**Документация к коду проекта «Библиотека»**

**Июль 2024 г.**

Оглавление

[1. Структура проекта 3](#_Toc172578326)

[2. Подробное описание кода 3](#_Toc172578327)

[2.1 Файл «main.py» 3](#_Toc172578328)

[2.2 Файл «processing\_main.py» 4](#_Toc172578329)

[2.2.1 Метод add\_book 4](#_Toc172578330)

[2.2.2 Метод delete\_book 5](#_Toc172578331)

[2.2.3 Метод search\_books 5](#_Toc172578332)

[2.2.4 Метод demonstration\_books 6](#_Toc172578333)

[2.2.5 Метод receiving\_issuing\_book 6](#_Toc172578334)

[2.3 Файл «processing\_json.py» 6](#_Toc172578335)

[2.3.1 Метод write\_json 7](#_Toc172578336)

[2.3.2 Метод read\_json 7](#_Toc172578337)

[2.3.3 Класс Books 7](#_Toc172578338)

[2.3.3.3 Метод search\_id 8](#_Toc172578339)

[2.3.3.4 Метод search\_books 8](#_Toc172578340)

[2.3.3.5 Метод receiving\_issuing\_book 9](#_Toc172578341)

[2.4 Файл «models.py» 9](#_Toc172578342)

[2.4.1 Метод check\_status 10](#_Toc172578343)

[2.4.2 Класс Book 10](#_Toc172578344)

[2.4.2.4 Метод book\_dict 11](#_Toc172578345)

[2.4.2.5 Метод demonstration 11](#_Toc172578346)

[2.5 Файл «constants.py» 11](#_Toc172578347)

[3. Как настроить проект 11](#_Toc172578348)

[4. Как запустить проект 12](#_Toc172578349)

[5. Как взаимодействовать с приложением 12](#_Toc172578350)

# 1. Структура проекта

В проекте «Библиотека» находится пять python файлов: «main.py» - основной файл необходимый для запуска и остановки приложения, а также распределения команд пользователя по нужным методам; «processing\_main.py» - файл для реализации главных функций приложения; «processing\_json.py» - файл для реализации хранения данных в формате json (т.к. проект реализован без сторонних библиотек), также в этом файле реализован класс Books; «models.py» - файл содержащий класс Book.

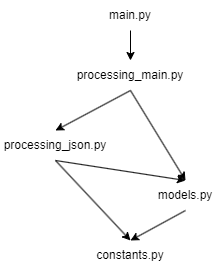


Рисунок 1 Диаграмма наследования файлов в проекте "Библиотека"

На рисунке 1 представлена диаграмма наследования файлов в проекте «Библиотека», кроме представленных на ней файлов в проекте присутствуют два файла отвечающие за хранение данных приложения: «current\_id.txt» - содержит в себе одно число, которое отвечает за уникальный идентификатор следующей добавляемой книги; «books.json» - содержит в себе информацию о книгах в формате json.

# 2. Подробное описание кода

## 2.1 Файл «main.py»

В файле «main.py» присутствует один метод main\_menu, представленный ниже.

def main\_menu(stage: bool) -> None:  
 if stage:  
 print("Добавление книги - 1, удаление книги - 2, поиск книги - 3, отображение всех книг - 4, изменение "  
 "статуса книги - 5\nВведите код: ")  
 else:  
 print("Добавление книги - 1, удаление книги - 2, поиск книги - 3, отображение всех книг - 4, изменение "  
 "статуса книги - 5, завершения программы - 6\nВведите код: ")  
 command: str or int = input()  
 if command.isdigit():  
 command = int(command)  
 match command:  
 case 1:  
 add\_book()  
 case 2:  
 delete\_book()  
 case 3:  
 search\_books()  
 case 4:  
 demonstration\_books()  
 case 5:  
 receiving\_issuing\_book()  
 case 6:  
 print("Программа завершенна. Спасибо за обращение, будем ждать Вас снова!")  
 global final  
 final = True  
 case \_:  
 print("К сожалению, нет такой команды. Прочтите инструкцию и обратитесь снова. Будем ждать!")

