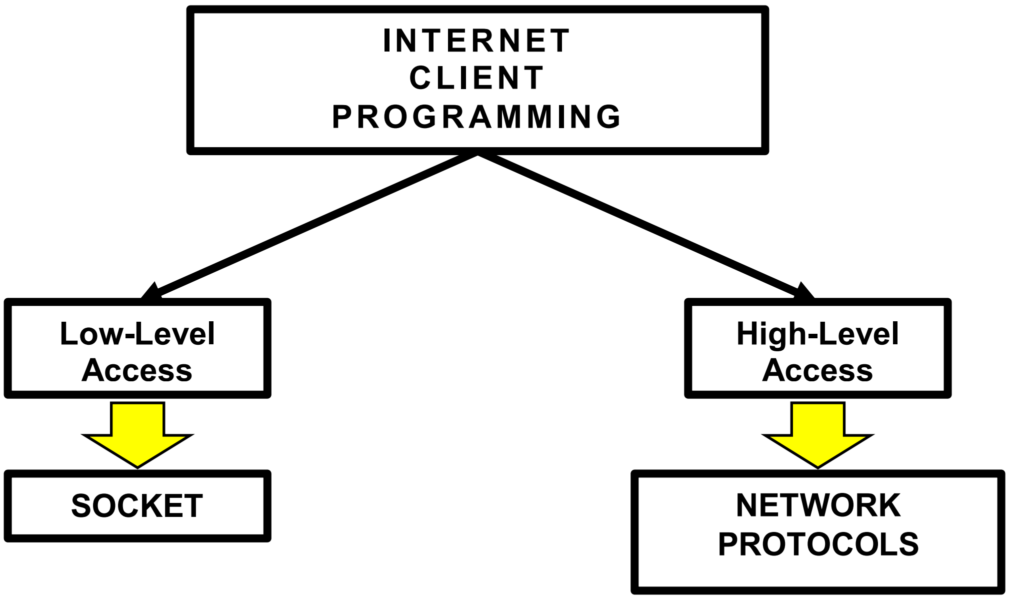
Co-funded by the Erasmus+ Programme of the European Union

INTERNET CLIENT PROGRAMMING

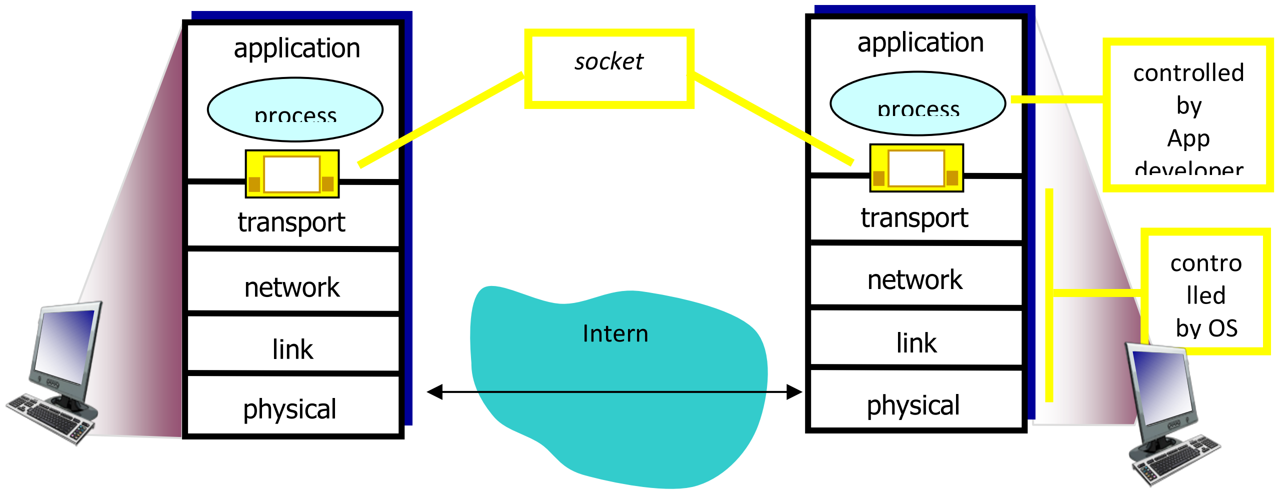


Python is an interpreted, cross-platform, object-oriented programming language that is popular for a wide range of applications, one of which is Internet programming.

