350-901$^{\text{Q\&As}}$

Developing Applications Using Cisco Core Platforms and APIs (DEVCOR)

Pass Cisco 350-901 Exam with 100% Guarantee

QUESTION 1

Which load balancing algorithm balances load based on the active sessions of a node?

A. weighted round-robin

B. IP source affinity

C. least connections

D. sticky session

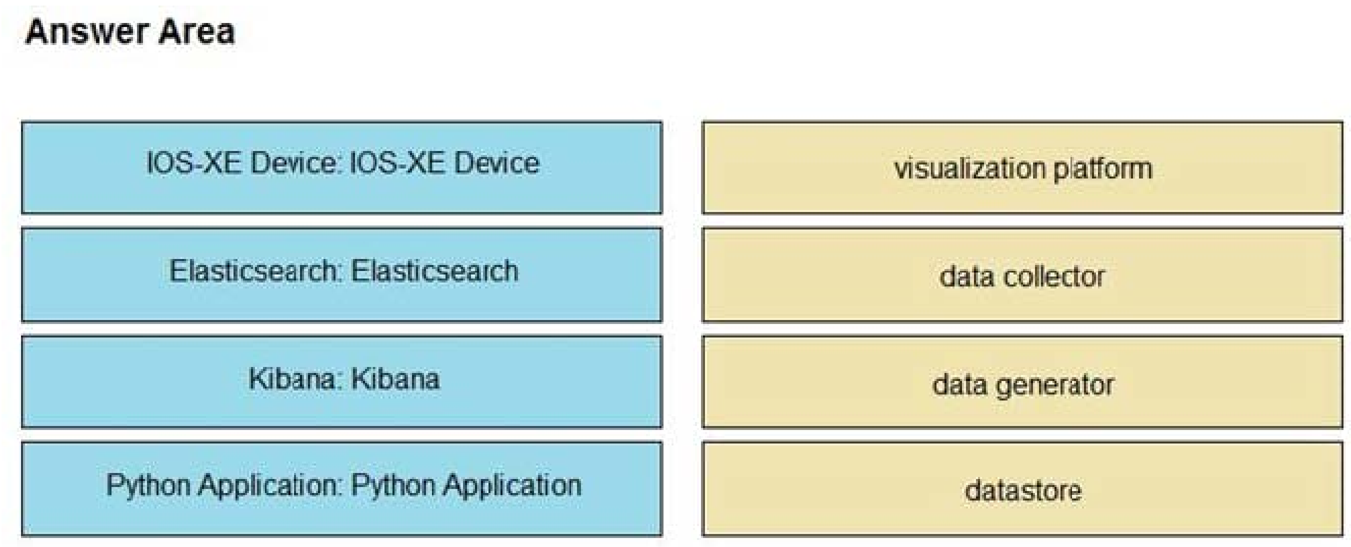
Correct Answer: C

QUESTION 2

DRAG DROP

An application is being built to collect and display telemetry streaming data. Drag and drop the elements of this stack from the left onto the correct element functions on the right.

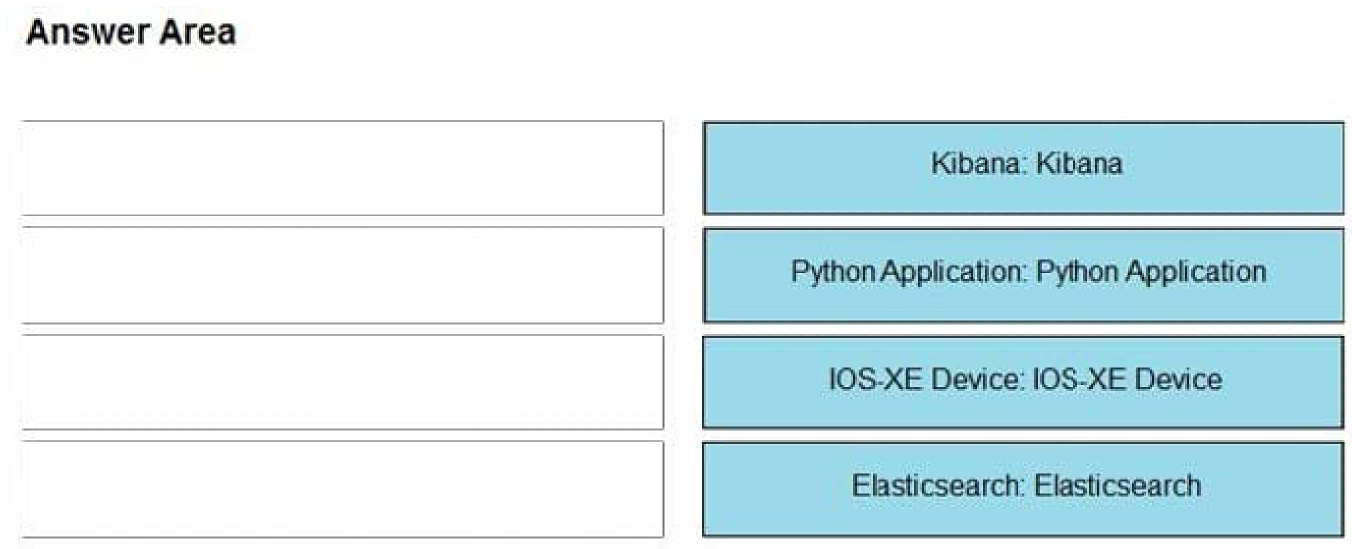
Select and Place:



