Reading and Writing CSV Files

In this exercise, you will:

- Read device information data from a CSV formatted file
- Create device objects with information for each device in the file
- Attempt to connect to the devices
- Print the results
- Output a CSV formatted file with information about each device from the list.

Read in the CSV device information data from the csv-devices file located in the PRNE/section15/csv folder. Create device objects based on the device type of each device in the file.

Answer

NOTE: The solution steps show code for each file denoted in parentheses. The file names are not part of the solution. The complete solution is posted in the PRNE/section15 folder.

```
(in main.py)

devices_list = read_devices_info('csv-devices')

(in util.py)

#======================================================================
def read_devices_info(devices_file):

    devices_list = []

    file = open(devices_file,'r')   # Open the CSV file
    csv_devices = csv.reader(file)  # Create the CSV reader for file

    # Use list comprehension to put CSV data into list of lists
    device_info_list = [dev_info for dev_info in csv_devices]

    for device_info in device_info_list:

        # Create a device object with this data
        if device_info[1] == 'ios':

            device = NetworkDeviceIOS(device_info[0],device_info[2],
                                      device_info[3],device_info[4])

        elif device_info[1] == 'ios-xr':

            device = NetworkDeviceXR(device_info[0],device_info[2],
                                     device_info[3],device_info[4])

        else:
            device = NetworkDevice(device_info[0],device_info[2],
                                   device_info[3],device_info[4])

        devices_list.append(device) # Append this device object to list

    return devices_list
```

Step 2

Attempt to connect to, and retrieve device information from each device.

```
(in main.py)

device.connect()          # connect to this specific device
device.get_interfaces()   # get interface info for this specific device

(in devclass.py)

#---- Class to hold information about an IOS network device --------
class NetworkDeviceIOS(NetworkDevice):

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        NetworkDevice.__init__(self, name, ip, user, pw)

    def connect(self):
        print '--- connecting IOS: telnet '+self.ip_address

        self.session = pexpect.spawn('telnet '+self.ip_address, timeout=20)
        result = self.session.expect(['Username:', pexpect.TIMEOUT])

        self.session.sendline(self.username)
        result = self.session.expect('Password:')

        # Successfully got password prompt, logging in with password
        self.session.sendline(self.password)
        self.session.expect('>')

    def get_interfaces(self):

        self.session.sendline('show interfaces summary')
        result = self.session.expect('>')

        self.interfaces = self.session.before
```

Step 3

Print the results of every device connection attempt.

Answer

```
(in main.py)

print_device_info(device)   # print device details for this device

(in util.py)

#=====================================================================
def print_device_info(device):

    print '-------------------------------------------------------'
    print '    Device Name:      ',device.name
    print '    Device IP:        ',device.ip_address
    print '    Device username:  ',device.username,
    print '    Device password:  ',device.password
    print '-------------------------------------------------------'
```

Step 4

Write the device information into a CSV file for every device.

Answer

```
(in main.py)

write_devices_info('csv-devices-out', devices_list)

(in util.py)

#======================================================================
def write_devices_info(devices_file, devices_list):

    print '---- Printing CSV output ------------------------------'

    # Create the list of lists with devices and device info
    devices_out_list = []  # create list for CSV output

    for device in devices_list:
        dev_info = [device.name,device.ip_address,device.interfaces != ""]
        devices_out_list.append(dev_info)

    pprint(devices_out_list)

    # Use CSV library to output our list of lists to a CSV file
    with open(devices_file, 'w') as file:
        csv_out = csv.writer(file)
        csv_out.writerows(devices_out_list)
```

