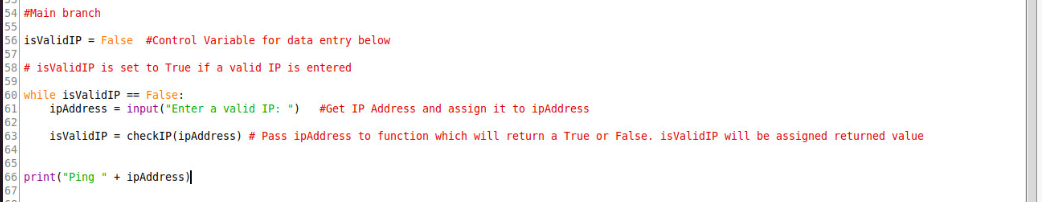
**Unit Three Lab**

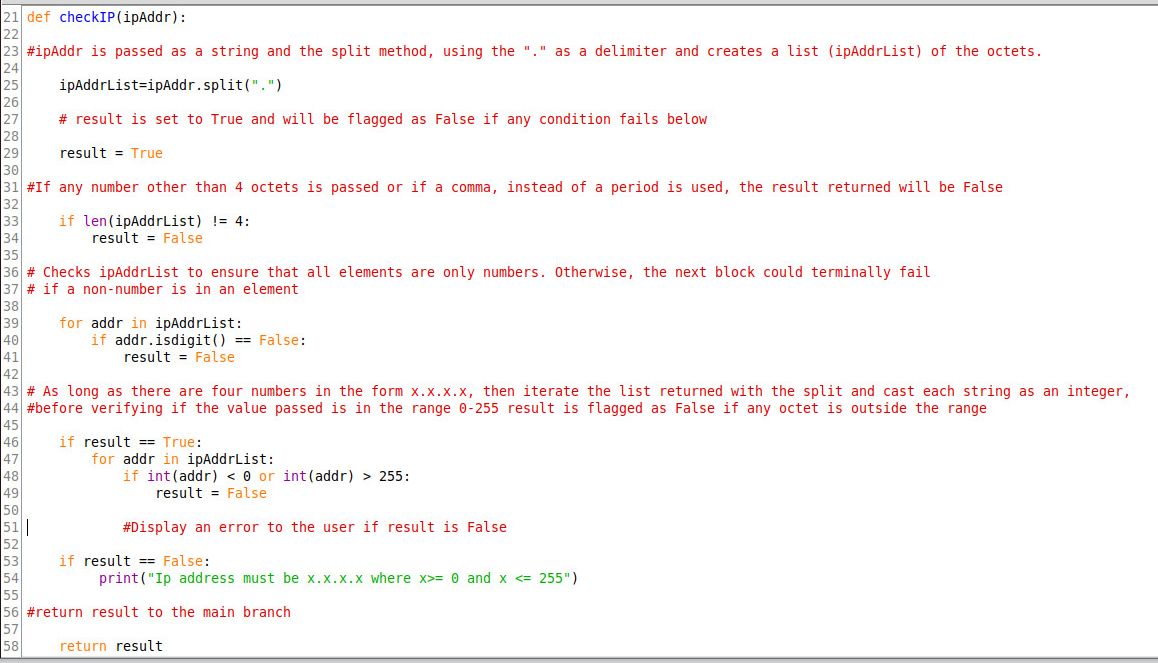
Directions: Complete each of the scripts below and submit them before the on the date specified in the assignment sheet and Blackboard. Submit **copies of your code** and **screenshots of the code running with each task**. Use your notes and scripts from your Python class and the website, <https://w3schools.com/python> or other Internet references as reference guides for this review assignment. Also, be sure to use the document, **Script Requirements as a guide** to writing good code. **Full credit will not be earned if you do not meet these script requirements.** **40 points**

**Functions**

Functions in programs and scripts are used to de-clutter your code, making it easier to read. They also provide a way to re-use code efficiently within a script or program. Functions can take a value or not, and they can return a value or not, depending on the script requirements. A function that just prints a report header, will likely not take a value or return a value. A function that does something to data or checks data, may return some information. Consider the code below:



In the code above, a Boolean control variable, isValidIP is used to control a while loop which has the main purpose of determining if a valid IP has been entered. Line 63 determines the value of isValidIP, True or False, using the checkIP() function. The checkIP() function was written by me to verify if a string is valid IP address (i.e., x.x.x.x where x >= 0 and x <=255). The function returns a value of True or False, which is passed through the assignment operator on line 63 to isValidIP. This part of the code is easy to read. The function is defined at the top of the code and looks like the following:



Line 21 is the function header and tells the function to expect a value (ipAddr). That value will be stored in the variable, ipAddr. Line 25 splits the string, ipAddr into a list. The comments explain the code thoroughly. Note that the variable, result, is returned to the calling branch on line 58. Line 58 literally means return the value of result to the calling branch of code. That is how isValidIP gets its value on line 63 in the first snippet of code. Functions can be a very powerful way to streamline your code. See the section on Functions on the W3Schools site for more examples and reference.