Answer Area

IOS-XE Device: IOS-XE Device	visualization platform
Elasticsearch: Elasticsearch	data collector
Kibana: Kibana	data generator
Python Application: Python Application	datastore

Correct Answer:



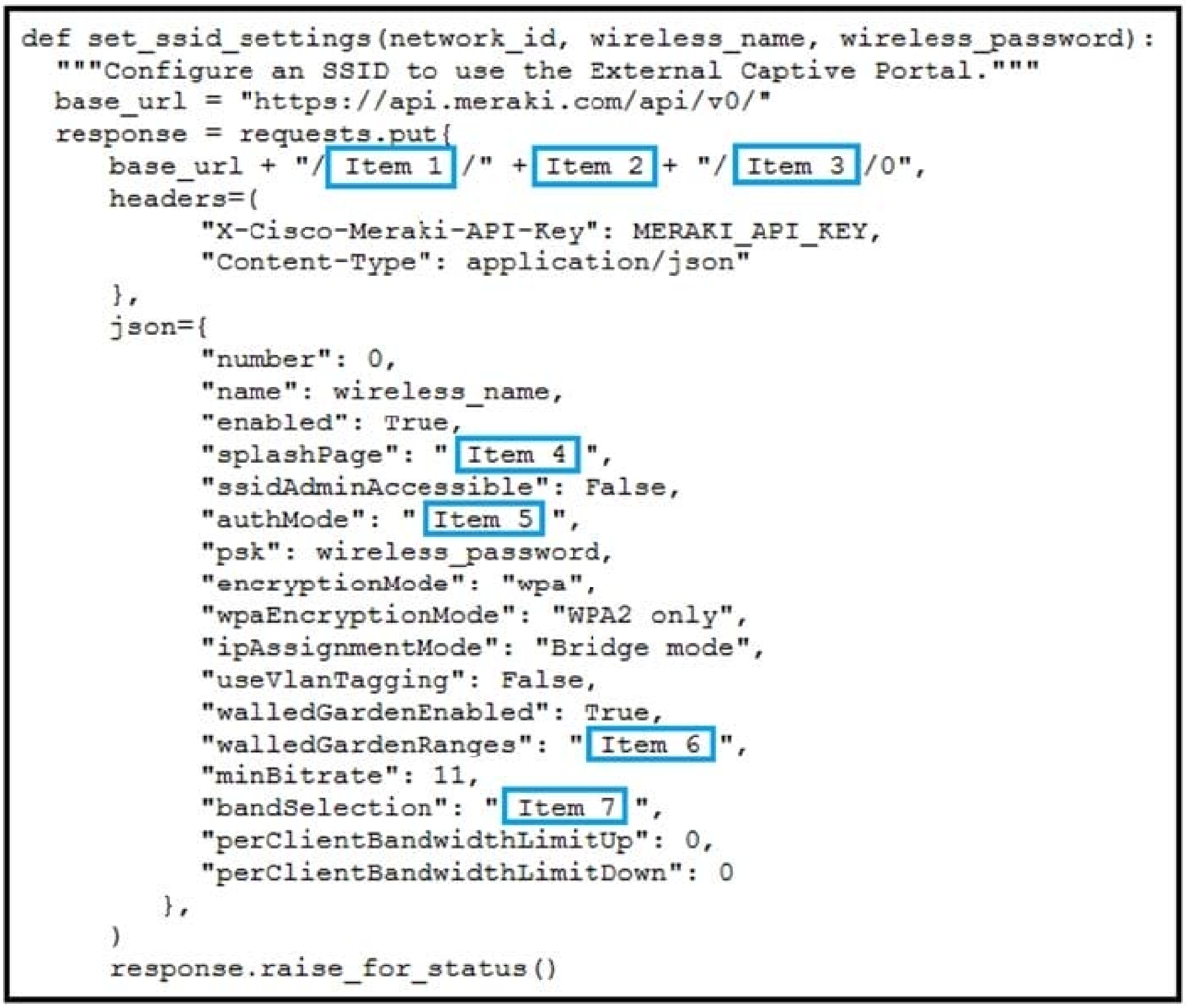
Answer Area

	Kibana: Kibana
	Python Application: Python Application
	IOS-XE Device: IOS-XE Device
	Elasticsearch: Elasticsearch

