

Reflections on creating a Python application consuming a RESTFUL web service (1801641)

by Jaromir Pauer

FILE	11677_JAROMIR_PAUER_REFLECTIONS_ON_CREATING_A_PYTHON_APPLICATION_CONSUMING_A_RESTFUL_WEB_SERVICE__1801641__314650_1062257955.DOCX (544.66K)		
TIME SUBMITTED	10-JAN-2021 01:59PM (UTC+0000)	WORD COUNT	1448
SUBMISSION ID	141398180	CHARACTER COUNT	7461

**Reflections on creating a Python application
consuming a RESTFUL web service.**

**Network Programmability
(ACCA5029)**

University of Wales Trinity Saint David

Jaromir Pauer (1801641)

10/1/2021

Contents

1. Introduction.	3
2. First LotR challenge.....	3
3. Second LotR challenge	4
4. Third LotR challenge.....	4
5. Fourth LotR challenge.	4
6. Conclusion.	5
7. Link to the GitHub repository:	5
8. Main and Sub Menu code snippets and screenshots:	6
8.1 Code.	6
8.2 Running the code screenshot	8
9. List of Books code snippets and screenshots:.....	9
9.1 Code.	9
9.2 Running the code screenshot:	9
10. List of Movies code snippets and screenshots:.....	10
10.1 Code.	10
10.2 Running the code screenshot:	11
11. List of Characters code snippets and screenshots:	12
11.1 Code.	12
11.2 Running the code screenshots:.....	13
12. List of Quotes code snippets and screenshots:.....	15
12.1 Code.	15
12.2 Running the code screenshots:.....	16
13. List of Chapters code and snippets:	17
13.1 Code.	17
13.2 Running the code screenshots:.....	18
14. List Chapters of a Book function snippets and screenshots:	20
14.1 Code.	20
14.2 Running the code screenshot:	21
15. Search for A Book function snippets and screenshots:	22
15.1 Code.	22
15.2 Running the code screenshot:	22

1. Introduction.

In this report the developer reflects on his experiences during the programming of the Python application. The developer realised that it was important to reflect on previous university assignments before starting work on this exercise. One important lesson previously learnt was that when designing a project careful consideration should be given to the customer's requirements for a product and how these can be delivered effectively, efficiently and economically. Following this and other academic advice, the developer sketched out an overview of the assignment setting out what would need to be addressed if the programme was to work successfully. On reflection this was a useful approach. Initially, a number of functions were successfully performed regarding the FavQs.com API, but difficulties were encountered, for example, the user session authorisation element could not be made to work completely. So, the Lord of the Rings API (LotR) was chosen as a replacement.

2. First LotR challenge.

Work commenced on the LotR API by adopting some of the functions that had been successfully performed on the FavQs.com API, which saved a great deal of time. The next step involved creating functions for the book, movie, character, quote and chapter end points listed in the API documentation. A difficulty was encountered at this stage if the user wished to print a list of quotes relating to specific movies. In short, a GET request sent to the /quote endpoint would elicit a response that consisted of JSON data including various keys and values. The values were largely set out in a string format that the user would be incapable of understanding, for example, the name of a movie would have read 5cd95395de30eff6ebccde5b. Eventually, this problem was resolved by sending another GET request to a different end point. The GET request included the URL, consisting of a URL link + the movieID, and the header information, consisting of an authorisation token and other information. A variable was created into which the response was stored. At this point it was possible to start looping through the data, stored in the variable, to obtain exact information about the movie. Crucially, this solution allowed the whole process to be automated behind the scene. Ultimately, the user was presented with intelligible script that paired the quote with the name of the relevant movie. The same approach was utilised a number of times, for example, the programme could quickly provide the user with the name and race of the character associated with a quote. So, the quality of service to the user was greatly improved. Looking back, work on this part of the code has improved the developer's understanding of how a programmer can link crucial data and refine a programme. This learning experience has also boosted the developer's self-confidence as a programmer.

