Assignment 18 – API & JSON

Trivia Challenge Application

In this assignment, we will be creating an application where the user is presented with a series of trivia questions and multiple choice for the answers until the user enters an incorrect answer.

For the questions, we will be retrieving them from a website (opentdb.com) using API calls and parsing the JSON response. But before we begin coding the application, we will need to have a basic understanding of what an API is and the JSON format that will be returned.

What is an API?

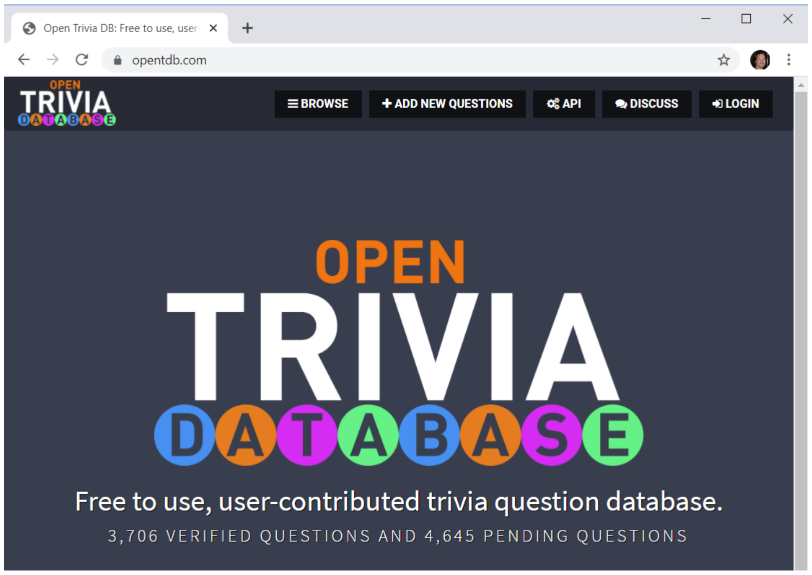
Technically speaking, API stands for **Application Programming Interface**. In very broad terms, companies have data that they want to make available to users, but they want to do so in a way that maintains the data security and integrity. To do this, they create an API that takes in parameters in a specific format and in return passes back data in a specific format. These specific input and output are known as a *contract*. The contract is between the data provider and the data consumer and as long as changes made by the data provider are not broken, the data consumer's application will not fail based on the data provided.

An example of this would be having gmail on your mobile phone. When you open the application, the application (client) connects to the Internet and sends data, typically a username and password, to the gmail server. On the server is an API that accepts the data, processes it, and send data back to the mobile phone application. The data that is returned must be in an interchange format that can be easily read by the client and two of the most popular are XML and JSON. In this example, the email server is the data provider and the email client on the mobile phone is the data consumer.

For our assignment, the data provider will be the website Open Trivia Database (www.opentdb.com) and the data will be returned in a JSON format.

Researching the APIs

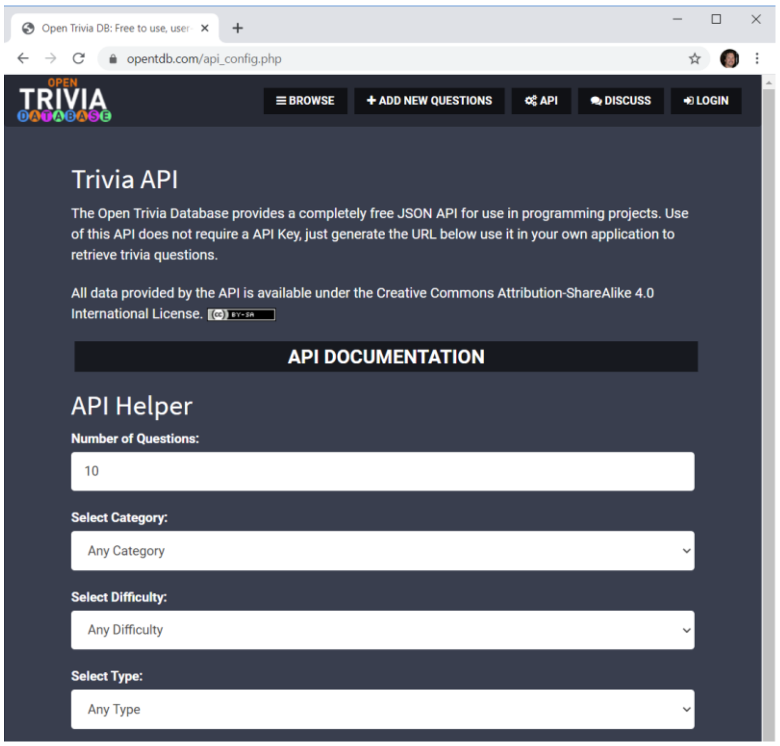
Before we can begin with creating the application, we need to understand the *contract* we will be using to get the trivia questions. The site, Open Trivia Database (www.opentdb.com) is a crowd sourced collection of trivia questions.



Free to use, user-contributed trivia question database.

3,706 VERIFIED QUESTIONS AND 4,645 PENDING QUESTIONS

Clicking on the API link at the top takes you to a page that includes documentation and various parameters for filtering the trivia questions which will then allow you to build a URL sample. By experimenting with this, we will be able to build a URL to fit our exact needs.



Trivia API

The Open Trivia Database provides a completely free JSON API for use in programming projects. Use of this API does not require a API Key, just generate the URL below use it in your own application to retrieve trivia questions.

All data provided by the API is available under the Creative Commons Attribution-ShareAlike 4.0 International License.

API Helper

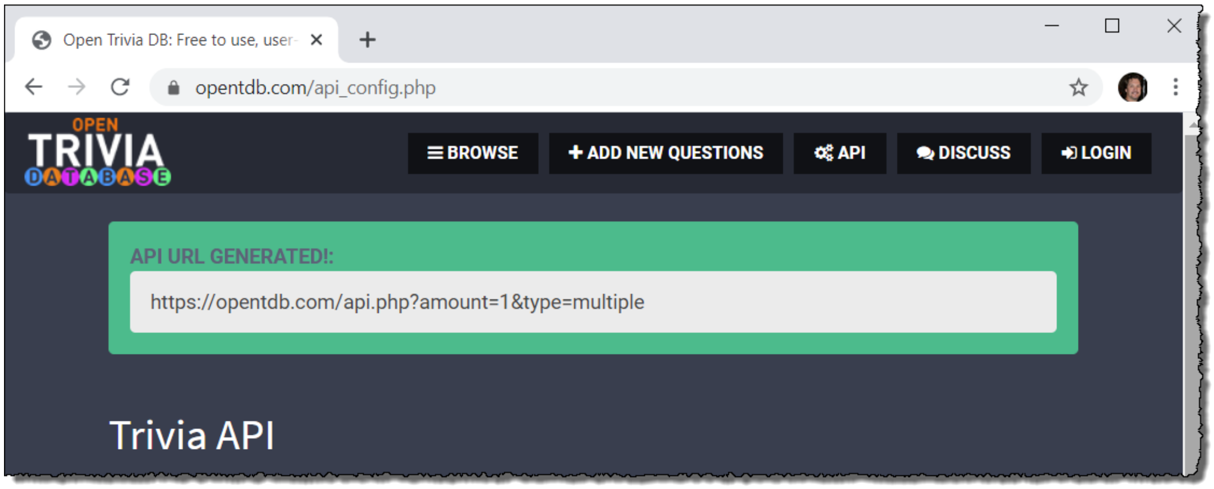
Number of Questions: 10

Select Category: Any Category

Select Difficulty: Any Difficulty

Select Type: Any Type

For our game, we will be retrieving the questions one question at a time, and we will only want multiple choice questions, so our URL will look like this:



