

Pelatihan Python : Standard Library



Divisi Riset POSS – UPI

Sabtu, 19 Oktober 2013

Lab Umum Ilmu Komputer. Gedung FPMIPA – C

Universitas Pendidikan Indonesia

Mengapa menggunakan STL ?

- Mempermudah programmer Python dalam mengembangkan perangkat lunak yang tidak tergantung platform.
- Menjadi abstraksi agar programmer Python tidak membuat ulang fungsional yang sama padahal sudah disediakan di STL.
- Merupakan kumpulan dari solusi – solusi standard yang bisa digunakan dalam pengembangan perangkat lunak berbasis Python.
- Memudahkan pengembangan perangkat lunak
- Biasanya memiliki modul khusus untuk sistem operasi tertentu. Jika ingin mengembangkan perangkat lunak yang spesifik.
- Ada yang sudah built-in ada juga yang harus diimport.

Modul - Modul STL di Python

String Services

Data Types

**Numeric and
Mathematical**

**File and Directory
Access**

**Data
Persitence**

**Data Compression
And Archiving**

**File
Formats**

**Generic Operating
System Service**

**Internet Data
Handling**

**Interprocess Communication
And Networking**

**Optional Operating
System Service**

**Structured Markup
Processing Tool**

**Internet Protocols and
Support**

Modul - Modul STL di Python

**Multimedia
Services**

**Importing
Modules**

**Program
Frameworks**

**Development
Tools**

**Restricted
Execution**

**Debugging and
Profiling**

**Python languages
services**

**Graphical User Interaces
With Tk**

**Structured Markup
Processing Tool**

**Python Compiler
Package**

**Miscellaneous
Services**

**Undocumented
Modules**

Modul - Modul STL Spesifik pada Sistem Operasi Tertentu

**MS Windows
Specific Services**

Unix Specific Services

**Mac OS X Specific
Services**

**SGI IRIX Specific
Services**

**SunOS Specific
Services**

**MacPython OSA
Modules**

Mari kita coba beberapa module
STL di Python :D

Time

Generic Operating System Services

Beberapa contoh method dasar yang akan digunakan :

- **ctime()**, mendapatkan waktu saat ini atau konversi detik menjadi waktu yang berformat
- **time()**, mendapatkan waktu saat ini dalam bentuk detik
- **strptime()**, mendapatkan bagian – bagian dari waktu
- **strftime()**, mencetak waktu sesuai dengan format yang diinginkan

Time

Generic Operating System Services

Catatan : simpan file berikut dengan nama utakatikjam.py

```
import time

# memanggil waktu sekarang
print 'sekarang jam : ', time.ctime()
print

# menambahkan 60 detik berikutnya
next_time = time.time() + 60
print '60 detik berikutnya adalah jam : ', time.ctime(next_time)
print

# jam yang terstruktur
jam_mentah = time.strptime(time.ctime())
print 'jam mentah : ', jam_mentah
print

# berbagai format waktu
print "tanggal dalam bentuk lengkap : ", time.strftime("%A, %d %B %Y", jam_mentah)
print "tanggal dalam bentuk singkat : ", time.strftime("%a, %d/%m/%y", jam_mentah)
print "jam dalam bentuk 24 jam : ", time.strftime("%H:%M:%S", jam_mentah)
print "jam dalam bentuk AM/PM : ", time.strftime("%I:%M:%S %p", jam_mentah)
```


Math

Numerical and Mathematical Modules

Beberapa contoh method dasar yang akan digunakan :

- **pow()**, memangkatkan suatu angka dengan pangkat yang diinginkan
- **sqrt()**, mendapatkan nilai akar dari suatu angka
- **acos()**, mendapatkan nilai acosinus dari sebuah nilai
- **degrees()**, mendapatkan nilai sudut
- **radians()**, mengubah sudut menjadi nilai radian

