QUESTION 16
Which HTTP status code indicates that a client application is experiencing intentional rate limiting by the server?

A. 202
B. 401
C. 429
D. 503

Answer: C

QUESTION 17
Which database type should be used to store data received from model-driven telemetry?

A. BigQuery database
B. Time series database
C. NoSQL database
D. PostgreSQL database

Answer: B

QUESTION 18
A heterogeneous network of vendors and device types needs automating for better efficiency and to enable future automated testing. The network consists of switches, routers, firewalls and load balancers from different vendors, however they all support the NETCONF/RESTCONF configuration standards and the YAML models with every feature the business requires. The business is looking for a buy versus build solution because they cannot dedicate engineering resources, and they need configuration diff and rollback functionality from day 1.
Which configuration management for automation tooling is needed for this solution?

A. Ansible
B. Ansible and Terraform
C. NSO
D. Terraform
E. Ansible and NSO

Answer: E

QUESTION 19
An automated solution is needed to configure VMs in numerous cloud provider environments to connect the environments to an SDWAN. The SDWAN edge VM is provided as an image in each of the relevant clouds and can be given an identity and all required configuration via cloud-init without needing to log into the VM once online.

Which configuration management and/or automation tooling is needed for this solution?

A. Ansible
B. Ansible and Terraform
C. NSO
D. Terraform
E. Ansible and NSO

Answer: E

QUESTION 20
Refer to the exhibit. The Ansible playbook is using the netconf_module to configure an interface using a YANG model. As part of this workflow, which YANG models augment the interface?

```
- name: Configure Interfaces
  with_items: "{{interfaces}}"
  netconf_config:
    <<: *host_info
    xml: |
      <config>
        <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interefaces">
          <interface>
            <name>{{item.interface_type}}{{item.interface_id}}</name>
            <description>{{item.description}}</description>
            <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
            <enabled>true</enabled>
            <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
              <address>
                <ip>{{item.ip_address}}</ip>
                <netmask>{{item.subnet_mask}}</netmask>
              </address>
            </ipv4>
          </interface>
        </interfaces>
      </config>
```

A. ietf-interfaces and ietf-ip
B. iana-if-type and ietf-interfaces
C. ietf-ip and openconfig-interface
D. ietf-ip and iana-if-type

Answer: B

QUESTION 21
Refer to the exhibit. Which key value pair from the ios_ntp Ansible module creates an NTP server peer?

```
---
- name: IOS XE Configuration
  hosts: ios_xe
  connection: local
  gather_facts: false

  tasks:
  - name: IOS NTP
    ios_ntp:
      provider: "{{ creds }}"
      server: 10.0.255.10
      source_int: GigabitEthernet2
      logging: false
    [          ]
```

A. state: present
B. state: True
C. config: present
D. config: True

Answer: A

QUESTION 22
Drag and Drop Question
Refer to the exhibit. The Python script is supposed to make an API call to Cisco DNA Center querying a wireless profile for the "ChicagoCampus" and then parsing out its enable FlexConnect value. Drag and drop the parts of the Python code from the left onto the item numbers on the right that match the missing sections in the exhibit.

GET `/dna/intent/api/v1/wireless/profile` Get Wireless Profile

Gets either one or all the wireless network profiles if no name is provided for network-profile.

Parameters

Name	Description
profileName `string` *(query)*	*Default value:*

Responses

Code	Description
200	*The request was successful. The result is contained in the response body.* **Example Value** Model

```
[
  {
    "profileDetails": {
      "name": "string",
      "sites": [
        "string"
      ],
      "ssidDetails": [
        {
          "name": "string",
          "type": "Guest",
          "enabledFabric": true,
          "flexConnect": {
            "enableFlexConnect": true,
            "localToVlan": 0
            },
            "InterfaceName": "string"
          }
        ]
      }
    }
  }
]
```