3. Second LotR challenge

The ListOfCharacters function was designed to print out information about characters. Sometimes during the process of looping through the data for each character, it was found that information was missing either for race, gender, name, or the Wikipedia URL. The problem resulted in this function breaking down and printing out the words 'KeyError'. It followed that the function could not provide the user with a complete list of characters and it did not therefore deliver all the desired information. It was eventually established that the root problem was a missing 'key' in a particular end point of the API. Research was undertaken to find a way of addressing this problem. It was decided that that it could be addressed by applying the 'try and except' method, which was designed to handle exceptions. Specifically, the programme was instructed to use the 'try' statement before printing the race, gender, name, and Wiki URL for each character. Then the 'except KeyError' statement was inserted so that the programme printed out a pre-defined text if a missing key was encountered. The experience gained during the resolution of this problem should be of great use in the future.

4. Third LotR challenge.

A less happy reflection was tackling a difficulty that remains unresolved, although it did not affect the functionality of the programme as far as the user was concerned. If the user selected option 7 of the main menu it took him to the submenu. At the submenu stage the user could choose to print a list of chapters from any of the listed books and the programme would have no problem delivering such requests. The difficulty remains that the programme has three functions regarding the submenu. Each function incorporates a great deal of repetitive lines of code. Ideally, the developer would have preferred a single streamlined function for the submenu options, but he was unable to find such a solution. In a real life scenario a long, complicated programme including unnecessary repetition might be considered inefficient, or indeed ineffective.

5. Fourth LotR challenge.

Another unresolved difficulty occurred when the user selected 'search for a book', which is option six of the main menu. This option instructs the user to type a capital letter at the start of every word, but otherwise he should use lowercase type. If the user does not comply by entering the name of a listed book in the case sensitive manner, the programme will perform the search and then inform him that 'sorry the book you were looking for was not found'. If on the other hand, the user types the correct value, complying with case sensitive rule, for a listed book, the programme will find the book and print out the search results. However, the function goes on to add the words 'Sorry, the book you were looking for was not found'. The developer could not simply delete the words 'Sorry the book you were looking for was not found' because these would not then appear when the user entered an invalid search value. It should be stressed however that the function did work even with these imperfections.

6. Conclusion.

As work proceeded on this assignment it was realised that designing a workable programme can be challenging. Challenges were therefore confronted as they arose. It became apparent that progress was better achieved if certain principles were applied. These principles included keeping the code short so that it could be read easily. Repetition and duplication, whenever possible, were avoided as lengthy codes can be difficult to test, maintain and extend. The application of these principles make it more likely that the finalised programme will be user friendly. During the assignment a number of difficulties were resolved. The next step is to discuss these unresolved matters with the course tutor.

7. Link to the GitHub repository:

URL: https://github.com/Trix1k/LordOfTheRings_API

8. Main and Sub Menu code snippets and screenshots:

8.1 Code.

8.1.1 (part1)

```
# Sub menu for the application
# 3 Options here.
def PrintSubMenu():
    print("\n")
    print("-----")
    print("----- Sub Menu -----")
    print("-----")
    print("1. Print a list of chapters for The Fellowship Of The Ring")
    print("2. Print a list of chapters for The Two Towers")
    print("3. Print a list of chapters for The Return Of The King")
    print("-----")

# Main Menu for the application
# 8 options including the Exit option.
def PrintMenu():
    print("\n")
    print("-----")
    print("----- Main Menu -----")
    print("-----")
    print("1. Print a list of characters")
    print("2. Print a list of books")
    print("3. Print a list of movies")
    print("4. Print a list of quotes")
    print("5. Print a list of chapters")
    print("6. Search for a book")
    print("7. Print chapters of a book")
    print("8. Exit")
    print("-----")
    print("\n")
```