QUESTION 3

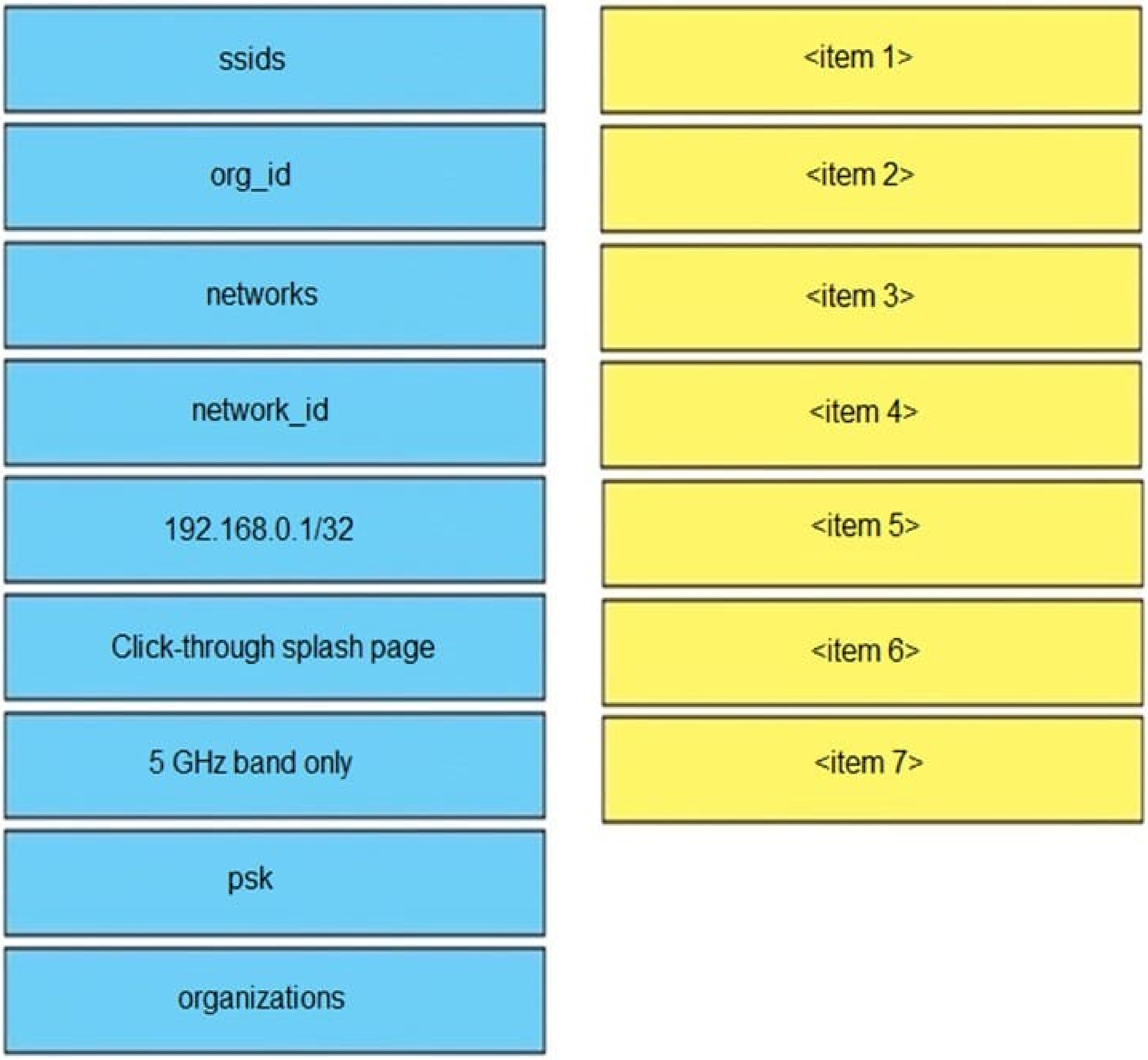
DRAG DROP

Refer to the exhibit above and click on the Meraki Resources tab in the top left corner to view Meraki documentation to help with this question. 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 to enable an SSID. Not all code parts are used.



```
def set_ssid_settings(network_id, wireless_name, wireless_password):
  """Configure an SSID to use the External Captive Portal."""
  base_url = "https://api.meraki.com/api/v0/"
  response = requests.put(
      base_url + "/ Item 1 /" + Item 2 + "/ Item 3 /0",
      headers={
            "X-Cisco-Meraki-API-Key": MERAKI_API_KEY,
            "Content-Type": application/json"
      },
      json={
            "number": 0,
            "name": wireless_name,
            "enabled": True,
            "splashPage": " Item 4 ",
            "ssidAdminAccessible": False,
            "authMode": " Item 5 ",
            "psk": wireless_password,
            "encryptionMode": "wpa",
            "wpaEncryptionMode": "WPA2 only",
            "ipAssignmentMode": "Bridge mode",
            "useVlanTagging": False,
            "walledGardenEnabled": True,
            "walledGardenRanges": " Item 6 ",
            "minBitrate": 11,
            "bandSelection": " Item 7 ",
            "perClientBandwidthLimitUp": 0,
            "perClientBandwidthLimitDown": 0
         },
      )
      response.raise_for_status()
```

