

SURVEY METHODOLOGY

SURVEY METHODOLOGY

This is the Subtitle

Robert M. Groves

Universitat de les Illes Balears

Floyd J. Fowler, Jr.

University of New Mexico



A JOHN WILEY & SONS, INC., PUBLICATION

Copyright ©2007 by John Wiley & Sons, Inc. All rights reserved.

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

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services please contact our Customer Care Department with the U.S. at 877-762-2974, outside the U.S. at 317-572-3993 or fax 317-572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print, however, may not be available in electronic format.

Library of Congress Cataloging-in-Publication Data:

Survey Methodology / Robert M. Groves . . . [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

To my parents

CONTRIBUTORS

MASAYKI ABE, Fujitsu Laboratories Ltd., Fujitsu Limited, Atsugi, Japan

L. A. AKERS, Center for Solid State Electronics Research, Arizona State University,
Tempe, Arizona

G. H. BERNSTEIN, 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

PART I SUBMICRON SEMICONDUCTOR MANUFACTURE

1	Home	3
2	Overview	5
3	Environment Setup	7
4	Basic Syntax	9
5	Variable Type	11
6	Basic Operator	13
7	Decision Making	15
8	Loop	17
9	Numbers	19
10	Strings	21
11	Lists	23
12	Tuples	25
13	Dictionary	27
14	Functions	29
		vii

15 Modules	31
16 Files I/O	33
17 Exceptions	35
18 Classess/Object	37
19 Reg Expression	39
20 CGI Programming	41
21 Databases Access	43
22 Sending Email	45
23 Multithreading	47
24 XML Processing	57
25 GUI Programming	67
26 Futher Expression	69

CONTENTS

List of Figures	xiii
List of Tables	xv
Foreword	xvii
Preface	xix
Acknowledgments	xxi
Acronyms	xxiii
Glossary	xxv
List of Symbols	xxvii
Introduction	xxix
<i>Catherine Clark, PhD.</i>	
References	xxix

PART I SUBMICRON SEMICONDUCTOR MANUFACTURE

1 Home	3
	ix

2	Overview	5
3	Enviromtment Setup	7
4	Basic Syntax	9
5	Variabel Type	11
6	Basic Operator	13
7	Desicion Making	15
8	Loop	17
9	Numbers	19
10	Strings	21
11	Lists	23
12	Tuples	25
13	Dictionary	27
14	Functions	29
15	Modules	31
16	Files I/O	33
17	Exceptions	35
18	Clasess/Object	37
19	Reg Expression	39
20	CGI Programming	41

21	Databases Access	43
22	Sending Email	45
23	Multithreading	47
24	XML Processing	57
25	GUI Programming	67
26	Futher Expression	69
	References	71
	References	73

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 during 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 numerically 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 [1]. His first chip is shown in Fig. I. For comparison, Fig. I.2 shows a modern microprocessor chip, [4].

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

$$ABC\mathcal{DE}\mathcal{F}\alpha\beta\Gamma\Delta\sum_{def}^{abc} \tag{I.1}$$

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

CHAPTER 2

OVERVIEW

CHAPTER 3

ENVIRONMENT SETUP

CHAPTER 4

BASIC SYNTAX

CHAPTER 5

VARIABLE TYPE

CHAPTER 6

BASIC OPERATOR

CHAPTER 7

DESICION MAKING

CHAPTER 8

LOOP

CHAPTER 9

NUMBERS

CHAPTER 10

STRINGS

CHAPTER 11

LISTS

CHAPTER 12

TUPLES

CHAPTER 13

DICTIONARY

CHAPTER 14

FUNCTIONS

CHAPTER 15

MODULES

CHAPTER 16

FILES I/O

CHAPTER 17

EXCEPTIONS

CHAPTER 18

CLASSESS/OBJECT

CHAPTER 19

REG EXPRESSION

CHAPTER 20

CGI PROGRAMMING

CHAPTER 21

DATABASES ACCESS

CHAPTER 22

SENDING EMAIL

CHAPTER 23

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 (interrupted)*
- Untuk sementara dapat ditunda sementara *thread* lainnya yang sedang berjalan ini disebut unggul.

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 (threadName, 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.

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 mengekspos 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 mengembalikan nama *thread*
- **setName()**
Metode menetapkan nama *thread*

1.3 Membuat *Thread* Menggunakan *Threading* 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 sebuah instance dari itu dan kemudian memulai *thread* baru dengan menerapkan *start()*, yang ada gilirannya 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

```

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 memungkinkan 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)
threads.append(thread2)

# Wait for all threads to complete
for 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
```

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:
```

```
        queueLock.acquire()
```

```
        if not workQueue.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

```

CHAPTER 24

XML PROCESSING

Python XML Processing

XML adalah bahasa open source portable yang memungkinkan pemrogram mengembangkan aplikasi yang dapat dibaca oleh aplikasi lain, terlepas dari sistem operasi dan bahasa pengembangannya.

Apa itu XML?

Extensible Markup Language (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.

2.1 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 memarsing 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 bekerjasama dengan file besar. Di sisi lain, menggunakan DOM secara eksklusif benar-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;

```

2.2 Parsing XML dengan API SAX

SAX adalah antarmuka standar untuk parsing XML berbasis event. Parsing XML dengan SAX umumnya mengharuskan untuk membuat *ControlHandler* 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 memarsing 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 *startElementNS* dan *endElementNS* dipanggil. Disini, tag adalah elemen dan atribut adalah atribut.

