Network Automation:

Network automation is the process of automating the configuring, managing, testing, deploying, and operating of physical and virtual devices within a network. Network automation is a methodology in which software automatically configures, provisions, manages and tests network devices. It is used by enterprises and service providers to improve efficiency and reduce human error and operating expenses. Every day network tasks and functions are performed automatically. Managing bandwidth and finding fast reroutes to implement the best computing paths. Automation is any process that is self-driven, that reduces and potentially eliminates, the need for human intervention. Automation was once confined to the manufacturing industry. Highly repetitive tasks, such as automobile assembly, were turned over to machines and the modern assembly line was born. Machines excel at repeating the same task without fatigue and without the errors that humans are prone to make in such jobs. Network Automation and Programmability are skills that network engineers of today and the future are going to need to know.

Python:

Python is widely used to perform network automation. Interacting with network devices. Python is widely used to perform network automation. Python is an open source scripting language, thus used to automate anything. Python is a popular programming language. It was created by Guido van Rossum and released in 1991. Python has a simple syntax similar to the English language. The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular. Python is an interpreted language, which means you just type in plain text to an interpreter, and things happen. There is no compilation step, as in languages such as c. Python is a general-purpose programming language which is dynamically typed, interpreted, and known for its easy readability with great design principles. Python is one of the easier languages to get started with and interpret. Python is an object-oriented programming language. Almost everything in Python is an object, with its properties and methods. A Class is like an object constructor, or a "blueprint" for creating objects. Python is a widely-used, interpreted, object-oriented, and high-level programming language with dynamic semantics, used for general-purpose programming. There are two main kinds of Python, called Python 2 and Python 3. Python 2 is an older version of the original Python. Python 3 is the newer (to be precise, the current) version of the language. It's going through its own evolution path, creating its own standards and habits.

String:

A string is simply one or more alphanumeric characters. A string can comprise many numbers or letters, depending on the Python version in use. String literals in python are surrounded by either single quotation marks, or double quotation marks. 'hello' is the same as "hello". You can display a string literal with the print() function:

Comments:

Comments start with a #, and Python will render the rest of the line as a comment:

```
#This is a comment
print("Hello, World!")
```

Variables:

Variables are containers for storing data values. Unlike other programming languages, Python has no command for declaring a variable. A variable is created the moment you first assign a value to it.

```
x = 6
y = "ali"
print(x)
print(y)
```

Data Types:

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things. Python has the following data types built-in by default, in these categories:

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview

```
x = 6
print(type(x))
```

Print Function:

The print() function is a built-in function. It prints/outputs a specified message to the screen/console window.

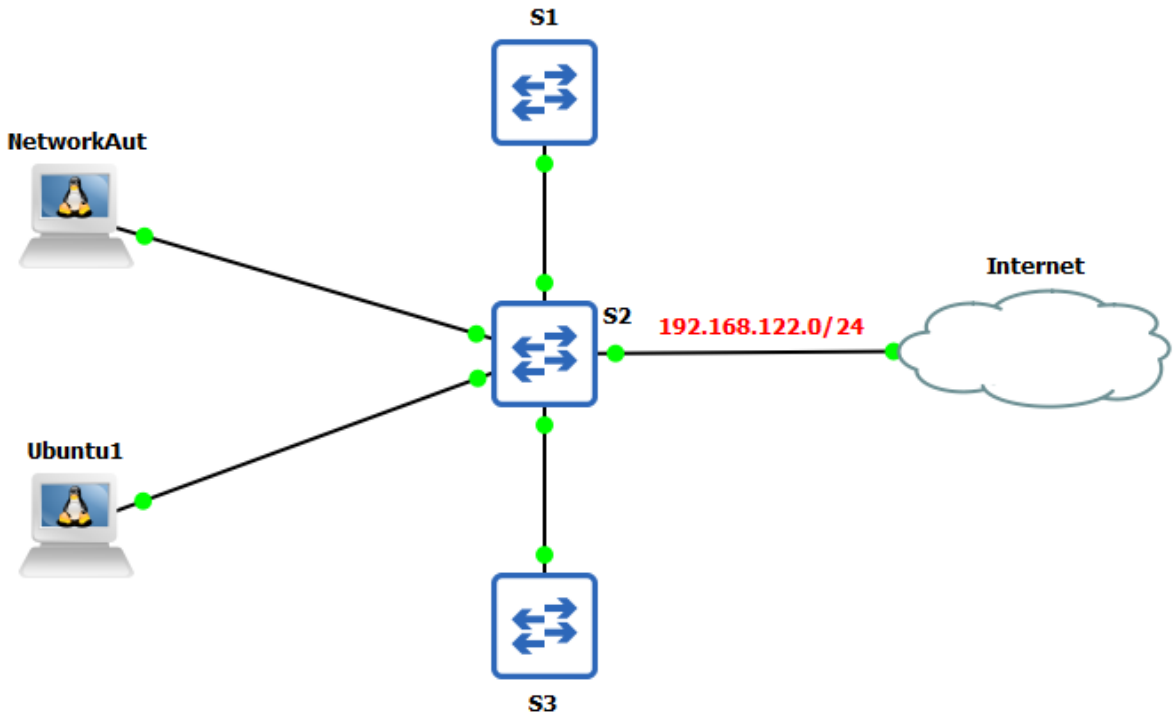
```
print(“Hello Network Engineers”)
```