Select and Place:



ssids

org_id

networks

network_id

192.168.0.1/32

Click-through splash page

5 GHz band only

psk

organizations

<item 1>

<item 2>

<item 3>

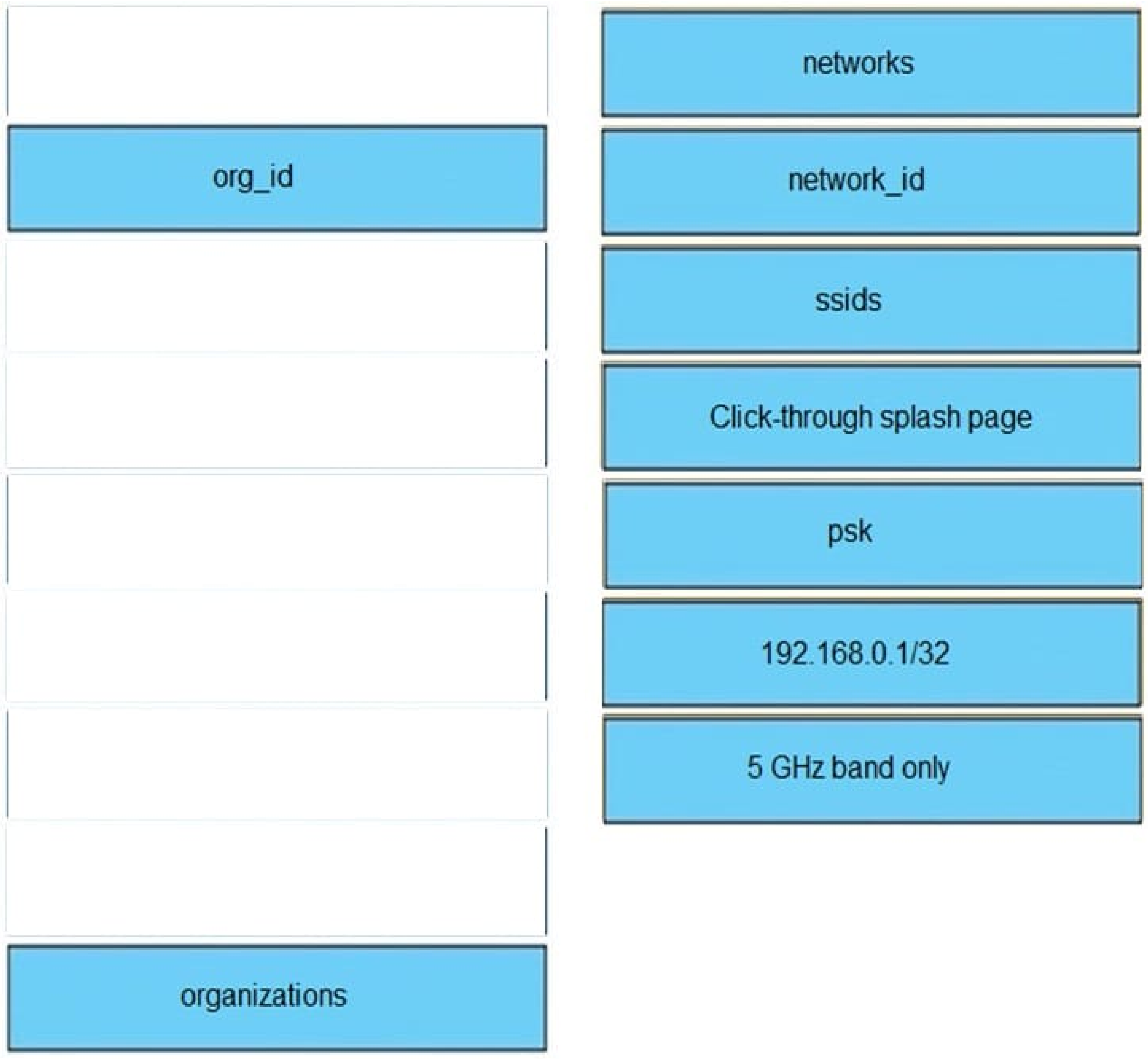
<item 4>

<item 5>

<item 6>

<item 7>

Correct Answer:



org_id

organizations

networks

network_id

ssids

Click-through splash page

psk

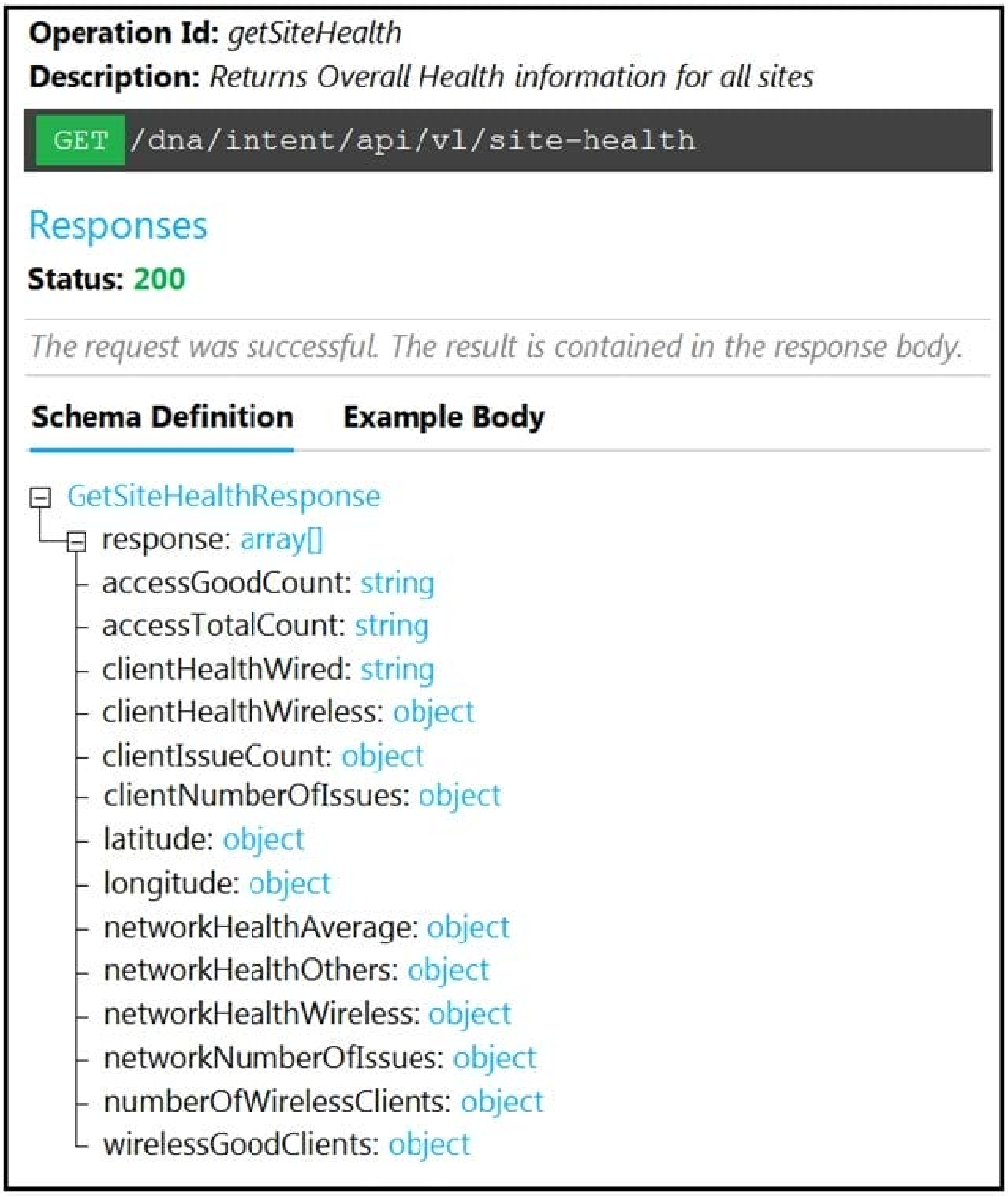
192.168.0.1/32

5 GHz band only

QUESTION 4

DRAG DROP

Refer to the exhibit. A developer is creating a Python script by using Cisco DNA Center APIs. Drag and drop the code from the bottom onto the box where the code is missing in the Python script to retrieve and display wireless health information for each site. Not all options are used.



Operation Id: *getSiteHealth*
Description: *Returns Overall Health information for all sites*

```
GET /dna/intent/api/v1/site-health
```