https://opentdb.com/api.php?amount=1&type=multiple

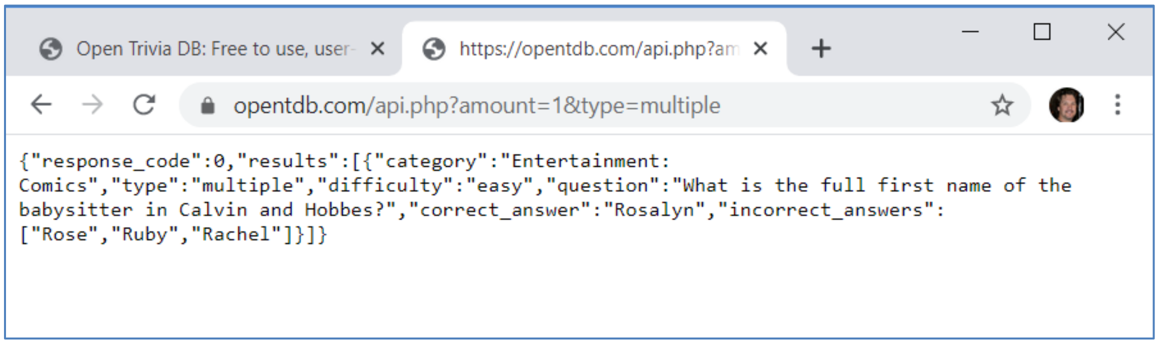
Trivia API

What is JSON?

JSON stand for JavaScript Object Notation and is a lightweight data-interchange format. In simple terms, JSON is a collection of name/value pairs.

Viewing the JSON Output

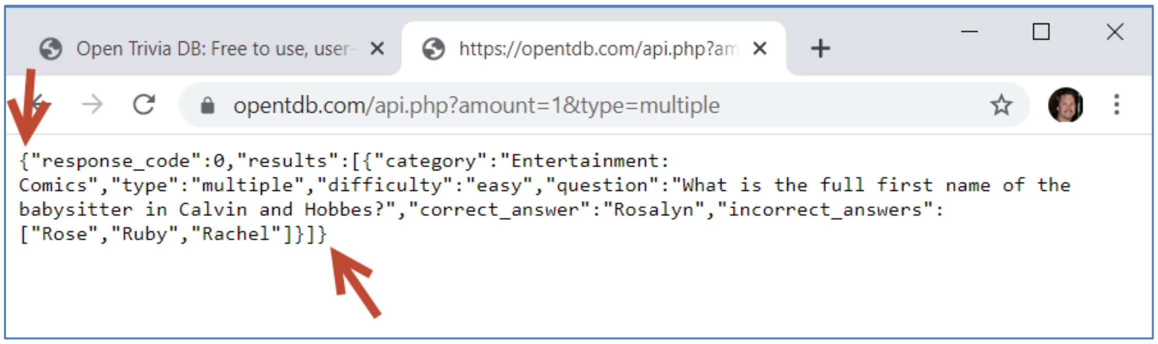
If we copy the URL from above and paste into a web browser, we'll be able to see a sample output.


Open Trivia DB: Free to use, user
https://opentdb.com/api.php?am

opentdb.com/api.php?amount=1&type=multiple

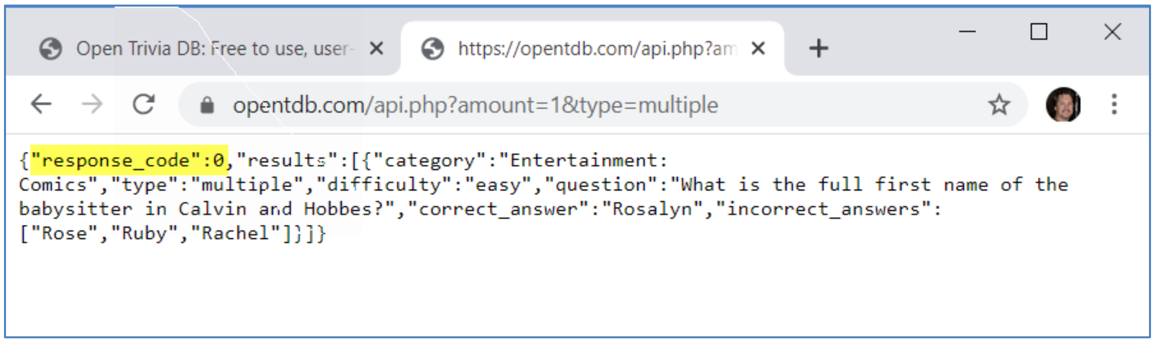
```
{"response_code":0,"results":[{"category":"Entertainment:
Comics","type":"multiple","difficulty":"easy","question":"What is the full first name of the
babysitter in Calvin and Hobbes?","correct_answer":"Rosalyn","incorrect_answers":
["Rose","Ruby","Rachel"]}]}
```

As mentioned, JSON is a collection of key/value pairs and these key value pairs are surrounded by curly braces.


Open Trivia DB: Free to use, user
https://opentdb.com/api.php?am
opentdb.com/api.php?amount=1&type=multiple

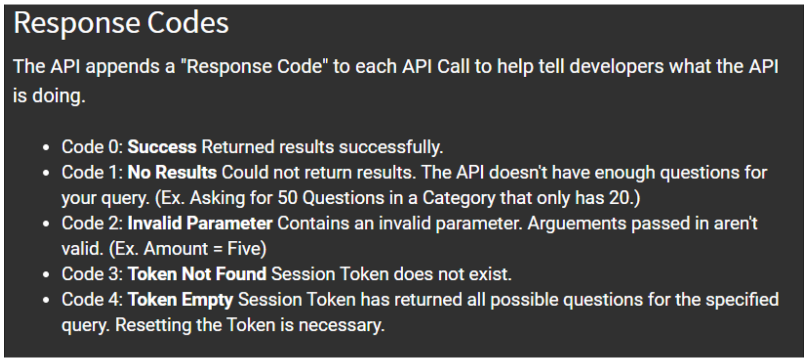
```
{"response_code":0,"results":[{"category":"Entertainment: Comics","type":"multiple","difficulty":"easy","question":"What is the full first name of the babysitter in Calvin and Hobbes?","correct_answer":"Rosalyn","incorrect_answers":["Rose","Ruby","Rachel"]}]}
```

The first key/value pair in our object is the “response_code”:0. Notice the “response_code is the key and 0 is the value, and the key and value are separated by a colon “:”.


Open Trivia DB: Free to use, user
https://opentdb.com/api.php?am
opentdb.com/api.php?amount=1&type=multiple

```
{"response_code":0,"results":[{"category":"Entertainment: Comics","type":"multiple","difficulty":"easy","question":"What is the full first name of the babysitter in Calvin and Hobbes?","correct_answer":"Rosalyn","incorrect_answers":["Rose","Ruby","Rachel"]}]}
```

In the API Documentation, you might have noticed a table of valid values for the response codes:



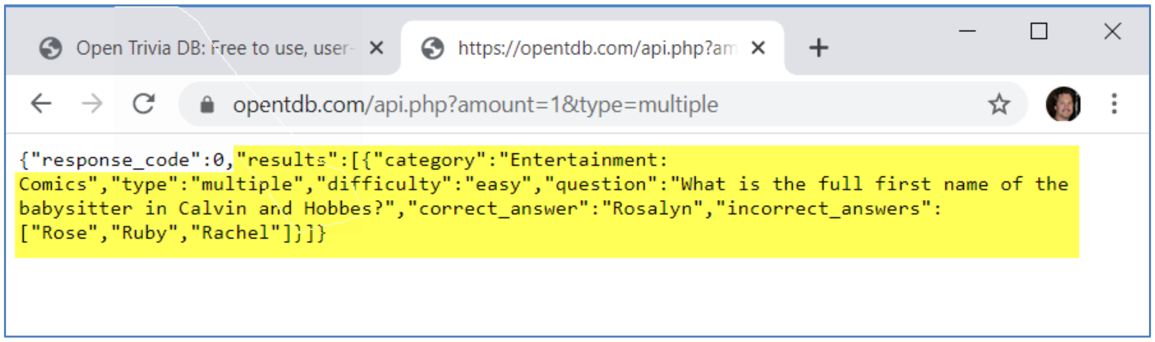
Response Codes

The API appends a "Response Code" to each API Call to help tell developers what the API is doing.

