $ " $Users $ " $)}

     userelement = xml.Element( $ " $user $ " $)}

     root.append(userelement)}
```

Bila menjalankan ini, akan menghasilkan sebagai berikut :

¡Users¿
¡user¿
¡user¿
¡/Users¿

Tambahkan anak-anak pegguna :

```
Uid = xml.SubElement(userelement,  $ " $uid $ " $)}

Uid.text =  $ " $1 $ " $}

FirstName = xml.SubElement(userelement,  $ " $FirstName
$ " $)}

FirstName.text =  $ " $testuser $ " $}

LastName = xml.SubElement(userelement,  $ " $LastName
$ " $}

LastName.text =  $ " $testuser $ " $}

Email = xml.SubElement(userelement,  $ " $Email $ " $)}

Email.text = {mailto:testuser@test.com}{testuser@test.com}
}

state = xml.SubElement(userelemet,  $ " $state $ " $)}

state.text =  $ " $xyz $ " $}

location = xml.SubElement(userelement,  $ " $location)}

location.text = abc}

tree = xml.ElementTree(root)}

with open(filename,  $ " $w $ " $) as fh:}
```

```
tree.write(fh)}
```

Pertama buat elemen root dengan mengunakan fungsi *ElementTree*. Kemudian membuat elemen pegguna dan menambahkannya ke root. Selanjutnya membuat *SubElement* dengan melewatkan elemen pengguna (userelement) ke *SubElemen* beserta namanya seperto ”FirstName ”. Kemudian untuk setiap *SubElement* tetapkan properti teks untuk memberi nilai. Di akhir, membuat *ElementTree* dan menggunakannya untuk menulis XML ke file. Jika menjalankan ini akan menjadi sebagai berikut :

¡users¿
¡user¿
¡uid¿1¡/uid¿
¡FirstName¿testuser¡/FirstName¿
¡LastName¿testuser¡/LastName¿
¡state¿xyz¡/state¿
¡location¿abc¡/location¿
¡/user¿
¡/Users¿

Parsing XML Documen :

```
import xml.etree.ElementTree as ET}

tree = ET.parse(Your $  \_  $XML $  \_  $file $  \_
$path)}

 root = tree.getroot()}
```

20.2 Kerentanan XML

Modul pemrosesan XML tidak aman terhadap data yang dibuat. Penyerang dapat menyalahgunakan kerentanan untuk penolakan serangan layanan, mengakses file lokal, menghasilkan koneksi jaringan ke mesin lain, atau menghindari firewall. Serangan terhadap penyalahgunaan XML fitur asing seperti inline DTD (document type definition) dengan

entitas. Tabel berikut memberikan gambaran umum tentang serangan yang diketahui dan jika berbagai modul rentan terhadapnya :

Table 20.1 Ukuran

Kind	Sax
billion laughs	Vulnerable
quadratic blowup	Vulnerable
external entity expansion	Vulnerable
DTD retrieval	Vulnerable
decompression bomb	Safe

20.3 XML Stream Parsing dengan Iterparse

Modul XML cenderung menjadi besar dalam memori yang mungkin bermasalah saat memilih modul,ini salah satu alasan untuk menggunakan API SAX sebagai alternatif DOM.

Menggunakan ET agar mudah membaca XML menjadi pohon memori dan memanipulasinya. Ini sebabnya mengapa paket tersebut menyediakan alat khusus untuk SAX-like, dengan penguraian XML yang cepat. Contoh iterparse dapat digunakan serta mengukur bagaimana tarif terhadap penguraian pohon standar 20.1 XML stream parsing dengan iterparse :

```
<?xml version="1.0" standalone="yes"?>
<site>
  <regions>
    <africa>
      <item id="item0">
        <location>United States</location>    <!-- Counting locations -->
        <quantity>1</quantity>
        <name>duteous nine eighteen </name>
        <payment>Creditcard</payment>
        <description>
          <parlist>
[...]
```

Figure 20.1 XML stream parsing dengan iterparse

CHAPTER 21

GUI PROGRAMMING

21.1 GUI Programming

Python menyediakan berbagai pilihan untuk mengembangkan antarmuka pengguna grafis (GUIs). Yang paling tercantum dibawah ini :

- Tkinter
  Antarmuka Python ke toolkit Tk GUI dikirimkan dengan Python.
- wxPython
  antarmuka Python open-source untuk wxWindows
- Jpython
  Port Python untuk java yang memberikan Python script akses tanpa batas ke perpustakaan kelas java pada mesin lokal

21.1.1 Tkinter Pemrograman

Tkinter adalah perpustakaan GUI standar untuk Python. Python bila dikombinasikan dengan Tkinter menyediakan cara yang mudah dan cepat untuk membuat aplikasi GUI. Tkinter menyediakan antarmuka berorientasi ojek yang kuat untuk toolkit Tk GUI.

Membuat aplikasi GUI menggunakan Tkinter adalah tugas yang mudah. Yang diperlukan adalah melakukan langkah-langkah sebagai berikut :

- Mengimpor Tkinter modul
- Buat jendela utama aplikasi GUI
- Tambahkan satu atau lebih dari widget tersebut diatas ke aplikasi GUI
- Masukkan acara loop utama untuk mengambil tindakan terhadap setiap peristiwa dipicu oleh pengguna

Contoh :