Step 5

Run your application and verify the CSV file is created.

```
File: util.py

import csv
from pprint import pprint

from devclass import NetworkDevice
from devclass import NetworkDeviceIOS
from devclass import NetworkDeviceXR

#==========================================================================
def read_devices_info(devices_file):

    devices_list = []

    file = open(devices_file,'r')   # Open the CSV file
    csv_devices = csv.reader(file)  # Create the CSV reader for file

    # Iterate through all devices in our CSV file
    for device_info in csv_devices:

        # Create a device object with this data
        if device_info[1] == 'ios':

            device = NetworkDeviceIOS(device_info[0],device_info[2],
                                      device_info[3],device_info[4])

        elif device_info[1] == 'ios-xr':

            device = NetworkDeviceXR(device_info[0],device_info[2],
                                     device_info[3],device_info[4])

        else:
            device = NetworkDevice(device_info[0],device_info[2],
                                   device_info[3],device_info[4])
        devices_list.append(device) # Append this device object to list

    return devices_list



#======================================================================
def print_device_info(device):

    print '--------------------------------------------------------'
    print '    Device Name:      ',device.name
    print '    Device IP:        ',device.ip_address
    print '    Device username:  ',device.username,
    print '    Device password:  ',device.password
    print '--------------------------------------------------------'

#======================================================================
def write_devices_info(devices_file, devices_list):

    print '---- Printing CSV output ------------------------------'

    # Create the list of lists with devices and device info
    devices_out_list = []  # create list for CSV output

    for device in devices_list:
        dev_info = [device.name,device.ip_address,device.interfaces != ""]
        devices_out_list.append(dev_info)

    pprint(devices_out_list)

    # Use CSV library to output our list of lists to a CSV file
    with open(devices_file, 'w') as file:
        csv_out = csv.writer(file)
        csv_out.writerows(devices_out_list)
```

File: devclass.py

```
import pexpect

#---- Class to hold information about a generic network device --------
class NetworkDevice():

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        self.name = name
        self.ip_address = ip
        self.username = user
        self.password = pw

    def connect(self):
        self.session = None

    def get_interfaces(self):
        self.interfaces = '--- Base Device, does not know how to get interfa

#---- Class to hold information about an IOS network device --------
class NetworkDeviceIOS(NetworkDevice):

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        NetworkDevice.__init__(self, name, ip, user, pw)

    def connect(self):
        print '--- connecting IOS: telnet '+self.ip_address

        self.session = pexpect.spawn('telnet '+self.ip_address, timeout=20)
        result = self.session.expect(['Username:', pexpect.TIMEOUT])

        self.session.sendline(self.username)
        result = self.session.expect('Password:')
        # Successfully got password prompt, logging in with password
        self.session.sendline(self.password)
        self.session.expect('>')

    def get_interfaces(self):

        self.session.sendline('show interfaces summary')
        result = self.session.expect('>')

        self.interfaces = self.session.before


#---- Class to hold information about an IOS-XR network device --------
class NetworkDeviceXR(NetworkDevice):

    #---- Initialize -----------------------------------------------
    def __init__(self, name, ip, user='cisco', pw='cisco'):
        NetworkDevice.__init__(self, name, ip, user, pw)

    #---- Connect to device -----------------------------------------
    def connect(self):

        print '--- connecting XR: ssh '+self.username+'@'+self.ip_address

        self.session = pexpect.spawn('ssh '+self.username+
                                     '@'+self.ip_address, timeout=20)
        result = self.session.expect(['password:', pexpect.TIMEOUT])

        # Check for failure
        if result != 0:
            print '--- Timout or unexpected reply from device'
            return 0
        # Successfully got password prompt, logging in with password
        print '--- password:',self.password
        self.session.sendline(self.password)
        self.session.expect('#')

    #---- Get interfaces from device ----------------------------------
    def get_interfaces(self):

        self.session.sendline('show interface brief')
        result = self.session.expect('#')

        self.interfaces = self.session.before
```

File: main.py

```
from util import read_devices_info
from util import print_device_info
from util import write_devices_info

#====================================================================
# Main program: connect to device, show interface, display

devices_list = read_devices_info('csv-devices')  # read CSV info for all dev

for device in devices_list:

    print '==== Device ==========================================================

    device.connect()          # connect to this specific device
    device.get_interfaces()   # get interface info for this specific device

    print_device_info(device)   # print device details for this device

write_devices_info('csv-devices-out', devices_list)   # write CSV entry for 
```

Reading and Writing JSON Files

In this exercise, you will:

- Read device information data from a JSON formatted file
- Create device objects with information for each device in the file
- Attempt to connect to the devices
- Output a JSON formatted file with information about each device from the list.

You will then read in the new JSON file and connect to the devices again to verify that the writing of data was successful.

