netmind

WeKnowIT

Roche

Python

TRABAJANDO EN RED

1. Sockets
2. Leyendo una URL
3. Accediendo a un Web Service
4. Mail
5. FTP

1 Sockets

Sockets

- Python proporciona dos niveles de acceso a los servicios de red.
  - En un bajo nivel, puede acceder al soporte de socket básico en el sistema operativo subyacente, lo que le permite implementar clientes y servidores tanto para protocolos orientados a conexión como sin conexión.
  - También tiene librerías que proporcionan un acceso de nivel superior a protocolos de red específicos de nivel de aplicación, como FTP, HTTP, etc.
- Los sockets son los puntos finales de un canal de comunicaciones bidireccional.
- Los sockets pueden comunicarse dentro de un proceso, entre procesos en la misma máquina, o entre procesos en diferentes continentes.
- Pueden implementarse en varios tipos de canales diferentes: sockets de dominio Unix, TCP, UDP, etc.
- La librería socket proporciona clases específicas para el manejo de los transportes comunes, así como una interfaz genérica para el manejo del resto.

El módulo socket

- https://docs.python.org/3/library/socket.html

- Para crear un socket, debe usar la función socket.socket () disponible en el módulo socket, que tiene la sintaxis:

  s = socket.socket (socket_family, socket_type, protocol = 0)

  - **socket_family**: Esto es AF_UNIX o AF_INET, como se explicó anteriormente.
  - **socket_type**: Esto es SOCK_STREAM o SOCK_DGRAM.
  - **protocolo**: Normalmente se omite, siendo por defecto 0.

Métodos socket

Método	Descripción
s.bind()	Este método enlaza la dirección (nombre de host, par de números de puerto) al socket.
s.listen()	Este método establece e inicia el oyente TCP.
s.accept()	Esto acepta pasivamente la conexión del cliente TCP, esperando hasta que llegue la conexión (bloqueo).
s.connect()	Inicia la conexión en la parte del cliente
s.recv()	Recibe el mensaje TCP
s.send()	Transmite el mensaje TCP
s.recvfrom()	Recibe el mensaje UDP
s.sendto()	Transmite el mensaje UDP

Sockets - Server

```
import socket

# create a socket object
serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# get local machine name
host = socket.gethostname()

port = 9999

# bind to the port
serversocket.bind(host, port)

# queue up to 5 requests
serversocket.listen(5)

while True:
    # establish a connection
    clientsocket, addr = serversocket.accept()

    print("Got a connection from %s" % str(addr))

    msg = 'Thank you for connecting' + "\r\n"
    clientsocket.send(msg.encode('ascii'))
    clientsocket.close()
```

Sockets - cliente

```
import socket

# create a socket object
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# get local machine name
host = socket.gethostname()

port = 9999

# connection to hostname on the port.
s.connect((host, port))

# Receive no more than 1024 bytes
msg = s.recv(1024)
s.close()
print (msg.decode('ascii'))
```

Jugando con sockets

- Crea una lista o una tupla numérica y envía todos los elementos de esta que sean múltiplos de dos desde el cliente al servidor, y que este los almacene en una lista.

- Cifra la comunicación parte a parte mediante una función de encriptado/desencriptado simple

2 Leyendo una URL

Módulo http

- http es un paquete que recopila varios módulos para trabajar con HyperText Transfer Protocol:
  - **http.client** es un cliente de protocolo HTTP de bajo nivel; para la apertura de URL de alto nivel urllib.request
  - **http.server** contiene clases básicas de servidores HTTP basadas en socketserver
  - **http.cookies** tiene utilidades para implementar la administración estatal con cookies
  - **http.cookiejar** proporciona persistencia de cookies

Implementando un servidor

```
from http.server import BaseHTTPRequestHandler,
HTTPServer


# HTTPRequestHandler class
class
testHTTPServer_RequestHandler(BaseHTTPRequestHand
ler):
    # GET
    def do_GET(self):
        # Send response status code
        self.send_response(200)

        # Send headers
        self.send_header('Content-type', 'text/html')
        self.end_headers()

        # Send message back to client
        message = "Hello world!"
        # Write content as utf-8 data
        self.wfile.write(bytes(message, "utf8"))
        return
```

```
def run():
    print('starting server...')

    # Server settings
    # Choose port 8080, for port 80, which is normally
used for a http server, you need root access
    server_address = ('127.0.0.1', 8081)
    httpd = HTTPServer(server_address,
testHTTPServer_RequestHandler)
    print('running server...')
    httpd.serve_forever()


run()
```

Implementando un servidor

```
# HTTPRequestHandler class
class testHTTPServer_RequestHandler(BaseHTTPRequestHandler):
    # GET
    def do_GET(self):
        # Send response status code
        self.send_response(200)

        # Send headers
        self.send_header('Content-type', 'text/html')
        self.end_headers()

        # Send message back to client
        message = "Hello world!"
        # Write content as utf-8 data
        self.wfile.write(bytes(message, "utf8"))
        return
```

