

# Python Project - My Library Database

Imran Hakim Roslan  
imrh@protonmail.com

January 30, 2021

## 1 Introduction

A Python application to monitor a PDF folder, analyze the PDF content for ISBN reference number, and interact with Google Books API to retrieve book's information with MariaDB act as a database to store the collected data.

## 2 Goals

Following are the main goals upon why the the application is created:

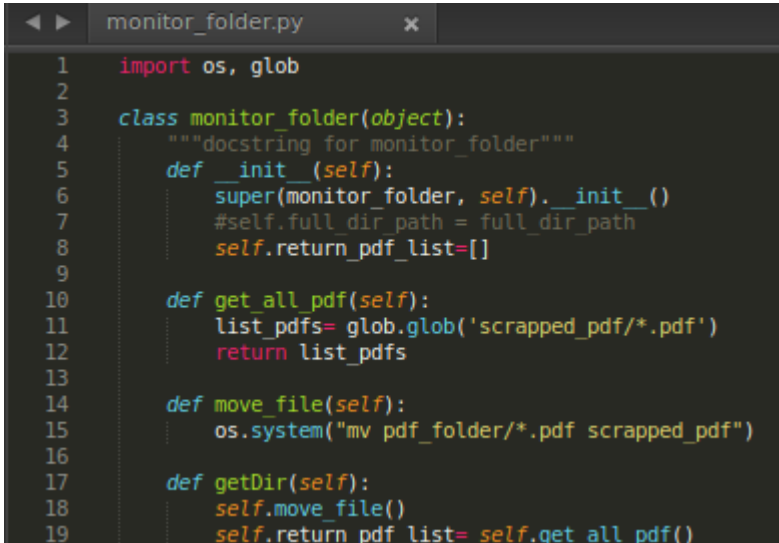
1. A personal library system to store e-books collection without having to worry about privacy concern of an entity collecting user data.
2. Able to analyze huge PDF or e-books collection and retrieve their respective information for references.
3. Able to effectively search, browse, and choose from the e-books collection that user personally want to read or refer to at any given time with no hassle of go through the collection one by one.
4. To easily store and make relational e-book category mapping for books recommendation.
5. Obtain the PDF path from storage for easy access.

## 3 Methods and Solutions

### 3.1 Problem 1: Monitoring Directory Path

A centralized folder or directory for the program to look for PDF file collections. Any folder path in system storage can be referred to. For simplicity, a folder named /pdf.folder in the program file directory was chosen.

The monitoring process procedures are being handled as below:




```
1 import os, glob
2
3 class monitor_folder(object):
4     """docstring for monitor_folder"""
5     def __init__(self):
6         super(monitor_folder, self).__init__()
7         #self.full_dir_path = full_dir_path
8         self.return_pdf_list=[]
9
10    def get_all_pdf(self):
11        list_pdfs= glob.glob('scrapped_pdf/*.pdf')
12        return list_pdfs
13
14    def move_file(self):
15        os.system("mv pdf_folder/*.pdf scrapped_pdf")
16
17    def getDir(self):
18        self.move_file()
19        self.return_pdf_list= self.get_all_pdf()
```

Figure 1: Monitoring a folder using Python

1. get\_all\_pdf() method is used to get all PDF document in the /pdf.folder directory.
2. getDir() will call move\_file() function to move the PDF document into scrapped\_pdf directory to indicate the document already been processed.