- Code 0: **Success** Returned results successfully.
- Code 1: **No Results** Could not return results. The API doesn't have enough questions for your query. (Ex. Asking for 50 Questions in a Category that only has 20.)
- Code 2: **Invalid Parameter** Contains an invalid parameter. Arguements passed in aren't valid. (Ex. Amount = Five)
- Code 3: **Token Not Found** Session Token does not exist.
- Code 4: **Token Empty** Session Token has returned all possible questions for the specified query. Resetting the Token is necessary.

In our sample, the response code is "0", meaning that the site was able to successful return a result for our URL.

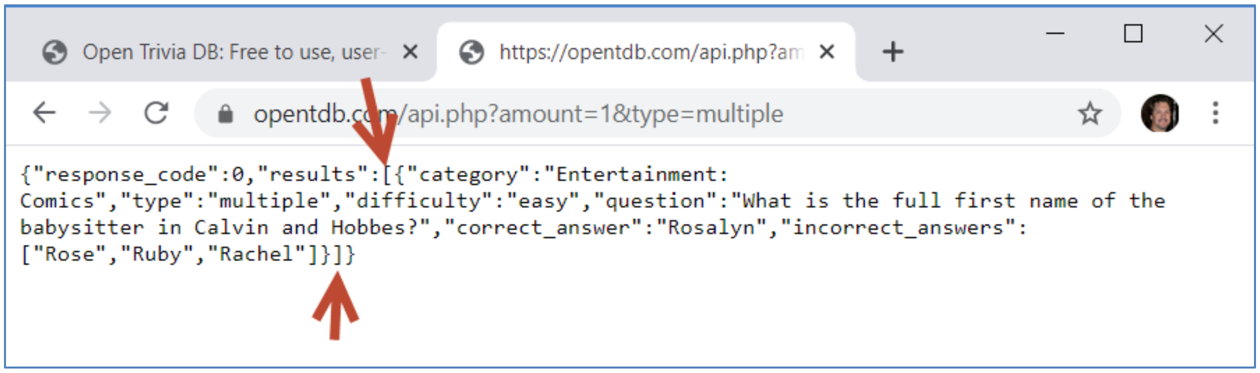
The key/value pairs are separated by a comma "," and the next key/value pair is "results" where results is the key, and the value is pretty much everything remaining in the output. The *everything else* is actually a nested collection of key/value pairs.


Open Trivia DB: Free to use, user-
https://opentdb.com/api.php?am
opentdb.com/api.php?amount=1&type=multiple

```
{"response_code":0,"results":[{"category":"Entertainment:
Comics","type":"multiple","difficulty":"easy","question":"What is the full first name of the
babysitter in Calvin and Hobbes?","correct_answer":"Rosalyn","incorrect_answers":
["Rose","Ruby","Rachel"]}]}
```

This might seem a little confusing at first so let's try to break this down.

To start with, remember in our URL, we can request 1 or more questions to be returned at a single time. We only selected to return 1 but if we returned more then 1, the questions would be returned as a list (sometimes also referred to as an array). This can be determined by the square brackets [] after the ":" which separate the key and the value.


Open Trivia DB: Free to use, user-
https://opentdb.com/api.php?am
opentdb.com/api.php?amount=1&type=multiple

```
{"response_code":0,"results":[{"category":"Entertainment:
Comics","type":"multiple","difficulty":"easy","question":"What is the full first name of the
babysitter in Calvin and Hobbes?","correct_answer":"Rosalyn","incorrect_answers":
["Rose","Ruby","Rachel"]}]}
```

If we know everything inside curly braces is a single entity, and we know this is a question being returned, we can break down the remaining key/value pairs as the data for the question. Examining the data, we can determine the question will look something like this:

Key	Value
category	Entertainment: Comics
type	multiple (this indicates that this is a multiple choice question)
difficulty	east
question	What is the full name of the babysitter in Calvin and Hobbes?
correct_answer	Rosalyn
incorrect_answers	[Rose, Ruby, Rachel]

Hopefully, you noticed the square brackets around the incorrect_answers value. Again, this indicates that the values are in the form of a list. After this question, if we returned more than 1 question, we would see a comma and 1 or more whole question with the same key/value pairs.

Creating the Trivia Challenge Application

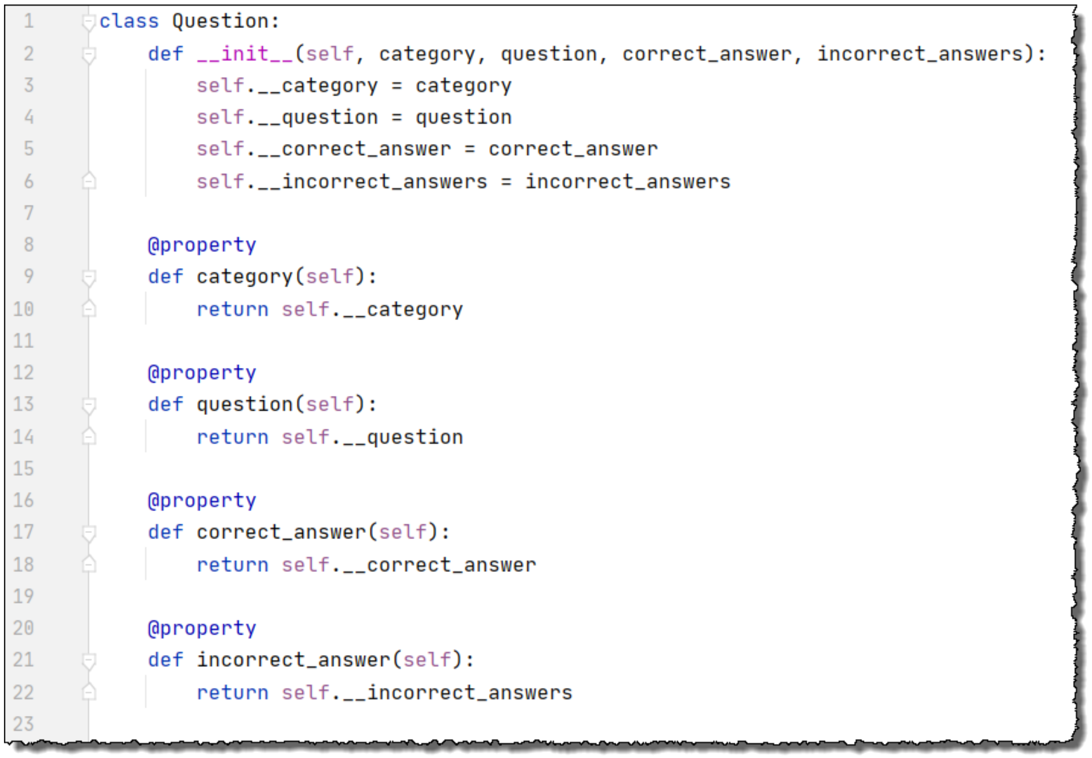
It's finally time to start working on our Trivia Challenge application. We're going to be sticking to the OOP principles we learned in previous assignments so we will have a separate module for the presentation, business, and data layers.

If you guessed that the API and JSON response are all part of the data layer, you're correct! Using the JSON structure, we can now create the class object for our question.

Business Layer

The JSON file is just an interchange file, a way of getting the data from the API to our application and we will want to convert the data from this file into some form that will be easier for us to work with in our application.

1. Start by creating a new python **trivia_challenge** project.
2. Create a new python file named **business.py**.
3. Code the Question class as follows:
    a. Notice that we're encapsulating the class (i.e. making all the properties *private*).
    b. We're also adding the @property annotation which does a few things for us. For one, it allows us to access these private properties in other modules as if the properties were public, and it also makes our code a little easier to read.

