НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ ТА СПЕЦІАЛІЗОВАНИХ КОМП’ЮТЕРНИХ СИСТЕМ

**Лабораторна робота №2  
з дисципліни «Архітектура комп’ютерів»**

**Варіант 7**

Виконали  
студенти 3-го курсу  
групи КВ-41  
*Яковенко Максим  
Курач Віктор  
Горпинич-Радуженко Іван*

Київ – 2017

**Постановка задачі**

1. Для програмних засобів, які було розроблено в лабораторній роботі №1 додати можливість збереження даних, що оброблюються, у текстовий файл. Передбачити можливість серіалізації та десеріалізації у форматах pickle, json, yaml. Для підтримки yaml скористатися бібліотекою pyyaml.

2. Для кожного способу серіалізації створити окремий модуль з уніфікованим інтерфейсом. Забеспечити кожний модуль документуючими коментарями. Для доступу до файлів використовувати контект­менеджери. Створити файл конфігурації та, використовуючи модуль configparser, розробити засоби для вибору способу серіалізації.

3. Створити додатковий модуль, який використовує для роботи з файлами клас io.StringIO. На основі цього модуля розробити модульні тести для засобів серіалізації.

4. Розроблений програмний код повинен складатися з окремих функцій, кожна з яких має не більше 10 рядків та не більше 3 рівнів вкладеності. За допомогою засобів статичного аналізу pep8 та pyflakes забезпечити сумісність програмного коду із стандартами кодування.

5. Розмістити розроблені програмні засоби у публічному сховищі програмного коду (github, bitbucket тощо).

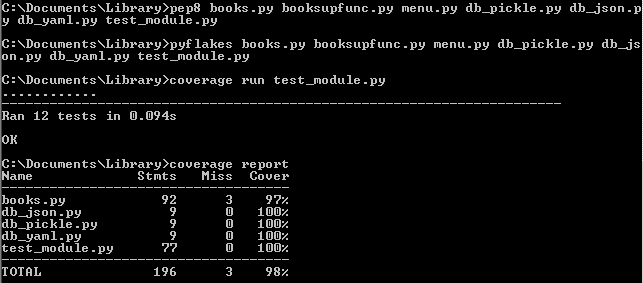
**Завдання для даного варіанту**

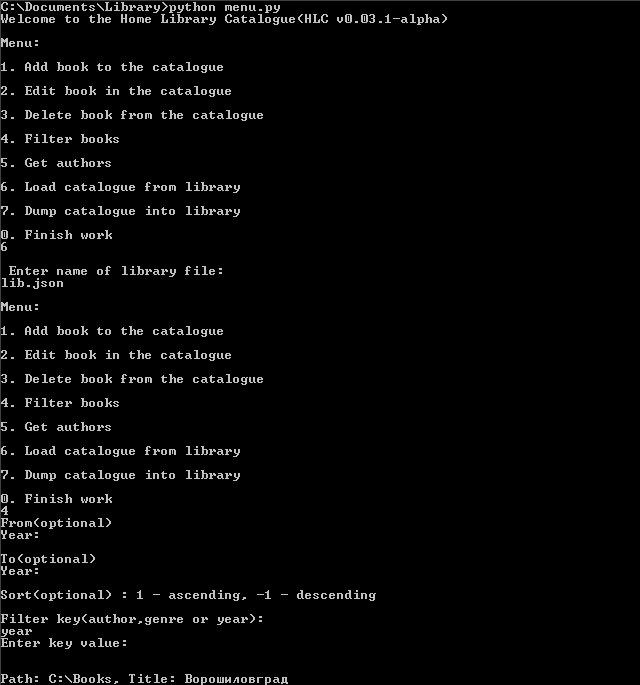
Каталог домашньої бібліотеки.

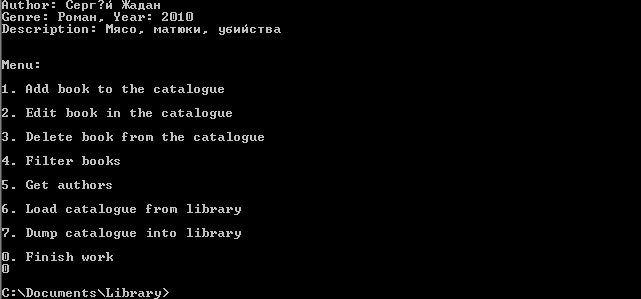
**Посилання на репозиторій**

<https://github.com/ViktorKurach/Library-Catalogue/>

**Перевірка відповідності стандартам та тестування**

****

**Приклад роботи програми**

****

**Документація**

Help on module books:

NAME

books

FUNCTIONS

add\_book(catalogue, new)

Adds new book to the catalogue.

:param catalogue: a list of books, which new book is added to.

