# Socket Names and DNS

# Socket module methods

- socket.gethostname()
- socket.gethostbyname('cern.ch')
- print(socket.gethostbyaddr('127.0.0.1'))
- print(socket.getprotobyname('UDP'))
- socket.getservbyname('www')
- print(socket.getservbyport(80))

# gethostname

- **Function Signature:**

  socket.gethostname()

- **Overview:**

  The Python function socket.gethostname() returns the host name of the current system under which the Python interpreter is executed.

- This Python function can be combined with socket.gethostbyname() to get the IP address of the local host.

- **Parameters:**   None

- **Return Value:**

  Host name of the localhost is returned.

# gethostbyname

- **Function Signature:**

  gethostbyname(hostname)

- **Overview:**

  Given a host name the gethostbyname() function returns the IP address of the host.

- gethostbyname() **returns** just one ip of the host even though the host could resolve to multiple IPs from a given location.

- The returned IP address is an IPv4 address.

- If the developer needs to resolve the hostname into IPv6 addresses or both IPv4 and IPv6 addresses are needed, socket.getaddrinfo() function can be used.

# getprotobyname

- **Function Signature:**

  getprotobyname(NameOfTheProtocol)

- **Function Overview:**

  - takes a protocol string like TCP, UDP or ICMP and returns the associated constant for the protocol as defined by the socket module.

  - For example, for TCP the getprotobyname() returns a constant value of 6, and for UDP the constant returned by getprotobyname() is 17 and icmp the returned constant value is 7.

# gethostbyaddr

► **Function Signature:**

  gethostbyaddr(ip_address)

► Function Overview:

Given an IP address the function socket.gethostbyaddr() returns a tuple containing

1. Host Name

2. Alias list for the IP address if any

3. IP address of the host

```
19    print(socket.getprotobyname('UDP'))
20    print(socket.gethostbyaddr("192.0.43.8"))
21

main
17
('43-8.any.icann.org', [], ['192.0.43.8'])
```
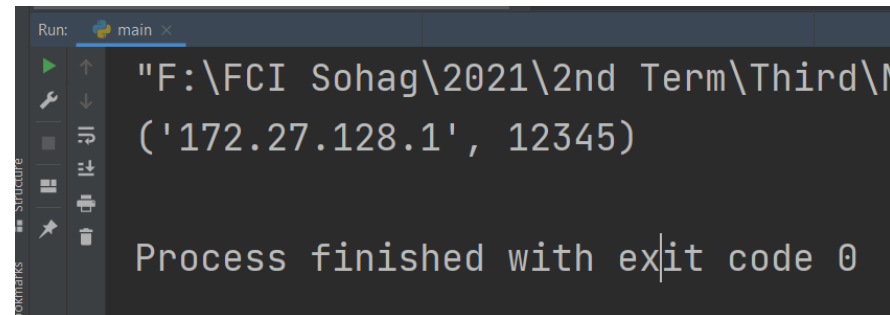
# socket object methods

▶ **mysocket.accept():** it returns a tuple whose second item is the remote address that has connected (the first item in the tuple is the new socket connected to that remote address

▶  **mysocket.bind(address):** This assigns the given local address to the socket

▶ **mysocket.connect(address):** This establishes that data sent through this socket will be directed to the given remote address.

  ▶ For UDP sockets, this simply sets the default address used if the caller uses send() rather than sendto() or recv() instead of recvfrom() but does not immediately perform any network communication. However,

  ▶  for TCP sockets, this actually negotiates a new stream with another machine using a three-way handshake and raises a Python exception if the negotiation fails.

# socket methods

- **mysocket.getpeername():** This returns the remote address to which this socket is connected.

- **mysocket.getsockname():** This returns the address of this socket's own local endpoint.

- **mysocket.recvfrom(...):** For UDP sockets, this returns a tuple that pairs a string of returned data with the address from which it was received.

- **mysocket.sendto(data, address):** An unconnected UDP port uses this method to fire off a data packet at a particular remote address.

# Getsockname method

```python
import socket
serv_sok=socket.socket(socket.AF_INET,so
host=socket.gethostname()
port=12345
serv_add=(host,port)
serv_sok.bind(serv_add)
serv_sok.listen()
print(serv_sok.getsockname())
```

```
Run:    main
    "F:\FCI Sohag\2021\2nd Term\Third\M
    ('172.27.128.1', 12345)

    Process finished with exit code 0
```

# getpeername Method from Client



```python
import socket
client_sok=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
host=socket.gethostname()
port=12345
serv_add=(host,port)
client_sok.connect(serv_add)
print(client_sok.getpeername())
```

```
('172.27.128.1', 12345)

Process finished with exit code 0
```

# Five Socket Coordinates

▶ import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('localhost', 1060))

▶ First, the *address family*
**AF_INET** is an address family that is used to designate the type of addresses that your socket can communicate with (in this case, Internet Protocol v4 addresses).

▶ The Linux kernel, for example, supports 29 other address families such as UNIX (AF_UNIX) sockets and IPX (AF_IPX), and also communications with IRDA and Bluetooth (AF_IRDA and AF_BLUETOOTH, but it is doubtful you'll use these at such a low level).
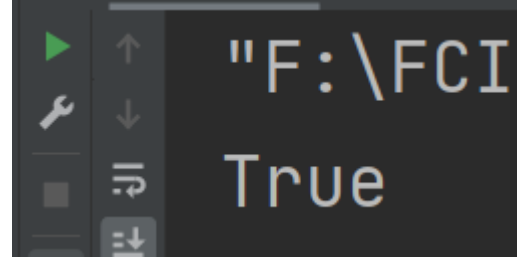
# Five Socket Coordinates con't

▶ Second, after the address family comes the *socket type*.