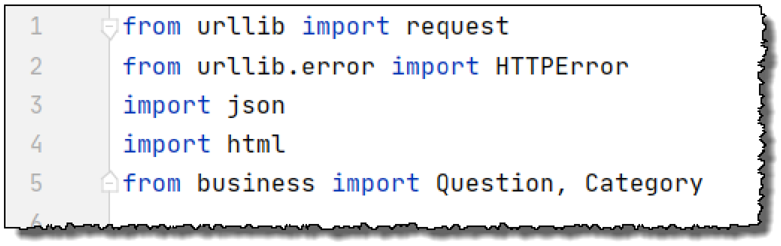


```python
class Question:
    def __init__(self, category, question, correct_answer, incorrect_answers):
        self.__category = category
        self.__question = question
        self.__correct_answer = correct_answer
        self.__incorrect_answers = incorrect_answers

    @property
    def category(self):
        return self.__category

    @property
    def question(self):
        return self.__question

    @property
    def correct_answer(self):
        return self.__correct_answer

    @property
    def incorrect_answer(self):
        return self.__incorrect_answers

```

Data Layer

Now that we have our class created, we can create the API call to get the data and instantiate (create) a new Question object.

4. Create a new python file named **requests.py**.
5. Start coding the imports as follows:

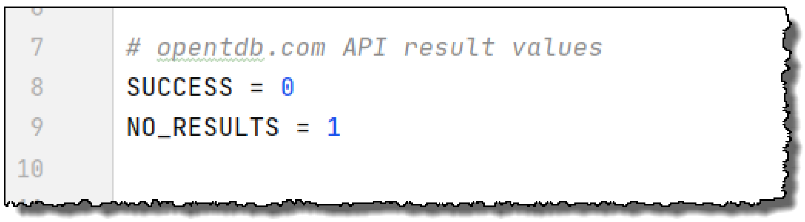


```python
from urllib import request
from urllib.error import HTTPError
import json
import html
from business import Question, Category
```

Lots of new imports here that you're probably not familiar with so let's review:

- The urllib import includes the functionality needed to make the API call.
- The json import includes the functionality needed to read the json file.
- The html import will provide functionality to convert "escaped characters".
    - To refresh, certain characters are not usable in html so they are converted to "escaped" characters and we want to convert those characters back for displaying in our application.

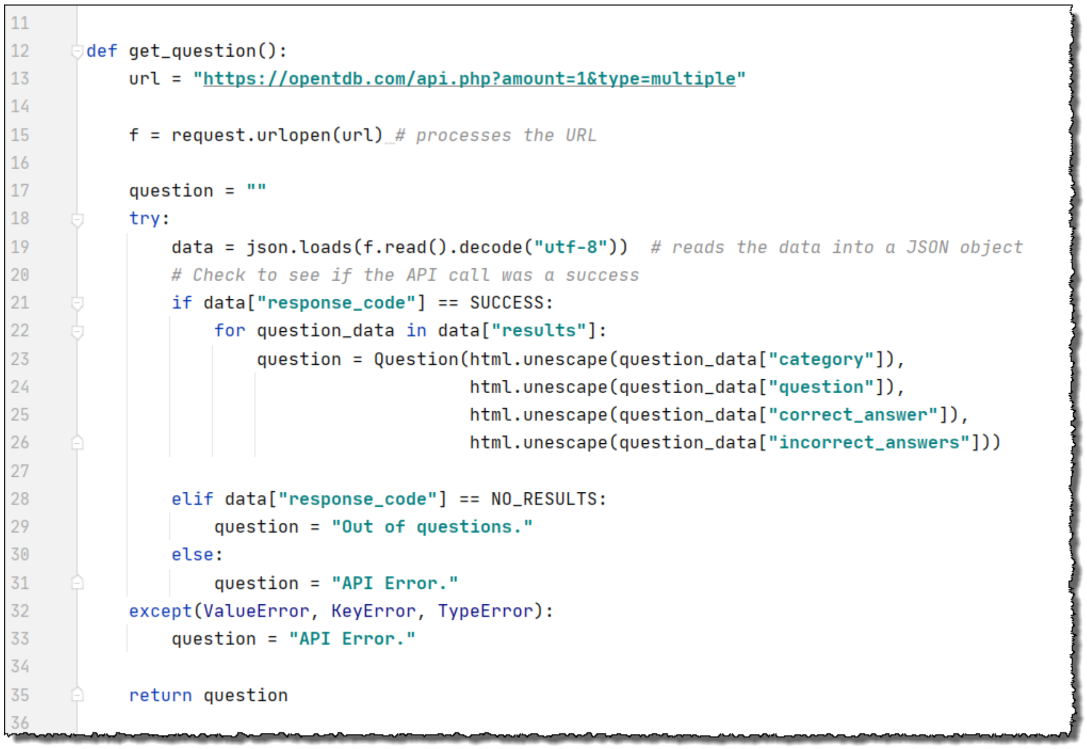
6. We next want to add a few constants to hold the API result codes. We will reference these later.



```python
# opentdb.com API result values
SUCCESS = 0
NO_RESULTS = 1
```

We are now ready to code the function that will get the question data from the API and return a question object.

7. Code as follows:



```
11
12  def get_question():
13      url = "https://opentdb.com/api.php?amount=1&type=multiple"
14
15      f = request.urlopen(url) # processes the URL
16
17      question = ""
18      try:
19          data = json.loads(f.read().decode("utf-8"))  # reads the data into a JSON object
20          # Check to see if the API call was a success
21          if data["response_code"] == SUCCESS:
22              for question_data in data["results"]:
23                  question = Question(html.unescape(question_data["category"]),
24                                      html.unescape(question_data["question"]),
25                                      html.unescape(question_data["correct_answer"]),
26                                      html.unescape(question_data["incorrect_answers"]))
27
28          elif data["response_code"] == NO_RESULTS:
29              question = "Out of questions."
30          else:
31              question = "API Error."
32      except(ValueError, KeyError, TypeError):
33          question = "API Error."
34
35      return question
36
```

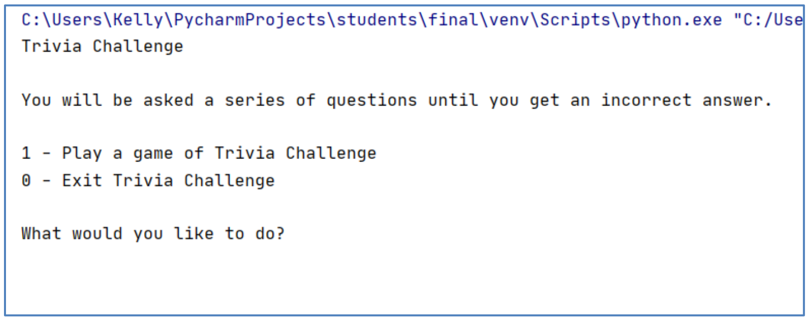
More new code so before continuing, let's discuss:

- The URL is the same as what we tested earlier in this document.
- The request.urlopen processes the API request with the f as a reference to the returned output.
- On line 19 of the code, the f.read() reads the returned data that was returned and the the json.loads converts that data into json format.
- On line 21, we now need to validate the API was a success and if so, start the process of taking the json data and creating a question object.
- On line 22, remember, the output could return multiple question entities so we need to process the data in a loop and we're getting all the data (value) for the key "results".
- Next, we'll pass the required data into the Question class constructor to instantiate the Question object.
- If everything went as expected, we should return be returning a valid question object on line 35

Presentation Layer

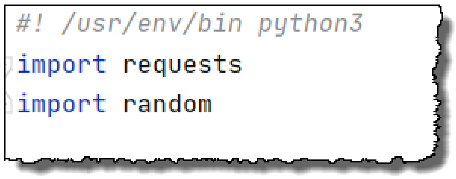
With the ability to retrieve trivia questions from the site, we can now turn our attention to creating the user interface.

