

Network Programming with Python

Required common installation modules: PIP and IDLE	
PIP (Python Package Installer)	\$ sudo apt-get install python-pip
IDLE (Integrated Development and Learning Environment)	\$ sudo apt-get install idle

Top Python Network Programming Libraries	
Django	High-level Python Web framework for rapid development and pragmatic
pycos (formerly asncoro)	Python framework for asynchronous, concurrent, network, distributed programming and distributed computing
Diesel	A clean API for writing network clients and servers. TCP and UDP supported. Bundles clients for HTTP, DNS, Redis, Riak and MongoDB.
Pulsar	Easy way to build scalable network programs
Twisted	Event-based framework for internet applications: HTTP clients and servers, SSHv2 and Telnet, IRC, XMPP, IMAPv4, POP3, SMTP, IMAPv4, POP3, SMTP, etc.
NAPALM	Network Automation and Programmability Abstraction Layer with Multivendor support - For dealing with dvice vendors
gevent	A coroutine -based Python networking library that uses greenlet to provide a high-level synchronous API on top of the libev or libuv event loop
Celery	Asynchronous task queue/job queue based on distributed message passing

Data Types	
Text	str - x = "Hello World"
Numeric	int, float, complex
Sequence	list, tuple, range
Mapping	dict
Set	set, frozenset
Boolean	bool
Binary	bytes, bytearray, memoryview
Socket Module (Berkley API interface)	
Primary Functions and Methods	socket() • ind() • listen() • accept() • connect() • connect_ex() • send() • recv() • close()

Math Operators	
**	Exponent 4 ** 2 = 16
%	Modulus/Remainder 43 % 5 = 3
//	Integer division 11 // 5 = 2
/	Division 11 / 5 = 2.2
*	Multiplication 3 * 3 = 9
-	Subtraction 8 - 3 = 5
+	Addition 2 + 2 = 4
==	Equal to
!=	Not equal to
<	Less than
>	Greater Than
<=	Less than or Equal to
>=	Greater than or Equal to
#	Can be used at the beginning of a line, or from within a line to the end of the line

Server-side socket example	
<pre>import socket s=socket.socket(socket.AF_INET,socket.SOCK_STREAM) host=socket.gethostname() port=1111 myserver.bind((host,port)) # replace myserver and myclient with respective IPs myserver.listen(5) while True: myclient,addr=myserver.accept() print("Connected to {str(addr)}") myclient.send(msg.encode("ascii")) myclient.close()</pre>	

Socket Errors / Exceptions	
exception socket.error	A deprecated alias of OSError, raised when a system function returns a system-related error
exception socket.herror	raised for address-related errors
exception socket.gaierror	raised for address-related errors by getaddrinfo() and getnameinfo()
exception socket.timeout	raised when a timeout occurs on a socket which has had timeouts enabled via a prior call to settimeout() (or implicitly through setdefaulttimeout())

Server-side socket example with Comments	
# Echo server program # Import socket module import socket # Create a socket object s = socket.socket() # Define the port on which you want to connect port=1111 # connect to the server on local computer s.connect(('172.18.0.1', port)) # receive data from the server print (s.recv(1024)) # close the connection s.close()	

Network forensics: Required python libraries and scripts	
EDDIE Tool	System and network monitoring, security, and performance analysis agent for python
pypcap	Small packet capture tool based on python and pcap
Paramiko	Implementation of the SSHv2 protocol, providing both client and server functionality
pip	Package installer for python
The Python Package Index (PyPI)	Repository of software for the Python

Python Keywords	
del	Delete objects
else	Conditional statement
except	Used in try-except blocks in error handling
finally	Used in try-except blocks in error handling, runs with or without errors
in	Check a value include in a sequence
import	Import module to current environment
lambda	Anonymous function
print	Standard print
try	Used in error handling

dnspython library	
Installation	
\$ pip install dnspython	
Basic DNS query	
import dns.resolver name = 'google.com' for qtype in 'A', 'AAAA', 'MX', 'NS', 'TXT', 'SOA': answer = dns.resolver.query(name,qtype, raise_on_no_answer=False) if answer.rdtype is not None: print(answer.rdtype)	
Get MX target and name preference	
import dns.resolver answers = dns.resolver.query('dnspython.org', 'MX') for rdata in answers: print ('Host', rdata.exchange, 'has preference', rdata.preference)	

Server-side socket example	
<pre>import socket HOST = '' # Symbolic name meaning all available interfaces PORT = 52542 # Arbitrary non-privileged port s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) s.bind((HOST, PORT)) s.listen(1) conn, addr = s.accept() print ('Connected by', addr) while 1: data = conn.recv(1024) if not data: break conn.sendall(data) conn.close()</pre>	