8.1.2 (part2)

```
bool = True

while bool:
    PrintMenu()
    choice = int(input("Please enter your desired option from the menu: "))

    if choice == 1:
        print("\nPrint a list of characters option has been selected.\n")
        ListOfCharacters()
    elif choice == 2:
        print("\nPrint a list of books option has been selected.\n")
        ListOfBooks()
    elif choice == 3:
        print("\nPrint a list of movies option has been selected.\n")
        ListOfMovies()
    elif choice == 4:
        print("\nPrint a list of quotes option has been selected.\n")
        ListOfQuotes()
    elif choice == 5:
        print("\nPrint a list of all book chapters option has been selected.\n")
        ListOfChapters()
    elif choice == 6:
        print("\nSearch for a book option has been selected.\n")
        SearchForBook()
    elif choice == 7:
        print("\nPrint chapters of a book option has been selected.\n")
        PrintSubMenu()
        subChoice = int(input("Please enter your option: "))
        if subChoice == 1:
            ListChaptersOfAbookA()
        elif subChoice == 2:
            ListChaptersOfAbookB()
        else:
            ListChaptersOfAbookC()
    elif choice == 8:
        bool = False
    else:
        input("\nThe option you've entered is invalid. Please try again: ")

print("\nThank you for using this programme.\n")
```

8.2 Running the code screenshot

```
-----  
----- Main Menu -----  
-----  
1. Print a list of characters  
2. Print a list of books  
3. Print a list of movies  
4. Print a list of quotes  
5. Print a list of chapters  
6. Search for a book  
7. Print chapters of a book  
8. Exit  
-----  
  
Please enter your desired option from the menu: 7  
Print chapters of a book option has been selected.  
  
-----  
----- Sub Menu -----  
-----  
1. Print a list of chapters for The Fellowship Of The Ring  
2. Print a list of chapters for The Two Towers  
3. Print a list of chapters for The Return Of The King  
-----  
Please enter your option:
```

9. List of Books code snippets and screenshots:

9.1 Code.

```
# Function to print out a list of books
def ListOfBooks():
    url = "https://the-one-api.dev/v2/book"

    # Including header to authorize the request with a bearer token.
    header = {
        "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",
        "Accept": "application/vnd.api+json"
    }

    # send a get request to this endpoint on the API.
    r = requests.get(url, headers=header)
    # store the content in a variable
    ListOfBooks = json.loads(r.content)

    # initiate for loop to print out the relevant data for this function
    for books in ListOfBooks['docs']:
        print("Book ID: {}".format(books['_id']))
        print("Name: {}\n".format(books['name']))
```

9.2 Running the code screenshot:

```
-----
----- Main Menu -----
-----
1. Print a list of characters
2. Print a list of books
3. Print a list of movies
4. Print a list of quotes
5. Print a list of chapters
6. Search for a book
7. Print chapters of a book
8. Exit
-----

Please enter your desired option from the menu: 2

Print a list of books option has been selected.

Book ID: 5cf5805fb53e011a64671582
Name: The Fellowship Of The Ring

Book ID: 5cf58077b53e011a64671583
Name: The Two Towers

Book ID: 5cf58080b53e011a64671584
Name: The Return Of The King
```

10. List of Movies code snippets and screenshots:

10.1 Code.

```
#function to print out a list of movies
def ListOfMovies():
    url = "https://the-one-api.dev/v2/movie"

    header = {
        "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",
        "Accept": "application/vnd.api+json"
    }

    r = requests.get(url, headers=header)
    MoviesList = json.loads(r.content)

    for movies in MoviesList['docs']:
        print("Movie ID: {}".format(movies['_id']))
        print("Name: {}".format(movies['name']))
        print("Movie Budget in Millions: {}".format(movies['budgetInMillions']))
        print("Academy Award Nominations: {}".format(movies['academyAwardNominations']))
        print("Academy Award Wins: {}".format(movies['academyAwardWins']))
        print("Rotten Tomatoes Score: {}\n".format(movies['rottenTomatoesScore']))
```