Этот метод представляет из себя основное меню, которое сообщает пользователю, что он может сделать. После ввода пользователем интересующей его команды основное меню запускает соответствующий метод.

Переменная stage – отвечает за первое или повторное включение, при первом включение основное меню не предлагает пользователю завершить работу. Переменная command – отвечает за введенную пользователем команду.

Кроме метода main\_menu в файле «main.py» присутствует свободный код, представленный ниже.

if \_\_name\_\_ == '\_\_main\_\_':  
 print('Добро пожаловать в главное меню приложения "Библиотека"!\nЧто вы желаете сделать?')  
 k: bool = True  
 final: bool = False  
 while True:  
 if final:  
 break  
 main\_menu(k)  
 k = False

Этот код отвечает за запуск приложения и его зацикливание, таким образом приложение будет работать пока пользователь не пожелает завершить его работу.

Переменная k – отвечает за первое включение, изначально принимает значение True. Переменная final – отвечает за конец работы, изначально принимает значение False.

## 2.2 Файл «processing\_main.py»

В файле «processing\_main.py» присутствует пять методов, которые отвечают за основные функции приложения: добавление новой книги, удаление книги, поиск книги, отображение всех книг и изменение статуса книги.

### 2.2.1 Метод add\_book

Метод add\_book используется для добавления новой книги и представлен ниже.

def add\_book() -> None:  
 global current\_books  
 print('Для добавления новой книги введите информацию о ней.\nВведите название книги: ')  
 current\_book\_title: str = input()  
 print('Введите автора книги: ')  
 current\_book\_author: str = input()  
 print('Введите год издания книги (только число): ')  
 current\_book\_year: str or int = input()  
 if current\_book\_year.isdigit():  
 current\_book\_year = int(current\_book\_year)  
 current\_book: Book = Book(current\_book\_title, current\_book\_author, current\_book\_year)  
 errors: set[str] = current\_book.check()  
 if errors:  
 print(f'Найденно {len(errors)} ошибок:')  
 for error in errors:  
 print(error)  
 else:  
 current\_books.add\_book(current\_book)  
 print('Книга добавлена! Спасибо за обращение!')

Этот метод сообщает, что пользователю необходимо ввести для добавления книги, после получения всех данных создает объект книги и проверяет ее. Если были найдены ошибки – они сообщаются пользователю, иначе – сообщается, что книга добавлена. Ошибки не прерывают работу приложения.

### 2.2.2 Метод delete\_book

Метод delete\_book используется для удаления книги и представлен ниже.

def delete\_book() -> None:  
 global current\_books  
 print('Для удаления книги введите ее уникальный идентификатор (id): ')  
 current\_book\_id: str or int = input()  
 if current\_book\_id.isdigit():  
 current\_book\_id = int(current\_book\_id)  
 if current\_books.delete\_book(current\_book\_id):  
 print('Книга удалена! Спасибо за обращение!')  
 else:  
 print('Ошибка! Книга не может быть удалена! Проверьте введеный id и попробуйте снова.\nБудем ждать вашего '  
 'обращения!')

Этот метод сообщает, что пользователю необходимо ввести для удаления книги, после получения всех данных запускает метод удаления книги. И проверяет была ли найдена искомая книга. Если книга была найдена – пользователю сообщается что книга удалена, иначе – сообщается, об ошибке. Ошибка не прерывают работу приложения.