Math

Numerical and Mathematical Modules

Contoh kasus mencari sudut dari dua buah vektor :

$$\cos \text{ teta} = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$$

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2$$

$$\|\mathbf{u}\|^2 = u_1^2 + u_2^2$$

$$\|\mathbf{v}\|^2 = v_1^2 + v_2^2$$

Math

Numerical and Mathematical Modules

Catatan : simpan file berikut dengan nama mencarisudut.py

```
import math

def get_degrees(a, b):
    print a
    print b

    sum_a = math.pow(a[0], 2) + math.pow(a[1], 2)
    len_a = math.sqrt(sum_a)

    sum_b = math.pow(b[0], 2) + math.pow(b[1], 2)
    len_b = math.sqrt(sum_b)

    adotb = a[0]*b[0] + a[1]*b[1]

    costeta = adotb / (len_a * len_b)
    teta = math.degrees(math.acos(costeta))

    return teta
```

Math

Numerical and Mathematical Modules

Catatan : simpan file berikut dengan nama mencarisudut.py

```
-----  
a = [10, 10]  
b = [30, 10]  
temp = []  
temp_x, temp_y = a[0], a[1]  
speed = 1  
  
teta = get_degrees(a, b)  
rad = math.radians(teta)  
print "teta : ", teta  
print "radians : ", rad
```

Random

Numerical and Mathematical Modules

Beberapa contoh method dasar yang akan digunakan :

- **randint()**, mendapatkan nilai random dari rentang yang telah kita tentukan.

Random

Numerical and Mathematical Modules

Catatan : simpan file berikut dengan nama tebakangka.py

```
import random

angka_ajaib = random.randint(1, 10)

angka_tebakan = raw_input('Hai manusia, akan kuberi kau satu permintaan\nberapa tebakanku angka tebakanku :D ? ')
jml_tebakan = 1
tanya = True
while tanya:
    if angka_ajaib == int(angka_tebakan):
        print "Anda mendapat hadiah dengan pajak ", jml_tebakan
        tanya = False
    elif int(angka_tebakan) > angka_ajaib:
        print "Kelebihan - kelebihan.."
        angka_tebakan = raw_input('Hai anakku, akan kuberi satu permintaan\nberapa tebakanku angka tebakanku :D ? ')
        jml_tebakan += 1
    elif int(angka_tebakan) < angka_ajaib:
        print "Kurang - kurang.."
        angka_tebakan = raw_input('Hai anakku, akan kuberi satu permintaan\nberapa tebakanku angka tebakanku :D ? ')
        jml_tebakan += 1
```

String

Built-in Type

Beberapa contoh method dasar yang akan digunakan :

- **capitalize()**, membuat kata pertama menjadi huruf kapital
- **center()**, menambahkan karakter tambahan sebelum huruf pertama
- **upper()**, mengubah semua huruf menjadi huruf besar
- **lower()**, mengubah semua huruf menjadi huruf kecil
- **title()**, mengubah huruf pertama pada setiap kata menjadi huruf besar
- **count()**, menghitung substring pada sebuah kalimat
- **replace()**, mengubah substring pada sebuah kalimat dengan substring baru
- **find()**, mencari substring

String

Built-in Type

Catatan : simpan file berikut dengan nama bermainkalimat.py

```
sebuah_kalimat = "Dunia tak selebar daun kelor."  
  
# cara mengakses huruf  
print sebuah_kalimat[0:5]  
  
# cara menambah kalimat  
sebuah_kalimat += "Dan langit tak setinggi jengkal."  
print sebuah_kalimat  
  
# membuat kata pertama menjadi huruf kapital  
print sebuah_kalimat.capitalize()  
  
# menambahkan karakter tambahan sebelum huruf pertama  
print sebuah_kalimat.center(len(sebuah_kalimat) + 10)  
  
# mengubah semua huruf menjadi huruf besar  
print sebuah_kalimat.upper()  
  
# mengubah semua huruf menjadi huruf kecil  
print sebuah_kalimat.lower()
```

String

Built-in Type

Catatan : simpan file berikut dengan nama bermainkalimat.py

```
# mengubah huruf pertama pada setiap kata menjadi huruf besar
print sebuah_kalimat.title()

# menghitung substring pada sebuah kalimat
print sebuah_kalimat.count("tak")

# mengubah substring pada sebuah kalimat dengan substring baru
print sebuah_kalimat.replace("tak", "tidak")

# mencari substring
print sebuah_kalimat.find("daun")
```