10.2 Running the code screenshot:

```
----- Main Menu -----
-----
1. Print a list of characters
2. Print a list of books
3. Print a list of movies
4. Print a list of quotes
5. Print a list of chapters
6. Search for a book
7. Print chapters of a book
8. Exit
-----

Please enter your desired option from the menu: 3

Print a list of movies option has been selected.

Movie ID: 5cd95395de30eff6ebccde56
Name: The Lord of the Rings Series
Movie Budget in Millions: 281
Academy Award Nominations: 30
Academy Award Wins: 17
Rotten Tomatoes Score: 94

Movie ID: 5cd95395de30eff6ebccde57
Name: The Hobbit Series
Movie Budget in Millions: 675
Academy Award Nominations: 7
Academy Award Wins: 1
Rotten Tomatoes Score: 66.33333333

Movie ID: 5cd95395de30eff6ebccde58
Name: The Unexpected Journey
Movie Budget in Millions: 200
Academy Award Nominations: 3
Academy Award Wins: 1
Rotten Tomatoes Score: 64

Movie ID: 5cd95395de30eff6ebccde59
Name: The Desolation of Smaug
Movie Budget in Millions: 217
Academy Award Nominations: 3
Academy Award Wins: 0
Rotten Tomatoes Score: 75

Movie ID: 5cd95395de30eff6ebccde5a
Name: The Battle of the Five Armies
Movie Budget in Millions: 250
Academy Award Nominations: 1
Academy Award Wins: 0
Rotten Tomatoes Score: 60

Movie ID: 5cd95395de30eff6ebccde5b
Name: The Two Towers
Movie Budget in Millions: 94
Academy Award Nominations: 6
Academy Award Wins: 2
Rotten Tomatoes Score: 96
```

11. List of Characters code snippets and screenshots:

11.1 Code.

```
#function to print out a list of characters
def ListOfCharacters():

    url = "https://the-one-api.dev/v2/character"

    header = {
        "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",
        "Accept": "application/vnd.api+json"
    }

    r = requests.get(url, headers=header)
    CharactersList = json.loads(r.content)

    for characters in CharactersList['docs']:
        try:
            print("Race: {}".format(characters['race']))
            print("Gender: {}".format(characters['gender']))
            print("Name: {}".format(characters['name']))
            print("Wiki URL: {}".format(characters['wikiUrl']))
            # create an exception if there is some data missing.
        except KeyError:
            print("Missing some information for this character. Moving on to the next one.\n")
```

11.2 Running the code screenshots:

11.2.1 (part1)

```
----- Main Menu -----
-----
1. Print a list of characters
2. Print a list of books
3. Print a list of movies
4. Print a list of quotes
5. Print a list of chapters
6. Search for a book
7. Print chapters of a book
8. Exit
-----

Please enter your desired option from the menu: 1

Print a list of characters option has been selected.

Race: Human
Gender: Female
Name: Adanel
Wiki URL: http://lotr.wikia.com/wiki/Adanel

Race: Human
Gender: Male
Name: Adrahil I
Wiki URL: http://lotr.wikia.com/wiki/Adrahil_I

Race: Human
Gender: Male
Name: Adrahil II
Wiki URL: http://lotr.wikia.com/wiki/Adrahil_II

Race: Elf
Gender: Male
Name: Aegnor
Wiki URL: http://lotr.wikia.com/wiki/Aegnor

Race: Human
Gender: Female
Name: Aerin
Wiki URL: http://lotr.wikia.com/wiki/Aerin
```

11.2.2 (part2)

Race: Wolfhound

Gender: Male

Name: Huan

Missing some information for this character. Moving on to the next one.

Race: Aínur

Gender: Female

Name: Estë

Missing some information for this character. Moving on to the next one.

Race: Men

Gender: Female

Name: Éowyn

Wiki URL: <http://lotr.wikia.com/wiki/%C3%89owyn>

Race: Men

Gender: Male

Name: Éomer

Wiki URL: <http://lotr.wikia.com/wiki/%C3%89omer>

Race: Half-elven

Gender: Female

Name: Elwing

Missing some information for this character. Moving on to the next one.

Race: Elves

Gender: Male

Name: Eluréd and Elurín

Missing some information for this character. Moving on to the next one.

Race: Half-elven

Gender: Male

Name: Elros

Missing some information for this character. Moving on to the next one.