Leyendo una URL

- Usando http.client

```
import http.client

conn = http.client.HTTPConnection("localhost", 8081)
conn.request("GET", "/")

r1 = conn.getresponse()
print(r1.status, r1.reason)
data1 = r1.read()
print(data1)
```

Usando urllib

- urllib.request es un módulo de Python para acceder a URLs (Uniform Resource Locators).
- Ofrece una interfaz muy simple, en la función **urlopen**, capaz de obtener URL usando una variedad de protocolos diferentes.
- También ofrece una interfaz ligeramente más compleja para manejar situaciones comunes - como autenticación básica, cookies, proxies y así sucesivamente. Estos son proporcionados por objetos handlers y openers.
- https://docs.python.org/3/howto/urllib2.html

Usando urllib

- Lectura básica

```
import urllib.request
with urllib.request.urlopen('http://python.org/') as response:
   html = response.read()
```

- Almacenaje en un fichero

```
import urllib.request
local_filename, headers = urllib.request.urlretrieve('http://python.org/')
html = open(local_filename)
```

Usando urllib

- Usando cabeceras y datos (POST)

```
import urllib.parse
import urllib.request

url = 'http://www.someserver.com/cgi-bin/register.cgi'
user_agent = 'Mozilla/5.0 (Windows NT 6.1; Win64; x64)'
values = {'name': 'Michael Foord',
          'location': 'Northampton',
          'language': 'Python' }
headers = {'User-Agent': user_agent}

data = urllib.parse.urlencode(values)
data = data.encode('ascii')
req = urllib.request.Request(url, data, headers)
with urllib.request.urlopen(req) as response:
   the_page = response.read()
```

Lee los titulares

- Lee los titulares del New York Times

3 Accediendo a un Web Service

Consultando un WebService

- Para ello podemos usar la librería **requests**
  - http://docs.python-requests.org/en/master/
- requests facilita el acceso a web services tipo REST, incluyendo autenticación

- Si el módulo no existe, se puede instalar usando pip

  pip install requests

Consultando un WebService

- import requests

- r = requests.get('https://api.github.com/user', auth=(username', password'))
- r.status_code
- r.headers['content-type']
- r.encoding
- r.text
- print(r.json())

Consultando un WebService

- Para autenticación kerberos existe la librería requests-kerberos
  - https://github.com/requests/requests-kerberos

```
import requests
from requests_kerberos import HTTPKerberosAuth

response = requests.get('http://thedataishere.com',
                        auth=HTTPKerberosAuth())
data = response.json()  # data es un objeto genérico
```

JSON en python

- La librería json de python permite condificar y decodificar json
  - https://docs.python.org/3/library/json.html
  - **json.dump**(obj, fp, *, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, cls=None, indent=None, separators=None, default=None, sort_keys=False, **kw)
    - Serializa un objeto como JSON
  - **json.dumps** – serializar un objeto a un string JSON formateado
  - **json.load**(fp, *, cls=None, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, object_pairs_hook=None, **kw)
    - Deserializa un fp a un objeto Python
- Todos estos métodos usan la tabla de conversión definida en:
  - https://docs.python.org/3/library/json.html#json-to-py-table

Consumiendo un WS REST

- Accede a la API de http://api.open-notify.org/iss-pass.json

4 Mail

Mail

- Python proporciona el módulo **smtplib**, que define un objeto de sesión de cliente SMTP que se puede utilizar para enviar correo a cualquier máquina de Internet con un daemon de escucha de SMTP o ESMTP.
  - https://docs.python.org/3.4/library/smtplib.html

- La sintaxis a usar es la siguiente

```
import smtplib
smtpObj = smtplib.SMTP( [host [, port [, local_hostname]]] )
```

  - **host**: el host SMTP (IP o dominio). Es opcional.
  - **puerto**: si se proporciona el host, se necesita especificar el puerto donde el servidor SMTP está escuchando. Normalmente el puerto 25.
  - **local_hostname**: Si el servidor SMTP se ejecuta en la máquina local, especificar localhost.

Mail

- Un objeto SMTP tiene un método de llamado sendmail, que normalmente se utiliza para realizar el envío de un mensaje.
- Se necesitan tres parámetros:
  - El remitente - Una cadena con la dirección del remitente.
  - Los receptores - Una lista de cadenas, una para cada destinatario.
  - El mensaje: un mensaje como una cadena formateada como se especifica en los distintos RFC.
- Por defecto el mensaje se enviará como texto, pero si encuentran tags html, se enviará como html

Mail

```
import smtplib

sender = 'from@fromdomain.com'
receivers = ['to@todomain.com']
message = """From: From Person <from@fromdomain.com>
To: To Person <to@todomain.com>
Subject: SMTP e-mail test
This is a test e-mail message.
"""
try:
   smtpObj = smtplib.SMTP('localhost')
   smtpObj.sendmail(sender, receivers, message)
   print "Successfully sent email"
except SMTPException:
   print "Error: unable to send email"
```