We will start with a very simple UI that will allow the user to play a game and to exit.



```
C:\Users\Kelly\PycharmProjects\students\final\venv\Scripts\python.exe "C:/Use
Trivia Challenge

You will be asked a series of questions until you get an incorrect answer.

1 - Play a game of Trivia Challenge
0 - Exit Trivia Challenge

What would you like to do?
```

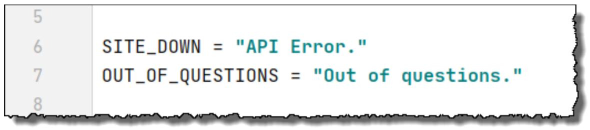
8. Create a new python file named **trivia.py**.
9. Start coding the imports as follows:
   a. We wi'll need the requests to get the trivia questions and the random to be able to randomize the order in which the answers are displayed.



```python
#! /usr/env/bin python3
import requests
import random
```

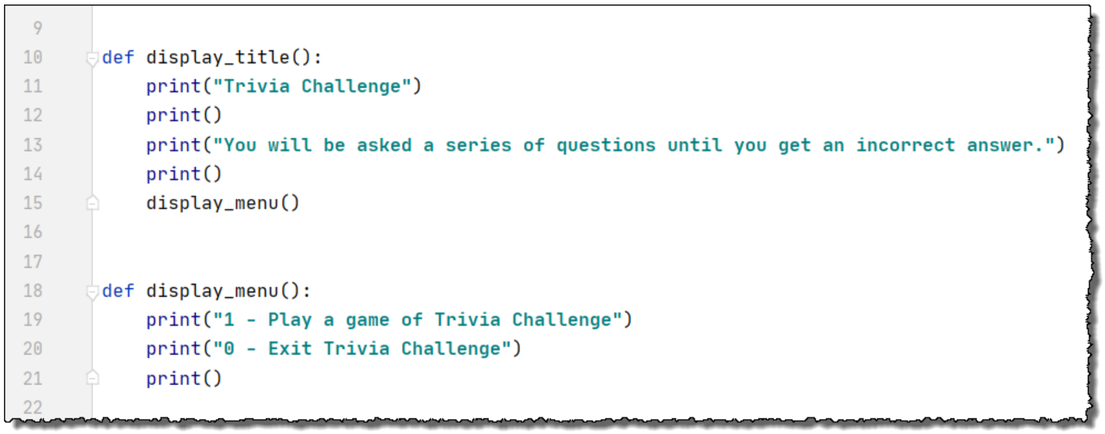
Next, we will add some constants which will be used to validate the API response.

10. Code the response constants as follows:



```python

SITE_DOWN = "API Error."
OUT_OF_QUESTIONS = "Out of questions."

```

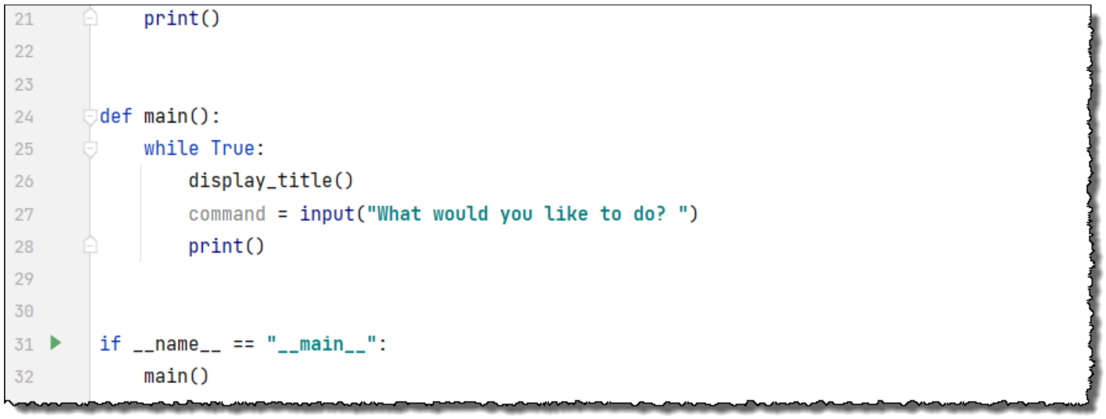
11. Now let's add the application header and menu:



```python

def display_title():
    print("Trivia Challenge")
    print()
    print("You will be asked a series of questions until you get an incorrect answer.")
    print()
    display_menu()


def display_menu():
    print("1 - Play a game of Trivia Challenge")
    print("0 - Exit Trivia Challenge")
    print()

```

Before we get too far in, let's add the menu and test the application to make sure everything is working so far.

12. Code the main function as follows:

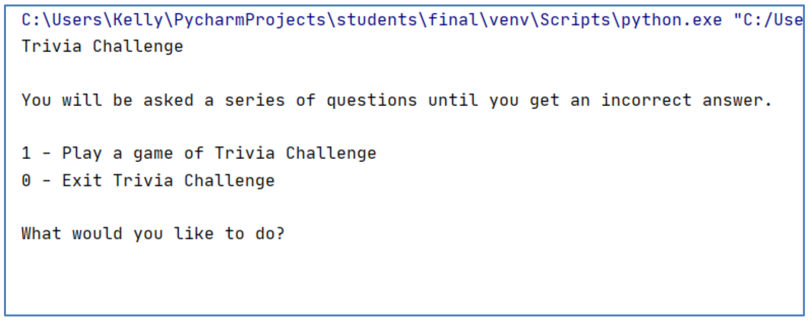


```
    print()


def main():
    while True:
        display_title()
        command = input("What would you like to do? ")
        print()


if __name__ == "__main__":
    main()
```

It's time to test!

13. Run the trivia application:

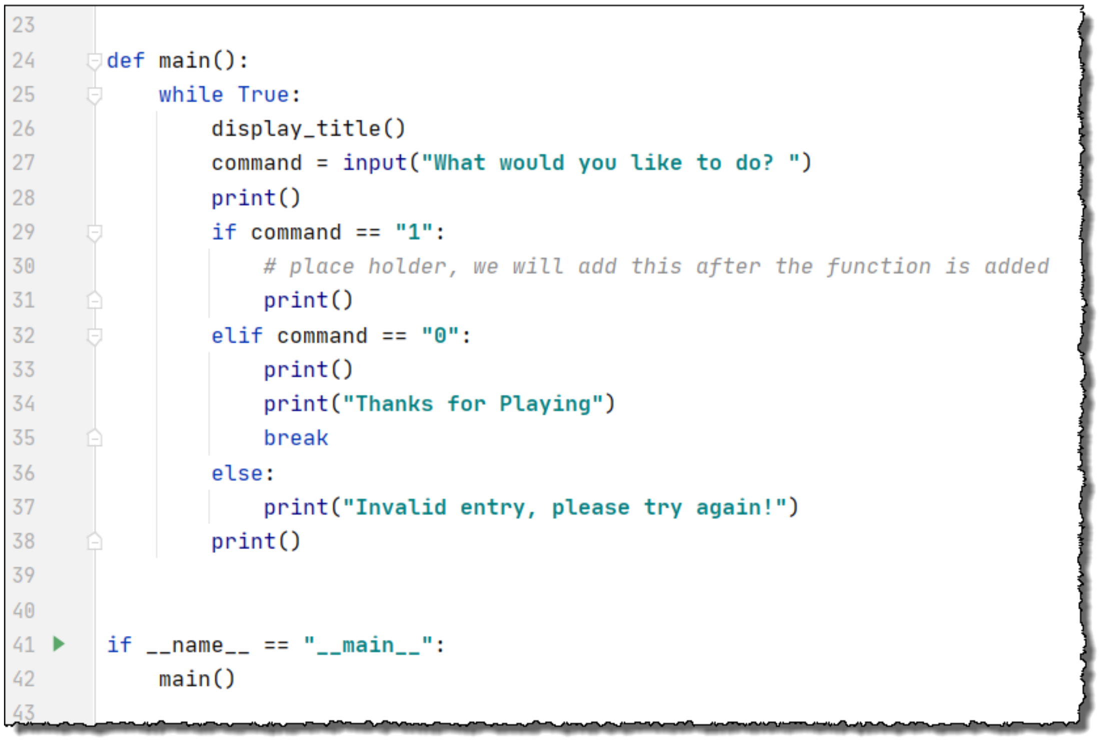
Screen Capture #1 (2 points)



```
C:\Users\Kelly\PycharmProjects\students\final\venv\Scripts\python.exe "C:/Use
Trivia Challenge

You will be asked a series of questions until you get an incorrect answer.

1 - Play a game of Trivia Challenge
0 - Exit Trivia Challenge

What would you like to do?
```

Next up, we will add the functionality to retrieve and display a question, get a response from the user, validate the response and if its correct, repeat. This will continue until the user enters an incorrect response.