```
$  \#  $!/usr/bin/python}
import Tkinter}

top = Tkinter.Tk()}

 $  \#  $ Code to add widgets will go here...}

top.mainloop()}
```

21.1.2 Tkinter Widget

Tkinter menyediakan berbagai kontrol seperti tombol, label dan kotak teks yang digunakan dalam aplikasi GUI. Kontrol ini biasanya disebut widget.

Saat ini ada 15 jenis widget di Tkinter. Menyajikan widget serta penjelasan singkat pada tabel berikut ini :

Table 21.1 Ukuran

Operator	Penjelasan
Button	SMenampilkan tombol dalam aplikasi
Canvas	Menggambar bentuk seperti garis, oval, poligon dan persegi panjang dalam aplikasi
Checkbutton	Menampilkan sejumlah pilihan sebagai kotak centang. Pengguna dapat memilih beberapa pilihan pada suatu waktu
Entry	Menampilka bidang garis teks tunggal untuk menerima nilai-nilai dari pengguna
Frame	Wadah untuk mengatur widget lainnya
Label	Memberikan keterangan garis single untuk widget lainnya. Hal ini berisi gambar
Listbox	Menyediakan daftar pilihan kepada pengguna
Menubutton	Menampilkan menu dalam aplikasi
Menu	Memberikan berbagai perintah untuk pengguna. Perintah-perintah ini terkandung di dalam MenuButton
Message	Menampilkan bidang teks multiline untuk menerima nilai-nilai dari pengguna
RadioButton	Menampilkan sejumah pilihan sebagai tombol radio. Pengguna dapat memilih hanya satu pilihan pada suatu waktu
Scale	Menyediakan widget slide
Scrollbar	Menambah kemampuan bergulir ke berbagai widget seperti kotak daftar
Text	Menampilka teks dalam beberapa garis
Toplevel	Menyediakan wajah jendela terpisah
PanedWindow	Wadah yang mengandung sejumlah panel disusun horizontal atau vertikal
LabelFrame	Wadah widget sederhana. Bertindak sebagai spacer atau wajah untuk layout jendela kompleks
TkMessageBox	Menampilkan kotak pesan dalam aplikasi
Spinbox	Memilih sejumlah tetap nilai-nilai

Beberapa atribut sebagai ukuran, warna dan font ditentukan. Berikut adalah beberapa atribut standar :

- Ukuran

  Berbagai panjang, lebar, dan dimensi lain dari widget digambarkan dalam banyak unit yang berbeda seperti :

- Jika menetapkan dimensi ke integer diasumsikan dalam piksel

- Menentukan unit dengan menentukan dimensi untuk string yang berisi sejumlah diikuti oleh :

Table 21.2 Ukuran

Karakter	Penjelasan
c	Sentimeter
i	Inci
m	Milimeter
p	Poin printer

Tkinter mengungkapkan panjang sebagai integer jumlah piksel. Berikut ini adalah daftar pilihan panjang umum:

- borderwidth

  Lebar batas yang memberikan tampilan tiga dimensi untuk widget

- highlightthickness

  Lebar puncak persegi panjang ketika widget memiliki fokus

- padX padY

  Ruang tambahan widget dari manajer tata letak luar minimum widget perlu menampilkan isinya di x dan y arah

- selectborderwidth

  Lebar perbatasan tiga dimensi disekitar dipilih item widget

- wraplength

  Panjang garis maksimum untuk widget yang melakukan kata membungkus

- height

  Tinggi diinginkan widget

- underline

  Indeks karakter untuk menggarisawahi dalam teks widget

- width

- Lebar diinginkan widget
- Warna

Tkinter memiliki warna dengan string. Ada dua cara umum untuk menentukan warna di Tkiter, yaitu :

- Menggunakan string menentukan proporsi merah, hijau dan biru didigit heksadesimal. Misalnya " #ffff " putih, " #000000 " hitam dan " #000fff000 " hijau.
- Menggunakan lokal standar nama warna . warna-warna "white ", "black ", "green " dan "magenta " akan selalu tersedia.

Pilihan warna umum :

- activebackground
  Warna latar berlakang untuk widget ketika widget aktif
- activeforeground
  Warna depan untuk widget ketika widget aktif
- background
  Merepresentasikan sebagai *bg*
- disableforeground
  Warna depan untuk widget ketika widget dinonaktifkan
- foreground
  Merepresentasikan fg
- highlightbackground
  Warna latar belakang dari daerah puncak ketika widget memiliki fokus
- hightlightcolor
  Warna depan dari wilayah puncak ketika widget memiliki fokus

- selectbackground
  Warna latar belakang untuk item yang dipilih dari widget
- selectforeground
  Warna depan untuk item yang dipilih dari widget
- Font
  Sebagai tupel yang elemen pertama adalah keluarga font diikuti dengan string yang berisi satu atau lebih gaya pengubah tebal,miring, garis bawah dan overstrike.

Dapat membuat ”font object ” dengan mengimpor modul tkFont dan menggunakan kelas konstruktor font nya : Import tkFont Font = tkFont.Font (option, ...) Berikut adalah daftar pilihan :

- Family
  Font nama keluarga sebagai string
- Size
  Font tinggi sebagai integer dalam poin
- Weight
  Bold untuk teal, normal untuk berat badan secara teratur
- Slant
  Italic untuk miring, roman untuk unstlanted
- Underline
  1 untuk teks yang digarisbawahi, 0 untuk normal
- Overstrike
  1 untuk teks telak, 0 untuk normal Jika berjalan di bawah X window system, dapat menggunakan salah satu nama font X. Sebagai contoh, font bernama ”-*lucidatypewriter-medium-r-*-*-*-140-*-*-* ” adalah favorit fixed-width font penulis untuk digunakan pada layar.

- Jangkar

  Jangkar digunakan untuk mendefinisikan mana teks diposisikan relatif terhadap titik acuan.

Jika menggunakan tengah sebagai jangkar tek, tek akan ditengahkan horizontal dan vertikal disekitar titik referensi. Jangkar NW akan posisi teks sehingga titik referensi bertepatan dengan laut sudut kotak berisi teks Jangakr W akan pusat teks secara vertikal disekitar titik referensi dengan tepi kiri kotak teks yang melewati titik itu dan sebagainya. Jika membuat widget kecil didalam bingkai besar dan menggunakan jangkar = SE pilihan, widget akan ditempatkan disudut kanan bawah gambar. Jika menggunakan anchor = N sebaliknya widget akan dipusatkan disepanjang tepi atas.

Widget mengacu pada efek 3-D simulasi terbaru disekitar bagian luar widget. Berikut adalah daftar konstanta yang mungkin dapat digunakan untuk atribut:

- Datar
- Dibesarkan
- Cekung
- Alur
- Punggung bukit

Contoh :