Numeric Type:

There are three numeric types in Python: Int, float and complex. Variables of numeric types are created when you assign a value to them. To verify the type of any object in Python, use the type() function: integer are numbers written without a fractional component. Floating are numbers that contain or are able to contain a fractional component.

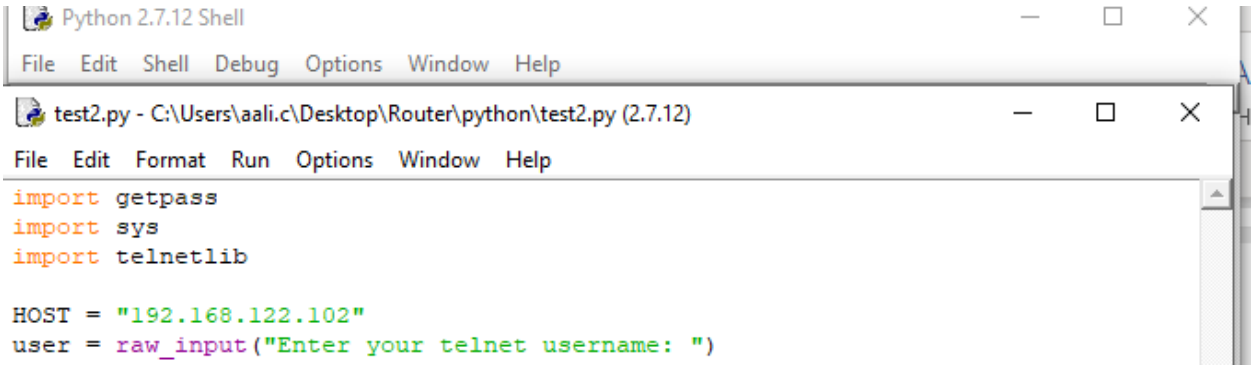
```
x = 1    # int
y = 2.8  # float
print(type(x))
print(type(y))
```

Lab Time:



NetworkAut and Ubuntu1 PC configuration	# DHCP config for eth0 auto eth0 iface eth0 inet dhcp
S1 Switch Configuration	S1(config)#interface vlan 1 S1(config-if)#ip address 192.168.122.101 255.255.255.0 S1(config-if)#no shutdown
S2 Switch Configuration	S2(config)#interface vlan 1 S2(config-if)#ip address 192.168.122.102 255.255.255.0 S2(config-if)#no shutdown
S3 Switch Configuration	S3(config)#interface vlan 1 S3(config-if)#ip address 192.168.122.103 255.255.255.0 S3(config-if)#no shutdown

Switch 1 Telnet Configuration
S1(config)#enable password secret S1(config)#username admin password cisco S1(config)#line vty 0 4 S1(config-line)#login local S1(config-line)#transport input all S1(config-line)#exit
Switch 2 Telnet Configuration
S2(config)#enable password secret S2(config)#username admin password cisco S2(config)#line vty 0 4 S2(config-line)#login local S2(config-line)#transport input all S2(config-line)#exit
Switch 3 Telnet Configuration
S3(config)#enable password secret S3(config)#username admin password cisco S3(config)#line vty 0 4 S3(config-line)#login local S3(config-line)#transport input all S3(config-line)#exit
Ubuntu PC
root@Ubuntu1:~# apt-get update root@Ubuntu1:~# apt-get install python root@Ubuntu1:~# python –version root@Ubuntu1:~# apt-get install vim root@Ubuntu1:~# python c.py
NetworkAuto PC
root@NetworkAut:~# apt-get update root@NetworkAut:~# python –version root@NetworkAut:~# apt-get install vim root@NetworkAut:~# python a.py



