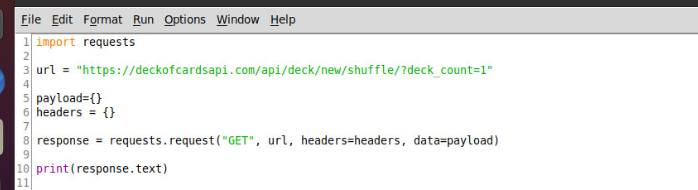
**Unit Four Lab**

Directions: Complete each of the scripts below and submit them by 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**

1. From Postman, go to your collections and load your shuffle. Click on the code link for it and generate a ‘Python Requests’ code block. Copy the block to a new Idle instance. It should look something like the screenshot below:



1. Before running the code, let’s look at what the above code does. If we understand it, we may be able to modify it for our own purposes. Line 1, imports the module that allows us to run line 8, which sends an API request via HTTPs. We will look closer at line 8 shortly. Line 3 is simply defining a data variable, url, to be used in our code on line 8.
   1. **The url variable can be called anything you want, but if you change it to another name like myURL or cardURL, what other part of the program above needs to change as well (be sure to submit your answer with your work)?**
      1. **The word URL in the response variable needs to be changed as well**

Lines 5 and 6 are also defining variables, named payload{} and headers{}, and their values are null in this script. In other scripts these may have values. **Can you see where these variables are used?**

**In the response variable, when calling url**

Line 8 is the request made using the requests.request() method, which takes 4 parameters, an HTTP verb, url, header, and data. Python uses the above code to send a request to generate a shuffled deck of cards. The response from the remote API is stored in the variable, response, on line 8 because of the “response =” assignment from the method. Line 10, simply prints the results for our benefit and displays the contents of the response variable.

1. Run the code, take screenshots of the following (you can capture one screenshot for this entire section (3) when you have completed it), and answer any questions:
   1. **A screenshot of the code results**
   2. The data is returned for us to work with. **What kind of python object does it look like?**
   3. From the interpreter, we can continue to act on the response variable because it is still in our stack. For example, we can issue another, print(response). **Try that and take a screenshot.**
   4. If our python object is what it looks like, we should be able to access the value of one of the returned attributes like the deck\_id in the form, print(response[“deck\_id”]). **Try this and take a screenshot of the result.**
   5. It didn’t work. Maybe it’s not the datatype that we think it is. **Try running the type() method on response and take a screenshot of the results.** Clearly, we haven’t seen this datatype before.
   6. **We will need to convert the data to one of our known datatypes. Maybe our object as some built in methods. Use the dir(variable) method to see a list of methods for response. Take a screenshot. Is there any method that looks interesting like it may convert from or to a format that Python can better work with?**
   7. **Type in deck = response.json()**
   8. **Re-run the type() method on deck and take a screenshot (note: deck is just new variable that we are using for our conversion)**
   9. **Type print(deck) and take a screenshot. It appears deck is now a familiar data type that we can use natively.**
   10. **Type print(deck[“deck\_id”]) and take a screenshot**
   11. You could now modify your original code to have it do something interesting.
   12. **Text

       Description automatically generated**
2. **Modify the original code in 3 that you copied from Postman, and instead of printing the entire object as in line 10, print something like: “Your deck is shuffled and ready. Your deck id = 1f45c3319b.” Note that deck id should be the actual id returned from your shuffle request. You will simply save the value of the deck\_id in a variable so that you can print it out. Note that in 3J above, if you can print the value, you can assign it to a variable.**

**Text

Description automatically generated**

1. Copy the code from the draw 3 request and add it to the unchanged shuffle request. Remove the redundant “import requests statement and the print(response) statement for the shuffle so that it looks like the following:



**It won’t run yet. Note that in the url assignment on line 11, you need your actual deck\_id from the first request, passed to the second. To do that, you will need to:**

1. **Convert your response variable on line 8 to a dictionary object as in number 3, above, using the variable.json() method like in 3G, above, and assign it to a variable.**
2. **Take that new dictionary variable and assign the “deck\_id” key value to a variable like you did above.**
3. **Modify the url variable with concatenation before the second request is sent to include the deck\_id variable (e.g., url=”**[**https://deckofcardsapi.com/api/deck/**](https://deckofcardsapi.com/api/deck/)**” + deck\_id + “/draw/?count=3”)**
4. **Take a screenshot of your output**
5. **Modify your program above where you ask your user how many cards from the deck they would like to draw. Then draw that many cards from the deck and print out the response unaltered (you will modify the count= part of the url variable before making the request).** Take a screenshot of your result and interaction with your user.