### 2.2.3 Метод search\_books

Метод search\_books используется для поиска книг и представлен ниже.

def search\_books() -> None:  
 global current\_books  
 print('Для поиска книги введите информацию о ней. Если информация Вам не известна - пропустите шаг, нажав клавишу '  
 '"Enter"\nВведите название книги или нажмите "Enter": ')  
 current\_book\_title: str = input()  
 print('Введите автора книги или нажмите "Enter": ')  
 current\_book\_author: str = input()  
 print('Введите год издания книги (только число) или нажмите "Enter": ')  
 current\_book\_year: str or int = input()  
 if current\_book\_year.isdigit():  
 current\_book\_year = int(current\_book\_year)  
 print('По вашему запросу найдены следующие книги:')  
 count\_books = current\_books.search\_books(current\_book\_title, current\_book\_author, current\_book\_year)  
 print(f'Всего книг найдено - {count\_books}. Спасибо за обращение!')

Этот метод сообщает, что пользователю необходимо ввести для поиска книги, после получения всех данных запускает метод поиска книг. После чего выводит общее количество книг, подходящих под описание, введённое пользователем.

### 2.2.4 Метод demonstration\_books

Метод demonstration\_books используется для отображения всех книг и представлен ниже.

def demonstration\_books() -> None:  
 global current\_books  
 print(f'В библиотеке всего {len(current\_books.books\_list)} книг:')  
 for book in current\_books.books\_list:  
 print(book.demonstration())  
 print('Спасибо за обращение!')

Этот метод сообщает количество книг, принадлежащих библиотеке и выводит их в столбик.

### 2.2.5 Метод receiving\_issuing\_book

Метод receiving\_issuing\_book используется для изменения статуса книги и представлен ниже.

def receiving\_issuing\_book() -> None:  
 global current\_books  
 print('Для изменения статуса книги введите необходимую информацию.\nВведите id книги: ')  
 current\_book\_id: str or int = input()  
 if current\_book\_id.isdigit():  
 current\_book\_id = int(current\_book\_id)  
 print('Введите новый статус книги: ')  
 current\_book\_status: str = input()  
 flag: bool or None = current\_books.receiving\_issuing\_book(current\_book\_id, current\_book\_status)  
 if isinstance(flag, type(None)):  
 print('Ошибка! Введен некорректный статус книги. Проверьте введеный статус и попробуйте снова.\nБудем ждать '  
 'вашего обращения!')  
 elif flag:  
 print(f'Статус книги изменен на {current\_book\_status}! Спасибо за обращение!')  
 else:  
 print('Ошибка! Введен некорректный id книги. Проверьте введеный id и попробуйте снова.\nБудем ждать вашего '  
 'обращения!')

Этот метод сообщает, что пользователю необходимо ввести для изменения статуса книги, после получения всех данных запускает метод изменения статуса книги. И проверяет была ли найдена искомая книга и если да, то правильно ли указан новый статус. Если книга была найдена и новый статус указан правильно – пользователю сообщается, что статус книги изменен, иначе – сообщается, об ошибке. Ошибка не прерывают работу приложения.

## 2.3 Файл «processing\_json.py»

В файле «processing\_json.py» присутствует два метода, которые отвечают за работу с json файлом, то есть считывание и запись данных; класс Books – для работы с множеством книг в библиотеке и свободный код, представленный ниже.

try:  
 f1 = open(constants.books)  
 f1.close()  
except FileNotFoundError:  
 raise FileNotFoundError(f"Ошибка! Файл {constants.books} не найден!")

Этот код служит для проверки наличия файла. Ошибка прерывает работу приложения, т.к. без доступа к файлу с данными приложение работать не сможет.

### 2.3.1 Метод write\_json

Метод write\_json используется для записи информации в json файл и представлен ниже.

def write\_json(json\_obj: str) -> None:  
 with open(constants.books, 'wb') as jsonFile:  
 jsonFile.write(bytes(json\_obj, 'utf-8'))

Этот метод получает информацию в формате строки, после чего открывает json файл и записывает в него полученную строку.

