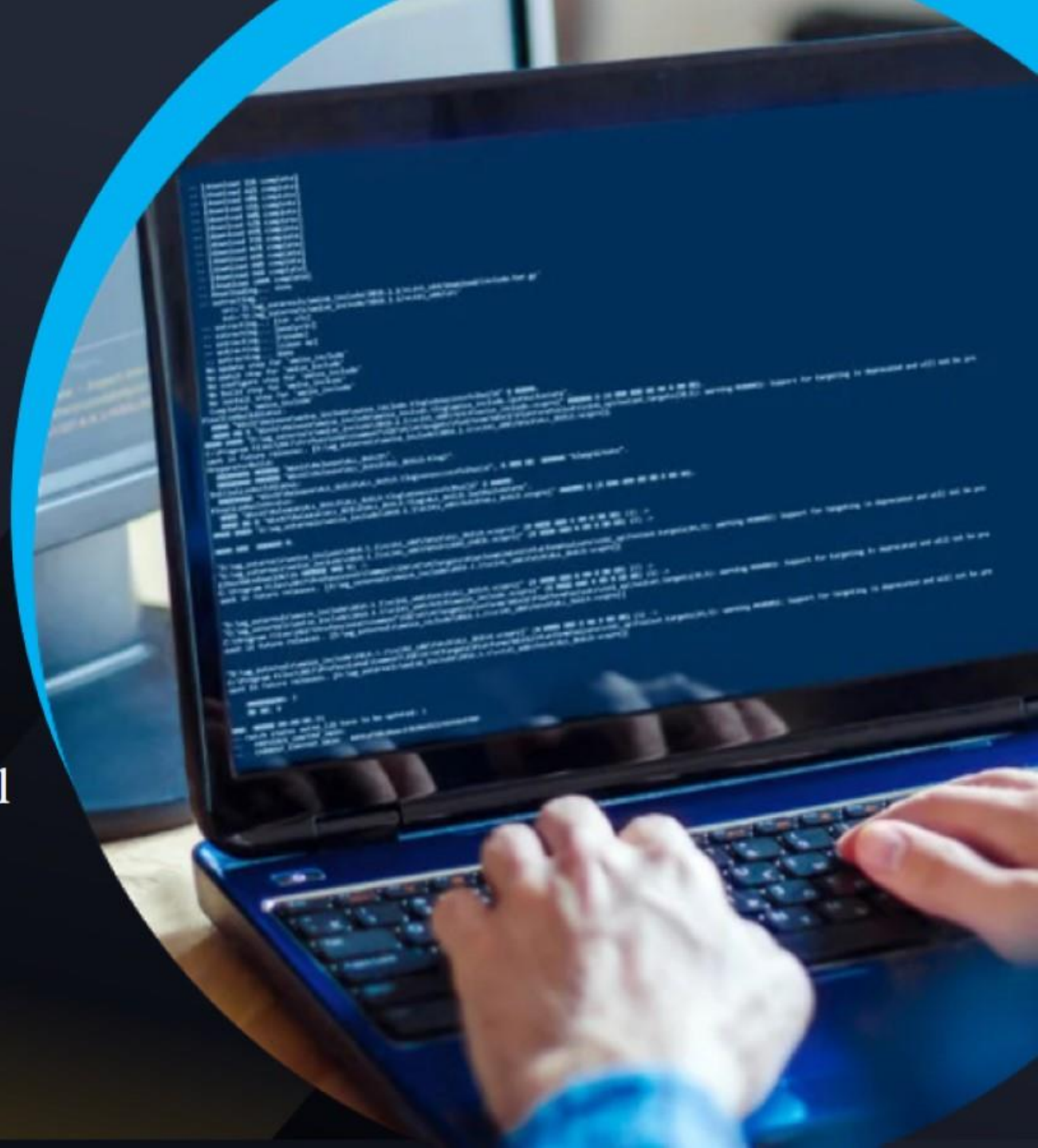


Network Programming in Python

Prepared by Dr. Hamada I. AbdulWakel

Lecture 7



The background features a series of vertical panels with rounded ends. The leftmost panel shows a person's hands typing on a laptop. The subsequent panels contain various white line-art icons on a semi-transparent background: a graduation cap, a play button, a target with an arrow, an open book, a magnifying glass, and a group of people. The title 'Domain Name System' is written in a large, blue, serif font across the middle of these panels.