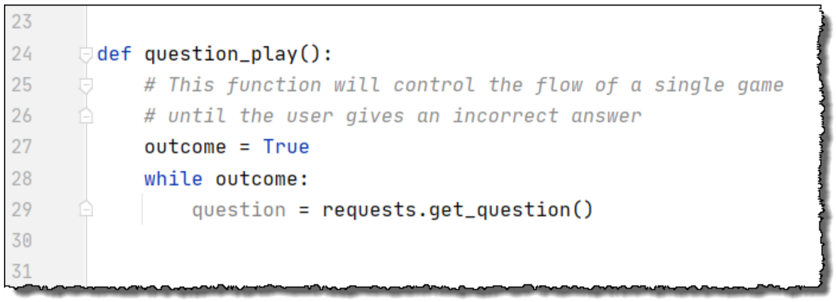
14. Code the main function as follows:
    a. Only thing to note here is the place holder on line 30 of the code. We will come back to this after we've coded out the function.



```
23
24  def main():
25      while True:
26          display_title()
27          command = input("What would you like to do? ")
28          print()
29          if command == "1":
30              # place holder, we will add this after the function is added
31              print()
32          elif command == "0":
33              print()
34              print("Thanks for Playing")
35              break
36          else:
37              print("Invalid entry, please try again!")
38          print()
39
40
41  if __name__ == "__main__":
42      main()
43
```

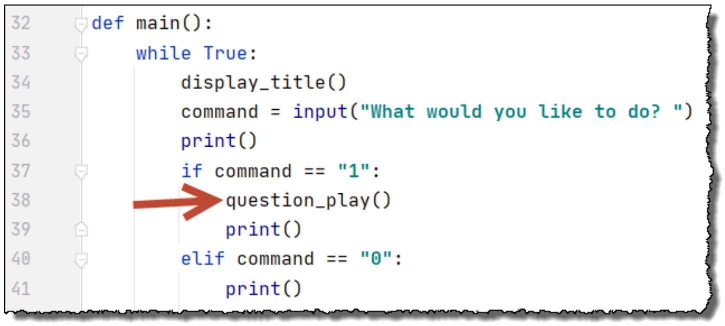
Let's start coding the function to control the flow of a single game.

15. Code the function as follows. Note: This is just a stub which we'll be building upon.



```

def question_play():
    # This function will control the flow of a single game
    # until the user gives an incorrect answer
    outcome = True
    while outcome:
        question = requests.get_question()


```

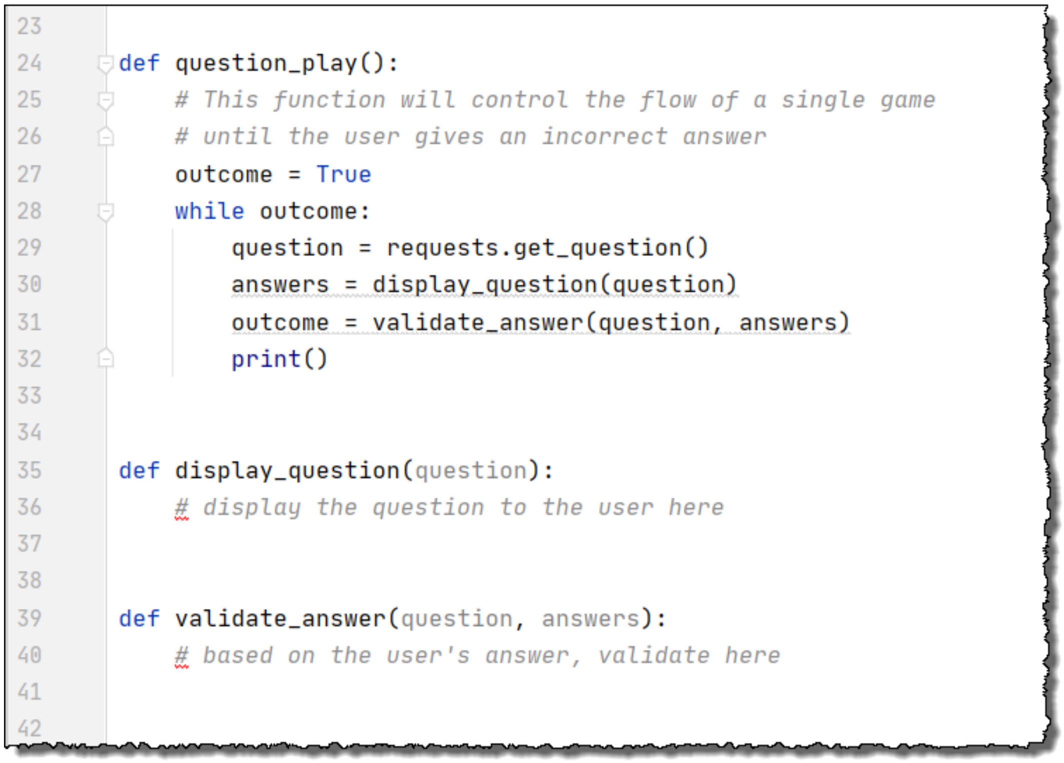
16. Time to revisit the main function and add a call to the new function.



```
def main():
    while True:
        display_title()
        command = input("What would you like to do? ")
        print()
        if command == "1":
            question_play()
            print()
        elif command == "0":
            print()
```

Let's stub out the rest of the question_play function. We know we need to display the question and get the user's guess and then validate the guess based on the correct answer.

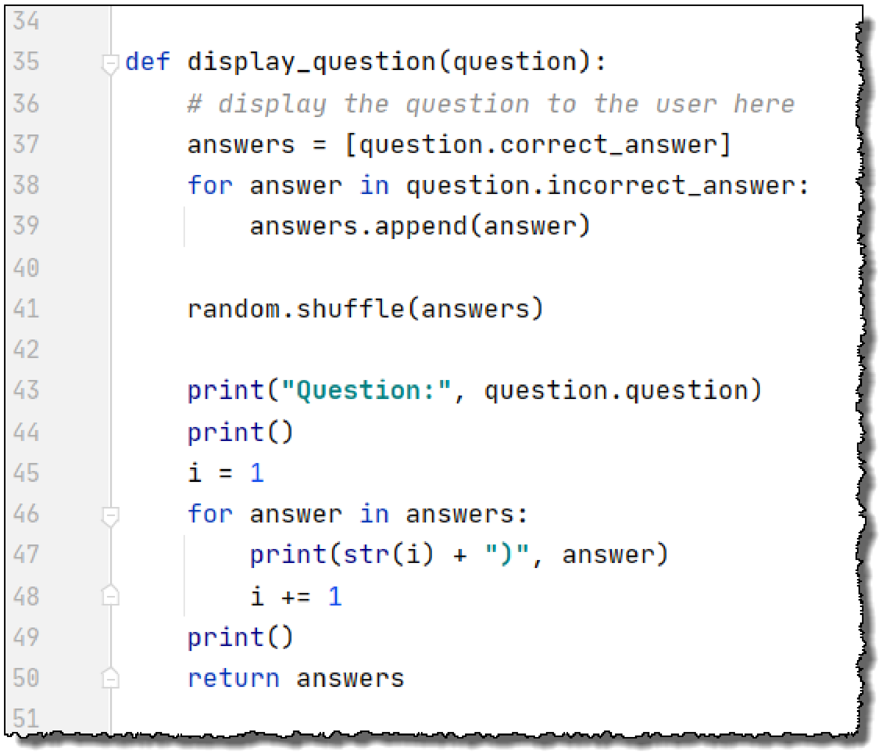
17. Code the logic as follows:



```

def question_play():
    # This function will control the flow of a single game
    # until the user gives an incorrect answer
    outcome = True
    while outcome:
        question = requests.get_question()
        answers = display_question(question)
        outcome = validate_answer(question, answers)
        print()


def display_question(question):
    # display the question to the user here


def validate_answer(question, answers):
    # based on the user's answer, validate here

```

Now let's code out the stubs starting with displaying the question.