JSON (JavaScript Object Notation):

JSON is a syntax for storing and exchanging data. JSON is text, written with JavaScript object notation. Python has a built-in package called json, which can be used to work with JSON data. Import the json module: import json. When exchanging data between a browser and a server, the data can only be text. JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server. JSON is a lightweight data-interchange format. Both JSON and XML can be used to receive data from a web server. JSON is a human readable data format used by applications for storing, transferring and reading data. JSON is a very popular format used by web services and APIs to provide public data. This is because it is easy to parse and can be used with most modern programming languages, including Python. Both JSON and XML can be used to receive data from a web server. XML is much more difficult to parse than JSON. JSON is parsed into a ready-to-use JavaScript object. JSON is light-weight as compare to XML. It is language independent. Easy to read and write. Text based, human readable data exchange format. These are some of the characteristics of JSON:

- It uses a hierarchical structure and contains nested values.
- It uses braces { } to hold objects and square brackets [ ] hold arrays.
- Its data is written as key/value pairs.

JSON Data:

In JSON, the data known as an object is one or more key/value pairs enclosed in braces { }. JSON data is written as name/value pairs, just like JavaScript object properties. A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value: JSON names require double quotes. JavaScript names do not.

```
"firstName":"Ali"
```

JSON Objects:

JSON objects are written inside curly braces. Just like in JavaScript, objects can contain multiple name/value pairs: Always surrounded by curly brackets. Composed of one-or-more name-value

```
{"firstName":"Ali", "lastName":"Khan"}
```

JSON Arrays:

JSON arrays are written inside square brackets. Just like in JavaScript, an array can contain objects: the object "employees" is an array. It contains three objects. Each object is a record of a person (with a first name and a last name).

```
"employees":[
   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter", "lastName":"Jones"}
]
```

JSON List of IPv4 Addresses:

```
{
 "addresses": [
  {
   "ip": "172.16.0.2",
   "netmask": "255.255.255.0"
  },
  {
   "ip": "172.16.0.3",
   "netmask": "255.255.255.0"
  },
  {
   "ip": "172.16.0.4",
   "netmask": "255.255.255.0"
  } ] }
```

Lab Time:

S1 Configuration
S1#terminal length 0
S1#show running-config | format

When run this command: show running-config | format it will show the running configuration in XML format copy the data go to any XML to JSON convertor in google such as https://codebeautify.org/xmltojson

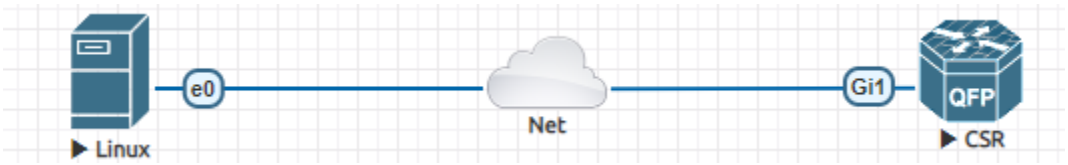
Data Models:

Data models describe the things you can configure, monitor, and the actions you can perform on a network device. So, what is a data model of a network device? To answer that, let's imagine a hypothetical scenario where your friend asks you what IP interface attributes can be configured on a specific router. You might say: "Well, to configure an interface on this router, you need to supply: an interface name, an IP address, and a subnet mask. You also need to enable the interface - the router will keep the interface disabled if you don't." Now, as simple as this response, what we just did was describe a data model for an IP interface. A YANG model will do the same but uses strict syntax rules to make the model standardized and easy to process with computers.

Yet Another Next Generation (YANG):

SNMP is widely used to monitor networks. You can use SNMP to configure network devices. YANG is a modeling language and uses data models that are similar to SNMP Management Information Base (MIBs). A language for building and defining data models. Can be used to build data models for any kind of data. These data models allow a uniform way for us to configure, monitor, and interact with network devices. Network automation tools like NETCONF, RESTCONF, and gRPC require YANG data models. YANG uses a hierarchical tree structure, similar to the XML data format. There is a clear distinction between configuration data and state information. A YANG module defines a data model through the data of a network device, and the hierarchical organization and constraints of that data. YANG identifies each module with a namespace URL. YANG modules to describe interface, Access-lists, Routing Tables etc. https://github.com/YangModels/yang