```
From Tkinter import *}

Import Tkinter}

 top = Tkinter.Tk()}
B1 = Tkinter.Button(top, text= $ " $FLAT $ " $, relief
=FLAT)}
B2 = Tkinter.Button(top, text= $ " $RAISED $ " $, relief
=RAISED)}
```

```
B3 =Tkinter.Button(top, text= $ " $SUNKEN $ " $, relief
=SUNKEN)}
B4=Tkinter.Button(top, text= $ " $GROOVE $ " $, relief
=GROOVE)}
B5=Tkinter.Button(top, text= $ " $RIDGE $ " $, relief
=RIDGE)}

 B1.pack()}
 B2.pack()}
 B3.pack()}
 B4.pack()}
 B5.pack()}
 top.mainloop()}
```

Ada beberapa jenis bitmap yang tersedia, diantaranya:

- Kesalahan
- Gray75
- Gray50
- Gray12
- Jam Pasir
- Info
- Questhead
- Perantanyaan
- Peringatan

Contoh:

```
From Tkinter import *}

Import Tkinter}

Top = Tkinter.Tk()}
```

```
B1 = Tkinter.Button(top, text = $ " $error $ " $, relief
=RAISED,  $    $ bitmap= $ " $error $ " $)}
B2 = Tkinter.Button(top, text = $ " $hourglass $ " $, relief
=RAISED,  $    $ bitmap= $ " $hourglass $ " $)}
B3 = Tkinter.Button(top, text = $ " $info $ " $, relief
=RAISED,  $    $ bitmap= $ " $info $ " $)}
B4 = Tkinter.Button(top, text = $ " $question $ " $, relief
=RAISED,  $    $ bitmap= $ " $question $ " $)}
B5 = Tkinter.Button(top, text = $ " $warning $ " $, relief
=RAISED,  $    $ bitmap= $ " $warning $ " $)}