12. List of Quotes code snippets and screenshots:

12.1 Code.

12.1.1 (part1)

```
# Function to print out a list of quotes
def ListOfQuotes():

    url = "https://the-one-api.dev/v2/quote"

    header = {
        "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",
        "Accept": "application/vnd.api+json"
    }

    r = requests.get(url, headers=header)
    QuotesList = json.loads(r.content)

    for quotes in QuotesList['docs']:
        print("Quote ID: {}".format(quotes['_id']))
        print("Dialogue: {}".format(quotes['dialog']))

        # At the current end point of the API it is only possible to print out a movie ID which is a long string, consisting of letters and numbers.
        # For this purpose, it is necessary to go into a different end point on the API to pull the relevant data.
        # Variable was created, and the value was given for that variable from movie ID in quotes['movie']
        movieID = quotes['movie']

        # specifying url (consisting of the URL link + movieID value) for the header of the request
        url = ("https://the-one-api.dev/v2/movie/" + movieID )

        # typical header consisting of authorization information
        header = {
            "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",
            "Accept": "application/vnd.api+json"
        }

        # sending a new get request to get that relevant data
        r1 = requests.get(url, headers=header)
        # storing the content in json format
        movieInfo = json.loads(r1.content)

        # for loop to print out the name of a movie
        for movie in movieInfo['docs']:
            print("Movie: {}".format(movie['name']))
```

12.1.2 (part2)

```
# This next part is the same process that was applied above to get the movie name.
# The only difference is that this time the purpose of this part of a code is to get some information about the character
charID = quotes['character']

url = ("https://the-one-api.dev/v2/character/" + charID )

header = {
    "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",
    "Accept": "application/vnd.api+json"
}

r2 = requests.get(url, headers=header)
charInfo = json.loads(r2.content)

for char in charInfo['docs']:
    print("Character Race: {}".format(char['race']))
    print("Character Name: {}\n".format(char['name']))
```

12.2 Running the code screenshots:

```
----- Main Menu -----  
-----  
1. Print a list of characters  
2. Print a list of books  
3. Print a list of movies  
4. Print a list of quotes  
5. Print a list of chapters  
6. Search for a book  
7. Print chapters of a book  
8. Exit  
-----  
  
Please enter your desired option from the menu: 4  
  
Print a list of quotes option has been selected.  
  
Quote ID: 5cd96e05de30eff6ebccea7e9  
Dialogue: Deagol!  
Movie: The Return of the King  
Character Race: Hobbit  
Character Name: Gollum  
  
Quote ID: 5cd96e05de30eff6ebccea7ea  
Dialogue: Deagol!  
Movie: The Return of the King  
Character Race: Hobbit  
Character Name: Gollum  
  
Quote ID: 5cd96e05de30eff6ebccea7eb  
Dialogue: Deagol!  
Movie: The Return of the King  
Character Race: Hobbit  
Character Name: Gollum  
  
Quote ID: 5cd96e05de30eff6ebccea7ec  
Dialogue: Give us that! Deagol my love  
Movie: The Return of the King  
Character Race: Hobbit  
Character Name: Gollum  
  
Quote ID: 5cd96e05de30eff6ebccea7ed  
Dialogue: Why?  
Movie: The Return of the King  
Character Race: Hobbit  
Character Name: Déagol  
  
Quote ID: 5cd96e05de30eff6ebccea7ee  
Dialogue: Because' , it's my birthday and I wants it.  
Movie: The Return of the King  
Character Race: Hobbit  
Character Name: Gollum  
  
Quote ID: 5cd96e05de30eff6ebccea7ef  
Dialogue: Arrghh!  
Movie: The Return of the King  
Character Race: Hobbit  
Character Name: Déagol
```

13. List of Chapters code and snippets:

13.1 Code.

Function to print out a list of chapters.

```
def ListOfChapters():

    url = "https://the-one-api.dev/v2/chapter"

    header = {
        "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",
        "Accept": "application/vnd.api+json"
    }

    r = requests.get(url, headers=header)
    ChaptersList = json.loads(r.content)

    for chapters in ChaptersList['docs']:
        print("Chapter ID: {}".format(chapters['_id']))
        print("Chapter name: {}".format(chapters['chapterName']))

        bookID = chapters['book']

        url = ("https://the-one-api.dev/v2/book/" + bookID )

        header = {
            "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",
            "Accept": "application/vnd.api+json"
        }

        r1 = requests.get(url, headers=header)
        bookInfo = json.loads(r1.content)

        for book in bookInfo['docs']:
            print("Name of a book: {}\n".format(book['name']))
```