Responses

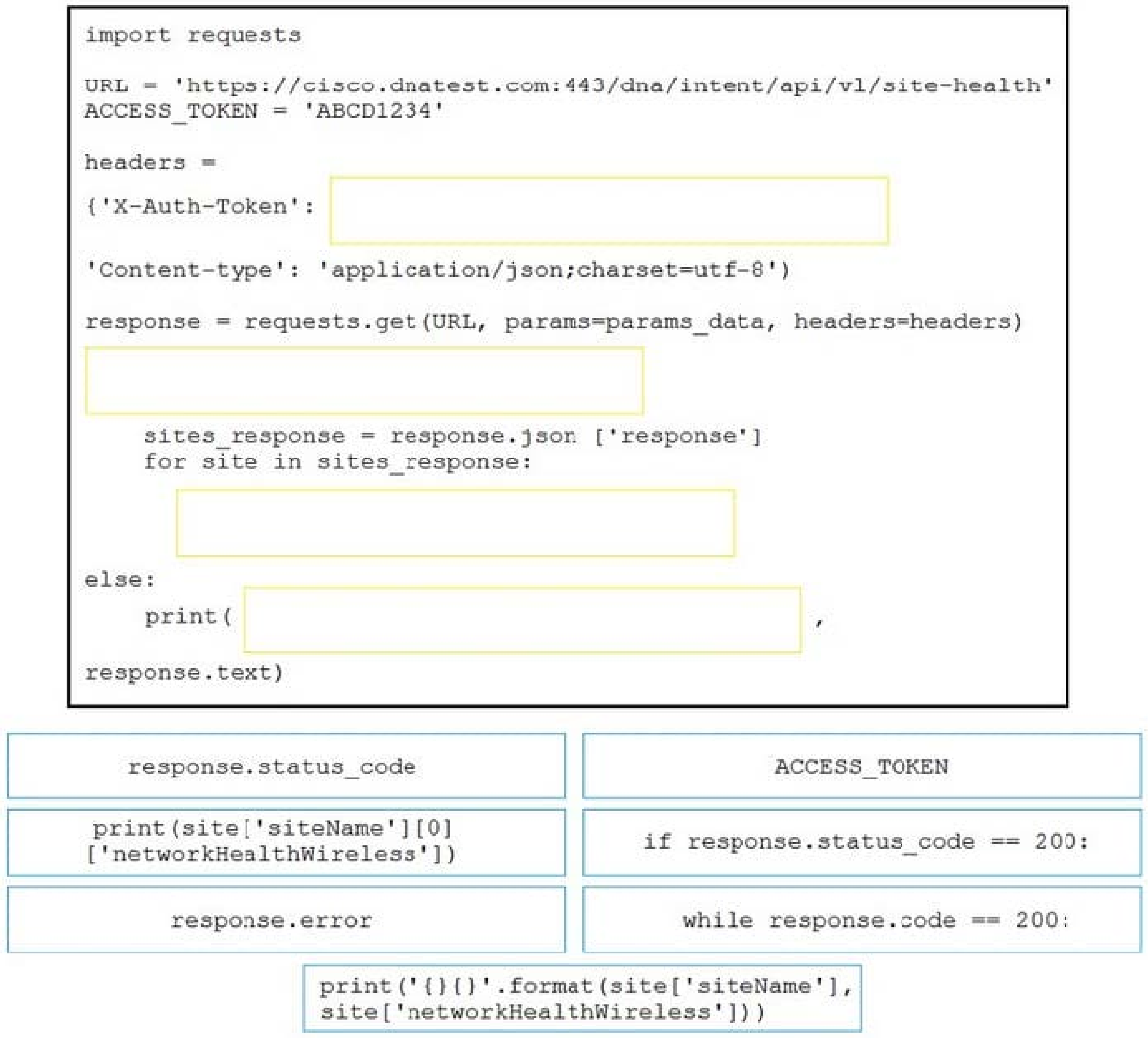
Status: 200

The request was successful. The result is contained in the response body.

Schema Definition **Example Body**

```
GetSiteHealthResponse
└ response: array[]
   ├ accessGoodCount: string
   ├ accessTotalCount: string
   ├ clientHealthWired: string
   ├ clientHealthWireless: object
   ├ clientIssueCount: object
   ├ clientNumberOfIssues: object
   ├ latitude: object
   ├ longitude: object
   ├ networkHealthAverage: object
   ├ networkHealthOthers: object
   ├ networkHealthWireless: object
   ├ networkNumberOfIssues: object
   ├ numberOfWirelessClients: object
   └ wirelessGoodClients: object
```

Select and Place:



```
import requests

URL = 'https://cisco.dnatest.com:443/dna/intent/api/v1/site-health'
ACCESS_TOKEN = 'ABCD1234'

headers =
{'X-Auth-Token': [                    ]

'Content-type': 'application/json;charset=utf-8'}

response = requests.get(URL, params=params_data, headers=headers)
[                    ]

    sites_response = response.json ['response']
    for site in sites_response:
        [                    ]

else:
    print( [                    ] ,

response.text)
```

```
response.status_code
```

```
ACCESS_TOKEN
```

```
print(site['siteName'][0]
['networkHealthWireless'])
```