B1.pack()}
B2.pack()}
B3.pack()}
B4.pack()}
B5.pack()}
top.mainloop()}
```

Berikut daftar menarik :

- Panah
- Lingkaran
- Jam
- Menyebrang
- Dotbox
- Bertukar
- Fluer
- Jantung
- Manusia
- Tikus
- Bajak laut
- Tamah

- Antar jemput
- Perekat
- Laba-laba
- Kaleng semprot
- Bintang
- Target
- Tcross
- Melakukan perjalanan
- Menonton

Contoh :

```
From Tkinter import *}

Import Tkinter}

Top = Tkinter.Tk()}

B1 = Tkinter.Button(top, text = $ " $circle $ " $, relief
=RAISED,  $    $ bitmap= $ " $circle $ " $)}
B2 = Tkinter.Button(top, text = $ " $plus $ " $, relief
=RAISED,  $    $ bitmap= $ " $plus $ " $)}

B1.pack()}
B2.pack()}
font top.mainloop()}
```

21.1.3 Manajemen Geometri

Semua widget tkinter memiliki akses ke metode manajemen geometri tertentu, yang memiliki tujuan menggorganisir widget diseluruh wilayah widget induk. Tkinter mengekspos kelas manager geometri berikut :

- Metode the *pack()*

  Manajer geometri ini mengatur widget diblok sebelum menempatkan mereka di widget induk

- Metode the *grid()*

  Manajer geometri ini mengatur widget dalam struktur tabel seperti di widget induk

- Metode the *place()*

Manajer geometri ini mengatur widget dengan menempatkan dalam posisi tertentu dalam widget induk

21.1.4 Manfaat Tkinter

Tkinter sangat sederhana. Erikut manfaat Tkinter dibandingkan GUI toolkit :

- Tkinter mudah diakses oleh siapa saja (Accessibilty)

  Tkinter merupakan toolkit yang ringan dan satu-satunya solusi GUI yang paling sederhana untuk Python sampai saat ini. Cukup menuliskan beberapa baris kode Python untuk membuat aplikasi GUI sederhana dengan Tkinter. Untuk menambahkan komponen baru pada Tkinter, dapat membuatnya dalam kode Python atau menambahkan paket ekstensi seperti Pmw, Tix, atau ttk.

- Tkinter mudah digunakan di semua platform (Portability)

  Sebuah program Python yang dibangun menggunakan Tkinter dapat berjalan dengan baik di semua platform sistem operasi seperti Microsoft Windows, Linux, dan Macintosh. Dan dari segi tampilan window, akan terlihat sama dengan standar platform yang digunakan.

- Tkinter selalu tersedia di Python (Availability)

  Tkinter merupakan modul standar pada pustaka Python. Sebagian besar paket instalasi Python sudah langsung berisi Tkinter. Khusus untuk beberapa distro Linux,

perlu menambahkan paket Tkinter secara terpisah. Pada Windows, bisa langsung menggunakan Tkinter sesaat setelah menginstal paket instalasi Python.

- Dokumentasi Tkinter (Documentation)

  Python (plus Tkinter) ini bersifat open-source, maka banyak sekali komunitas-komunitas yng membahas Python dan Tkinter dan bisa belajar dan bertanya langsung dengan para ahli

21.2 Contoh GUI Programming

Dalam contoh ini, menulis naskah yang membuka jendela yang memiliki dua kotak masuk diantaranya satu nama depan berlabel dan nama belakang lainnya. Jendela memiliki dua tombol yaitu Greeting yang menampilkan kotak pesan selamat datang ke pengguna dan Close yang menutup aplikasi

```
import tkinter
from tkinter import *

det DisplayMsgBox():

    tkinter.messagebox.showinfo("Your Name", "Welcome"+
    Entry.get()+ "" +Entry2.get())

mainwindow = Tk()
Label(mainwindow, text="Firstname").grid(row=0)
Label(mainwindow, text="Lastname").grid(row=0)

Entry1 = Entry(mainwindow)
Entry2 = Entry(mainwindow)

Entry1.grid(row=0, column=1)
Entry2.grid(row=1, column=1)
Entry1.focus()