13.2 Running the code screenshots:

13.2.1 (part1)

```
----- Main Menu -----
1. Print a list of characters
2. Print a list of books
3. Print a list of movies
4. Print a list of quotes
5. Print a list of chapters
6. Search for a book
7. Print chapters of a book
8. Exit

Please enter your desired option from the menu: 5

Print a list of all book chapters option has been selected.

Chapter ID: 5cdc25d4bc17e929cf2461eb
Chapter name: A Long-expected Party
Name of a book: The Fellowship Of The Ring

Chapter ID: 5cdc25d4bc17e929cf2461ec
Chapter name: The Shadow of the Past
Name of a book: The Fellowship Of The Ring

Chapter ID: 5cdc25d4bc17e929cf2461ed
Chapter name: Three is Company
Name of a book: The Fellowship Of The Ring

Chapter ID: 5cdc25d4bc17e929cf2461ee
Chapter name: A Short Cut to Mushrooms
Name of a book: The Fellowship Of The Ring

Chapter ID: 5cdc25d4bc17e929cf2461ef
Chapter name: In the House of Tom Bombadil
Name of a book: The Fellowship Of The Ring

Chapter ID: 5cdc25d4bc17e929cf2461f0
Chapter name: The Old Forest
Name of a book: The Fellowship Of The Ring

Chapter ID: 5cdc25d4bc17e929cf2461f1
Chapter name: A Conspiracy Unmasked
Name of a book: The Fellowship Of The Ring

Chapter ID: 5cdc25d5bc17e929cf2461f2
Chapter name: At the Sign of The Prancing Pony
Name of a book: The Fellowship Of The Ring

Chapter ID: 5cdc25d5bc17e929cf2461f3
Chapter name: Fog on the Barrow-Downs
Name of a book: The Fellowship Of The Ring

Chapter ID: 5cdc25d5bc17e929cf2461f4
Chapter name: Strider
Name of a book: The Fellowship Of The Ring

Chapter ID: 5cdc25d5bc17e929cf2461f5
Chapter name: Flight to the Ford
Name of a book: The Fellowship Of The Ring
```

13.2.2 (part2)

Chapter ID: 5cdc25d5bc17e929cf246213
Chapter name: The Window on the West
Name of a book: The Two Towers

Chapter ID: 5cdc25d5bc17e929cf246214
Chapter name: The Stairs of Cirith Ungol
Name of a book: The Two Towers

Chapter ID: 5cdc25d5bc17e929cf246215
Chapter name: Shelob's Lair
Name of a book: The Two Towers

Chapter ID: 5cdc25d5bc17e929cf246216
Chapter name: The Choices of Master Samwise
Name of a book: The Two Towers

Chapter ID: 5cdc25d5bc17e929cf246217
Chapter name: The Passing of the Grey Company
Name of a book: The Return Of The King

Chapter ID: 5cdc25d5bc17e929cf246218
Chapter name: The Muster of Rohan
Name of a book: The Return Of The King

Chapter ID: 5cdc25d5bc17e929cf246219
Chapter name: Minas Tirith
Name of a book: The Return Of The King

Chapter ID: 5cdc25d5bc17e929cf24621a
Chapter name: The Ride of the Rohirrim
Name of a book: The Return Of The King

Chapter ID: 5cdc25d5bc17e929cf24621b
Chapter name: The Battle of the Pelennor Fields
Name of a book: The Return Of The King

Chapter ID: 5cdc25d5bc17e929cf24621c
Chapter name: The Pyre of Denethor
Name of a book: The Return Of The King

Chapter ID: 5cdc25d5bc17e929cf24621d
Chapter name: The Siege of Gondor
Name of a book: The Return Of The King

Chapter ID: 5cdc25d5bc17e929cf24621e
Chapter name: The Houses of Healing
Name of a book: The Return Of The King

Chapter ID: 5cdc25d5bc17e929cf24621f
Chapter name: The Last Debate
Name of a book: The Return Of The King

Chapter ID: 5cdc25d5bc17e929cf246220
Chapter name: The Black Gate Opens
Name of a book: The Return Of The King