Python provides two levels of access to network services. At a low level, you can access the basic **socket** support in the underlying operating system, which allows you to implement clients and servers for both connection-oriented and connectionless protocols.

Python also has **libraries** that provide higher-level access to specific application-level network protocols, such as FTP, HTTP, and so on.

SOCKET



The socket Module

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

Sockets are the endpoints of a bidirectional communications channel.

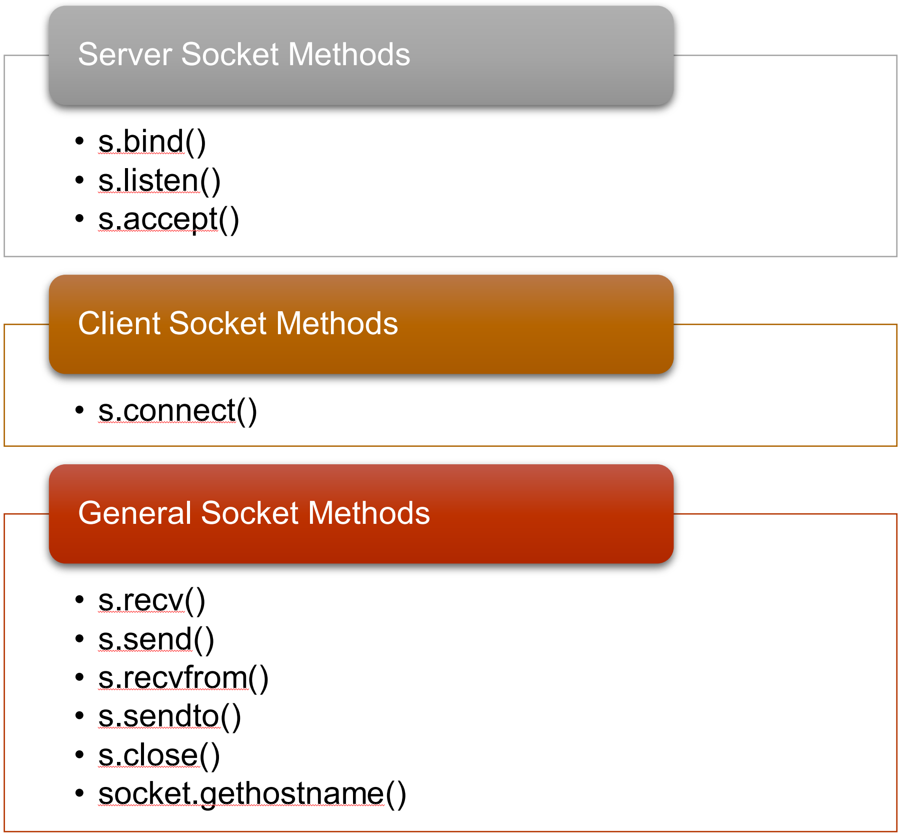
Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on.

This document has been produced with the support of the EUROPEAN COMMISSION under the ERASMUS+ Programme, KA2 – Capacity Building in the Field of Higher Education: 598092-EPP-1-2018-1-BG-EPPKA2-CBHE-SP.
It reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

The **socket** library provides specific classes for handling the common transports as well as a generic interface for handling the rest.

To create a socket, you must use the ***socket.socket()*** function available in ***socket*** module, which has the general syntax.



Server Socket Methods

- s.bind()

This method binds address such as hostname or port number pair to socket.