XML DOM Minidom

Structured Markup Processing Tools

Beberapa contoh objek yang akan digunakan :

- **Document()**, membuat dokumen XML baru.
- **childNodes**, list yang berisi daftar element anak pada element

Beberapa contoh method yang akan digunakan :

- **appendChild()**, menambahkan elemen pada element atau dokumen
- **createComment()**, menulis komen pada dokumen atau elemen
- **createElement()**, menulis elemen root
- **createTextNode()**, menulis teks pada
- **toprettyxml()**, menulis dokumen xml di konsol dan berupa string
- **parse()**, membaca file xml
- **getElementsByTagName()**, membaca tag di dokumen atau setiap element
- **documentElement()**, membaca element root pada xml

XML DOM Minidom

Structured Markup Processing Tools

Catatan : simpan file berikut dengan nama tulisxml.py

```
from xml.dom import minidom, Node

tokoh_anime = [
    {'nama' : 'Monkey D. Luffy', 'anime':'One Piece', 'rating':'8'},
    {'nama' : 'Rivaille', 'anime':'Shingeki No Kyojin', 'rating':'9'},
    {'nama' : 'Uzumaki Naruto', 'anime':'Naruto', 'rating':'8'},
    {'nama' : 'Chinmi', 'anime':'Kungfu Boy', 'rating':'7'},
    {'nama' : 'Kenshin Himura', 'anime':'Samurai X', 'rating':'9'},
    {'nama' : 'Toriko', 'anime':'Toriko', 'rating':'7'},
    {'nama' : 'Sasuke', 'anime':'Naruto', 'rating':'9'},
    {'nama' : 'Sagara Sasuke', 'anime':'Full Metal Panic', 'rating':'8'}
]

doc = minidom.Document()
doc.appendChild(doc.createComment("Tokoh Anime"))

koleksi = doc.createElement('koleksi')
doc.appendChild(koleksi)
```

XML DOM Minidom

Structured Markup Processing Tools

Catatan : simpan file berikut dengan nama tulisxml.py

for karakter in tokoh_anime:

```
tokoh = doc.createElement('tokoh')
koleksi.appendChild(tokoh)
```

```
nama = doc.createElement('nama')
nama.appendChild(doc.createTextNode(karakter['nama']))
tokoh.appendChild(nama)
```

```
anime = doc.createElement('anime')
anime.appendChild(doc.createTextNode(karakter['anime']))
tokoh.appendChild(anime)
```

```
rating = doc.createElement('rating')
rating.appendChild(doc.createTextNode(karakter['rating']))
tokoh.appendChild(rating)
```

```
print doc.toprettyxml(indent = '  ')
```

```
f = open('tokohanime.xml', 'w')
f.write(doc.toprettyxml(indent='  '))
f.close()
```

XML DOM Minidom

Structured Markup Processing Tools

Catatan : simpan file berikut dengan nama bacaxml.py

```
from xml.dom.minidom import parse
import xml.dom.minidom

tokoh_anime = xml.dom.minidom.parse('tokohanime.xml')
koleksi = tokoh_anime.documentElement

daftar_tokoh = koleksi.getElementsByTagName("tokoh")

print "Daftar Tokoh Anime \n-----\n"

for tokoh in daftar_tokoh:
    nama = tokoh.getElementsByTagName('nama')[0]
    print "Nama : %s" % nama.childNodes[0].data.strip()

    anime = tokoh.getElementsByTagName('anime')[0]
    print "Anime : %s" % anime.childNodes[0].data.strip()

    rating = tokoh.getElementsByTagName('rating')[0]
    print "Rating : %s" % rating.childNodes[0].data.strip()

print
```

ZipFile

Data Compression and Archiving

Beberapa contoh objek dasar yang akan digunakan :

- **ZipFile()**, membuat objek file zip baru yang akan digunakan untuk membaca atau menulis file zip.