### 2.3.2 Метод read\_json

Метод read\_json используется для считывания информации из json файла и представлен ниже.

def read\_json() -> list and str:  
 with open(constants.books, 'rb') as jsonFile:  
 json\_obj: str = jsonFile.read().decode('utf-8')  
 str\_line: str = json\_obj.replace('[', '').replace(']', '')  
 number\_lines: int = str\_line.count('}')  
 list\_line: list = list()  
 if number\_lines:  
 for number in range(number\_lines):  
 line: str = str\_line[str\_line.find('{') + 1:str\_line.find('}')]  
 dict\_line: dict = dict()  
 while line:  
 name: str = line[line.find('"') + 1:line.find(':') - 1]  
 line = line[line.find(':') + 2:]  
 if line[0] == '"':  
 line = line[1:]  
 dict\_line[name] = line[:line.find('"')]  
 line = line[line.find('"') + 1:]  
 else:  
 if line.find(',') == -1:  
 dict\_line[name] = int(line)  
 break  
 else:  
 dict\_line[name] = int(line[:line.find(',')])  
 list\_line.append(Book(data=dict\_line))  
 str\_line = str\_line[str\_line.find('}') + 1:]  
  
 return list\_line, json\_obj

Этот метод открывает json файл и считывает хранящийся в нем текст, после чего текст подвергается обработке для преобразования в список объектов класса Book. В итоге метод возвращает считанную информацию в двух форматах – в формате списка объектов класса Book и строковом. При внешнем вмешательстве в содержимое файла – возможна некорректная обработка.

### 2.3.3 Класс Books

Класс Books представляет собой класс множества книг и при инициализации определяет две переменные books\_str и books\_list, множество книг в формате строки и списка соответственно, с помощью метода считывания информации из json файла. Данный класс имеет пять функций: добавление новой книги в множество книг, удаление книги из множества книг, поиск книги в множестве книг (по уникальному идентификатору), поиск книг в множестве книг (по названию, автору и/или году издания) и изменение статуса одной книги из множества книг.

#### 2.3.3.1 Метод add\_book

Метод add\_book используется для добавления новой книги в множество и представлен ниже.

def add\_book(self, book: Book) -> None:  
 self.books\_list.append(book)  
 data: dict or str = book.book\_dict()  
 data = str(data).replace('\'', '"')  
 if len(self.books\_str) > 2 and self.books\_str[0] == '[':  
 self.books\_str = f"{self.books\_str[:-1]}, {data}]"  
 else:  
 self.books\_str = f"[{data}]"  
 write\_json(self.books\_str)

Этот метод получает объект Book добавляет его в список объектов book (books\_list), после чего преобразует в строку, объединяет со строкой множества книг (books\_str) и записывает в json файл.

#### 2.3.3.2 Метод delete\_book

Метод delete\_book используется для удаления одной книги из множества и представлен ниже.

def delete\_book(self, book\_id: int) -> bool:  
 flag: bool  
 book: Book  
 flag, book = Books.search\_id(self, book\_id)  
 if not flag:  
 return flag  
 old\_book: str = str(book.book\_dict())  
 self.books\_str.replace(old\_book, old\_book + '||').replace(old\_book, '').replace('||, ', '').replace(', ||', '')  
 self.books\_list.pop(self.books\_list.index(book))  
 write\_json(self.books\_str)  
 return flag

Этот метод получает уникальный идентификатор книги, которую необходимо удалить, после чего ищет эту книгу в множестве книг с помощью готового метода (2.3.3.3). Если книга не найдена – метод возвращает значение False, иначе – преобразует объект Book книги, которую необходимо удалить, в строковый формат и несколько раз преобразует множество книг в строковом формате для удаления книги без нарушения структцры json формата. Далее удаляет книгу из множества книг в формате списка объектов Book, записывает в json файл новое содержимое (без удаленной книги) и возвращает значение True.