▶ The third field in the socket() call, **the *protocol***, is rarely used because once you have specified the address family and socket type, you have usually narrowed down the possible protocols to only one major option. Thus, programmers usually leave this unspecified, or they provide the value 0 to force it to be chosen automatically. If you want a stream under IP, the system knows to choose TCP. If you want datagrams, then it selects UDP.

▶ Finally, the fourth and fifth values used to make a connection are the IP address and port number

# IPv6

- The address family for IPv6, named AF_INET6
- To test platform supports IPv6 by checking the has_ipv6

```
print(socket.has_ipv6)
```

# getaddrinfo Method
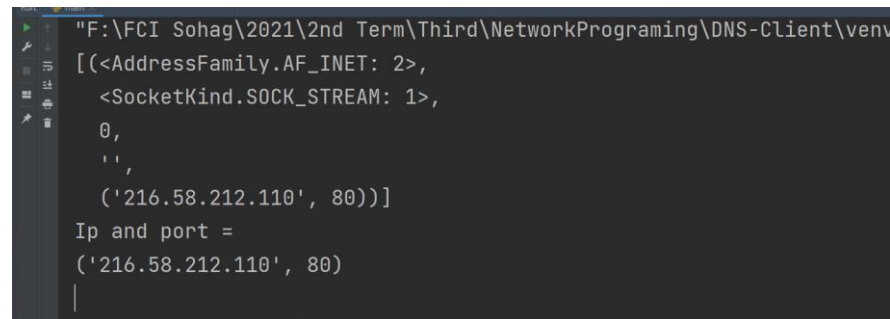
- returns a list of tuples for a given network service with which sockets can be created to connect to that service.

- use to transform the hostnames and port numbers that your users specify into addresses that can be used by socket methods.

- **Function Signature:**
  - **socket.getaddrinfo(host, port, family=0, type=0, proto=0, flags=0)**

- **Parameters**:

- host:     The host parameter takes either an ip address or a host name.

- port:    The port number of the service. If zero, information is returned for

-              all supported socket types.

# Getaddrinfo Example

```python
import socket

from pprint import pprint

infolist = socket.getaddrinfo('google.com', 'www',socket.AF_INET)
pprint(infolist)
print("Ip and port =")
pprint(infolist[0][4])
```

```
"F:\FCI Sohag\2021\2nd Term\Third\NetworkPrograming\DNS-Client\venv
[(<AddressFamily.AF_INET: 2>,
  <SocketKind.SOCK_STREAM: 1>,
  0,
  '',
  ('216.58.212.110', 80))]
Ip and port =
('216.58.212.110', 80)
```

# Getaddrinfo Example

```python
import socket
from pprint import pprint

infolist =
socket.getaddrinfo('gatech.edu',
'www')
pprint(infolist)
info = infolist[0]
print(info[0:3])

s = socket.socket( *info[0:3])
pprint(info[4])
s.connect(info[4])
```

```
"F:\FCI Sohag\2021\2nd Term\Third\NetworkPrograming\DNS-Client\
[(<AddressFamily.AF_INET: 2>,
  <SocketKind.SOCK_STREAM: 1>,
  0,
  '',
  ('3.214.16.8', 80))]
(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>, 0)
('3.214.16.8', 80)
```

# Using getaddrinfo() to Bind Your Server to a Port

► If you want an address to provide to bind(), then you will call getaddrinfo() with None as the hostname but with the port number and socket type filled in.

► **infolist=socket.getaddrinfo(None, 'smtp', 0, socket.SOCK_STREAM, 0)**

►

# Using getaddrinfo() to Connect to a Service

- ```python
  from pprint import pprint

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

  pprint(socket.getaddrinfo('iana.org', 'www', 0,
  socket.SOCK_STREAM, 0,socket.AI_ADDRCONFIG |
  socket.AI_V4MAPPED))

  pprint(socket.getaddrinfo('iana.org', 'www', 0,
  socket.SOCK_STREAM,0))
  ```

# AI_ADDRCONFIG and AI_V4MAPPED

▶ AI_ADDRCONFIG. It turns off AAAA lookups if the machine isn't configured for IPv6 (and similarly for IPv4, theoretically). This is especially important when behind gateways whose DNS forwarder silently filter AAAA requests. Without AI_ADDRCONFIG, every DNS request has to wait for 4 AAAA request timeouts before it even attempts an A request.

▶ If the AI_V4MAPPED flag is specified with the *ai_family* field using the value of AF_INET6 or AF_UNSPEC, then the caller will accept IPv4-mapped IPv6 addresses.

# Using getsockaddr() in Your Own Code Client

```python
infolist=socket.getaddrinfo(None, 'smtp', socket.AF_INET, socket.SOCK_STREAM,
0,socket.AI_ADDRCONFIG)

info = infolist[0]
print(info[0:3])
s = socket.socket(*info[0:3])
s.connect(info[4])
msg=s.recv(1024).decode()
print(msg)
s.send("Client: I recieved your msg".encode())
```

# Using getsockaddr() in Your Own Code Server

```python
import socket
infolist=socket.getaddrinfo(None, 'smtp', socket.AF_INET, socket.SOCK_STREAM)
info = infolist[0]
print(info[0:3])
s = socket.socket(*info[0:3])
s.bind(info[4])
s.listen()
conn,add=s.accept()
conn.send("Server: Hello".encode())
msg=conn.recv(1024).decode()
print(msg)
```

# Thank You