Beberapa contoh method dasar yang akan digunakan :

- **write()**, memasukkan file kedalam file zip
- **close()**, mengakhiri proses penulisan atau pembacaan file zip
- **namelist()**, mendapatkan daftar file yang ada di dalam file zip
- **read()**, membaca file yang ada di dalam file zip

ZipFile

Data Compression and Archiving

Catatan : simpan file berikut dengan nama ngepakfile.py

```
import zipfile

# membuat dulu beberapa file
try:
    fo = open("perabotan.txt", "w")
    fo.write("1. kemoceng, sapu, lap pel\n")
    fo.write("2. ember, jolang, baskom\n")
    fo.write("3. piring, mangkok, pinggan\n")
    fo.write("4. sendok, garpu, spatula\n")
except IOError, e:
    print "terjadi error : ", e
finally:
    fo.close()

try:
    fo = open("makanan.txt", "w")
    fo.write("1. bolu, brownies, bika ambon\n")
    fo.write("2. nastar, bulan, kacang\n")
    fo.write("3. kupat tahu, ketoprak, lontong kari\n")
    fo.write("4. lotek, gado - gado, karedok\n")
except IOError, e:
    print "terjadi error : ", e
finally:
    fo.close()
```

ZipFile

Data Compression and Archiving

Catatan : simpan file berikut dengan nama ngepakfile.py

```
-----  
# melakukan kompresi terhadap kedua file diatas  
print "lagi ngepak filenya dulu yah :D ..."  
hasil_ngepak = zipfile.ZipFile('hasil_ngepak.zip', mode='w')  
try:  
    print "menambakan file perabotan.txt"  
    hasil_ngepak.write('perabotan.txt')  
    print "menambakan file makanan.txt"  
    hasil_ngepak.write('makanan.txt')  
finally:  
    print "selesai ..."  
    hasil_ngepak.close()
```


ZipFile

Data Compression and Archiving

Catatan : simpan file berikut dengan nama ngekstrakzip.py

```
import zipfile

hasil_ekstrak = zipfile.ZipFile('hasil_ngepak.zip')
for fname in hasil_ekstrak.namelist():
    try:
        fcontent = hasil_ekstrak.read(fname)
    except (IOError, KeyError, IndexError), e:
        print "Terjadi ERROR : ", e
    else:
        print fname, ' : '
        print repr(fcontent)
        try:
            fo = open(fname, "w")
            fo.write(fcontent)
        except IOError, e:
            print "terjadi error : ", e
        finally:
            fo.close()
print
```

Socket

Interprocess Communication and Networking

Beberapa contoh objek dasar yang akan digunakan :

- **socket()**, membuat objek socket

Beberapa contoh method dasar yang akan digunakan :

- **gethostbyaddr()**, mendapatkan data host, alias, dan adress dari sebuah IP Address
- **bind()**, memasang socket IP Address
- **listen()**, menyalakan socket dalam bentuk server
- **accept()**, menangkap koneksi yang datang
- **recv()**, membaca data dari koneksi yang datang
- **send()**, mengirim data lewat socket
- **close()**, menutup socket
- **connect()**, koneksi ke socket yang sedang aktif

Socket

Interprocess Communication and Networking

Catatan : simpan file berikut dengan nama chating-server.py

```
import socket

s = socket.socket()
#host = socket.gethostname()
host, alias, addr = socket.gethostbyaddr('127.0.0.1')
port = 50000
s.bind((host, port))

s.listen(5)
while True:
    c, addr = s.accept()
    print 'Got connection from ', addr
    while True:
        print "from client : ", c.recv(1024)
        pesan = raw_input("server : ")
        c.send(pesan)

c.close()
```

Socket

Interprocess Communication and Networking

Catatan : simpan file berikut dengan nama chating-client.py

```
import socket

s = socket.socket()
host, alias, addr = socket.gethostbyaddr('127.0.0.1')
port = 50000

s.connect((host, port))
while True:
    pesan = raw_input("client : ")
    s.send(pesan)
    print "from server : ", s.recv(1024)

s.close
```

Simple XMLRPC Server

Internet Protocols and Support

Beberapa contoh objek dasar yang akan digunakan :