Berikut ini metode penting untuk memahami sebelum melanjutkan ke materi berikutnya :

Metode

Metode berikut membuat objek parsing baru dan mengembalikannya. Objek parsing dibuat akan menjadi tipe parsing pertama yang ditemukan sistem.

```
xml.sax.make_parser([parser_list])
```

Berikut adalah detail parameternya :

Parser *_list* : pilihan argumen 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 *ContentHandler*

- *ErrorHandler*

Jika ditentukan, *errorhandler* harus menjadi objek *ErrorHandler* SAX

- Metode *parseString*

Membuat parsing SAX dan mengurai string XML yang ditentukan :

```
xml.sax.parsestring(xmlstring, contenthandler[, errorhandler])
```

Berikut ini adalah detail nama dan parameter :

XMLstring

Nama dari string yang bisa dibaca

ContentHandler

Menjadi objek ContentHandler

ErrorHandler

Menjadi objek ErrorHandler 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 element 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 namespaces
    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

2.3 Parsing XML dengan API DOM

Document Object 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.

```
#!/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

2.4 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 digunakan saat sebuah dokumen diurai untuk mengetahui bagaimana menghasilkan bentuk serial dari isinya kemudian dapat ditulis ke sebuah file.

Untuk membuat instance elemen gunakan fungsi elemen constructor dan *SubElement()* pabrik.

Import `xml.etree.ElementTree` as `xml`

```
filename = "/home/abc/Desktop/test_xml.xml "
```

```
root = xml.Element( "Users "
```

```
    uselement = xml.Element( "user "
```

```
root.append(uselement)
```

Bila menjalankan ini, akan menghasilkan sebagai berikut :

```
<Users>
  <user>
  <user>
</Users>
```

Tambahkan anak-anak pengguna

```
Uid = xml.SubElement(uselement, "uid ")
```

```
Uid.text = "1 "
```

```
FirstName = xml.SubElement(uselement, "FirstName ")
```

```
FirstName.text = "testuser "
```

```
LastName = xml.SubElement(uselement, "LastName "
```

```
LastName.text = "testuser "
```

```
Email = xml.SubElement(uselement, "Email ")
```

```
Email.text = "mailto:testuser@test.comtestuser@test.com"
```

```
state = xml.SubElement(uselement, "state ")
```

```
state.text = "xyz "
```

```
location = xml.SubElement(uselement, "location")
```

```
location.text = "abc"
```

```
tree = xml.ElementTree(root)
```

```
with open(filename, "w ") as fh:
```

```
tree.write(fh)
```

Pertama buat elemen root dengan menggunakan fungsi *ElementTree*. Kemudian membuat elemen pengguna dan menambahkannya ke root. Selanjutnya membuat

SubElement dengan melewati elemen pengguna (userelement) ke *SubElement* beserta namanya seperti "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 Document :

```

import xml.etree.ElementTree as ET
tree = ET.parse(Your_XML_file_path)
root = tree.getroot()

```


Disini *getroot()* akan mengembalikan elemen dari dokumen XML

```
¡Users version=" 1.0 " language=" SPA "¿  
  ¡user¿  
    ¡uid¿1¡/uid¿  
    ¡FirstName¿testuser¡/FirstName¿  
    ¡LastName¿testuser¡/LastName¿  
    ¡Email¿testuser@tes.com/Email¿  
    ¡state¿xyz¡/state¿  
    ¡location¿abc¡/location¿  
  ¡/user¿  
¡/Users¿
```

CHAPTER 25

GUI PROGRAMMING

CHAPTER 26

FUTHER EXPRESSION

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).
4. A. Berenbaum, B. W. Colbry, D.R. Ditzel, R. D Freeman, and K.J. O'Connor, "A Pipelined 32b Microprocessor with 13 kb of Cache Memory," in Int. Solid State Circuit Conf., Dig. Tech. Pap., p. 34 (1987).

REFERENCES

- [Kil76] J. S. Kilby, "Invention of the Integrated Circuit," *IEEE Trans. Electron Devices*, **ED-23**, 648 (1976).
- [Ham62] R. W. Hamming, *Numerical Methods for Scientists and Engineers*, Chapter N-1, McGraw-Hill, New York, 1962.
- [Hu86] 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).
- [Ber87] A. Berenbaum, B. W. Colbry, D.R. Ditzel, R. D Freeman, and K.J. O'Connor, "A Pipelined 32b Microprocessor with 13 kb of Cache Memory," in Int. Solid State Circuit Conf., Dig. Tech. Pap., p. 34 (1987).