**Text

Description automatically generated**

1. Run your working script and choose 3 cards. From the interpreter, after your script runs, do the following as shown below:



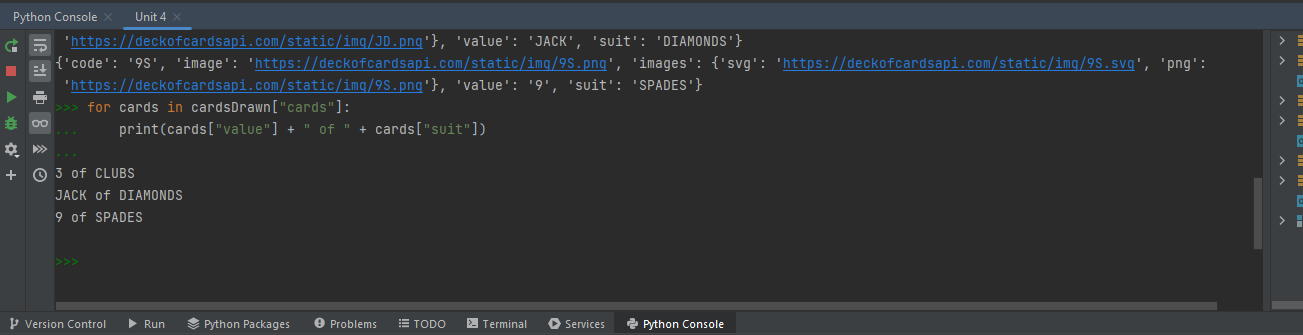
Note that your cards will be different. The original response returned a “requests.models.Response” custom datatype. I converted it to a dictionary called cardsDrawn. The dictionary contained nested objects, including a nested Dictionary with a nested List of Dictionaries with key: “value” pairs. I have highlighted the constructions. Note that cardsDrawn[“cards”] is a list, in this case a list of three dictionaries. See the square bracket right after ‘cards’: and the square bracket in the last highlight. That is the list. Within that Python List, there are three dictionaries returned, one for each card. We can iterate this list as follows (which you need to do):



In the above code, I am iterating through the cardsDrawn[“cards”] list. For each iteration, a Dictionary is returned with the card information. In my case, three cards were returned, and I have highlighted the first card. Since, card is a dictionary, I can access individual key pairs of the card Dictionary. See below:



I have shown in the gray highlighted area that card is a dictionary, therefore, I should be able to access key/pairs as in card[“value”] and card[“suit”], yellow highlighted in the original object print. Note the output of the three cards iterated in the last three lines. Take a screenshot of your output after you try the above.



1. **Write a script that does the following (don’t forget to use functions, validation checking, comments, etc., that meets scripting requirements rubric):**
   1. **Shuffles a single deck at the beginning of the script using the deckofcardsapi.com APIs that we have been using. Also, display a quick explanation of how the game is played. You will understand this by the end of the explanation.**
   2. **Ask the user how many cards, from 0 to 5 (0 means quit and exit), they wish to challenge the computer to a war with. Be sure to check to ensure proper values are entered and challenge the user if an incorrect value, like 6 or “g” is entered.**
   3. **Once the value is determined to be a correct value, have the computer draw that many cards. Display the text of the cards in some format something like: “ I drew a two of clubs and King of diamonds. That is 12 points for me.” (Note that for the values, Ace, King, Queen, and Jack, you will have to convert the name to an appropriate value).**
   4. **Next have the computer draw the same number of cards from the same deck for the user, repeating the above, but totaling the points for the user.**
   5. **Have the script declare who won, based on who had the highest score. If it was a tie, have the computer declare that as well.**
   6. **Have fun with this. You are almost ready to start working with network devices.**