Button(mainwindow, text='Greeting', command=DisplayMsgBox)
```

```
.grid(row=3, column=0)
Button(mainwindow, text='Close', command=DisplayMsgBox)
.grid(row=3, column=1)

mainloop()
```

Ketik nama depan dan nama belakang, lalu tekan "Greeting", 21.1 Lihat Contoh dibawah ini :

Figure 21.1 Contoh

Tekan "Close" akan menghentikan aplikasi
****

Tkinter berisi sebagian besar kelas dan metode yang dibutuhkan untuk membuat aplikasi GUI yang bagus. Menggunakan kelas Tk untuk membuat master (jendela utama) dan instantiated objek untuk memasukkan berbagai kontrol pada jendela aplikasi.

CHAPTER 22

FUTHER EXPRESSION

22.1 Further Expression

Setiap kode yang dituliskan menggunakan bahasa yang dikompilasi seperti C, C++ atau Java dapat diintegrasikan ke skrip Python lainnya. Kode ini dianggap sebagai ektensi.

Modul ekstensi Python tidak lebih dari sekedar perpustakaan C biasa. Pada mesin Unix, perpustakaan ini biasanya diakhiri dengan .so (untuk objek bersama). Pada mesin windows, biasanya melihat .dll (untuk perpusatkaan yang terhubung secara dinamis).

22.1.1 Pra-Persyaratan untuk Menulis Ekstensi

Untuk memulai ektensi, memerlukan file header Python. Pada mesin Unix, biasanya memerlukan instalasi paket khusus pengembang seperti python 2-5.

Pengguna window mendapatkan header ini sebagai bagian dari paket saat menggunakan pemasang Python biner.

Harus memiliki pengetahuan yang baik tentang C atau C++ untuk menulis ekstensi Python menggunakan pemrograman C.

Untuk melihat modul ekstensi Python, perlu mengelompokkan kode menjadi empat bagian :

- File header Python
- Fungsi C yang ingin ditampilkan sebagai antarmuka dari modul
- Sebuah tabel memetakan nama-nama fungsi saat pengembang Python melihat ke fungsi C didalam modul ekstensi
- Fungsi inilisasi

Perlu menyertakan file header Python.h di file sumber C memberi akses ke API Python internal digunakan untuk menghitung modul ke penerjamah. Menyertakan header Python.h sebelum header lain yang mungkin dibutuhkan. Mengikuti termasuk dengan fungsi yang ingin dipanggil dari Python. Tanda tangan penerapan C fungsi selalu mengambil salah satu dari tiga bentuk berikut :

```
static PyObject *MyFunction( PyObject *self, PyObject *ar
gs);}

static PyObject *MyFunctionWithKeywords(PyObject *self,}
                                        PyObject *args,}
                                        PyObject *kw);}
```

```
static PyObject *MyFunctionWithNoArgs( PyObject *self );}
```

Masing-masing deklarasi seelumnya mengembalikan objek Python. Tidak ada yang namanya fungsi void dengan Python seperti ada di C. Jika ingin fungsi mengembalikan nilai, Python. Header Python mendefinisikan makro.

Nama-nama fungsi C bisa menjadi apapun yang disuka karena tidak pernah diluar modul ekstensi mendefinisikan sebagai statis.

Fungi Cbiasanya diberi nama dengan menggabnungkan modul dan fungsi Python bersama-sama yang ditunjukan disini :

```
static PyObject *{module $  \_  $func}(PyObject *self,
PyObject *args) $ \{ $

    /* Do your stuff here. */

    Py $  \_  $RETURN $  \_  $NONE;

 $  \}  $
```

Ini adalah fungsi Python yang disebut func didalam modul-modul. Memasukkan petunjuk ke fungsi C ke dalam tabel metode untuk modul yang biasanya muncul selanjutnya dikode sumber tael pemetaan metode.

Tabel metode ini adalah susunan sederhana dari struktur PyMethodSef. Struktur itu terlihat seperti ini :

```
struct PyMethodDef  $  \{  $

   char *ml $  \_  $name;

   PyCFunction ml $  \_  $meth;

   int ml $  \_  $flags;

   char *ml $  \_  $doc;
```

```
$  \}  $;
```

Ini nilai uraian anggota struktur ini :

- Ml _name

  Nama fungsi yang digunakan penafsir Python saat digunakan dalam program Python

- Ml _meth

  Menjadi alamaat ke fungsi yang memiliki salah satu tanda tangan yang dijelaskan dalam penelusuran sebelumnya

- Ml _flags

  Memberitahu penafsir yang mana dari tiga tanda tangan yang digunakan ml _meth. Bendera ini biasanya mmiliki nilai meth _varargs. Bendera ini dapat digandakan dengan ored dengan meth _keywords jika ingin memiarkan argumen kata kunci masuk ke fungsi. Ini juga bisa memiliki nilai meth _noargs yang menunjukan bahwa tidak ingin menerima argumen apa pun.

- Ml _doc

  Ini adalah docstring untuk fungsi yang bisa jadi NULL jika tidak ingin menulisnya.

Tabel ini perlu diakhiri dengan sentinel yang terdiri dari NULL dan 0 untuk anggota yang sesuai.

Contoh:

```
static PyMethodDef {module} $  \_  $methods[]
=  $  \{  $

   $  \{  $ "{func}", (PyCFunction){module $
\_  $func}, METH $  \_  $NOARGS, NULL  $  \}  $,