18. Code the display question as follows:

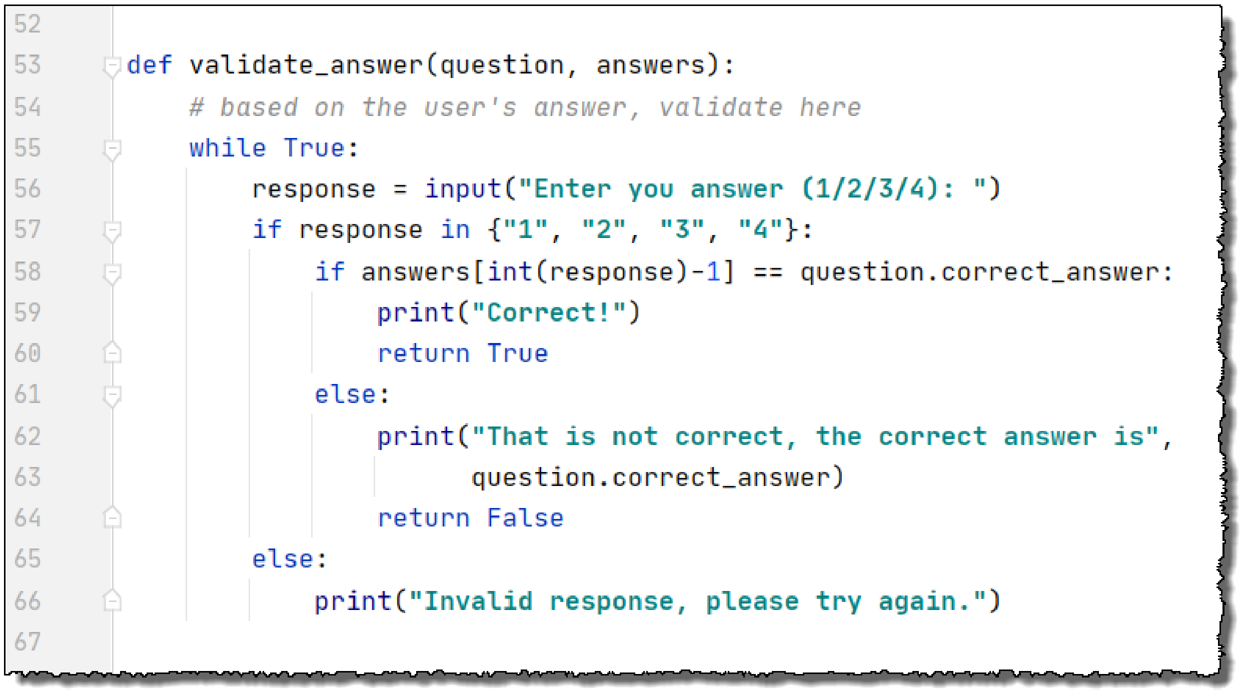


```
34
35  def display_question(question):
36      # display the question to the user here
37      answers = [question.correct_answer]
38      for answer in question.incorrect_answer:
39          answers.append(answer)
40
41      random.shuffle(answers)
42
43      print("Question:", question.question)
44      print()
45      i = 1
46      for answer in answers:
47          print(str(i) + ")", answer)
48          i += 1
49      print()
50      return answers
51
```

Let's review the logic of the function.

- First, we're going to create a list of all the answers, starting with the correct answer. (line 37)
- Next, we'll add all the incorrect answer to the list. (line 38-39)
- Now that we have a list with all the answers in it, we'll shuffle the list for display (line 41)
- On line 43, we'll display the question to the user and finally from lines 45 to 48, we'll display the answers.

19. Finally, let’s code out the validation function as follows:



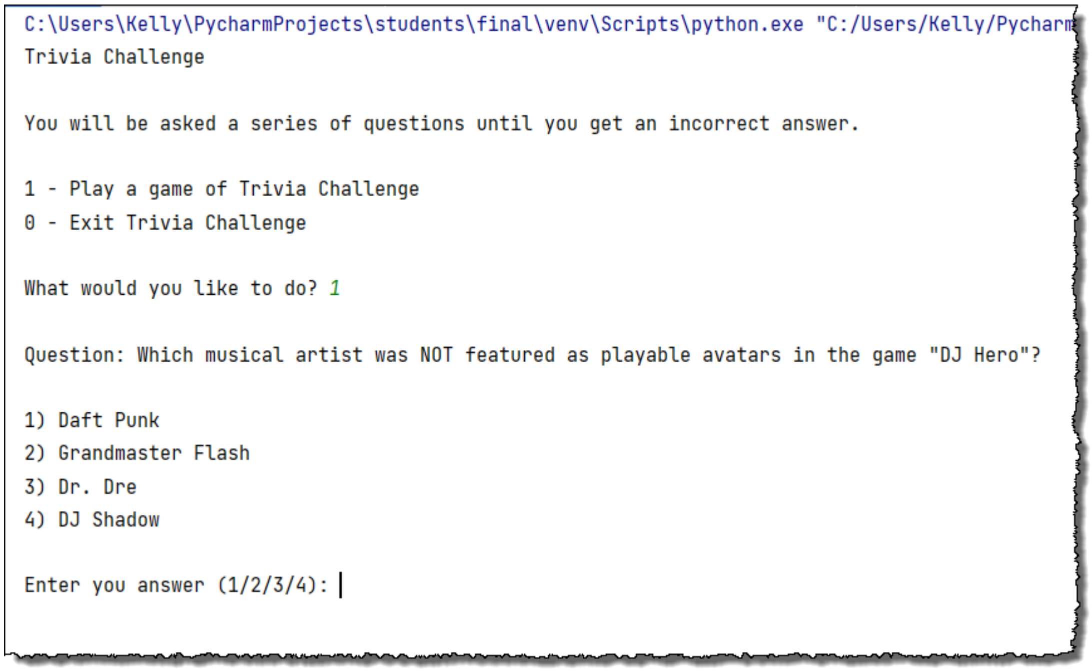
```
52
53  def validate_answer(question, answers):
54      # based on the user's answer, validate here
55      while True:
56          response = input("Enter you answer (1/2/3/4): ")
57          if response in {"1", "2", "3", "4"}:
58              if answers[int(response)-1] == question.correct_answer:
59                  print("Correct!")
60                  return True
61              else:
62                  print("That is not correct, the correct answer is",
63                        question.correct_answer)
64                  return False
65          else:
66              print("Invalid response, please try again.")
67
```

Quick explanation of the code:

- This function will take in 2 parameters, the question object and the user’s guess.
- The question is displayed with multiple choice options for the answer so continue to display the answers until the user makes a valid select.
- Lines 58 to 64 will determine it the user’s guess was correct returning the corresponding value (true for correct, false for an incorrect guess)