**Assignments**

1. **Modify your script from the lab, Python Review 1, where you ask the user for their first and last name. In step 2, you were asked to verify that only two names, consisting of alpha characters was entered. Move this logic of verifying that a valid first/last name pair was entered into a function that is called when a name is entered.**
2. **Write a script that uses a dictionary that stores information about NTP servers and their addresses, named ntpServer. You presently have the following servers that can be defined with key: value pairs, which can be created initially when the script runs:**

**Server1 = 221.100.250.75**

**Server2 = 201.0.113.22**

**Server3 = 58.23.191.6**

**Have the script print out the information above, using a For loop, that looks something like the following:**

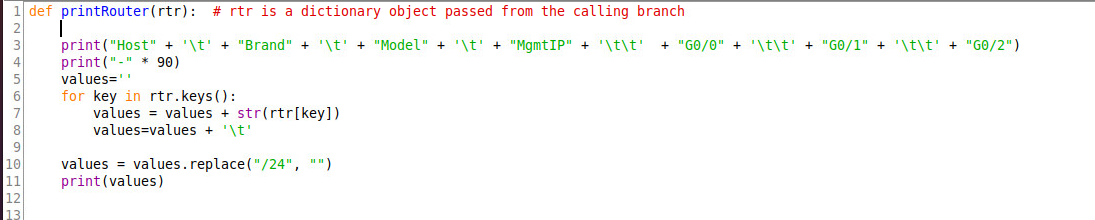
Server Name Address

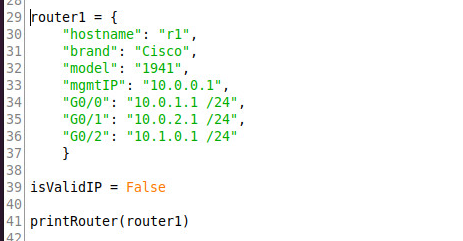
Server1 221.100.250.75

Server2 201.0.113.22

Server3 58.23.191.6

**Have report print out in a function. Pass the dictionary to the function and have the function print the above report. The Function does not need to return values. Note a similar example below using the dictionary object shown below as well that you can use as a hint:**





**Note that line 41 calls the function above. Also, note that since no value is returned, the function is called without an assignment operator and no return statement exists in the calling function.**

1. **Modify the script in number 2 by adding a function called PingPrep(ipList) that takes a list of IP addresses generated from the ntpServers dictionary object and prepends Ping to the list items as shown below. Do not change the dictionary object. Just read it and generate a list from the values.**

**List before Ping Prepended**

|  |
| --- |
| 221.100.250.75 |
| 201.0.113.22 |
| 58.23.191.6 |

**List after Ping Prepended**

|  |
| --- |
| Ping 221.100.250.75 |
| Ping 201.0.113.22 |
| Ping 58.23.191.6 |

**Finally, have the function print out the list on the screen:**

Ping 221.100.250.75

Ping 201.0.113.22

Ping 58.23.191.6

Note that while we are not actually pinging, we have manipulated our data such that it could be passed to something like a ping utility.

1. **You will be modifying an earlier script you wrote. Note the initial requirements for the script from Python Dictionaries, number 6 in blue:**
2. To help make your scripts run more efficiently, you need to create a nested dictionary object for your devices. Right now, you only have two switches and two routers, but your network will be growing soon. Call the dictionary, “devices”. Devices will have the following structure:

devices

R1

type = router

hostname = R1

mgmtIP = 10.0.0.1

R2

type = router

hostname = R2

mgmtIP = 10.0.0.2

S1

type = switch

hostname = S1

mgmtIP = 10.0.0.3

S2

type = switch

hostname = S2

mgmtIP = 10.0.0.4

**Write a python script that iterates through devices in the above nested dictionary and creates a Python list of strings for each device with “ping” prepended to it. Your output should look like the following after iterating through the list (note that you do not need to actually ping; just create strings as shown below):**

ping 10.0.0.1

ping 10.0.0.2

ping 10.0.0.3

ping 10.0.0.4

Hint: Note that the device names, needed to access the nested dictionaries, are keys in the device dictionary.

1. **Modify your script as follows:**
   1. **Move your ping prep code that prints the above ping commands to a function.**
   2. **Ask your user if they want to add a new device.**
   3. **If they say choose add, you will prompt them for all of the information. Write a function that verifies is the IP address added is valid. Make sure that no key: pair value is left blank or null. Then add the new nested dictionary object to the original devices and re-run the ping function. An example on how to add a nested dictionary object with values is shown below:**



**Lines 1 – 16 define the initial nested dictionary object. Lines 19 – 22 just assign values to variables that are passed in the assignment lines of 24-30 for the new nested dictionary, S3.**

**While validDevice == False:**

**deviceDict = GetDeviceInfo()**

**validIP = validateIP(deviceDict[“mgmtIP”])**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Graphical user interface, text, application, email

Description automatically generated**

**Def printDevices(**

**deviceDict = GetDeviceInfo(“device)**

**Graphical user interface, text, application, email

Description automatically generated**

**Calendar

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated**

**Text

Description automatically generated**

**Graphical user interface, text, application, whiteboard

Description automatically generated**