   $  \{  $ NULL, NULL, 0, NULL  $  \}  $
```

```
$  \}  $;
```

Bagian terakhir dari modul ekstensi adalah fungsi inialisasi. Fungsi ini dipanggil oleh juru bahasa Python saat modul diisikan. Hal ini diperlukan agar fungsi diberi nama intiModule dimana modul adalah nama modul.

Fungsi inialisasi perlu diekspor dari perpustakaan yang akan dibangun. Header Python mendefinisikan PyMODINIT _Func untuk memasukkan mantra yang sesuai agar terjadi pada lingkungan tertentu tempat menyuusun. Yang harus dilakukan adalah mengunakan saat menentukan fungsinya. Fungsi inialisasi C umumnya memiliki strktur keseluruhan berikut :

```
PyMODINIT $  \_  $FUNC init{Module}()  $  \{  $

   Py $  \_  $InitModule3({func}, {module} $  \_
   $methods, "docstring...");

 $  \}  $
```

Berikut adalah penjelasan fugsi Py _IntiModule :

- Func
  Ini adalah fungsi yang akan diekspor
- Module
  Ini adalah nama tabel pemetaan yang didefinisikan diatas
- Docstring
  Ini adalah komentar yang ingin diberikan diekstensi

Menempatkan ini semua bersama-sama terlihat sebagai berikut :

```
$  \#  $include <Python.h>
```

```
static PyObject *{module $  \_  $func}(PyObject *self,
PyObject *args)  $  \{  $

   /* Do your stuff here. */

   Py $  \_  $RETURN $  \_  $NONE;

 $  \}  $

static PyMethodDef {module} $  \_  $methods[] =  $
\{  $

    $  \{  $ "{func}", (PyCFunction){module $  \_
    $func}, METH $  \_  $NOARGS, NULL  $  \}  $,

    $  \{  $ NULL, NULL, 0, NULL  $  \}  $

 $  \}  $;

PyMODINIT $  \_  $FUNC init {Module}()  $  \{  $

   Py $  \_  $InitModule3({func}, {module} $  \_
   $methods, "docstring...");

 $  \}  $
```

Contoh:

```
 $  \#  $include <Python.h>

static PyObject* helloworld(PyObject* self)

 $  \{  $

   return Py $  \_  $BuildValue("s", "Hello, Python
   extensions!!");

 $  \}  $
```

```
static char helloworld $  \_  $docs[] =
static PyMethodDef helloworld $  \_  $funcs[] =  $
\{  $

    $  \{  $"helloworld", (PyCFunction)helloworld,

    METH $  \_  $NOARGS, helloworld $  \_  $docs $
    \}  $,

    $  \{  $NULL $  \}  $

 $  \}  $;

void inithelloworld(void)

 $  \{  $

   Py $  \_  $InitModule3("helloworld", helloworld $
   \_  $funcs,

              "Extension module example!");

 $  \}  $
```

Disini fungsi Py_BuildValue digunakan untuk membangun nilai Python.

22.1.2 Membangun dan Menginstal Ekstensi

Paket membuatnya sangat mudah mendistribusikan modul Python, baik Python murni dan modul ekstensi dengan cara standar. Modul didistribusikan dalam bentuk sumber dan dibangun dan diinstal melalui skrip setup yang iasa disebut setup.py sebagai berikut :

```
from distutils.core import setup, Extension \par
      ext $  \_  $modules=[Extension('helloworld',
      ['hello.c'])]).
```

Sekarang gunakan perintah berikut yang aka melalakukan semua kompilasi dan langkah penghubunh yang diperlukan dengan perintah dan bendera penyusun dan penghubung yang benar dan menyalin perpustakaan dinamis yang dihasilkan ke dalam direktori yang sesuai .

Contoh :

```
$  \$  $ python setup.py install
```

Pada sistem berbasis Unix kemungkinan besar perlu menjalankan perintah ini sebagai root agar meminta izin untuk menulis ke direktori paket situs. Ini biasanya tidak menjadi masalah pada window. Setelah menginstal ekstensi, akan dapat mengimpor dan memanggil ekstensi tersebut di skrip Python sebagai berikut :

```
$  \#  $!/usr/bin/python

  import helloworld

  print helloworld.helloworld()
```

Ini akan menghasilkan hasil sebagai berikut : Hello, Python extensions!!

Seperti kemungkinan besar ingin mendefinisikan fungsi yang menerima argumen, dapat menggunakan salah satu tanda tangan lain untuk fungsi C. Sebagai contoh, fungsi berikut yang menerima beberapa parameter akan didefinisikan seperti ini :