### 2.3.3.3 Метод search\_id

Метод search\_id используется для поиска одной книги в множестве по ее уникальному идентификатору и представлен ниже.

def search\_id(self, book\_id: int) -> bool and Book:  
 flag: bool = False  
 book: Book or None = None  
 for book in self.books\_list:  
 if book.id == book\_id:  
 flag = True  
 break  
 return flag, book

Этот метод получает уникальный идентификатор книги после чего проходит по списку объектов Book и сверяет id. Если искомая книга нашлась, то поиски прекращаются и возвращаются значения True и объект класса Book, иначе – возвращаются значения False и None.

### 2.3.3.4 Метод search\_books

Метод search\_books используется для поиска нескольких книг в множестве и представлен ниже.

def search\_books(self, book\_title: str, book\_author: str, book\_year: int) -> int:  
 found\_books: int = 0  
 for book in self.books\_list:  
 if book\_title:  
 if book.title != book\_title:  
 continue  
 if book\_author:  
 if book.author != book\_author:  
 continue  
 if book\_year:  
 if book.year != book\_year:  
 continue  
 found\_books += 1  
 print(book.demonstration())  
 return found\_books

Этот метод получает название, автора и год издания книги и определяет счетчик найденных книг. После чего проходит по всем книгам в формате объектов класса Book и при наличии параметра сверяет его с книжным. Если хоть один параметр не совпал – переходит к просмотру следующей книги, иначе – увеличивает счетчик и выводит информацию о книги на экран (найденные книги никуда не сохраняются, т.к. нет необходимости). В конце метод возвращает количество найденных книг.

### 2.3.3.5 Метод receiving\_issuing\_book

Метод receiving\_issuing\_book используется для изменения статуса одной книги из множества и представлен ниже.

def receiving\_issuing\_book(self, book\_id: int, status: str) -> bool or None:  
 flag: bool  
 book: Book  
 flag, book = Books.search\_id(self, book\_id)  
 if not flag:  
 return flag  
 index: int = self.books\_list.index(book)  
 old\_book: str = str(book.book\_dict()).replace('\'', '"')  
 if not book.receiving\_issuing(status):  
 return None  
 self.books\_list[index] = book  
 new\_book: str = str(book.book\_dict()).replace('\'', '"')  
 write\_json(self.books\_str.replace(old\_book, new\_book))  
 return flag

Этот метод получает уникальный идентификатор и новый статус книги, которую необходимо изменить, после чего ищет эту книгу в множестве книг с помощью готового метода (2.3.3.3). Если книга не найдена – метод возвращает значение False, иначе - определяет старую книгу в формате строки и ее индекс в списке объектов Book. Дальше изменяет статус книги и проверяет корректность нового статуса, если он некорректен – метод возвращает значение None, иначе – в списке объектов класса Book заменяет старую книгу на новую, определяет новую книгу в формате строки и записывает в json файл строку, в которой заменяет старую книгу на новую. В конце возвращает значение True.

## 2.4 Файл «models.py»

В файле «models.py» присутствует метод, которые отвечают за проверку корректности статуса книги, класс Book для определения класса одной книги в библиотеке и свободный код, представленный ниже.

try:  
 f1 = open(constants.current\_id)  
 f1.close()  
except FileNotFoundError:  
 raise FileNotFoundError(f"Ошибка! Файл {constants.current\_id} не найден!")

Этот код служит для проверки наличия файла. Ошибка прерывает работу приложения, т.к. без доступа к файлу с данными приложение работать не сможет.

### 2.4.1 Метод check\_status

Метод check\_status используется для проверки корректной записи статуса книги и представлен ниже.

def check\_status(status: str) -> str or None:  
 if status not in {'в наличии', 'выдана'}:  
 return "Некорректный статус книги"  
 return None

Этот метод получает информацию в формате строки, после чего проверяет вхождение строки в множество возможных значений статуса. Возвращает либо ошибку, либо None (если все нормально). Ошибка не прерывает работу приложения.

### 2.4.2 Класс Book

Класс Book представляет собой класс одной книги и при инициализации определяет параметры книги: уникальный идентификатор, название, автора, год издание и статус. Инициализировать объект класса можно как новый – по названию, автору и году издания, так и старый – по словарю, в котором находятся: уникальный идентификатор, название, автора, год издание и статус. Для определения уникального идентификатора для новых объектов класс до инициализации считывает максимальный неиспользованный id из текстового файла.