- **SimpleXMLRPCServer()**, membuat objek server xmlrpc

Beberapa contoh method dasar yang akan digunakan :

- **register_instance()**, mendaftarkan sebuah objek ke server xmlrpc agar bisa diakses oleh client
- **serve_forever()**, menyalakan server xmlrpc

XML-RPC Lib

Internet Protocols and Support

Beberapa contoh objek dasar yang akan digunakan :

- **ServerProxy()**, konek ke server xmlrpc

Simple XMLRPC Server

Internet Protocols and Support

Catatan : simpan file berikut dengan nama objektersebar-server.py

```
from SimpleXMLRPCServer import SimpleXMLRPCServer
import time

server = SimpleXMLRPCServer(('localhost', 9000), logRequests=True)

class PersegiPanjang:
    def __init__(self):
        self.log_perhitungan = []

    def keliling(self, p, l):
        return 2*p + 2*l

    def luas (self, p, l):
        return p * l

    def baca_log_perhitungan(self):
        return self.log_perhitungan

    def simpan_hasil(self, luas, keliling):
        titi_mangsa = time.ctime()
        self.log_perhitungan.append(list([titi_mangsa, luas, keliling]))
        return str('disimpan pada : %s' % titi_mangsa)
```

Simple XMLRPC Server

Internet Protocols and Support

Catatan : simpan file berikut dengan nama objektersebar-server.py

```
-----  
server.register_instance(PersegiPanjang())  
  
try:  
    print 'Gunakan Control+C untuk berhenti..'  
    server.serve_forever()  
except KeyboardInterrupt:  
    print 'Berhenti..'
```


XML-RPC Lib

Internet Protocols and Support

Catatan : simpan file berikut dengan nama objektersebar-client.py

```
from pprint import pprint
import xmlrpclib
import time

proxy = xmlrpclib.ServerProxy('http://localhost:9000')
keliling = proxy.keliling(10, 10)
luas = proxy.luas(10, 10)
print "Menghitung Persegi Panjang"
print "-----"
print "luas      : ", luas
print "keliling   : ", keliling

print "info : ", proxy.simpan_hasil(luas, keliling)
print
print "hasil yang sudah disimpan : "
pprint(proxy.baca_log_perhitungan())
```

OS

Generic Operating System Services

Beberapa contoh method dasar yang akan digunakan :

- **system()**, memanggil perintah dari sistem operasi tertentu
- **walk()**, menelusuri folder dan file pada sebuah folder secara keseluruhan
- **getcwd()**, melihat direktori yang sedang dikunjungi
- **listdir()**, melihat daftar folder pada sebuah folder
- **makedirs()**, membuat folder baru
- **join()**, menyambungkan folder dan file dengan tanda '/' atau '\' sesuai sistem operasi yang sedang digunakan.

OS

Generic Operating System Services

Catatan : simpan file berikut dengan nama ngaksesos-lihatisi.py

```
import os, pprint

# memanggil perintah pada sistem operasi
os.system('ls -l')
print "\n"

# melihat daftar folder dan file pada suatu direktori
for dir_name, sub_dirs, files in os.walk('/usr/share/example-content'):
    print dir_name
    for dir in sub_dirs:
        print '\t/%s' % dir
    for file in files:
        print "\t", file
print "\n"

# melihat direktori yang sedang dikunjungi
print os.getcwd()
print "\n"

# melihat daftar direktori pada suatu direktori
pprint.pprint(os.listdir('/var/www'))
```

OS

Generic Operating System Services

Catatan : simpan file berikut dengan nama ngaksesos-bikinfolder.py

```
import os, pprint

# membuat direktori baru
os.makedirs('testing')
print "\n"

# membuat file dalam folder 'testing'
nama_file = os.path.join('testing', 'contoh.txt')
f = open(nama_file, 'w')
try:
    f.write('ini di dalam folder testing')
    f.write('nanti akan kita coba hapus')
finally:
    f.close
```

OS

Generic Operating System Services

Catatan : simpan file berikut dengan nama ngaksesos-hapusfolder.py

```
import os, pprint

nama_folder = 'testing'
os.system('rm %s -R' % nama_folder)
```

Begitu banyak module yang ada di
Python STL

Untuk eksplorasi lebih lanjut Anda bisa mengunjungi link berikut ini

Python Modules of The Week

Website : <http://www.pymotw.com>



PyMOTW

- Home
- Blog
- The Book
- About
- Site Index

If you find this information useful, consider picking up a copy of my book, *The Python Standard Library By Example*.