- s.listen()

This method sets up and start TCP listener.

- s.accept()

This passively accept TCP client connection, waiting until connection arrives or blocking.

Client Socket Methods

- s.connect()

This method actively initiates TCP server connection.

General Socket Methods

- s.recv()

This method receives TCP message.

- s.send()

This method transmits TCP message.

- s.recvfrom()

This method receives UDP message.

- s.sendto()

This method transmits UDP message.

- s.close()

This method closes socket.

- socket.gethostname()

This document has been produced with the support of the EUROPEAN COMMISSION under the ERASMUS+ Programme, KA2 – Capacity Building in the Field of Higher Education: 598092-EPP-1-2018-1-BG-EPPKA2-CBHE-SP.
It reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Returns the hostname.

A Simple Server

import socket	# Import socket module
s = socket.socket()	# Create a socket object
host = socket.gethostname()	# Get local machine name
port = 12345	# Reserve a port for your service.
s.bind((host, port))	# Bind to the port
s.listen(5)	# Now wait for client connection.
while True:	
c, addr = s.accept()	# Establish connection with client.
print 'Got connection from', addr	
c.send('Thank you for connecting')	# Close the connection
c.close()	

To write Internet servers, we use the **socket** function available in socket module to create a socket object. A socket object is then used to call other functions to setup a socket server.

Now call **bind(hostname, port)** function to specify a port for your service on the given host.

Next, call the ***accept*** method of the returned object. This method waits until a client connects to the port you specified, and then returns a ***connection*** object that represents the connection to that client.

A Simple Client

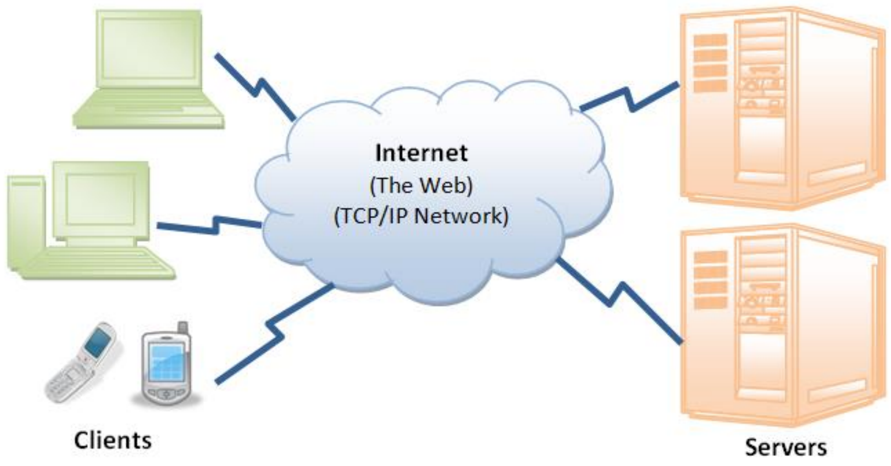
import socket	# Import socket module
s = socket.socket()	# Create a socket object
host = socket.gethostname()	# Get local machine name
port = 12345	# Reserve a port for your service.
s.connect((host, port))	
print s.recv(1024)	
s.close()	# Close the socket when done

Let us write a very simple client program which opens a connection to a given port 12345 and given host. This is very simple to create a socket client using Python's ***socket*** module function.

The ***socket.connect(hosname, port)*** opens a TCP connection to ***hostname*** on the port. Once you have a socket open, you can read from it like any IO object. When done, remember to close it, as you would close a file.