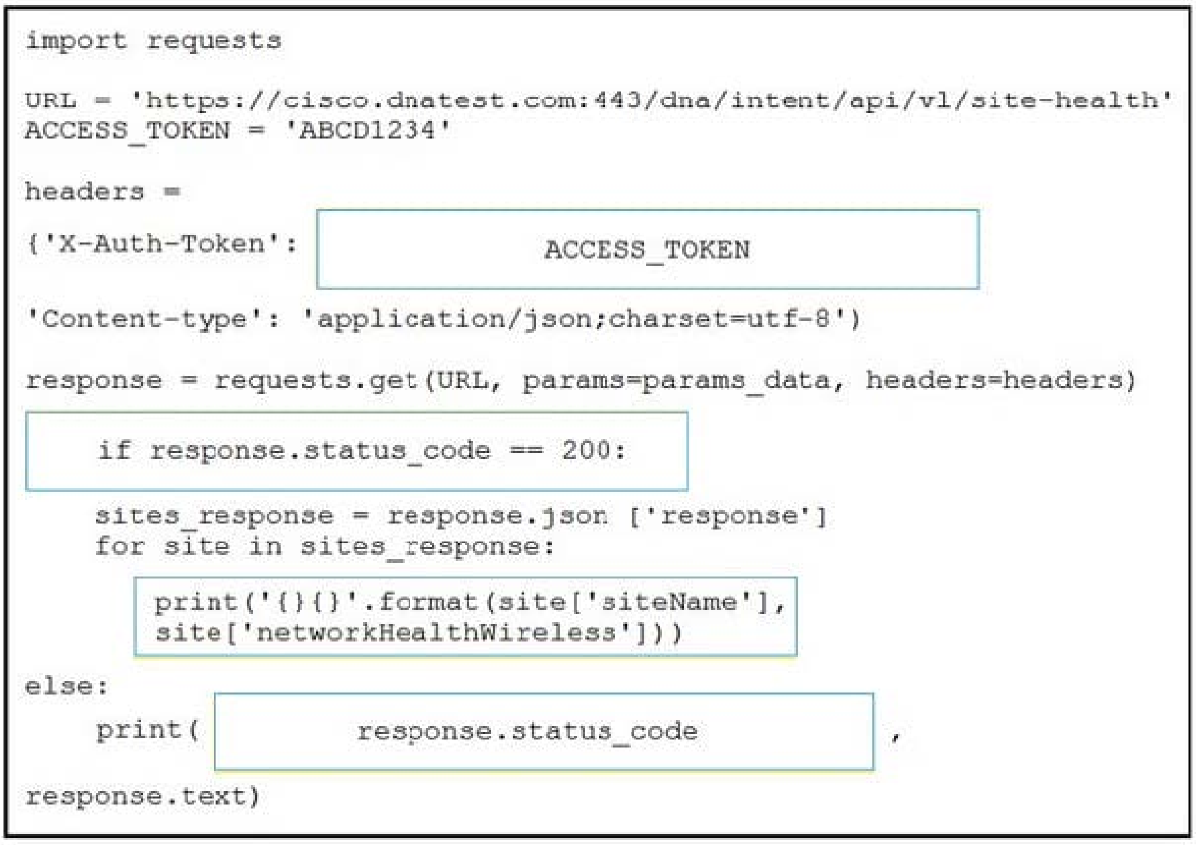
```
if response.status_code == 200:
```

```
response.error
```

```
while response.code == 200:
```

```
print('{}{}'.format(site['siteName'],
site['networkHealthWireless']))
```

Correct Answer:



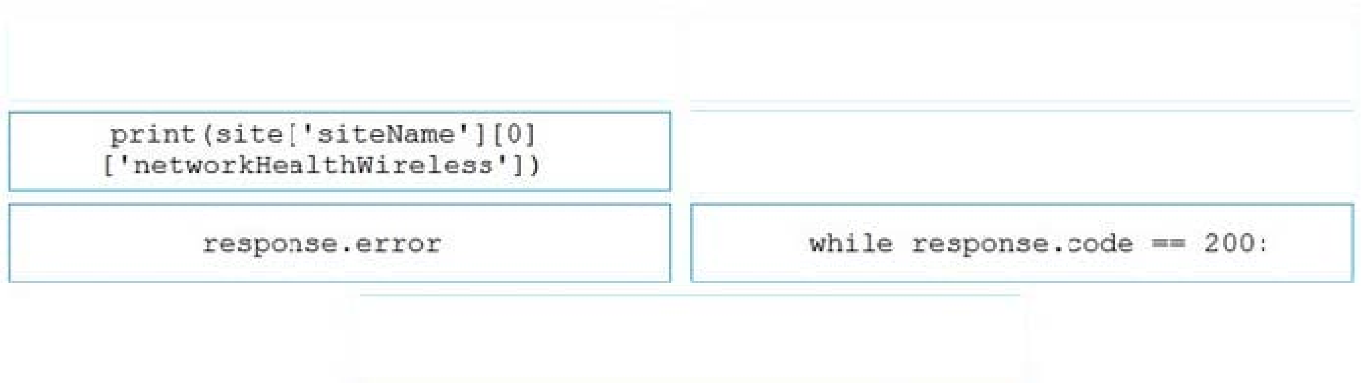
```
import requests

URL = 'https://cisco.dnatest.com:443/dna/intent/api/v1/site-health'
ACCESS_TOKEN = 'ABCD1234'

headers =
{'X-Auth-Token':        ACCESS_TOKEN

'Content-type': 'application/json;charset=utf-8'}

response = requests.get(URL, params=params_data, headers=headers)
    if response.status_code == 200:
    sites_response = response.json ['response']
    for site in sites_response:
        print('{}{}'.format(site['siteName'],
        site['networkHealthWireless']))
else:
    print(         response.status_code         ,
response.text)
```