Step 6

Read device information from the file `json-devices` that is located in the PRNE/section15/json folder. Create device objects for each device in the file.

```
(in main.py)

devices_list = read_devices_info('json-devices')  # read JSON info for all d

(in util.py)

import json
from pprint import pprint

from devclass import NetworkDevice
from devclass import NetworkDeviceIOS
from devclass import NetworkDeviceXR

#======================================================================
def read_devices_info(devices_file):

    devices_list = []

    # Open the device file with JSON data and read into string
    json_file = open(devices_file,'r')   # open the JSON file
    json_device_data = json_file.read()  # read in the JSON data from file

    # Convert JSON string into Python data structure
    devices_info_list = json.loads(json_device_data)

    for device_info in devices_info_list:

        # Create a device object with this data
        if device_info['os'] == 'ios':

            device = NetworkDeviceIOS(device_info['name'],device_info['ip'],
                                      device_info['user'],device_info['passw
        elif device_info['os'] == 'ios-xr':

            device = NetworkDeviceXR(device_info['name'],device_info['ip'],
                                     device_info['user'],device_info['passwo

        else:
            device = NetworkDevice(device_info['name'],device_info['ip'],
                                   device_info['user'],device_info['password

        devices_list.append(device) # Append this device object to list

    return devices_list
```

Step 7

For every device, connect using Pexpect and get interface data.

```
(main.py)

for device_info in devices_info_list:

     # Create a device object with this data
     if device_info['os'] == 'ios':

         device = NetworkDeviceIOS(device_info['name'],device_info['ip'],
                                   device_info['user'],device_info['password

     elif device_info['os'] == 'ios-xr':

         device = NetworkDeviceXR(device_info['name'],device_info['ip'],
                                  device_info['user'],device_info['password'

     else:
         device = NetworkDevice(device_info['name'],device_info['ip'],
                                device_info['user'],device_info['password'])

        devices_list.append(device) # Append this device object to list

    return devices_list
```

```
#---- Class to hold information about an IOS-XR network device --------
class NetworkDeviceXR(NetworkDevice):

    #---- Initialize -----------------------------------------------------
    def __init__(self, name, ip, user='cisco', pw='cisco'):
        NetworkDevice.__init__(self, name, ip, user, pw)
        self.os_type = 'ios-xr'

    #---- Connect to device -------------------------------------------
    def connect(self):

        print '--- connecting XR: ssh '+self.username+'@'+self.ip_address

        self.session = pexpect.spawn('ssh '+self.username+
                                     '@'+self.ip_address, timeout=20)
        result = self.session.expect(['password:', pexpect.TIMEOUT])

        # Check for failure
        if result != 0:
            print '--- Timout or unexpected reply from device'
            return 0

        # Successfully got password prompt, logging in with password
        print '--- password:',self.password
        self.session.sendline(self.password)
        self.session.expect('#')

    #---- Get interfaces from device ----------------------------------
    def get_interfaces(self):

        self.session.sendline('show interface brief')
        result = self.session.expect('#')

        self.interfaces = self.session.before
```

Step 8

Print the device information for every device.

Answer

```
(main.py)

for device in devices_list:

    print '==== Device =======================================================

    device.connect()          # connect to this specific device
    device.get_interfaces()   # get interface info for this specific device

    print_device_info(device)   # print device details for this device

(util.py)

#=====================================================================
def print_device_info(device):

    print '--------------------------------------------------------'
    print '    Device Name:      ',device.name
    print '    Device IP:        ',device.ip_address
    print '    Device username:  ',device.username,
    print '    Device password:  ',device.password
    print '--------------------------------------------------------'
```

Step 9

Write device information to a JSON formatted file, which will hold the information about each device from your device objects.

Answer

```
(main.py)

write_devices_info('json-devices-out', devices_list)   # write JSON entry fo


(util.py)

#======================================================================
def write_devices_info(devices_file, devices_list):

    print '---- Printing JSON output ------------------------------'

    # Create the list of lists with devices and device info
    devices_out_list = []  # create list for JSON output

    for device in devices_list:
        dev_info = {'name':device.name,'ip':device.ip_address,'os':device.os_
                    'user':device.username,'password':device.password}
        devices_out_list.append(dev_info)

    pprint(devices_out_list)

    # Convert the python device data into JSON for output to file
    json_device_data = json.dumps(devices_out_list)

    # Output the JSON string to a file
    with open(devices_file, 'w') as json_file:
        json_file.write(json_device_data)
```

Step 10

Read in the new file, creating device objects for each device.