NETWORK PROTOCOLS

This document has been produced with the support of the EUROPEAN COMMISSION under the ERASMUS+ Programme, KA2 – Capacity Building in the Field of Higher Education: 598092-EPP-1-2018-1-BG-EPPKA2-CBHE-SP.
It reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



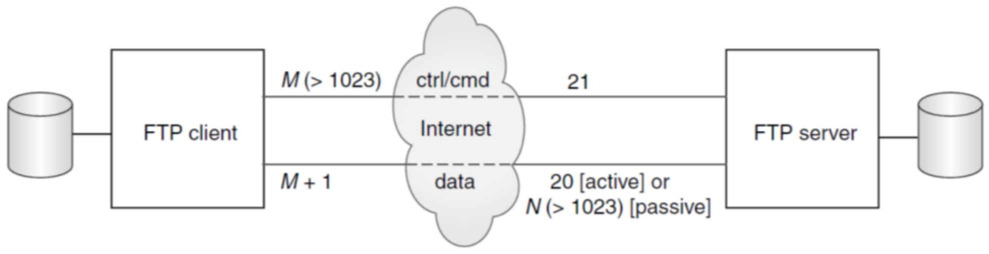
The Internet Protocol is a scheme for imposing a uniform system of addresses on all of the Internet-connected computers in the entire world, and to make it possible for packets to travel from one end of the Internet to the other.

PYTHON INTERNET MODULES

Protocol	Common function	Port No	Python module
HTTP	Web pages	80	httplib, urllib, xmlrpclib
NNTP	Usenet news	119	nntplib
FTP	File transfers	20	ftplib, urllib
SMTP	Sending email	25	smtplib
POP3	Fetching email	110	poplib
IMAP4	Fetching email	143	imaplib
Telnet	Command lines	23	telnetlib
Gopher	Document transfers	70	gopherlib, urllib

A list of some important modules in Python Network Internet programming.

FILE TRANSFERRING PROTOCOLS



FTP Clients and Servers on the Internet. The client and server communicate using the FTP protocol on the command or control port data; is transferred using the data port.

The File Transfer Protocol (FTP) was once among the most widely used protocols on the Internet, invoked whenever a user wanted to transfer files between Internet-connected computers.

The File Transfer Protocol (FTP) was developed by the late Jon Postel and Joyce Reynolds in the Internet Request for Comment (RFC) 959 document and published in October 1985. It is primarily used

This document has been produced with the support of the EUROPEAN COMMISSION under the ERASMUS+ Programme, KA2 – Capacity Building in the Field of Higher Education: 598092-EPP-1-2018-1-BG-EPPKA2-CBHE-SP.
It reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

to download publicly accessible files in an anonymous fashion. It can also be used to transfer files between two computers, especially when you're using a Unix-based system for file storage or archiving and a desktop or laptop PC for work. Before the Web became popular, FTP was one of the primary methods of transferring files on the Internet, and one of the only ways to download software and/or source code.

The protocol is diagrammed in the picture and works as follows:

1. Client contacts the FTP server on the remote host;
2. Client logs in with username and password;
3. Client performs various file transfers or information requests;
4. Client completes the transaction by logging out of the remote host and FTP server.

Python FTP Interface: *ftplib*

Name	Description
login()	FTP login
quit()	Close connection and quit
retrlines/binary()	Get text or binary file
storlines/binary()	Put text or binary file
dir()	Request directory listing
cwd()	Change working directory
delete()	Delete remote file

Python **ftplib** is a module that implements the client side of the FTP protocol. It contains an FTP client class and some helper functions.

This is a list of the most popular methods of **ftplib.FTP class.**

The methods you will most likely use in a normal FTP transaction include login(), cwd(), dir(), storlines(), retrlines() and quit().