Network Analysis with Python	
Use NMAP with port scanner	\$ pip install python-nmap
Commands to run NMAP scan	
import nmap nmScan = nmap.PortScanner() nmScan.scan('10.1.0.0', '25-443')	
NMAP commands used with python	
nmScan.scaninfo() # {'tcp': {'services': '25-80', 'method': 'connect'}}	
nmScan.all_hosts()	
nmScan['10.1.0.0'].hostname()	
nmScan['10.1.0.0'].state()	
nmScan['10.1.0.0'].all_protocols()	
nmScan['10.1.0.0']['tcp'].keys() # Results -[80, 25, 22, 135]	
nmScan['10.1.0.0'].has_tcp(25) # Result -True/False	
nmScan['10.1.0.0'].has_tcp(21) # Result False/True	

Parsing Modules	
argparse()	The argparse module makes it easy to write user-friendly command-line interfaces. The program defines what arguments it requires, and argparse will figure out how to parse those out of sys.argv
Creating a parser	>>> parser = argparse.ArgumentParser(description='Process some integers.')
Adding arguments	>>> parser.add_argument('integers', metavar='N', type=int, nargs='+', ... help='an integer for the accumulator') >>> parser.add_argument('--sum', dest='accumulate', action='store_const', ... const=sum, default=max, ... help='sum the integers (default: find the max)')
Parsing arguments	>>> parser.parse_args(['--sum', '7', '-1', '42']) Namespace(accumulate=<built-in function sum>, integers=[7, -1, 42])

Socket Types	
SOCK_STREAM	For TCP protocols • Reliable transmission • Packet sequence • Connection-oriented • Bidirectional
SOCK_DGRAM	For UDP protocols • Unreliable transmission • No sequence of packets • Connectionless(UDP) • Not Bidirectional
Create a socket	
import socket # Imports the socket method	
socket.socket() # Function that creates socket	
sock = socket.socket(socket.family, socket.type, protocol=value)	
Socket Family	AF_UNIX or AF_INET
Socket Type	SOCK_STREAM or SOCK_DGRAM for TCP & UDP respectively • e.g. TCP - UDP2 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) • e.g. UDP - TCP2 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
Client socket method	connect()
Server socket method	bind() • listen(backlog) • accept()
TCP socket methods	s.recv() # Receive TCP packets s.send() #Send TCP packets
UDP socket methods	s.recvfrom() # Receives UDP packets s.sendto() # Transmits UDP packets
More Socket Methods	
close()	Close the socket connection
gethostname()	Returns a string which includes the hostname of the current PC
gethostbyname()	Returns a string which includes the hostname and IP address of the current PC
listen()	Setup and start TCP listener
bind()	Attach (host-name, port number) to the socket
accept()	TCP client connection wait
connect()	Initiate TCP server connection
TCP Socket Methods	
mysocket.accept()	Returns a tuple with the remote address that has connected
mysocket.bind(address)	Attach the specified local address to the socket
mysocket.connect(address)	Data sent through the socket assigns to the given remote address
mysocket.getpeername()	Returns the remote address where the socket is connected
mysocket.getsockname()	Returns the address of the socket's own local endpoint
mysocket.sendto(data, address)	Force a data packet to a specific remote address
Socket Blocking	
setblocking(1)	Setup block
setblocking(0)	Remove / un-setup block
Get port number using domain name	
import socket	
socket.getservbyname('domain name')	
Check support for IPV6	
import socket socket.has_ipv6 # Answer is TRUE or FALSE	
getaddrinfo() - Bind Server to a Port	
from socket import getaddrinfo getaddrinfo(None, 'FTP', 0, socket.SOCK_STREAM, 0, socket.AI_PASSIVE) [(2, 1, 6, '', ('0.0.0.0', 21)), (10, 1, 6, '', ('::', 21, 0, 0))]	
Script Examples	
Create list of devices	
>>>devices = ['SW1', 'SW2', 'SW3']	
Create VLAN dictionary list	
vlans = [{'id': '100', 'name': 'staff'}, {'id': '200', 'name': 'VOICE'}, {'id': '300', 'name': 'wireless'}]	
Write functions to collect commands and push to the network	
>>>def get_commands(vlan, name): commands = [] commands.append('vlan ' + vlan) commands.append('name ' + name) return commands >>> def push_commands(device, commands): print('Connecting to device: ' + device) for cmd in commands: print('Sending command: ' + cmd)	
Create VLANs in multiple switches using python script	
>>>for vlan in vlans: id = vlan.get('id') name = vlan.get('name') print('\n') print('Configure VLAN:' + id) commands = get_commands(id, name) for device in devices: push_commands(device, commands) print('\n')	
Citation: https://www.oreilly.com/library/view/network-programmability-and/9781491931240/ch04.html	
Disable router interface using python command	
>>> from push import push_commands device = 'router2' commands = ['interface Eth0/1', 'shutdown'] push_commands(device, commands)	