Answer

```
# Do it again, reading from your output file, to prove we did it correctly

print '---------------------------------------------------------------------'
print '---------- Reading from our output file, doing it again -----------'

devices_list = read_devices_info('json-devices-out')  # read JSON info for a
```

Step 11

For every device, connect using Pexpect and get interface data. Print the device information.

Answer

```
for device in devices_list:

    print '==== Device ================================================'

    device.connect()          # connect to this specific device
    device.get_interfaces()   # get interface info for this specific device

    print_device_info(device)   # print device details for this device
```

Step 12

Run your application and verify that the JSON output file was created.

```
File: util.py

import json
from pprint import pprint

from devclass import NetworkDevice
from devclass import NetworkDeviceIOS
from devclass import NetworkDeviceXR

#========================================================================
def read_devices_info(devices_file):

    devices_list = []

    # Open the device file with JSON data and read into string
    json_file = open(devices_file,'r')   # open the JSON file
    json_device_data = json_file.read()  # read in the JSON data from file

    # Convert JSAON string into Python data structure
    devices_info_list = json.loads(json_device_data)

    for device_info in devices_info_list:

        # Create a device object with this data
        if device_info['os'] == 'ios':

            device = NetworkDeviceIOS(device_info['name'],device_info['ip'],
                                      device_info['user'],device_info['passw

        elif device_info['os'] == 'ios-xr':

            device = NetworkDeviceXR(device_info['name'],device_info['ip'],
                                     device_info['user'],device_info['passwo

        else:
            device = NetworkDevice(device_info['name'],device_info['ip'],
                                   device_info['user'],device_info['password

        devices_list.append(device) # Append this device object to list

    return devices_list
```

```
#=====================================================================
def print_device_info(device):

    print '-------------------------------------------------------'
    print '    Device Name:      ',device.name
    print '    Device IP:        ',device.ip_address
    print '    Device username:  ',device.username,
    print '    Device password:  ',device.password
    print '-------------------------------------------------------'

#=====================================================================
def write_devices_info(devices_file, devices_list):

    print '---- Printing JSON output ------------------------------'

    # Create the list of lists with devices and device info
    devices_out_list = []  # create list for JSON output

    for device in devices_list:
        dev_info = {'name':device.name,'ip':device.ip_address,'os':device.os
                    'user':device.username,'password':device.password}
        devices_out_list.append(dev_info)

    pprint(devices_out_list)

    # Convert the python device data into JSON for output to file
    json_device_data = json.dumps(devices_out_list)

    # Output the JSON string to a file
    with open(devices_file, 'w') as json_file:
        json_file.write(json_device_data)
```

File: devclass.py

```
import pexpect

#---- Class to hold information about a generic network device --------
class NetworkDevice():

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        self.name = name
        self.ip_address = ip
        self.username = user
        self.password = pw
        self.os_type = None

    def connect(self):
        self.session = None

    def get_interfaces(self):
        self.interfaces = '--- Base Device, does not know how to get interfa
```

```
#---- Class to hold information about an IOS network device --------
class NetworkDeviceIOS(NetworkDevice):

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        NetworkDevice.__init__(self, name, ip, user, pw)
        self.os_type = 'ios'

    def connect(self):
        print '--- connecting IOS: telnet '+self.ip_address

        self.session = pexpect.spawn('telnet '+self.ip_address, timeout=20)
        result = self.session.expect(['Username:', pexpect.TIMEOUT])

        self.session.sendline(self.username)
        result = self.session.expect('Password:')

        # Successfully got password prompt, logging in with password
        self.session.sendline(self.password)
        self.session.expect('>')

    def get_interfaces(self):

        self.session.sendline('show interfaces summary')
        result = self.session.expect('>')

        self.interfaces = self.session.before
```

```
#---- Class to hold information about an IOS-XR network device --------
class NetworkDeviceXR(NetworkDevice):

    #---- Initialize -------------------------------------------------
    def __init__(self, name, ip, user='cisco', pw='cisco'):
        NetworkDevice.__init__(self, name, ip, user, pw)
        self.os_type = 'ios-xr'

    #---- Connect to device -------------------------------------------
    def connect(self):

        print '--- connecting XR: ssh '+self.username+'@'+self.ip_address

        self.session = pexpect.spawn('ssh '+self.username+
                                     '@'+self.ip_address, timeout=20)
        result = self.session.expect(['password:', pexpect.TIMEOUT])

        # Check for failure
        if result != 0:
            print '--- Timout or unexpected reply from device'
            return 0

        # Successfully got password prompt, logging in with password
        print '--- password:',self.password
        self.session.sendline(self.password)
        self.session.expect('#')

    #---- Get interfaces from device ----------------------------------
    def get_interfaces(self):

        self.session.sendline('show interface brief')
        result = self.session.expect('#')

        self.interfaces = self.session.before
```