Domain Name System

Lecture 7

Prepared by
Dr. Hamada I. AbdulWakel

www.midocean.edu.km

Lecture

07

RECAP: LECTURE 6

- Broadcasting client/server model.
- Securing Sockets and SSL module.

Lecture

07

LECTURE MAIN POINTS

- Domain Name System (DNS).
- DNS module in Python.
- The `getaddrinfo()` method.

Lecture

07

NAME RESOLUTION

- Although TCP/IP uses IP addresses to locate and connect to hosts, users typically prefer to use friendly names.



Lecture

07

NAME RESOLUTION

Cont'd



- **Name resolution systems** provide the translation between alphanumeric names and numerical addresses.

NAME RESOLUTION

Cont'd

- There are two common methods for implementing name resolution:
 - A **static file** on each host on the network, containing all the name-to-address translations.
 - Like HOSTS file in Windows/Linux.
 - If a change occurs, every device's HOSTS file must be updated.
 - A **centralized server** that all hosts on the network connect to for name resolution.
 - All devices point to this centralized DNS server for name resolution, ensuring that changes only need to occur in one place.

Lecture

07

DOMAIN NAME SYSTEM

- **DNS** is used on the Internet to provide a standard naming convention for locating IP-based computers.
- **DNS servers:** connected servers have powerful computing resources, storing data, and providing IP addresses for requested domains.
- On Windows/Linux
 - **NOTE:** IP may vary
`nslookup google.com`

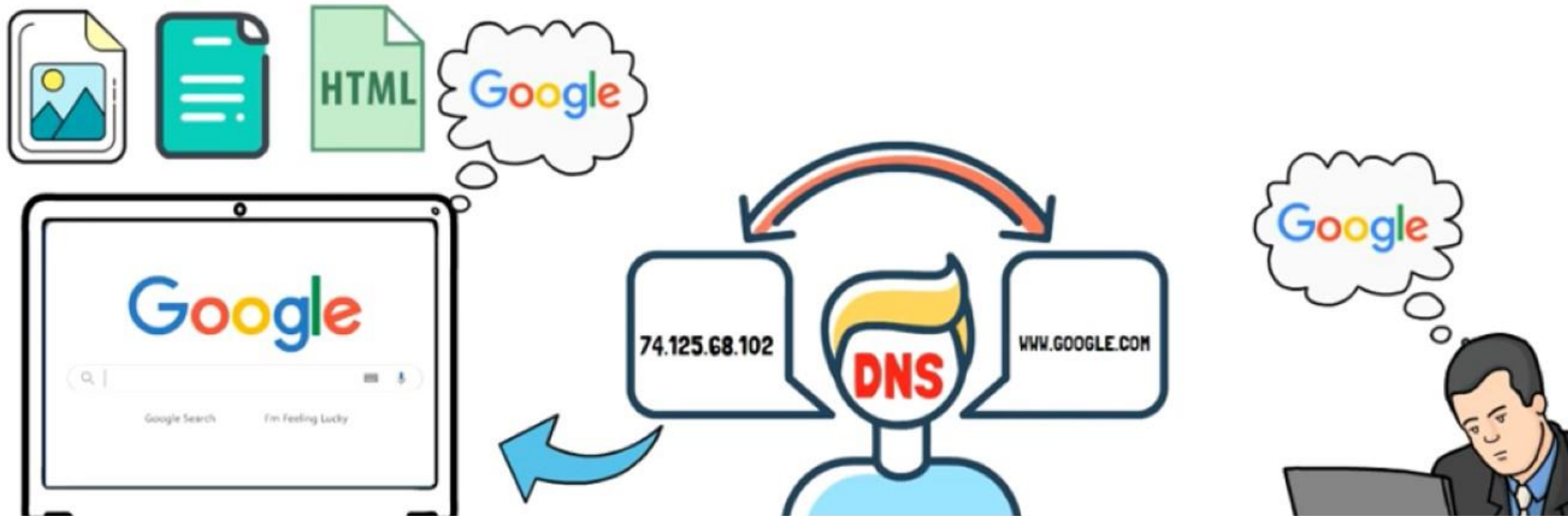
```
drhamada@drhamada-Latitude-E6520:~$ nslookup google.com
Server:          127.0.0.53
Address:         127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.200.206
Name:   google.com
Address: 2a00:1450:4006:80c::200e
```