PyMOTW »

[modules](#) | [index](#)

Python Module of the Week

The [Python Module of the Week](#) series, or PyMOTW, is a tour of the [Python](#) standard library through short examples. This is version 1.132, last updated Jun 11, 2013 to cover the [ConfigParser](#) module.

[Complete Table of Contents](#)
lists all articles and subsections

[Global Module Index](#)
quick access to all modules

[About PyMOTW](#)
background on the project

[General Index](#)
all functions, classes, terms

Download

Download [version 1.132](#), including all source files with examples and HTML versions of the documentation.

There is also a PDF of this entire document [available for download](#).

Translations

[Several translations](#) are in process.

Recent Blog Posts

- [PyMOTW Moving](#) 2/19/2013
I am in the process of moving my web site to a new server at DreamHost, and in the process...


Subscribe

 [Subscribe in a reader](#)

 [Subscribe by Email](#)



[The Python Standard Library by Example](#)
Doug Hellmann
Best Price \$31.43
or Buy New \$32.80
[Buy from amazon.com](#)
Privacy Information


\$5/Mo. SSD Virtual Server

Python Documentation

Website : <http://docs.python.org/release/2.7/library/index.html>

The screenshot shows a web browser window displaying the Python v2.7 documentation page for 'The Python Standard Library'. The browser's address bar shows the URL `docs.python.org/release/2.7/library/index.html`. The page has a dark blue header with navigation links: 'previous | next | modules | index'. On the left side, there is a sidebar with 'Previous topic' (9. Full Grammar specification), 'Next topic' (1. Introduction), and 'This Page' (Report a Bug, Show Source). Below the sidebar is a 'Quick search' box with a 'Go' button and a search prompt: 'Enter search terms or a module, class or function name.' The main content area features the title 'The Python Standard Library' and a metadata box with 'Release: 2.7' and 'Date: July 03, 2010'. The text describes the library's scope, its extensive facilities, and its availability in different operating systems. It also mentions the Python Package Index and provides a list of links: 1. Introduction, 2. Built-in Functions, 3. Non-essential Built-in Functions, and 4. Built-in Constants. At the bottom of the page, there is a search bar with the text 'Find: send' and navigation controls: '< Previous', 'Next >', 'Highlight all', and 'Match case'.

Python v2.7 documentation » [previous](#) | [next](#) | [modules](#) | [index](#)

The Python Standard Library

Release: 2.7
Date: July 03, 2010

While *The Python Language Reference* describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

