10
Standardmoduler och externa moduler

Översikt

- Standardmoduler
  - Nätverkskommunikation
  - HTML- och XML-parsning
  - Loggning
  - Debug
- Exempel på externa (”tredjeparts”) moduler

Nätverkskommunikation

- *socketserver* - högnivåmodul för att bygga en nätverkstjänst/server
  - Flera olika klasser av servertyper, forkande, trådad, tcp, udp etc
- *http.server*
- *urllib, urllib.parse* - webbklient
- *xmlrpc.client* - klientaccess XML-RPC
- *xmlrpc.server* - XML-RPC-server
- Många andra protokoll; smtp, nntp, pop3, imap4, telnet, ftp, snmp
- Lågnivå socket-kommunikation via modulen *socket*
- *asyncio* och *aiohttp* (den senare finns ej i standardbiblioteket)
- Se även modulerna *asyncore* och *asynchat*
- Se vidare *kapitel 7 - File Handling* och *kapitel 10 - Networking*

Portscanning med asyncore

```
import asyncore, time, Scanner

host = '5.178.76.122'
ports = range(70, 900)
socks = { p: Scanner.Scanner(host, p) for p in ports }
deadline = time.time() + 10.0

while asyncore.socket_map:
    time_left = deadline - time.time()
    if time_left <= 0:
        break
    asyncore.loop(timeout = time_left, count = 1)

for p, s in socks.items():
    if s.status != "connecting":
        print("Port", p, s.status)
```

Hanterare i asyncore

```
class Scanner(asyncore.dispatcher):
    def __init__(self, host, port):
        asyncore.dispatcher.__init__(self)
        self.create_socket()
        self.connect( (host, port) )
        self.status = "connecting"
    def handle_connect(self):
        self.status = "connected"
    def handle_close(self):
        self.close()
    def handle_error(self):
        self.close()
    def writable(self):
        return True
    def handle_write(self):
        self.status = "writing"
        self.close()
```

Hämta filer på webben med aiohttp

```
import asyncio, aiohttp
async def load_url(session, url):
    try:
        async with session.get(url) as resp:
            txt = await resp.text()
            if resp.status == 200:
                return txt
            else:
                return 'error ' + resp.status + \
                    ': ' + txt
    except asyncio.CancelledError:
        return 'cancelled'
    except Exception as err:
        return 'error: ' + str(err)
```

Många parallella hämtningar med aiohttp

```
from random import randint
srv = "http://lab04.bredbandskollen.se/"

async def program():
    async with aiohttp.ClientSession() as session:
        urls = [ srv + "ping/" + str(randint(1,999999)) +
                 ".txt" for _ in range(50) ]
        dl = [ asyncio.ensure_future(load_url(session, url))
               for url in urls ]
        for future in asyncio.as_completed(dl):
            res = await future
            print(res.strip())

asyncio.run(program())
```

Använda proxy med aiohttp

```
async with aiohttp.ClientSession() as session:
    my_auth = aiohttp.BasicAuth('user', 'pass')
    async with session.get("http://python.org",
                           proxy="http://proxy.com",
                           proxy_auth=my_auth) as resp:
        print(resp.status)
```

HTML- och XML-parsning

- *html.parser* - parsa HTML- och XHTML-kod
- XML-parsning (och generering)
  - *xml.etree*
  - *xml.dom/xml.dom.minidom*
  - *xml.sax*
  - xml.parsers.expat
  - Se vidare *kapitel 7 - File Handling*

html.parser: Definiera hanterare

```
import html.parser
class MinParser(html.parser.HTMLParser):
    def handle_starttag(self, tag, attrs):
        if attrs:
            print(' '*self.nivå + "Tag:", tag, "Attr:", attrs)
        else:
            print(' '*self.nivå + "Tag:", tag)
        self.nivå += 1
    def handle_endtag(self, tag):
        self.nivå -= 1
        print(' '*self.nivå + "Slut", tag)
    def handle_data(self, data):
        print(' '*self.nivå + "Innehåll:", data)
    def reset(self):
        self.nivå = 0
        super().reset()
```

html.parser, forts.

```
>>> parser = MinParser()
>>> html = '''<html><body>
... <a href="http://127.0.0.1/">Hem</a>
... </body></html>'''
>>> parser.feed(html)
Tag: html
 Tag: body
  Innehåll:

  Tag: a Attr: [('href', 'http://127.0.0.1/')]
   Innehåll: Hem
  Slut a
  Innehåll:

 Slut body
Slut html
>>> parser.close()
```