Creating FTP Clients

- **Connect to server**
- **Login**
- **Make service request (and hopefully get reply)**
- **Quit**

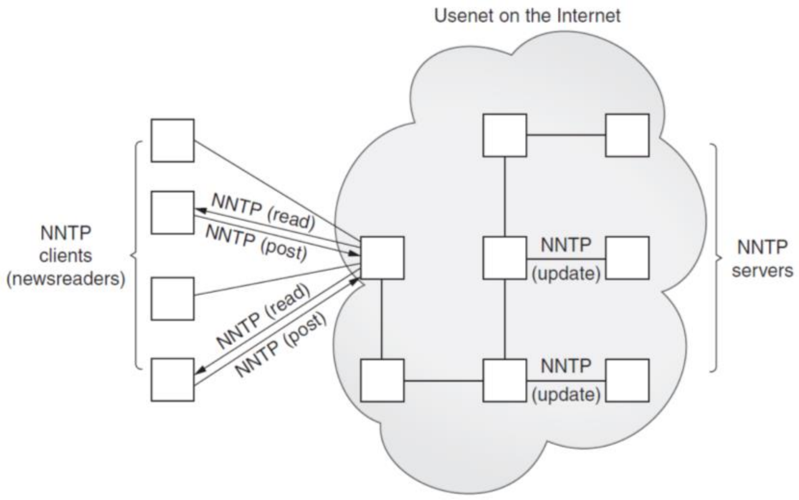
```
from ftplib import FTP

f = FTP(your_FTP_server)

f.login('anonymous', 'guess@who.org')

...

f.quit()
```

When using Python's FTP support, all you do is import the **ftplib** module and instantiate the **ftplib.FTP class.**

All FTP activity—logging in, transferring files, and logging out—will be accomplished using your object.

NETWORK NEWS TRANSFER PROTOCOL (NNTP)

This document has been produced with the support of the EUROPEAN COMMISSION under the ERASMUS+ Programme, KA2 – Capacity Building in the Field of Higher Education: 598092-EPP-1-2018-1-BG-EPPKA2-CBHE-SP.
It reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



NNTP Clients and Servers on the Internet. Clients mostly read news but can also post. Articles are then distributed as servers update each other.

The method by which users can download newsgroup postings or articles or perhaps post new articles, is called the Network News Transfer Protocol (NNTP). It was authored by Brian Kantor (University of California, San Diego) and Phil Lapsley (University of California, Berkeley) in RFC 977, published in February 1986. The protocol has since been updated in RFC 2980, published in October 2000.

As another example of client/server architecture, NNTP operates in a fashion similar to FTP; however, it is much simpler. Rather than having a whole set of different port numbers for logging in, data, and control, NNTP uses only one standard port for communication, 119. You give the server a request, and it responds appropriately, as shown in the picture.

Let's review the protocol briefly:

1. Connect to server;
2. Log in (if applicable);
3. Make service request(s);
4. Quit.

Python NNTP Interface: *nntplib*

Name	Description
group()	Choose newsgroup
quit()	Close connection and quit
article/head/body()	Get entire article or just head or body
stat/next/last()	Set article "pointer," move to next/last
post()	Post article
list()	Request list of valid newsgroups
xhdr()	Retrieve specific headers from articles

This module provides a *Network News Transfer Protocol* (NNTP) client implementation.

Network News, also known as Usenet News, is mostly transmitted with the Network News Transport Protocol (NNTP). The Python standard library supports this protocol in its module nntplib. The nntplib module supplies a class NNTP to connect to an NNTP server.

This is a list of the most popular methods of **nntplib.NNTP class**.