File: main.py

```
from util import read_devices_info
from util import print_device_info
from util import write_devices_info

#=====================================================================
# Main program: connect to device, show interface, display

devices_list = read_devices_info('json-devices')  # read JSON info for all d

for device in devices_list:

    print '==== Device =========================================================

    device.connect()          # connect to this specific device
    device.get_interfaces()   # get interface info for this specific device

    print_device_info(device)   # print device details for this device

write_devices_info('json-devices-out', devices_list)   # write JSON entry fo

# Do it again, reading from our output file, to prove we did it correctly

print '----------------------------------------------------------------------
print '---------- Reading from our output file, doing it again ------------

devices_list = read_devices_info('json-devices-out')  # read JSON info for a

for device in devices_list:

    print '==== Device =========================================================

    device.connect()          # connect to this specific device
    device.get_interfaces()   # get interface info for this specific device

    print_device_info(device)   # print device details for this device
```

Reading Binary Files

In this exercise, you will read information about network devices, and then for each device will read a binary file associated with that device representing its SSH key.

For the purposes of this lab exercise, the contents of the binary file are not relevant – the exercise is only covering the operation of reading binary data and storing it in Python objects. This exercise is simulating binary SSH key files, one for each network device.

You will then read in the new JSON file and connect to the devices again to verify that the writing of data was successful.

Step 13

Read device information from the `json-devices` file located in the PRNE/section15/binary folder. Create device objects containing the device name, IP address, username, and password.

Answer

NOTE: Only the new code is shown for each step. The entire solution is available at the end of the exercise.

```
#=======================================================================
def read_devices_info(devices_file):

   devices_list = []

   # Open the device file with JSON data and read into string
   json_file = open(devices_file,'r')   # open the JSON file
   json_device_data = json_file.read()  # read in the JSON data from file

   # Convert JSAON string into Python data structure
   devices_info_list = json.loads(json_device_data)

   for device_info in devices_info_list:

      # Create a device object with this data
      if device_info['os'] == 'ios':

         device = NetworkDeviceIOS(device_info['name'],device_info['ip'],
                                   device_info['user'],device_info['password

      elif device_info['os'] == 'ios-xr':

         device = NetworkDeviceXR(device_info['name'],device_info['ip'],
                                  device_info['user'],device_info['password'

      else:
         device = NetworkDevice(device_info['name'],device_info['ip'],
                                device_info['user'],device_info['password'])
```

Step 14

For each device, use the device IP address to read SSH key data (just a binary file) located in the PRNE/section15/binary/ sshkeys/<device-IP-address>/<keyname> folder.

Answer

```
    # Open SSH key file for this device
       key_file_path = "sshkeys/"+device_info['ip']+"/"+device_info['key']
       key_file = open(key_file_path,'rb')

       key_data = key_file.read()    # read ssh key data
```

Step 15

After reading the file, pass the binary data object to the device object. The device object will need to implement a method such as 'set_sshkey' which is passed the binary data object you have just read.

Answer

```
    device.set_sshkey(key_data)   # store ssh key data in device object
```

Step 16

Show Me

When complete, print the information for each device, including the binary key data file. Print the key data in hexadecimal.

Note: printing hexadecimal values for your binary data can be accomplished by using a for loop, iterating through every character in the 'string' (which is how the binary data is stored), and using the 'encode()' function to convert each character to hex for output.

```
    for c in device.sshkey: print c.encode('hex'),
```