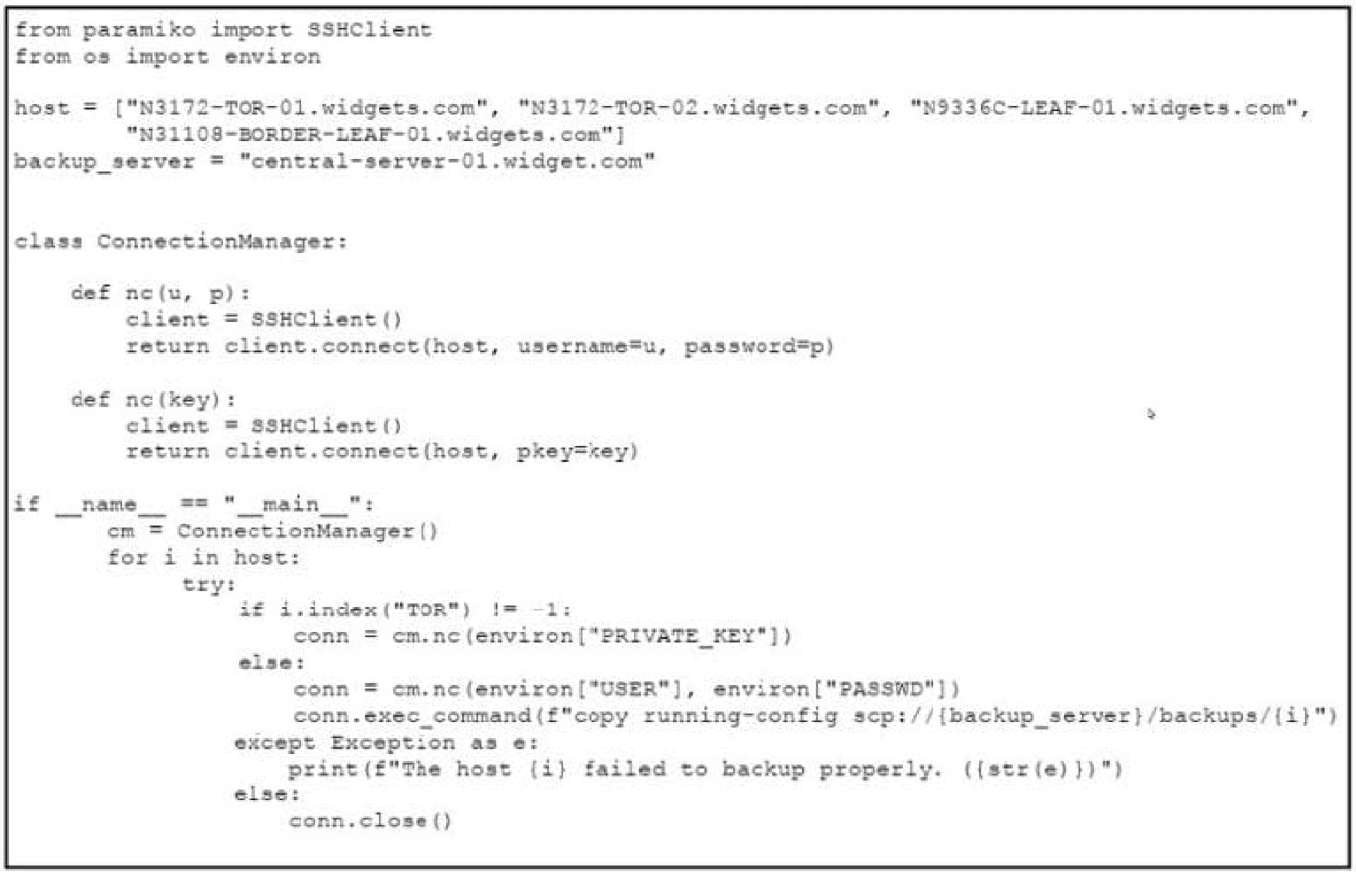


```
print(site['siteName'][0]
['networkHealthWireless'])

response.error                while response.code == 200:
```

QUESTION 5

Refer to the exhibit.



```
from paramiko import SSHClient
from os import environ

host = ["N3172-TOR-01.widgets.com", "N3172-TOR-02.widgets.com", "N9336C-LEAF-01.widgets.com",
        "N31108-BORDER-LEAF-01.widgets.com"]
backup_server = "central-server-01.widget.com"


class ConnectionManager:

    def nc(u, p):
        client = SSHClient()
        return client.connect(host, username=u, password=p)

    def nc(key):
        client = SSHClient()
        return client.connect(host, pkey=key)

if __name__ == "__main__":
        cm = ConnectionManager()
        for i in host:
            try:
                if i.index("TOR") != -1:
                    conn = cm.nc(environ["PRIVATE_KEY"])
                else:
                    conn = cm.nc(environ["USER"], environ["PASSWD"])
                    conn.exec_command(f"copy running-config scp://{backup_server}/backups/{i}")
                except Exception as e:
                    print(f"The host {i} failed to backup properly. ({str(e)})")
                else:
                    conn.close()
```

A developer must review an intern\\'s code for a script they wrote to automate backups to the storage server. The script must connect to the network device and copy the running- config to the server.

When considering maintainability, which two changes must be made to the code? (Choose two.)

A. Rename the class to "ArchiveManager".

B. The code is incorrect because the class does not have an__init__() method.

C. The command sent to the network device is incorrect.

D. Refactor the code placing the "for" loop steps inside a single nc method.

E. The intern must use IP addresses because DNS is unreliable.

Correct Answer: CD

[Latest 350-901 Dumps](https://www.geekcert.com/350-901.html) [350-901 VCE Dumps](https://www.geekcert.com/350-901.html) [350-901 Braindumps](https://www.geekcert.com/350-901.html)