```
static PyObject *{module $  \_  $func}(PyObject
*self, PyObject *args)  $  \{  $

    /* Parse args and do something interesting here. */

    Py $  \_  $RETURN $  \_  $NONE;

 $  \}  $
```

Tabel metode yang berisi entri untuk fungsi baru akan terlihat seperti ini :

```
static PyMethodDef {module} $  \_  $methods[] =  $
\{  $

    $  \{  $ "{func}", (PyCFunction) {module
    $  \_  $func}, METH $  \_  $NOARGS, NULL  $  \}  $,

    $  \{  $ "{func}", {module $  \_  $func
    }, METH $  \_  $VARARGS, NULL  $  \}  $,

    $  \{  $ NULL, NULL, 0, NULL  $  \}  $

 $  \}  $;
```

Menggunakan fungsi API PyArg _ParseTuple untuk mengekstrak argumen dari satu pointer PyObject yang dikirimkan ke fungsi C. Argumen pertama untuk PyArg _ParseTuple adalah args argumen. Ini adalah objek yang akan parsing. Argumen kedua adalah string format yang menggambarkan argumen saat mengharapkannya muncul. Setiap argumen diwakili oleh satu atau lebih karakter dalam format string sebagai berikut :

```
static PyObject *{module $  \_  $func}(PyObject *self,
PyObject *args)  $  \{  $

    int i;

   double d;

   char *s;

   if (!PyArg $  \_  $ParseTuple(args, "ids",  $  \&  $i,
   $  \&  $d,  $  \&  $s))  $  \{  $

   return NULL;
```

```
$  \}  $

/* Do something interesting here. */

Py $  \_  $RETURN $  \_  $NONE;

$  \}  $
```

Mengkompilasi versi baru dari modul dan mengimpornya memungkinkan untuk memanggil fungsi baru dengan sejumlah argumen dari jenis apa pun :

```
module.func(1, s="three", d=2.0)

module.func(i=1, d=2.0, s="three")

module.func(s="three", d=2.0, i=1)
```

Berikut adalah tanda tangan standar untuk fungsi PyArg _ParseTuple:

```
int PyArg $  \_  $ParseTuple(PyObject* tuple,char* format,...)
```

Fungsi ini mengembalikan 0 untuk kesalahan, dan nilai tidak sama dengan 0 untuk kesuksesan. Tuple adalah PyObject * yang merupakan argumen kedua dari fungsi C. Format berikut adalah string C yang menggambarkan argumen wajib dan opsional. Berikut adalah daftar kode format untuk fungsi PyArg _ParseTuple:

Table 22.1 Ukuran

Karakter	Penjelasan
c	Sentimeter
i	Inci
m	Milimeter
p	Poin printer

Py _BuildValue mengambil format string seperti PyArg _ParseTuple. Alih-alih menyampaikan alamat nilai yang sedang bangun, melewati nilai sebenarnya. Berikut adalah contoh yang menunjukkan bagaimana menerapkan fungsi tambah :

```
static PyObject *foo $  \_  $add(PyObject *self, PyObject
*args)  $  \{  $

   int a;

   int b;
\vspace{12pt}

   if (!PyArg $  \_  $ParseTuple(args, "ii",  $  \&  $a,
   $  \&  $b))  $  \{  $

    return NULL;

    $  \}  $

    return Py $  \_  $BuildValue("i", a + b);

 $  \}  $

 def add(a, b):

 return (a + b)

 static PyObject *foo $  \_  $add $  \_  $subtract(PyObject
 *self, PyObject *args)  $  \{  $

  int a;

  int b;

  if (!PyArg $  \_  $ParseTuple(args, "ii",  $  \&  $a,  $
  \&  $b))  $  \{  $
```

```
  return NULL;

  $  \}  $

  return Py $  \_  $BuildValue("ii", a + b, a - b);

$  \}  $

  def add $  \_  $subtract(a, b):