Lab Time:



CSR Router Configuration
Router(config)#hostname CSR CSR(config)#interface gigabitEthernet 1 CSR(config)#ip address 192.168.122.105 255.255.255.0 CSR(config-if)#no shutdown CSR(config)#ip route 0.0.0.0 0.0.0.0 192.168.122.2 CSR(config)#username admin privilege 15 password 123 CSR(config)#netconf-yang
CSR# show platform software yang-management process CSR# show netconf session CSR# clear netconf sessions

Ubuntu Virtual Machine
root@test-virtual-machine# ssh -s admin@192.168.122.105 netconf
root@test-virtual-machine# apt-get install python
root@test-virtual-machine# apt install python-pip
root@test-virtual-machine# pip install ncclient

The router responds with a “hello” message:

```
root@test-virtual-machine:/home/test/Desktop# ssh -s  admin@192.168.122.100 netc
onf
Password:
<?xml version="1.0" encoding="UTF-8"?><hello xmlns="urn:ietf:params:xml:ns:netco
nf:base:1.0"><capabilities><capability>urn:ietf:params:netconf:base:1.0</capabil
ity><capability>urn:ietf:params:netconf:capability:writeable-running:1.0</capabi
lity><capability>urn:ietf:params:netconf:capability:startup:1.0</capability><cap
ability>urn:ietf:params:netconf:capability:url:1.0</capability><capability>urn:c
isco:params:netconf:capability:pi-data-model:1.0</capability><capability>urn:cis
co:params:netconf:capability:notification:1.0</capability></capabilities><sessio
n-id>53497312</session-id></hello>]]>]]>
```

This hello message contains the capabilities that the router supports. We need to reply to the router with a hello message. This hello message contains the capabilities that we support:

```
<?xml version="1.0"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"xmlns:cpi="http://www.cisco.
com/cpi_10/schema"
message-id="101">
   <get-config>
        <source>
             <running/>
         </source>
          <filter>
              <config-format-text-cmd>
               <text-filter-spec>
interface GigabitEthernet1
                 </text-filter-spec>
              </config-format-text-cmd>
          </filter>
      </get-config>
</rpc>
```

Interacting with NETCONF manually over SSH is difficult method. Instead, we should use clients or code libraries that do most of the work for us. Ncclient is a popular python library we can use for NETCONF.

Ubuntu Virtual Machine
root@test-virtual-machine# apt-get install python
root@test-virtual-machine# apt install python-pip
root@test-virtual-machine# pip install ncclient
root@test-virtual-machine# python get-run-conf.py

NETCONF:

NETCONF is a protocol developed by IETF to “install, manipulate, and delete the configuration of network devices”. The goal of NETCONF is to make network automation easier. It uses XML for data encoding and Remote Procedure Call (RPC) for messages. It runs over SSH. NETCONF is an IETF standard protocol that uses the YANG data models to communicate with the various devices on the network. NETCONF runs over SSH.

Operation	Description
<get>	Retrieve running configuration and device state information.
<get-config>	Retrieve all or part of a specified configuration datastore.
<edit-config>	The <edit-config> operation loads all or part of a specified configuration to the specified target configuration datastore.
<copy-config>	Create or replace an entire configuration datastore with the contents of another complete configuration datastore.
<delete-config>	Delete a configuration datastore. The <running> configuration datastore cannot be deleted.
<commit>	The <commit> operation instructs the device to implement the configuration data contained in the candidate configuration.
<lock>	The <lock> operation allows the client to lock the entire configuration datastore system of a device.
<unlock>	The <unlock> operation is used to release a configuration lock, previously obtained with the <lock> operation.
<close-session>	Request graceful termination of a NETCONF session.
<kill-session>	Force the termination of a NETCONF session.