Данный класс имеет пять функций: проверка корректности года издания книги, проверка книги на корректность данных, изменение статуса книги, преобразование книги из объекта класса Book в словарь и преобразование книги из объекта класса Book в строку для красивой демонстрации пользователю.

#### 2.4.2.1 Метод check\_year

Метод check\_year используется для проверки корректности года издания книги и представлен ниже.

def check\_year(self) -> str or None:  
 if not isinstance(self.year, int) or self.year < 1000 or self.year > 10000:  
 return "Некорректный год издания книги"  
 return None

Этот метод проверяет, что год издания книги является числом и состоит из четырех цифр. Возвращает либо ошибку, либо None (если все нормально). Ошибка не прерывает работу приложения.

#### 2.4.2.2 Метод check

Метод check используется для удаления одной книги из множества и представлен ниже.

def check(self) -> set:  
 error: set[str or None] = set()  
 error.add(Book.check\_year(self))  
 error.add(check\_status(self.status))  
 error.discard(None)  
 return error

Этот метод определяет множество ошибок, запускает проверки года издания и статуса книги, удаляет значение None и возвращает множество ошибок.

#### 2.4.2.3 Метод receiving\_issuing

Метод receiving\_issuing используется для изменения статуса книги и представлен ниже.

def receiving\_issuing(self, status: str) -> bool:  
 check: str or None = check\_status(status)  
 if check:  
 return False  
 self.status = status  
 return True

Этот метод получает статус в формате строки и проверяет его на корректность. Если новый статус корректен – изменяет у объекта статус и возвращает значение True, иначе возвращает значение False.

### 2.4.2.4 Метод book\_dict

Метод book\_dict используется для преобразования книги из объекта класса Book в словарь и представлен ниже.

def book\_dict(self) -> dict:  
 return {'id': self.id, 'title': self.title, 'author': self.author, 'year': self.year, 'status': self.status}

Этот метод получает преобразует книгу в формат словаря с ключами: id, title, author, year и status. Метод возвращает словарь.

### 2.4.2.5 Метод demonstration

Метод demonstration используется для преобразования книги из объекта класса Book в строку для красивой демонстрации пользователю и представлен ниже.

def demonstration(self) -> str:  
 return f'Книга №{self.id}. {self.author} "{self.title}" {self.year} года издания - {self.status}'

Этот метод преобразует книгу в формат строки, которая содержит в себе всю информацию о книги в приятном для чтения варианте. Метод возвращает строку.

## 2.5 Файл «constants.py»

В файле «constants.py» хранится две переменный current\_id и books, каждая приравнена строке – названию файла с данными. Current\_id – текстовый файл, содержащий одно число – максимальный не использованный id, books – json файл, содержащий информацию о всех книгах библиотеки в формате json.

# 3. Как настроить проект

1. Скачайте и установите python 3.10 (или более новой версии).
2. Убедитесь, что скачали данный проект (library).
3. Проверьте файлы хранения данных. Файл books.json должен хранить перечень книг (если приложение уже использовалось) или быть пустым (если приложение еще не использовалось). Файл current\_id.txt хранит в себе одно число - минимальный не использованный номер для идентификации книг (в самом начале это 0).

# 4. Как запустить проект

1. Откройте приложение Командная строка. Перейдите в папку library, для этого напишите команду cd и путь к папке library.
2. Запустите приложение, для этого напишите команду python main.py.
3. Готово - приложение запущенно.

# 5. Как взаимодействовать с приложением

Интерфейс приложения максимально удобен, поэтому после запуска приложения – приложение само объясняет вам как с ним взаимодействовать. Для ввода данных необходима клавиатура, набираете необходимый текст и нажимаете клавишу «Enter». Удачи!