## 3.2 Problem 2: Extracting PDF Text Content

To analyze the PDF document content, two python libraries have been used, *pdfminer.six* and *PyPDF2*. Two libraries have been used to perform the text extraction were due to different PDF versions that require different approach of text extraction. More libraries can be added later as needed. ISBN extraction processes from PDF document are discussed below:



```
1 from pdfminer.high_level import extract_text
2 from PyPDF2 import PdfFileReader
3 import re, glob
4
5 class ISBN_class(object):
6     """docstring for ISBN_class"""
7     def __init__(self, path):
8         super(ISBN_class, self).__init__()
9         self.path = path
10        self.re_ISBN= re.compile('\d\d\d\d\-\?\d\-\?\d\d\d\d\-\?\d\d\d\d\-\?\d')
11
12    def get_all_pdf(self):
13        list_pdf_in_dir= glob.glob(str(self.path)+'/*.pdf')
14        return list_pdf_in_dir
15
16    def parse_ISBN_pdfminer(self, pdf): ...
17
18    def parse_ISBN_PyPDF2(self, pdf): ...
19
20    def normalize_ISBN(self, raw_ISBN):
21        return ISBN= raw_ISBN.replace('-', '')
22        return return_ISBN
23
24    def get_ISBN(self):
25        list_returned_ISBN=[]
26
27        if type(self.path) != list:
28            returned_ISBN= self.parse_ISBN_PyPDF2(self.path)
29            if returned_ISBN == None:
30                returned_ISBN= self.parse_ISBN_pdfminer(self.path)
31            if returned_ISBN == None:
32                print(f'Failed to get ISBN for {self.path} using available libraries')
33            else:
34                returned_ISBN= self.normalize_ISBN(returned_ISBN)
35            list_returned_ISBN.append(returned_ISBN)
36        else:
37            for pdf in self.path:
38                returned_ISBN= self.parse_ISBN_PyPDF2(pdf)
39                if returned_ISBN == None:
40                    returned_ISBN= self.parse_ISBN_pdfminer(pdf)
41                if returned_ISBN == None:
42                    print(f'Failed to get ISBN for {pdf} using available libraries')
43                else:
44                    returned_ISBN= self.normalize_ISBN(returned_ISBN)
45                list_returned_ISBN.append(returned_ISBN)
46
47        return list_returned_ISBN
```

Figure 2: Extracting ISBN from PDF document using Python

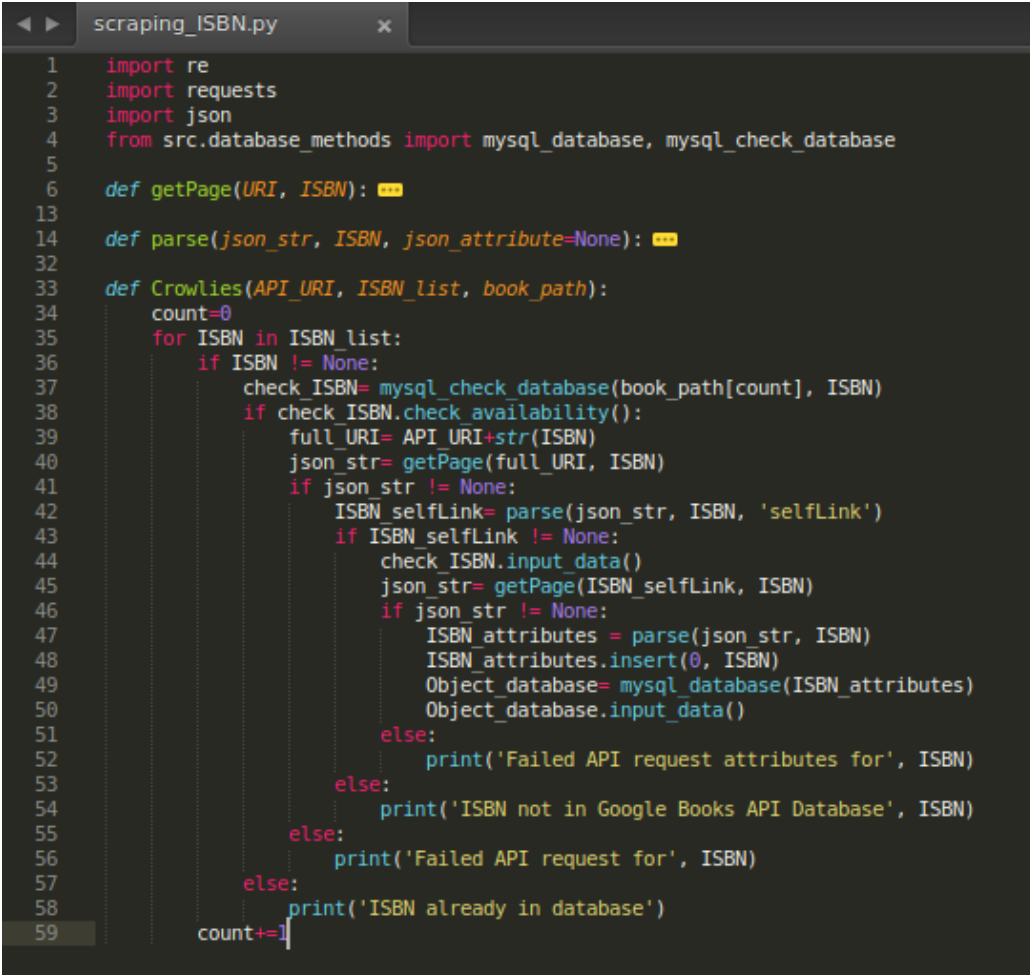
1. `get_ISBN()` method from `ISBN_class` class object in `extract_ISBN.py`