Lecture

07

DOMAIN NAME SYSTEM

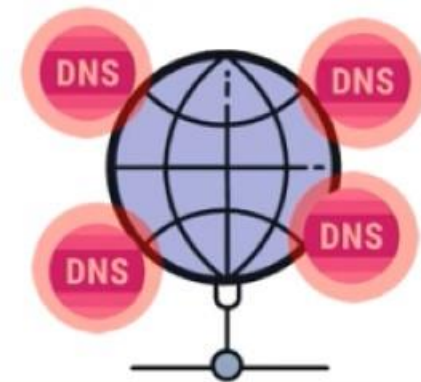
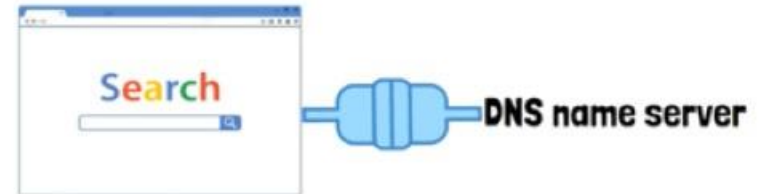


Lecture

07

DNS SERVERS

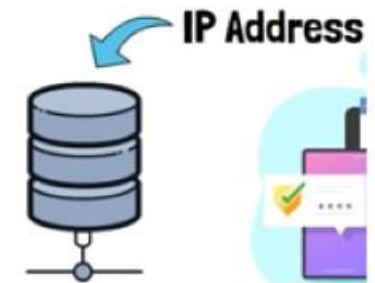
- There are four types of DNS servers:
 - **DNS Resolver:**
 - Provided by Internet Service Provider (ISP).
 - Connects machines with DNS name server.
 - **Root Name Server (RNS):**
 - Identical 13 DNS servers around the world.
 - Keep tracking of new added domains names.