XML-parsning

- Exempel med ElementTree:

```
#!/usr/bin/env python3
import xml.etree.ElementTree as ET

report = """<?xml version="1.0"?>
<bbk-resultset type="int">
  <data>212345678</data>
</bbk-resultset>"""

xmlroot = ET.fromstring(report)
tag = xmlroot[0].tag
res_id = xmlroot[0].text
print("Tag:", tag, " Text:", res_id)
```

Modulen logging

- Kraftfull modul - stort antal sätt att logga, t.ex.
  - Till fil (med inbyggd rotation)
  - Via nätverk; tcp/udp, smtp, socket
  - Via syslog
- Efterliknar syslog i sitt koncept:
  - Olika nivåer; debug, info, warning, error, critical
  - Olika avsändare (t.ex. applikationsnamn)
  - Styra olika meddelanden (t.ex. nivåer) till olika destinationer

Exempel: modulen logging

```
import logging

logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(levelname)-8s %(message)s',
                    datefmt='%a, %d %b %Y %H:%M:%S',
                    filename='/tmp/myapp.log',
                    filemode='w')
logging.debug('A debug message')
logging.info('Some information')
logging.warning('A shot across the bows')
```

Genererar följande logg-fil:

```
Fri, 02 Jul 2004 13:06:18 DEBUG    A debug message
Fri, 02 Jul 2004 13:06:18 INFO     Some information
Fri, 02 Jul 2004 13:06:18 WARNING  A shot across the bows
```

Debug mode - pdb

- Ett program kan köras i debug-läge, enklast via följande kommando:

```
$ python3 -m pdb ./myprog.py
```

- Kommandon, exempel:
  - *h/help* – skriv ut hjälp
  - *l/list* – skriv ut var i koden vi befinner oss
  - *n/next* – stega fram (över) en rad i programmet
  - *s/step* – stega fram (in i) en rad i programmet
  - *q/quit*
  - *p/print* – skriv ut variabelinnehåll
- Om ett program kraschar med ett ohanterat undantag går debuggern in i "post-mortem"-mod där möjlighet finns att undersöka variabelvärden, återstarta programmet mm
- Se vidare pythondokumentationen

Andra externa moduler och ramverk

- NumPy
  - Matlab-liknande modul för t.ex. flerdimensionella arrayer
  - Se vidare http://numpy.scipy.org/
- pandas
  - Analys och visualisering av data
- Twisted
  - Händelsestyrd nätverksmotor med stort antal nätverkstjänster/protokoll
  - Se vidare http://twistedmatrix.com
- PyGame
  - Grafikramverk för att skriva spel/multimedia/grafik-applikationer
  - Se vidare http://www.pygame.org

Webb-ramverk

- cherryPy - http://www.cherrypy.org
  - Webbapplikationsserver i lättviktsformat
- Zope - http://www.zope.org
  - Webbapplikationsserver
- TurboGears - http://www.turbogears.org/ och Django - http://www.djangoproject.com/
  - Webbutvecklingsramverk som består av flera andra opensource-komponenter, bl.a. webbapplikationsserver, javascriptbibliotek, templatingbibliotek
- Plone - http://plone.org/
  - Stort och spritt Content Management-system baserat på Zope

Grafiska gränssnitt

- Tkinter
  - Vanligt grafikramverk för Python
  - IDLE är skriven i Tkinter
- PyGTK
  - Python-ramverk ovanpå Gnome GTK+
  - Se vidare http://www.pygtk.org/
- WXPython
  - Python-ramverk ovanpå wxWidgets
  - Se vidare http://www.wxpython.org
- PyQT4, pyqt5
  - Python-ramverk ovanpå Qt

Övning

- **ovn1001/slurp.py** Skriv funktionen `webbsida` som tar en URL som parameter. Funktionen ska hämta URL:en och returnera innehållet som en sträng.
  - Använd t.ex. `urllib.request` eller `aiohttp`
  - Förutsätt att sidan är kodad i utf-8 eller latin-1
  - Returnera `None` ifall det inte gick att hämta sidan
- **ovn1002/leta.py** Skriv funktionen `allalankar` som tar en URL som parameter. Funktionen ska returnera en mängd (`set`) med alla länkar som webbsidan innehåller, dvs. den ska söka efter `<a href="...">`
  - Använd t.ex. `html.parser`

Övning

- **ovn1003/accelerator.py**** Skapa funktionen webbot som tar två strängar som parametrar. Den första strängen ska vara en URL, den andra sökvägen till en katalog.
  - Kontrollera att katalogen finns, annars ska funktionen skapa den.
  - Hämta URL:en och spara innehållet i filen ”main.html” i katalogen.
  - Hitta alla länkar i den hämtade sidan och parallellt hämta innehållet i alla länkar och spara i olika filer i katalogen.
    - Använd t.ex. trådar eller korutiner.
    - Ignorera länkar som inte börjar med `http`.