14. List Chapters of a Book function snippets and screenshots:

14.1 Code.

```
def ListChaptersOfAbookA():  
    url = "https://the-one-api.dev/v2/book/5cf5805fb53e011a64671582/chapter"  
  
    header = {  
        "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",  
        "Accept": "application/vnd.api+json"  
    }  
    r = requests.get(url, headers=header)  
    bookA = json.loads(r.content)  
  
    print("\nHere is a list of all chapters from The Fellowship Of The Ring:")  
    for chapters in bookA['docs']:  
        print(" ", chapters['chapterName'], sep = "-")
```

```
def ListChaptersOfAbookB():  
    url = "https://the-one-api.dev/v2/book/5cf58077b53e011a64671583/chapter"  
  
    header = {  
        "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",  
        "Accept": "application/vnd.api+json"  
    }  
    r = requests.get(url, headers=header)  
    bookB = json.loads(r.content)  
  
    print("\nHere is a list of all chapters from The Two Towers:")  
    for chapters in bookB['docs']:  
        print(" ", chapters['chapterName'], sep = "-")
```

```
def ListChaptersOfAbookC():  
    url = "https://the-one-api.dev/v2/book/5cf58080b53e011a64671584/chapter"  
  
    header = {  
        "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",  
        "Accept": "application/vnd.api+json"  
    }  
    r = requests.get(url, headers=header)  
    bookC = json.loads(r.content)  
  
    print("\nHere is a list of all chapters from The The Return Of The King:")  
    for chapters in bookC['docs']:  
        print(" ", chapters['chapterName'], sep = "-")
```

14.2 Running the code screenshot:

```
----- Sub Menu -----  
-----  
1. Print a list of chapters for The Fellowship Of The Ring  
2. Print a list of chapters for The Two Towers  
3. Print a list of chapters for The Return Of The King  
-----  
Please enter your option: 2  
  
Here is a list of all chapters from The Two Towers:  
- The Uruk-Hai  
- The Riders of Rohan  
- Treebeard  
- The White Rider  
- The Departure of Boromir  
- Helm's Deep  
- The King of the Golden Hall  
- The Road to Isengard  
- Flotsam and Jetsam  
- The Voice of Saruman  
- The Palantir  
- The Passage of the Marshes  
- The Taming of Smeagol  
- The Black Gate is Closed  
- Of Herbs and Stewed Rabbit  
- The Forbidden Pool  
- Journey to the Cross-roads  
- The Window on the West  
- The Stairs of Cirith Ungol  
- Shelob's Lair  
- The Choices of Master Samwise
```

15. Search for A Book function snippets and screenshots:

15.1 Code.

```
# Function to search for a book
def SearchForBook():

    url = "https://the-one-api.dev/v2/book"

    header = {
        "Authorization": "Bearer NoLE7ZRyH0Vy4xRdr2px",
        "Accept": "application/vnd.api+json"
    }

    bookID = input("Please enter the name of the book you want to find(Start with capital letter, otherwise use lowercase type!): ")

    r = requests.get(url, headers=header)
    bookList = json.loads(r.content)

    for books in bookList['docs']:

        if books['name'] == bookID:
            print("\nHere are the details about the book you were looking for: \n")
            print("Book ID: {}".format(books['_id']))
            print("Name of a Book: {}".format(books['name']))
            break
        else:
            continue

    print("\nSorry the book you were looking for was not found.")
```

15.2 Running the code screenshot:

```
Please enter your desired option from the menu: 6

Search for a book option has been selected.

Please enter the name of the book you want to find(Start with capital letter, otherwise use lowercase type!): The Fellowship Of The Ring

Here are the details about the book you were looking for:

Book ID: 5cf5805fb53e011a64671582
Name of a Book: The Fellowship Of The Ring

Sorry the book you were looking for was not found.
```

Reflections on creating a Python application consuming a RESTFUL web service (1801641)

GRADEMARK REPORT

FINAL GRADE

92/100

GENERAL COMMENTS

Instructor

Implementation (70): 66

Report (30): 26

A really excellent implementation, Jaromir. You've clearly put a ton of time and effort into this - lots of functionality and an application that goes well beyond the scope of what was expected for this assignment - really, really well done. Git has been used to great effect also.

A good report overall too - though as a minor comment, consider adding a tiny bit of narrative with the images (beyond the title) as there are so many, just to tie it together a little more. This is a very, very minor consideration, however.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22