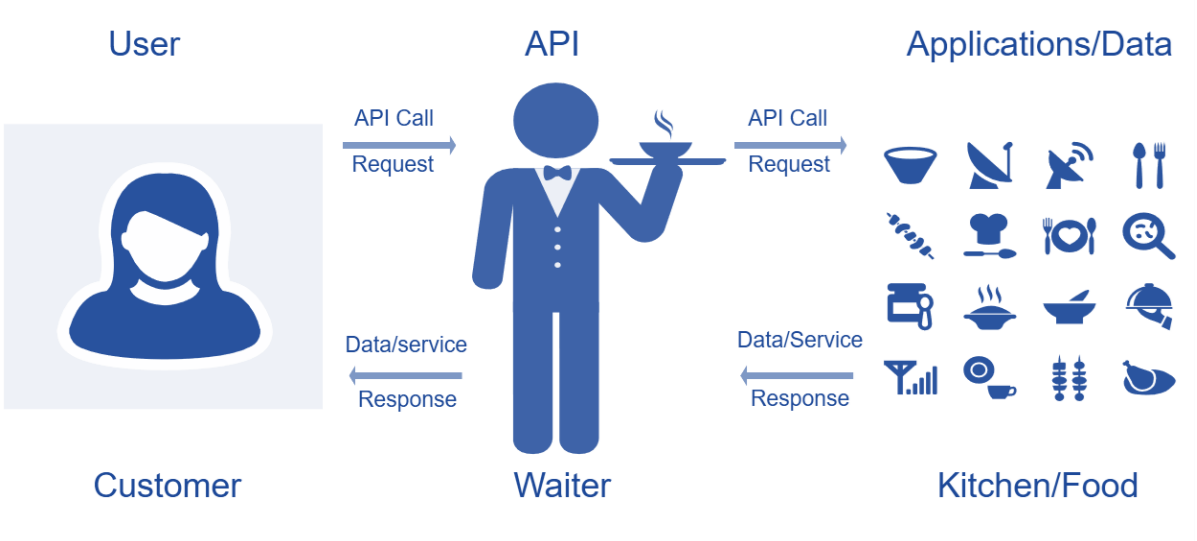
RESTCONF:

RESTCONF is protocol which works similar to a REST API. It maps a YANG specification to a RESTful interface and uses the HTTPS protocol for transport. You can use JSON or XML as data formats. RESTCONF is newer than NETCONF but not a replacement. It’s more of a lightweight option for engineers who are familiar with REST APIs.

RESTCONF	NETCONF
GET	<get>, <get-config>
POST	<edit-config> (operation=”create”)
PUT	<edit-config> (operation=”create/replace”)
PATCH	<edit-config> (operation=”merge”)
DELETE	<edit-config> (operation=”delete”)

API (Application Programming Interface):

To interact with applications or network devices, we can use an Application Programming Interface (API). An API is a software interface which allows other applications to communicate with our application. APIs are found almost everywhere. Amazon Web Services, Facebook, and home automation devices such as thermostats, refrigerators, and wireless lighting systems, all use APIs. They are also used for building programmable network automation. An API is software that allows other applications to access its data or services. It is a set of rules describing how one application can interact with another, and the instructions to allow the interaction to occur. The user sends an API request to a server asking for specific information and receives an API response in return from the server along with the requested information. XML and JSON are two of the most common data formats that are used with APIs. An API is similar to a waiter in a restaurant. A customer in a restaurant would like to have some food delivered to the table. The food is in the kitchen where it is cooked and prepared. The waiter is the messenger, similar to an API. The waiter (the API) is the person who takes the customer's order (the request) and tells the kitchen what to do. When the food is read, the waiter will then deliver the food (the response) back to the customer.



REST and RESTful API:

It's an acronym for **RE**presentational **S**tate **T**ransfer:

Representational means we transfer the *representation* of a resource between a server and a client. We use a data format for this representation, typically JSON or XML.

State Transfer means that each operation with a REST API is **self-contained**. Each request carries (transfers) all information (state) to complete the operation.

REST APIs **typically use HTTP methods** to retrieve or send information between applications. We use the same HTTP methods when we use a web browser to visit a website, but now we use them to interact with an application. HTTP has multiple methods, but these four are the most common ones:

GET: A read-only method to retrieve a specified resource.
POST: Submits data to the specified resource to process. The POST method can also create new resources.
PUT: Updates the specified resource by replacing the existing data.
DELETE: Deletes the specified resource.
HTTP is popular so you can use REST APIs in almost any programming language.
REST has no built-in security features, but if needed we can add these. For example, we can use HTTPS for encryption and usernames or tokens for authentication.

Embedded Event Manager (EEM):

Embedded Event Manager (EEM) is a very flexible and powerful Cisco IOS tool. EEM allows engineers to build software applets that can automate many tasks. EEM enables you to build custom scripts using Tcl. Scripts can automatically execute based on the output of an action or an event on a device. EEM is all contained within the local device. There is no need to rely on an external scripting engine or monitoring device in most cases.