DNS SERVERS

Cont'd

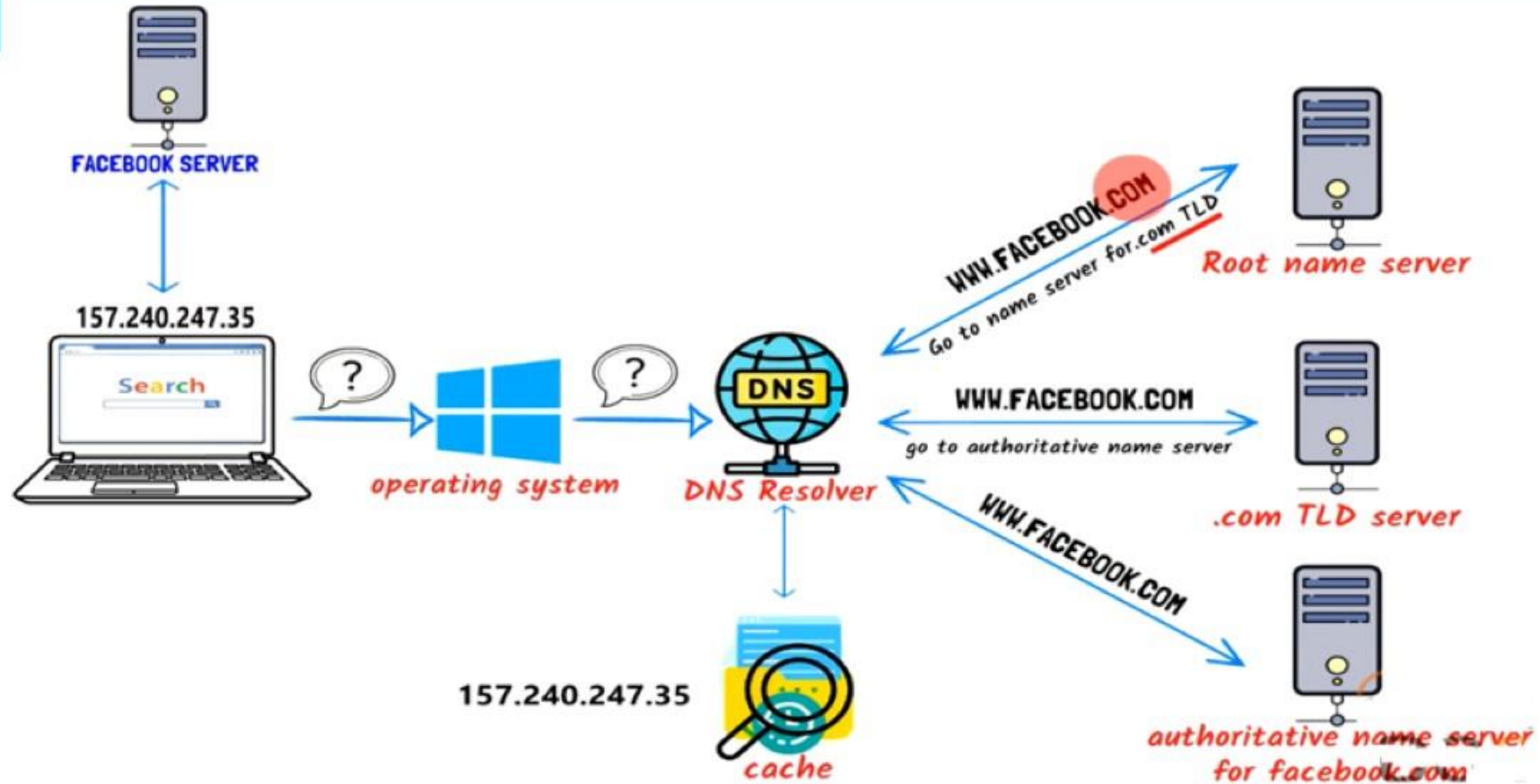
- There are four types of DNS servers:
 - **Top-level Domain Server (TLD):**
 - Stores all domain names with the same extension.
 - Domain extensions like .com, .net, .info, .org, .edu,
 - Extension together with country code like ae form the set of possible suffixes for valid domain names.
 - **Authoritative Name Server (ANS):**
 - Stores the IP address of the requested website.
 - Ensures if the user has the permissions to access IP or not.