  return (a + b, a - b)
```

Berikut daftar tabel string kode yang umum digunakan, yang nol atau lebihnya digabungkan ke dalam format string :

Table 22.2 Ukuran

Code	C Type	Meaning
c	char	String Python dengan panjang 1 menjadi huruf C.
d	double	Pelampung Python menjadi C ganda.
f	float	Pelampung Python menjadi pelampung C.
i	int	Int Python menjadi int int
l	long	Sebuah int Python menjadi panjang C.
L	long long	Sebuah int Python menjadi C panjang panjang
O	PyObject*	Gets non-NULL meminjam referensi ke argumen Python
s	char*	Python string tanpa nulls tertanam ke C char *
s	char*+int	Setiap string Python ke alamat dan panjang C
t	char*+int	Read-only penyangga segmen tunggal ke alamat C dan panjangnya
U	PyUNICODE*	Python Unicode tanpa nulls tertanam ke C
u	PPyUNICODE*int+	Setiap alamat dan panjang Python Unicode C
w	char*+int	Membaca / menulis penyangga segmen tunggal ke alamat dan panjang C
Z	char*	Seperti s, juga menerima None (set C char * ke NULL).
z	char*+int	Seperti s, juga menerima None set C char * ke NULL
(...)	Poin printer	Urutan Python diperlakukan sebagai satu argumen per item.
—	as per ...	Argumen berikut bersifat opsional.
:		Format akhir, diikuti dengan nama fungsi untuk pesan error.
;		Format akhir, diikuti oleh seluruh pesan kesalahan teks.

Kode {... } membangun kamus dari sejumlah nilai C, kunci dan nilai bergantian. Misalnya, Py _BuildValue (" {issi }", 23, "zig", "zag", 42) mengembalikan kamus seperti {23: 'zig', 'zag': 42 } Python. Setiap blok memori yang dialokasikan dengan malloc () pada akhirnya harus dikembalikan ke genangan memori yang tersedia dengan satu panggilan untuk membebaskan (). Penting untuk menelepon gratis () pada waktu yang tepat. Jika alamat blok dilupakan tapi gratis () tidak dipanggil untuk itu, memori yang ditempatinya tidak dapat digunakan kembali sampai program berakhir. Ini disebut kebocoran memori. Di sisi lain, jika sebuah program memanggil gratis () untuk satu blok dan kemudian terus menggunakan blok tersebut, itu menciptakan konflik dengan penggunaan ulang blok melalui panggilan malloc () yang lain. Ini disebut dengan menggunakan memori yang dibebaskan. Ini memiliki konsekuensi buruk yang sama seperti merujuk pada data yang tidak diinisiasi - dump inti, hasil yang salah, crash misterius. Karena Python membuat penggunaan malloc () dan gratis (), dibutuhkan strategi untuk menghindari kebocoran memori dan juga penggunaan memori yang bebas. Metode yang dipilih disebut penghitungan referensi. Prinsipnya sederhana: setiap objek berisi sebuah counter, yang bertambah saat referensi ke objek disimpan di suatu tempat, dan yang dikurangi saat referensi itu dihapus. Saat counter mencapai nol, referensi terakhir ke objek telah dihapus dan objeknya dibebaskan.

22.2 Simbol Further Expression

Simbol khusus yang paling penting adalah garis miring terbalik yang digunakan untuk dua tujuan. Pertama, backslash dapat digunakan untuk membuat lebih banyak karakter meta dalam ekspresi reguler. Kedua, jika simbol khusus diawali dengan garis miring terbalik, maka artinya khusus akan dihapus dan dengan demikian menghasilkan kecocokan literal dari simbol khusus tersebut. Garis miring terbalik menciptakan beberapa masalah karena merupakan simbol khusus dengan ekspresi Python dan juga string

Python. Ekspresi hanya akan menghasilkan kecocokan untuk .⌢ Untuk mengatasi masalah ini, ekspresi Python menggunakan notasi string mentah yang membuat ekspresi tetap sederhana. Dalam notasi string mentah, setiap string ekspresi diawali dengan r sehingga tidak perlu menambahkan backslash beberapa kali.Berikut 22.1 SimbolFurhertExpression :

```
>>> import re
>>> pat = re.compile('a{3}')
>>> execfile('run.py')
Enter File Name to Process: file2.txt
aaa
aaa
aaa
```

Figure 22.1 SimbolFurhertExpression

Simbol adalah operator atau ekspresi biasa. Simbol khusus, tanda kurung tutup dan penutup, digunakan untuk mencari pola berulang.

REFERENCES

1. B. Santoso, G. Serpong, and I. Tangerang, "Bahasa pemrograman python di platform gnu/linux," *Jurnal Program Studi Teknik Informatika, Fakultas Teknologi Informasi dan Komunikasi Universitas Multimedia Nusantara Gading Serpong Tanggerang*, 2009.
2. S. Suparno, D. Fisika-FMIPA, and U. Indonesia, "Komputasi untuk sains dan teknik," *Departemen Fisika-FMIPA Universitas Indonesia*, 2013.
3. G. Van Rossum *et al.*, "Python programming language." in *USENIX Annual Technical Conference*, vol. 41, 2007, p. 36.
4. T. E. Oliphant, "Python for scientific computing," *Computing in Science & Engineering*, vol. 9, no. 3, 2007.
5. M. J. N. Yudianto, "Jaringan komputer dan pengertiannya," *Jurnal Komunitas elearning IlmuKomputer. Com*, 2007.