



300-835^{Q&As}

Automating Cisco Collaboration Solutions (CLAUTO)

Pass Cisco 300-835 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

<https://www.geekcert.com/300-835.html>

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Cisco
Official Exam Center

- ⚙️ **Instant Download** After Purchase
- ⚙️ **100% Money Back** Guarantee
- ⚙️ **365 Days** Free Update
- ⚙️ **800,000+** Satisfied Customers





QUESTION 1

What is a benefit of using Python virtual environments?

- A. It isolates dependencies of every project from the system and each other.
- B. It allows Python to differentiate between package versions.
- C. It frees the developer from installing the project dependencies.
- D. It puts dependent packages in a common site-packages directory.

Correct Answer: A

QUESTION 2

DRAG DROP

This Python script automates the creation of a Webex Teams space and adds participants to the space. Drag and drop code on the snippet to complete the script. Not all options are used.

Select and Place:

Answer Area

```
import requests

head = { 'Content-Type': '[ ] ',
        'Authorization': 'Bearer NWU4NjQ0YWUtNjYItMWMY_-417f-ae0e10f' }

res = requests.post(url = 'https://api.ciscospark.com/v1/rooms/',
                    headers = head, json = { '[ ] ': 'Populate members' })

spaceId = res.json()[ 'id' ]

members = [ 'ToEnglish@webex.bot', 'ToSpanish@webex.bot' ]
for members in members:
    requests.post(url='https://api.ciscospark.com/v1/[ ] ',
                  headers = head,
                  json = { 'roomId': spaceId, '[ ] ': member })
```

/memberships

person

title

/rooms

/spaces

personEmail

application/xml

application/json

name

/members



Correct Answer:

Answer Area

```
import requests

head = { 'Content-Type': ' application/json ',
        'Authorization': 'Bearer NWU4NjQ0YWUtNjYItMWMY_-417f-ae0e10f' }

res = requests.post(url = 'https://api.ciscospark.com/v1/rooms/',
                    headers = head, json = { ' name ': 'Populate members' })

spaceId = res.json()[ 'id' ]

members = [ 'ToEnglish@webex.bot', 'ToSpanish@webex.bot' ]
for members in members:
    requests.post(url='https://api.ciscospark.com/v1/ personEmail /',
                  headers = head,
                  json = { 'roomId': spaceId, ' /memberships ': member})
```

person

title

/rooms

/spaces

application/xml

/members

QUESTION 3

Which two functions are provided by the Java-based computer telephony applications API? (Choose two.)

- A. Provide call blocking and screening for applications.
- B. Provide analytics about Cisco Unified Communications Manager endpoints and users.
- C. Control and observe Cisco Unified Communications Manager phones.
- D. Route calls by using computer telephony integration ports and route points (virtual devices).
- E. Allow provisioning of Cisco Unified Communications Manager endpoints and users.

Correct Answer: DE

QUESTION 4



DRAG DROP

Drag and drop the bootstrap functions from the left onto the correct actions on the right.

Select and Place:

Answer Area

hears()	Adds a question to the thread and evaluate the answer.
ask()	Sends a message with no pause for response.
say()	Replies to an incoming message.
reply()	Listens for keyword, phrases, or patterns in messages from users.

Correct Answer:

Answer Area

ask()
say()
reply()
hears()

QUESTION 5

DRAG DROP

A small Python script is constructed that creates a Webex Meeting for John Doe scheduled for the current time. Drag and drop the code snippets at the bottom onto the areas of the source code exhibit to create a program. Not all options are used.



Select and Place:

Answer Area

```
import requests, datetime
timestamp = datetime.datetime.now()
startDate = timestamp.strftime('%m/%d/%Y %H:%M:%S')
xml = f'''<message><header><securityContext>

</securityContext></header>
<body><bodyContent xsi:type="java:com.webex.service.binding.meeting.CreateMeeting">

</bodyContent></body></message>'''
print(requests.post('https://api.webex.com/WBXService/XMLService', xml))

<siteName>apidemoeu</siteName>
<webExID>johndoe@example.com</webExID>
<token>AAABbSKM1UYAOgAKEkdfUOhBMjU2XOFMR09SSVRI TV8=</token>

<schedule><startDate>(startDate)</startDate></schedule>
</bodyContent></body></message>'''
print(requests.post('https://api.webex.com/WBXService/XMLService', xml))

<metaData>
<confName>Sample Meeting</confName><meetingType>105</meetingType>
</metaData>

<siteName>apidemocu</siteName>
<webExID>johndoe@example.com</webExID>
<sessionTicket>AAABbSKM1UYAOgAKEkdfUOhBMjU2XOFMR09SSVRI TV8=</sessionTicket>

<metaData>
<confName>Sample Meeting</confName><meetingType>Training Center</meetingType>
</metaData>
```

Correct Answer:

**Answer Area**

```
import requests, datetime
timestamp = datetime.datetime.now()
startDate = timestamp.strftime('%m/%d/%Y %H:%M:%S')
xml = f'''<message><header><securityContext>

<siteName>apidemoeu</siteName>
<webExID>johndoe@example.com</webExID>
<sessionTicket>AAABbSKM1UYAOgAKEkdfUOhBMjU2XOFMR09SSVRITV8=</sessionTicket>

</securityContext></header>
<body><bodyContent xsi:type="java:com.webex.service.binding.meeting.CreateMeeting">

<metaData>
<confName>Sample Meeting</confName><meetingType>105</meetingType>
</metaData>

<schedule><startDate>{startDate}</startDate></schedule>
</bodyContent></body></message>'''
print(requests.post('https://api.webex.com/WBXService/XMLService', xml))
```

```
<siteName>apidemoeu</siteName>
<webExID>johndoe@example.com</webExID>
<token>AAABbSKM1UYAOgAKEkdfUOhBMjU2XOFMR09SSVRITV8=</token>
```

```
<metaData>
<confName>Sample Meeting</confName><meetingType>Training Center</meetingType>
</metaData>
```

[300-835 PDF Dumps](#)[300-835 Study Guide](#)[300-835 Exam Questions](#)