Mail con archivos adjuntos

```
#!/usr/bin/python

import smtplib
import base64

filename = "/tmp/test.txt"

# Read a file and encode it into base64 format
fo = open(filename, "rb")
filecontent = fo.read()
encodedcontent = base64.b64encode(filecontent)  # base64

sender = 'webmaster@tutorialpoint.com'
reciever = 'amrood.admin@gmail.com'

marker = "AUNIQUEMARKER"

body ="""
This is a test email to send an attachement.
"""
# Define the main headers.
part1 = """From: From Person <me@fromdomain.net>
To: To Person <amrood.admin@gmail.com>
Subject: Sending Attachement
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=%s
--%s
""" % (marker, marker)

# Define the message action
part2 = """Content-Type: text/plain
Content-Transfer-Encoding:8bit

%s
--%s
""" % (body,marker)

# Define the attachment section
part3 = """Content-Type: multipart/mixed; name=\"%s\"
Content-Transfer-Encoding:base64
Content-Disposition: attachment; filename=%s

%s
--%s--
""" %(filename, filename, encodedcontent, marker)
message = part1 + part2 + part3

try:
   smtpObj = smtplib.SMTP('localhost')
   smtpObj.sendmail(sender, reciever, message)
   print "Successfully sent email"
except Exception:
   print "Error: unable to send email"
```

Usando gmail

- Intenta enviar un mail usando tu cuenta de gmail
  - Deberás activar la opción menos segura de tu cuenta
    - https://www.google.com/settings/security/lesssecureapps
  - Usar el puerto 587
  - Activar la opción ehlo y starttls
  - usar el método login para indicar tu usuario/contraseña

Usando mailgun

- Crea una cuenta en mailgun (https://www.mailgun.com/) y úsala para enviar un mail
- Usa la guía: http://mg-documentation.readthedocs.io/en/latest/quickstart-sending.html#send-via-smtp

5 FTP

FTP

- La librería **ftplib** implementa el protocol FTP.
  - https://docs.python.org/3/library/ftplib.html
- Usando FTP podemos crear y acceder a archivos remotos a través de llamadas de función usando la sintaxis:

```
from ftplib import FTP
ftplib.FTP(host='', user='', passwd='', acct='', timeout=None,
source_address=None)
```

- También existe la posibilidad de usar TLS con la subclase FTP_TLS:

```
ftplib.FTP_TLS(host='', user='', passwd='', acct='', keyfile=None, certfile=None,
context=None, timeout=None, source_address=None)
```

FTP

- Listar un directorio

```
import ftplib

ftp = ftplib.FTP("ftp.nluug.nl")
ftp.login("anonymous", "ftplib-example-1")

data = []

ftp.dir(data.append)

ftp.quit()

for line in data:
    print("-", line)
```

FTP

- Cambiar y listar un directorio: ftp.cwd('/')

```
import ftplib

ftp = ftplib.FTP("ftp.nluug.nl")
ftp.login("anonymous", "ftplib-example-1")

data = []

ftp.cwd('/pub/')         # change directory to /pub/
ftp.dir(data.append)

ftp.quit()

for line in data:
    print("-", line)
```

FTP

- Descargar un archivo

```
import ftplib


def getFile(ftp, filename):
    try:
        ftp.retrbinary("RETR " + filename, open(filename, 'wb').write)
    except:
        print("Error")


ftp = ftplib.FTP("ftp.nluug.nl")
ftp.login("anonymous", "ftplib-example-1")

ftp.cwd('/pub/')         # change directory to /pub/
getFile(ftp, 'README.nluug')

ftp.quit()
```

FTP

- Subir un archivo

```
import ftplib
import os


def upload(ftp, file):
    ext = os.path.splitext(file)[1]
    if ext in (".txt", ".htm", ".html"):
        ftp.storlines("STOR " + file, open(file))
    else:
        ftp.storbinary("STOR " + file, open(file, "rb"), 1024)


ftp = ftplib.FTP("127.0.0.1")
ftp.login("username", "password")

upload(ftp, "README.nluug")
```

SFTP

- Para tratar con sftp existe un módulo que resuelve la mayor parte de las casuísticas: **pysftp**
  - https://pypi.python.org/pypi/pysftp
- Para usarlo es necesario instalarlo usando pip

  pip install pysftp

- Ejemplo

```
import pysftp

with pysftp.Connection('hostname', username='me', password='secret') as
sftp:
    with sftp.cd('public'):           # temporarily chdir to public
        sftp.put('/my/local/filename')  # upload file to public/ on remote
        sftp.get('remote_file')         # get a remote file
```

Jugando con ftp

- Genera un script que se conecte a un servidor ftp, muestre el listado raíz y a partir de allí te permita cambiar de directorio, descargar un archivo (o varios) o subir uno (o varios)
- Los archivos pueden ser texto o binarios

netmind

WeKnowIT

Barcelona

C. Almogàvers, 123
08018 Barcelona
Tel. 93 304.17.20
Fax. 93 304.17.22

Madrid

Plaza Carlos Trías Bertrán, 7
28020 Madrid
Tel. 91 442.77.03
Fax. 91 442.77.07

www.netmind.es