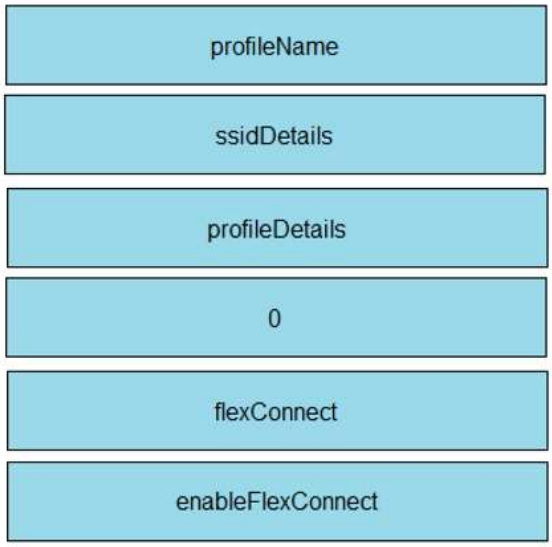
```
import requests
import json

def get_dnac_wireless_profiles():
    try:
      url = "https://sandboxdnac2.cisco.com/dna/intent/api/v1" \
      + "/wireless/profile?<item1>=ChicagoCampus|"

      print(token)
      payload = {}
      headers = {
      'x-auth-token': token
      }

      response = requests.request("GET", url, headers=headers, data = payload)
      response.raise_for_status()
      return response.json()[0]['<item 2>'][' <item 3>'] \
                          [<item 4>]['<item 5>']["<item 6>"]
    except Exception as e:
      print(e)

def create_dnac_token():
    try:
      url = "https://sandboxdnac2.cisco.com/dna/system/api/v1/auth/token"

      payload = {}
      headers = {
      'Authorization': 'Basic ZGV2bmV0dXNlcjpDaXNjbzEyMyE= ',
      'Content-Type': 'application/json'
      }

      response = requests.request("POST", url, headers=headers, data = payload)
      response.raise_for_status()
      return response.json()["Token"]
    except Exception as e:
      print(e)

if ____name____ == "___main____":
    token = create_dnac_token()
    print(get_dnac_wireless_profiles())
```

Answer Area

0	<item 1>
ssidDetails	<item 2>
profileDetails	<item 3>
profileName	<item 4>
flexConnect	<item 5>
enableFlexConnect	<item 6>

Answer:

Answer Area



QUESTION 23
Drag and Drop Question
Drag and drop the expressions from below onto the code to implement error handling. Not all options are used.

Answer Area

```
base_url = "https://api.meraki.com/api/v0"
posturl = '%s/networks/%s/staticRoutes' % ((str(base_url), str(networked)))
headers = {
      'x-cisco-meraki-api-key': api_key,
      'Content-Type': 'application/json'
}
routes = [ {
      "subnet": "10.16.4.0/22",
      "gatewayIp": "10.1.0.20",
      "name": "ROUTE1",
      "enabled": true
    }
    {
      "subnet": "10.253.254.0/24",
      "gatewayIp": "10.1.0.20",
      "name": "ROUTE2",
      "enabled": true
    }
    {
"subnet": "10.168.0.0/21",
      "gatewayIp": "10.1.0.20",
      "name": "ROUTE3",
      "enabled": true
    } ]

for route in routes:
      print("Adding static: " + str(route['subnet'])
      response = requests.post(posturl, json=route, headers=headers)
      [          ]
      print("Done!")
      [          ]
      print("Failed to add static: " + str(route['subnet']) + "\n" + response.text)
```

if response == 601:	else:	when:
if response == 201:	elif:	

Answer:

Answer Area

```
base_url = "https://api.meraki.com/api/v0"
posturl = '%s/networks/%s/staticRoutes' % ((str(base_url), str(networked)))
headers = {
     'x-cisco-meraki-api-key': api_key,
     'Content-Type': 'application/json'
}
routes = [ {
     "subnet": "10.16.4.0/22",
     "gatewayIp": "10.1.0.20",
     "name": "ROUTE1",
     "enabled": true
   }
   {
     "subnet": "10.253.254.0/24",
     "gatewayIp": "10.1.0.20",
     "name": "ROUTE2",
     "enabled": true
   }
   {
"subnet": "10.168.0.0/21",
     "gatewayIp": "10.1.0.20",
     "name": "ROUTE3",
     "enabled": true
   } ]

for route in routes:
     print("Adding static: " + str(route['subnet']))
     response = requests.post(posturl, json=route, headers=headers)
     [if response == 201:]
     print("Done!")
     [else:]
     print("Failed to add static: " + str(route['subnet']) + "\n" + response.text)
```

[if response == 601:] [when:] [elif:]

QUESTION 24
Drag and Drop Question
Refer to the exhibit. Drag and drop the code snippets from the left onto the item numbers on the right that match the missing sections in the exhibit to complete the script to implement control flow.