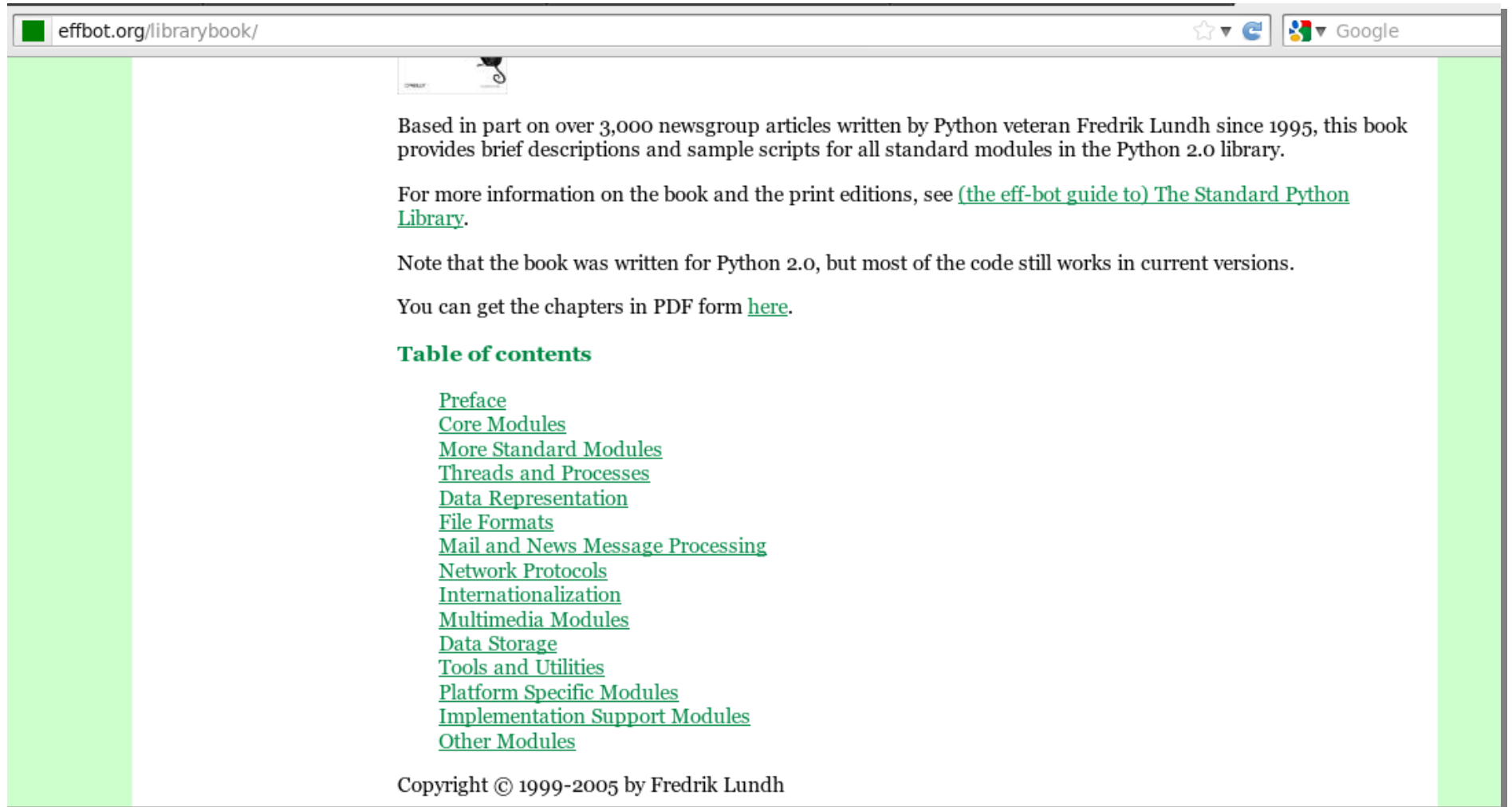
In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the [Python Package Index](#).

- 1. [Introduction](#)
- 2. [Built-in Functions](#)
- 3. [Non-essential Built-in Functions](#)
- 4. [Built-in Constants](#)

Find: < Previous > Next > Highlight all Match case

The Standard Python Library

Website : <http://effbot.org/librarybook/>



Based in part on over 3,000 newsgroup articles written by Python veteran Fredrik Lundh since 1995, this book provides brief descriptions and sample scripts for all standard modules in the Python 2.0 library.

For more information on the book and the print editions, see [\(the eff-bot guide to\) The Standard Python Library](#).

Note that the book was written for Python 2.0, but most of the code still works in current versions.

You can get the chapters in PDF form [here](#).

Table of contents

- [Preface](#)
- [Core Modules](#)
- [More Standard Modules](#)
- [Threads and Processes](#)
- [Data Representation](#)
- [File Formats](#)
- [Mail and News Message Processing](#)
- [Network Protocols](#)
- [Internationalization](#)
- [Multimedia Modules](#)
- [Data Storage](#)
- [Tools and Utilities](#)
- [Platform Specific Modules](#)
- [Implementation Support Modules](#)
- [Other Modules](#)

Copyright © 1999-2005 by Fredrik Lundh