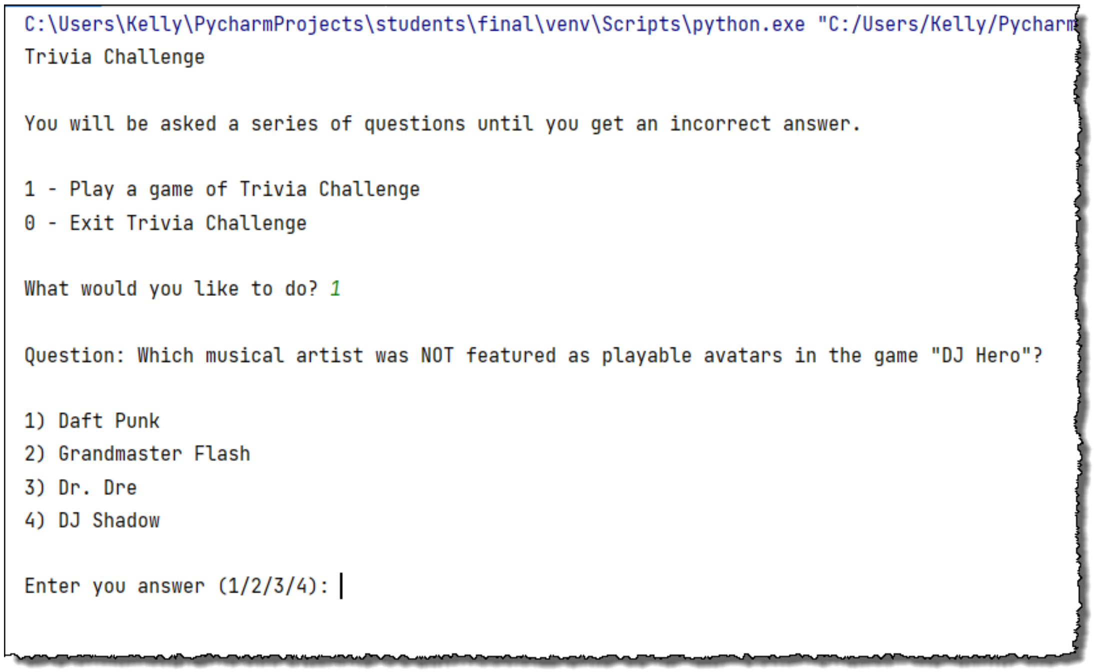
Once again, it’s time to test!

20. Run the application.
21. Select 1, Play a game of Trivia Challenge.
    Screen Capture #2 (3 points)



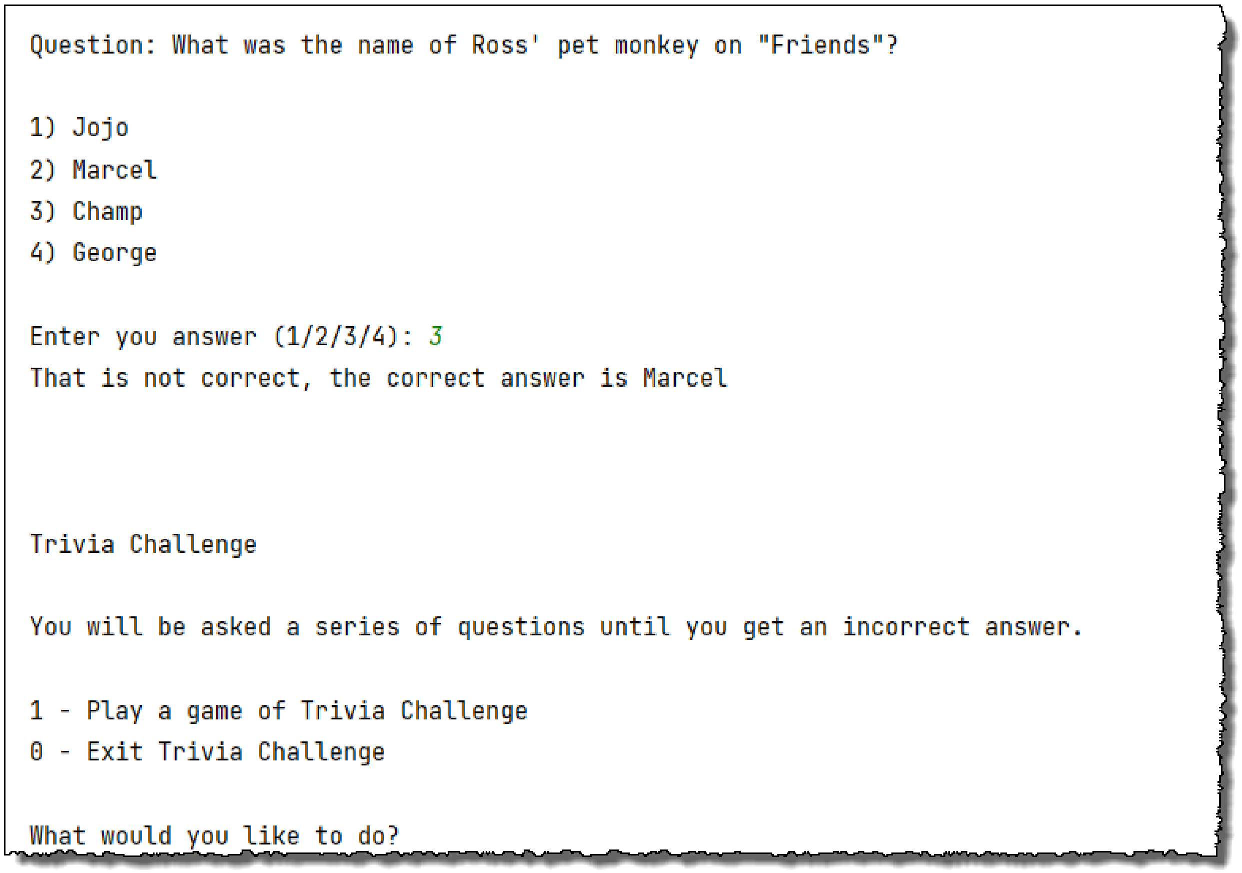
```
C:\Users\Kelly\PycharmProjects\students\final\venv\Scripts\python.exe "C:/Users/Kelly/Pycharm
Trivia Challenge

You will be asked a series of questions until you get an incorrect answer.

1 - Play a game of Trivia Challenge
0 - Exit Trivia Challenge

What would you like to do? 1

Question: Which musical artist was NOT featured as playable avatars in the game "DJ Hero"?

1) Daft Punk
2) Grandmaster Flash
3) Dr. Dre
4) DJ Shadow

Enter you answer (1/2/3/4): 
```

22. Make a guess and validate the answer is correct or incorrect.
    Screen Capture #3 (3 points) – Show when the guess is correct.



```
C:\Users\Kelly\PycharmProjects\students\final\venv\Scripts\python.exe "C:/Users/Kelly/Pycharm
Trivia Challenge

You will be asked a series of questions until you get an incorrect answer.

1 - Play a game of Trivia Challenge
0 - Exit Trivia Challenge

What would you like to do? 1

Question: Which musical artist was NOT featured as playable avatars in the game "DJ Hero"?

1) Daft Punk
2) Grandmaster Flash
3) Dr. Dre
4) DJ Shadow

Enter you answer (1/2/3/4): 
```

Screen Capture #4 (2 points) – Show when the guess is incorrect.



```
Question: What was the name of Ross' pet monkey on "Friends"?

1) Jojo
2) Marcel
3) Champ
4) George

Enter you answer (1/2/3/4): 3
That is not correct, the correct answer is Marcel



Trivia Challenge

You will be asked a series of questions until you get an incorrect answer.

1 - Play a game of Trivia Challenge
0 - Exit Trivia Challenge

What would you like to do?
```

On an incorrect answer, the title and menu should redisplay, and the user should be prompted with “What would you like to do?”

Code Validation (10 points) – Include all 3 python files with your submission.

Extra Credit

To get full points for each extra credit, you must include the python (.py) code files.

Extra Credit #1 – Add a session token (2 points)

Without a session token, opentdb can return previously asked questions. By adding the session token, this will ensure a new question is returned each time during a given session. You can find details on how to get and use the session tokens in the API Documentation section in the API section of the opentdb site.

For this you will need to:

- Create a function in the request layer to retrieve a session token (the url can be found on the opentdb site)
- Save and add the session token to your get question url