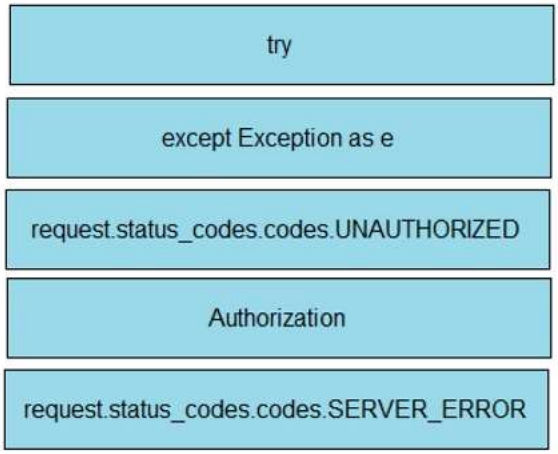
```
import request
import json
import sys

token = ""

def get_dnac_devices():
     <item 1>:
          url = "https://sanboxdnac.cisco.com/dna/intent/api/v1/network-device"

          print(token)
          payload = {}
          headers = {
          'Content-Type': 'application/json',
          "Accept": 'application/json',
          'x-auth-token': token
          }

          response = requests.request("GET", url, headers=headers, data = payload)
          response.raise_for_status()
          return response.text

     <item 2>:
          print(e)
          if str(<item 3>) in str(e):
             create_dnac_token()

def create_dnac_token():
     try:
          url = "https://sanboxdnac.cisco.com/dna/system/api/v1/auth/token"

          payload = {}
          headers ={
          '<item4>': 'Basic ZGV2bmV0dXNlcjpDaXNjbzEyMyE=',
          'Content-Type': 'application/json'
          }

          response = requests.request("POST", url, headers=headers, data = payload)
          response.raise_for_status()
          return response.json()["Token"]
     except Exception as e:
          print(e)
          if str(<item 5>) in str(e):
               sys.exit("DNAC Service is not reachable")

if __name__ == "__main__":
     token = create_dnac_token()
     print(get_dnac_devices())
```

Answer Area

except Exception as e	<item 1>
try	<item 2>
Authorization	<item 3>
request.status_codes.codes.SERVER_ERROR	<item 4>
request.status_codes.codes.UNAUTHORIZED	<item 5>

Answer:

Answer Area

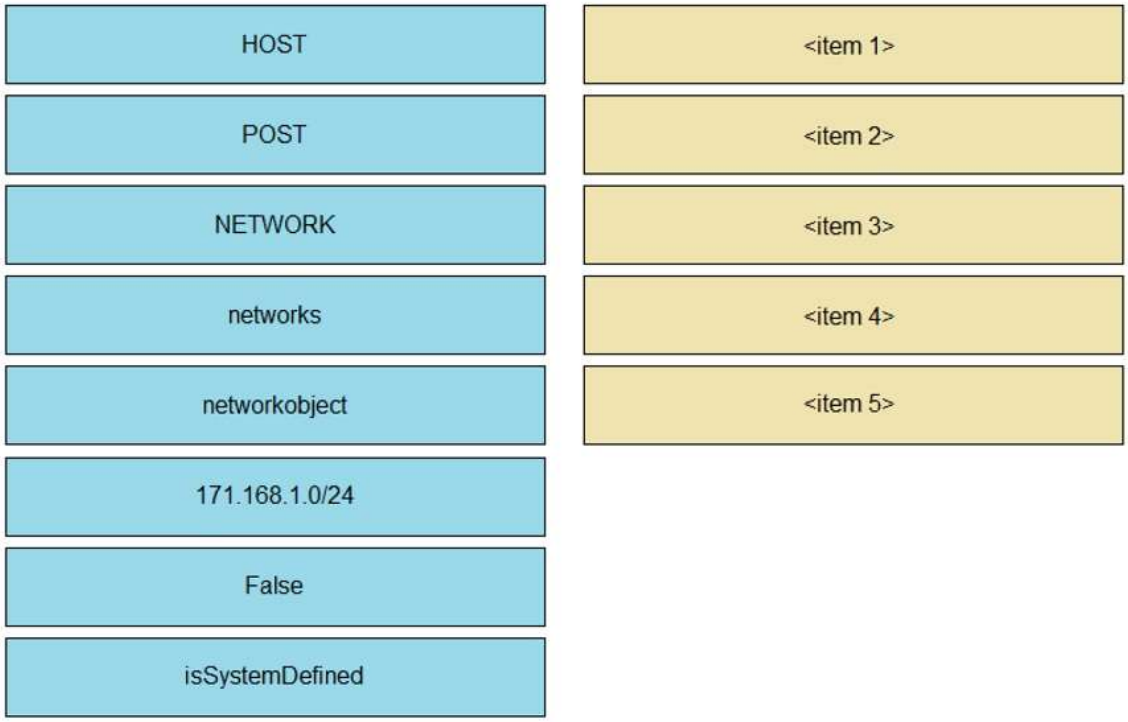


QUESTION 25

Drag and Drop Question

Refer to the exhibit. Drag and drop the code snippets from the left onto the item numbers on the right that match the missing sections in the cURL exhibit to complete the cURL request to FirePower Device Manager API to create objects. Not all code snippets are used.

Answer Area



Answer:

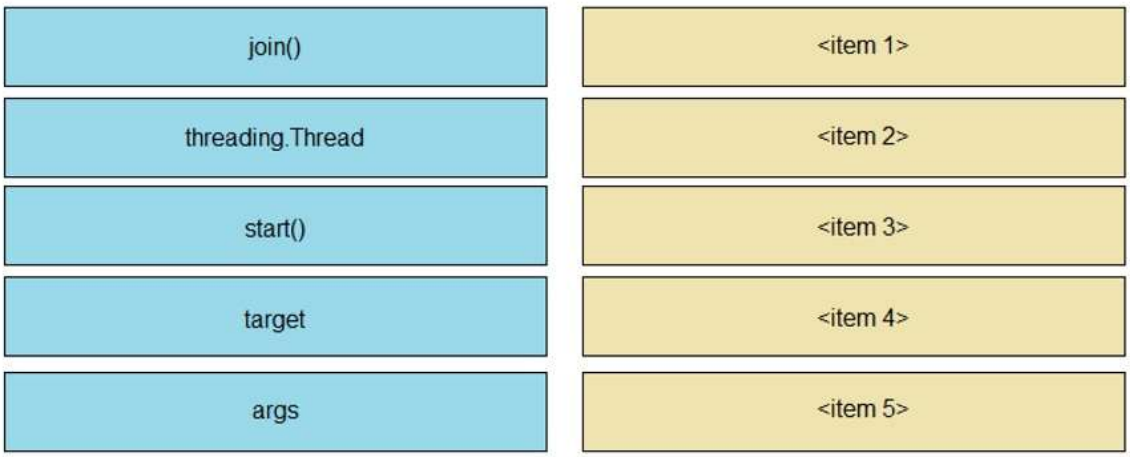
Answer Area

HOST	POST
	171.168.1.0/24
NETWORK	False
	networks
networkobject	isSystemDefined

QUESTION 26
Drag and Drop Question
Refer to the exhibit. Python threading allows a developer to have different parts of a program run concurrently and simplify a design. Drag and drop the code snippets from the left onto the item numbers on the right that match the missing sections in the exhibit to create a thread instance.

```
import threading
import requests

def get_device_list(endpoint, apikey):
     url = "https://api.meraki.com/api/v0/networks/" + endpoint
     hdr = {'x-cisco-meraki-api-key': format(str(apikey)), 'Content-Type':
'application/json'}
     response = requests.get(url=url, headers=hdr)
     print(response.json())

if __name__ == "__main__":
     # creating thread
     thread = <item 1>(<item2>=get_device_list,

     <item 3>=("NETWORK_ID/devices","API_TOKEN"))

     thread.<item 4>
     thread.<item 5>
```

Answer Area



Answer:

Answer Area