Creating NNTP Clients

- Connect to server
- Choose newsgroup

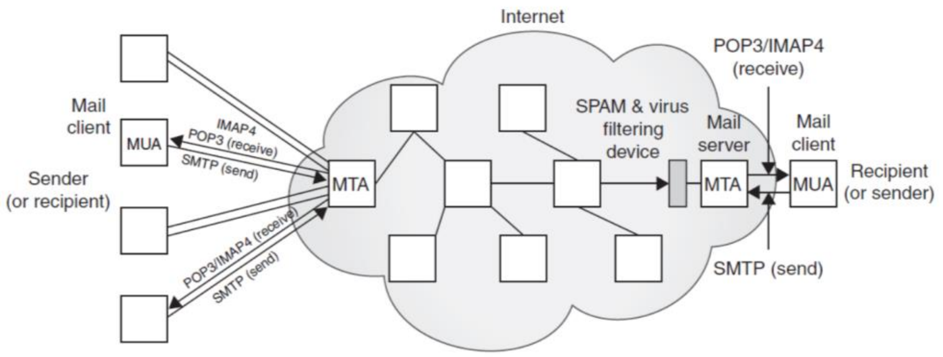
This document has been produced with the support of the EUROPEAN COMMISSION under the ERASMUS+ Programme, KA2 – Capacity Building in the Field of Higher Education: 598092-EPP-1-2018-1-BG-EPPKA2-CBHE-SP.
It reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- group() returns reply, count, first, last, group #
- Perform action:
  - Scroll through (and read) articles
  - article() returns reply, article #, entire message
  - Get or post article
- Quit

```
from nntplib import NNTP
n = NNTP(your_NNTP_server)
r,c,f,l,g = n.group('comp.lang.python')
…
n.quit()
```

Typically, once you log in, you will choose a newsgroup of interest and call the group() method. It returns the server reply, a count of the number of articles, the ID of the first and last articles, and superfluously, the group name again. Once you have this information, you will then perform some sort of action, such as scroll through and browse articles, download entire postings (headers and body of article), or perhaps post an article.

POST OFFICE PROTOCOL VERSION 3 (POP3)



E-Mail Senders and Recipients on the Internet. Clients download and send mail via their MUAs, which talk to their corresponding MTAs. E-mail "hops" from MTA to MTA until it reaches the correct destination.

The first protocol developed for downloading was the Post Office Protocol. As stated in the original RFC document, RFC 918 published in October 1984, "The intent of the Post Office Protocol (POP) is to allow a user's workstation to access mail from a mailbox server. It is expected that mail will be posted from the workstation to the mailbox server via the Simple Mail Transfer Protocol (SMTP)." The most recent version of POP is version 3, otherwise known as POP3. POP3, defined in RFC 1939, is still widely used today.

Protocol consists:

import **poplib** and instantiate the **poplib.POP3 class;**

the standard conversation is as expected:

1. Connect to server;
2. Log in;
3. Make service request(s);
4. Quit.

Python POP3 Interface: poplib

Name	Description
user()	Login to mail server with user name
pass_()	Send user password to server

This document has been produced with the support of the EUROPEAN COMMISSION under the ERASMUS+ Programme, KA2 – Capacity Building in the Field of Higher Education: 598092-EPP-1-2018-1-BG-EPPKA2-CBHE-SP.
It reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

list()	List messages and message sizes
retr()	Retrieve an e-mail message
dele()	Delete an e-mail message
quit()	Close connection and quit
stat()	Get number of messages & mbox size

This is a list of the most popular methods of **poplib.POP3{,SSL} classes**.

Creating POP3 Clients

- **Connect to server**
- **Login**
- **Make service requests**
- **Quit**

```
from poplib import POP3
p = POP3(your_POP_server)
p.user('wesley')
…
p.pass_('secret')
…
p.quit()
```

The protocol of creating POP3 Clients consists:

- **Connect to server**
- **Login**
- **Make service requests**
- **Quit**

Python pseudocode shows creating POP3 Clients.

REFERENCES

1. Internet Programming т with Python by Wesley J. Chun. 1998-2009 CyberWeb Consulting. All rights reserved.
2. Link: [https://www.tutorialspoint.com/python/python_networking.htm]
3. Application Layer and Socket Programming, Hakim Weatherspoon, September 3, 2014
4. Core Python Applications programming (Third edition) by Wesley J.Chun

This document has been produced with the support of the EUROPEAN COMMISSION under the ERASMUS+ Programme, KA2 – Capacity Building in the Field of Higher Education: 598092-EPP-1-2018-1-BG-EPPKA2-CBHE-SP.
It reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.