Lecture

07

INTERNAL WORK OF DNS



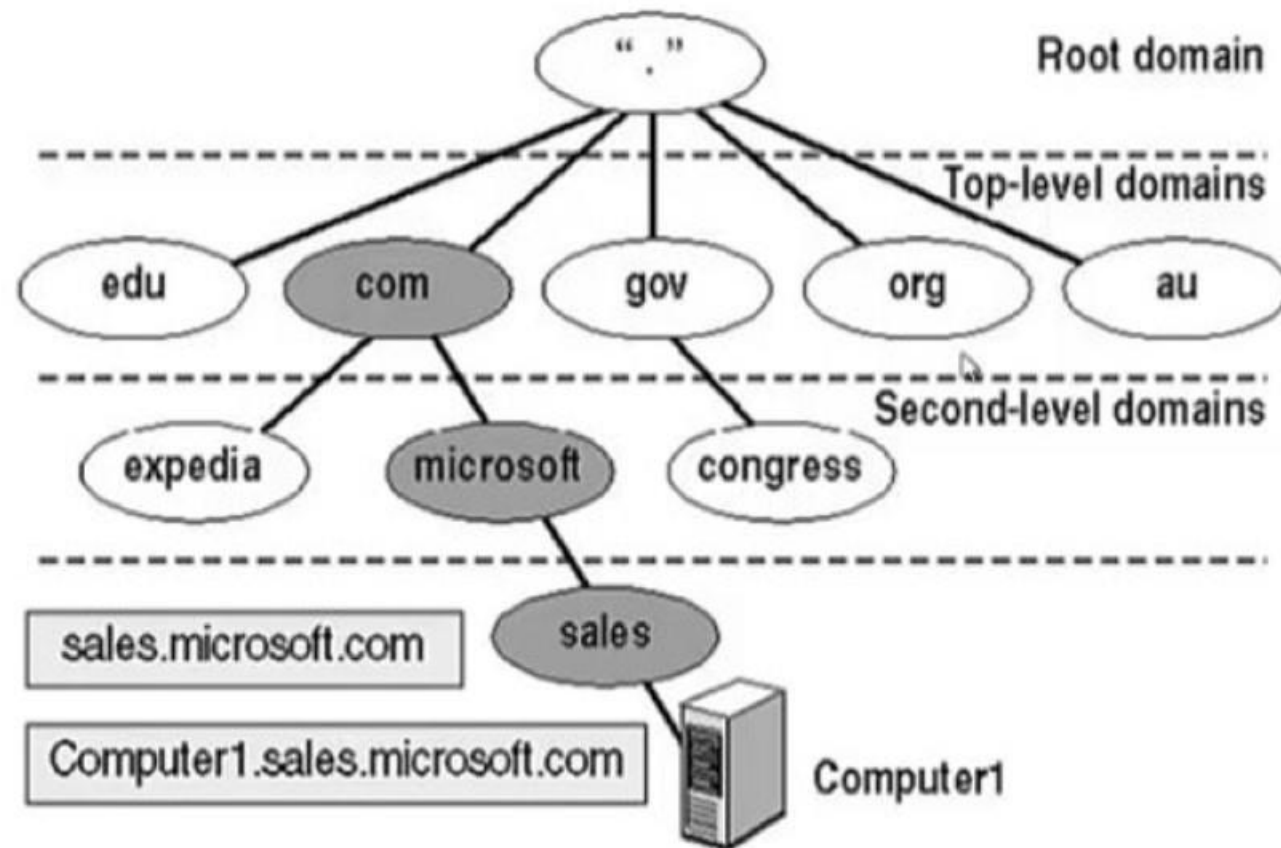
INTERNAL WORK OF DNS

Cont'd

- Writing a website URL, browsers does not understand it, it needs IP address, So:
 - Browser sends a query to OS which forwards it to the DNS resolver.
 - DNS resolver checks if it has the requested IP in its cache or not.
 - If it is in the cache, it gets IP address and sends it back to the browser.
 - If not, the query will be sent to the RNS which checks for the domain extension and sends back the IP address for the corresponding TLD.
 - The DNS resolver query the given TLD to get the IP address for the authoritative name server, which contains the IP address for the requested website.

INTERNAL WORK OF DNS

Cont'd



Lecture

07

INTERNAL WORK OF DNS

Cont'd

- The naming system on which DNS is based is a hierarchical and logical tree structure called **the domain namespace**.
- Every node in the DNS domain tree can be identified by a fully qualified domain name (FQDN).
- To allow computers to properly recognize a fully qualified domain name:
 - Dots are placed between each part of the name.
 - All resolvers start their lookups at the root.
 - The resolver navigates its way down the tree until it gets to the left-most part of the domain name.

Lecture

07

FULLY QUALIFIED DOMAIN NAME

- Fully qualified domain name (FQDN): names an internet site or host by appending its organization's full domain name to the name of a particular machine in that organization.



Lecture

07

DNS RECORD INFORMATION

- **A:** used for converting domain name to IPv4 address.
- **AAAA:** used for converting domain name to IPv6 host address.
- **PTR** (Pointer) : used for converting IP to a domain name.
- **CNAME** (Canonical Name): used to map a domain name as an alias for another domain.
- **MX** (Mail eXchanger): specifies a mail server responsible for accepting email messages on behalf of a recipient's domain.
- **NS** (Name Server): gives a name for each group of servers used in a domain.
- **TXT** (Descriptive Text): lets the administrator insert into the DNS record.

Lecture

07

DNS RECORD INFORMATION

Cont'd

- Install DNS corresponding module.