will instantiate the extraction process.

2. From `get_ISBN()` method, two PDF document text parsing method will be called to examine the document for ISBN value and returns discovered ISBN value if any.

### 3.3 Problem 3: Requesting Book's Info from an API

Google Books API has been used for collecting e-book details as the API contains vast majority of e-books repository and easily accessible without requiring subscription or API key authentication. The preceding steps show the e-books data collection through the API.



```
1  import re
2  import requests
3  import json
4  from src.database_methods import mysql_database, mysql_check_database
5
6  def getPage(URI, ISBN): ...
13
14  def parse(json_str, ISBN, json_attribute=None): ...
32
33  def Crawlies(API_URI, ISBN_list, book_path):
34      count=0
35      for ISBN in ISBN_list:
36          if ISBN != None:
37              check_ISBN= mysql_check_database(book_path[count], ISBN)
38              if check_ISBN.check_availability():
39                  full_URI= API_URI+str(ISBN)
40                  json_str= getPage(full_URI, ISBN)
41                  if json_str != None:
42                      ISBN_selfLink= parse(json_str, ISBN, 'selfLink')
43                      if ISBN_selfLink != None:
44                          check_ISBN.input_data()
45                          json_str= getPage(ISBN_selfLink, ISBN)
46                          if json_str != None:
47                              ISBN_attributes = parse(json_str, ISBN)
48                              ISBN_attributes.insert(0, ISBN)
49                              Object_database= mysql_database(ISBN_attributes)
50                              Object_database.input_data()
51                          else:
52                              print('Failed API request attributes for', ISBN)
53                      else:
54                          print('ISBN not in Google Books API Database', ISBN)
55                  else:
56                      print('Failed API request for', ISBN)
57              else:
58                  print('ISBN already in database')
59          count+=1
```

Figure 3: Requesting ISBN details from an API

1. Crowlies() method in scraping\_ISBN.py is responsible for initiating API request for each ISBN. Crowlies() method will accept three parameters, an API URI link path, ISBN values in a python list with their respective book path in the local system as the third parameter.
2. for loop in Crowlies() method will check for every ISBN value in the ISBN python list.
3. ISBN value checking will be made for each ISBN value in database before proceed with API request.
4. The API get request will be handled by getPage() method which expecting a json format and return in json data format.
5. Function parse() will then identify the needed data by the program in the json data format and returns the data in a python list.
6. After the e-book data being collected, database method is called to store the data.

### **3.4 Problem 4: Interfacing MariaDB Database with Python**

A python module pymysql is used to perform the python and MariaDB SQL database connection. database\_methods.py has been utilized for interfacing with MariaDB database and handling all database query methods. The database\_methods.py consists of two class object mysql\_database and check\_mysql\_database class. The latter class object mentioned inherits the mysql\_database class methods.

Due to complexity of the class objects in database\_methods.py, the connection of the database with python procedures are too lengthy to be discussed here. Please head to my github page at <https://github.com/imhak/mylib> for the code base.

## 4 Appendix

### A Dependencies

Environment dependencies:

1. python 3.x
2. MariaDB database

Below are extra python modules required for the application to run. Those can be installed using python pip module.

1. pdfminer.six
2. PyPDF2
3. requests
4. pymysql

### B Improvements

Clearly plenty improvements can be made to optimize the python application and more user friendly. Below are some improvements to name a few that can be made.

1. Develop a GUI to make the application user friendly.
2. Using machine learning to analyze PDF document and identify the PDF subject category without needing to rely on public API.
3. Using machine learning to better recommend user what books are available in user's library and related to search query in the database.
4. Provide server based service for the application to be accessible remotely.