- In Windows:

```
python -m pip install dnspython3
```

- In Linux:

```
pip install dnspython3
```

```
python3 -m pip install dnspython3
```

- Do not name your code file with **dns.py**, because the import statement will give more errors.

Lecture

07

DNS RECORD INFORMATION

Cont'd

- Code:

```
import dns.resolver as dns
def lookup(name):
    NSrecord = dns.query(name, 'NS')
    for ns in NSrecord:
        print(ns)
```

```
lookup('google.com')
lookup('matrix219.com')
```

Lecture

07

DNS RECORD INFORMATION

Cont'd

• Code:

```
import dns.resolver
def lookup(name):
    for qtype in 'A', 'AAAA', 'CNAME', 'MX', 'NS':
        answer = dns.resolver.query(name, qtype, raise_on_no_answer=False)
        for val in answer:
            print(val)

lookup('google.com')
```

Lecture

07

SOCKETS WITH GETADDRINFO METHOD

- The `getaddrinfo()` uses the DNS protocol to talk to your name servers, which are configured using their IP address.
- The `getaddrinfo()` method returns a list of tuples - `[(0, 0)]` - for a given network service with which socket objects can be created to connect to that service.

```
socket.getaddrinfo(host, port, family=0, type=0, proto=0, flags=0)
```

SOCKETS WITH GETADDRINFO METHOD

Cont'd

```
socket.getaddrinfo(host, port, family=0, type=0, proto=0, flags=0)
```

- Host: IP address or host name.
- Port: could be a number or service name like http and www.
- Family: IP version.
- Type: Stream or Datagram.
- Proto: TCP or UDP.
- Flags: used for filtering the addresses.

Lecture

07

**SOCKETS WITH GETADDRINFO
METHOD**

Cont'd

• Code.

```
import socket
infolist = socket.getaddrinfo('google.com', 'www')
print(infolist)
info = infolist[0]
print(*info[0:2])
sok = socket.socket(*info[0:2])
print(info[4])
sok.connect(info[4])
```

Lecture

07

**CLIENT/SERVER USING
GETADDRINFO METHOD**

- If you want an address to provide to `bind()`, then you will call `getaddrinfo()` with `None` as the hostname.

```
Infolist = socket.getaddrinfo(None, 'www', socket.AF_INET)
```

CLIENT/SERVER USING GETADDRINFO METHOD

Cont'd

- Code: the server.

```
import socket
infolist = socket.getaddrinfo(None, 'www', socket.AF_INET)
info = infolist[0]
soc = socket.socket(*info[0:2])
soc.bind(info[4]); soc.listen();
con, addr = soc.accept()
con.send("Server: Hello".encode())
msg = con.recv(1024).decode()
print(msg)
```

CLIENT/SERVER USING GETADDRINFO METHOD

Cont'd

- Code: the client.

```
import socket
infolist = socket.getaddrinfo(None, 'www', socket.AF_INET)
info = infolist[0]
soc = socket.socket(*info[0:2])
soc.connect(info[4])
msg = soc.recv(1024).decode()
print(msg)
soc.send("client: I received your msg".encode())
```

Lecture

07

SOCKETS FILTERS

- To able to use `AF_INET6` with socket object, you should first test if the python supports IPv6 by checking the `has_ipv6`.
`print(socket.has_ipv6)`
- Using the `getaddrinfo()` with `AI_ADDRCONFIG` flag, will filter out any IPv6 addresses that are impossible for your computer to reach.
- If the service that you are connecting supports only IPv4, you will also want to specify `AI_V4MAPPED` to return the IPv4 addresses.

SOCKETS FILTERS

Cont'd

- Code:

```
import socket
print(socket.getaddrinfo('fedoraproject.org', 'ftp')
print(socket.getaddrinfo('fedoraproject.org',
                           'ftp',
                           socket.AF_INET,
                           socket.SOCK_STREAM,
                           0,
                           socket.AI_ADDRCONFIG |
                           socket.AI_V4MAPPED))
```

Lecture

07

QUESTIONS FOR ONLINE DISCUSSION

- Try to modify the `getaddrinfo()` client/server program to allow the server to accept multiple clients.



Thank You