:param new: a dictionary with next keys: 'path', 'title', 'author',

'publication', 'genre', 'description', 'year'.

:returns 1 if the book is already in catalogue, or 0 otherwise.

delete\_book(catalogue, title, author)

Removes a book from the catalogue.

:param catalogue: a list of books.

:param title: a string - title of the book, which will be deleted.

:param author: a string - author of the book, which will be deleted.

:returns 0 in case of success, or 1, if the book is not found in catalogue.

dump\_library(file, catalogue, test\_mode=False)

Dumps list of books into .pkl, .json or .yaml file.

:param file: a file object; file must be opened in "wb" mode, if it is

.pkl, or in "w" mode if it is .json or .yaml; file type must be the same,

as set in 'db\_file\_type' option in 'config.cfg'. Otherwise function's work

is unpredictable.

:param catalogue: a list of books.

:param test\_mode: if True, 'test\_file\_type' option instead of

'db\_file\_type' is used.

:returns catalogue in case of success.

edit\_book(catalogue, old\_title, old\_author, new)

Edits a book in the catalogue.

:param catalogue: a list of books.

:param old\_title: a string - title of the book, which will be edited.

:param old\_author: a string - author of the book, which will be edited.

:param new: a dictionary, containing new information about the book;

must have keys: 'path', 'title', 'author', 'publication', 'genre',

'description', 'year'.

:returns new book in case of success, or nothing, if old book is not

found in the catalogue.

filter\_books(catalogue, key, value, year\_from=None, year\_to=None, desc=0)

Filters books by the key and/or publishing year and sorts them.

:param catalogue: a list of books, that is filtered.

:param key: "genre", "author", or "year".

:param value: a string, genre's name, if key="genre"; a string, author's

name, if key="author"; if key="year", any value is allowed.

:param year\_from: an integer, lower limit of publishing years' diapason;

if year\_from=None, filtering by year is not performed.

:param year\_to: an integer, higher limit of publishing year's diapason;

if year\_to=None and year\_from is not None, searches only books, published

in year\_from; if year\_from=None, any year\_to is allowed.

:param desc: if 1, sorts selected books by publishing year increasing,

if -1 - by publishing year decreasing; otherwise sorting is not performed.

:returns a filtered and sorted list of books.

filter\_by\_year(book\_list, year\_from, year\_to, desc)

Filters books by publishing year and/or sorts them.

:param book\_list: a list of books, that is filtered.

:param year\_from: an integer, lower limit of publishing years' diapason;

if year\_from=None, filtering is not performed.

:param year\_to: an integer, higher limit of publishing year's diapason;

if year\_to=None and year\_from is not None, searches only books, published

in year\_from; if year\_from=None, any year\_to is allowed.

:param desc: if 1, sorts selected books by publishing year increasing,

if -1 - by publishing year decreasing; otherwise sorting is not performed.

:returns a filtered and sorted list of books.

In next versions function will be private: usage outside this module is

not recommended.

get\_authors(catalogue, genre=None)

Selects authors, whose books relate to genre.

:param catalogue: a list of books.

:param genre: a string - name of the genre, authors must relate to;

if None, returns all of the authors in catalogue.

:returns list of strings - authors' names.

get\_config(filename, section, option)

Parses .cfg file and gets some value from it.

:param filename: a string - file name.

:param section: a string - name of section.

:param option: a string - name of option.

:returns value of option in section in case of success, or nothing

otherwise.

import\_db\_module(file\_type)

Imports and returns one of db\_xxx modules depending on file\_type.

:param file\_type: "pkl", "json" or "yaml"

:return: if file\_type="pkl", returns module db\_pickle; if file\_type="json",

returns module db\_json; if file\_type="yaml", returns module db\_yaml.

In next versions function will be private: usage outside this module is not

recommended.

load\_library(file, test\_mode=False)

Loads a book list from .pkl, .json or .yaml file.

:param file: a file object; file must be opened in "rb" mode, and be the

same type, as set in 'db\_file\_type' option in 'config.cfg'. Otherwise

function's work is unpredictable.

:param test\_mode: if True, 'test\_file\_type' option instead of

'db\_file\_type' is used.

:returns loaded list of books in case of success.

search\_book(catalogue, title, author)

Searches a book in the catalogue.

:param catalogue: a list of books, which the search is performed in.

:param title: a string - title of the book, which is searched.

:param author: a string - name of author of the book, which is searched.

:returns a dictionary, corresponding to searched book, or nothing, if book

is not found.

set\_config(filename, section, option, new\_value)

Parses .cfg file and sets new value for some option.

:param filename: a string - file name.

:param section: a string - name of section.

:param option: a string - name of option.

:param new\_value: a new value of option in section.

:returns new\_value in case of success, or nothing otherwise.

sort\_by\_year(book\_list, desc)

Sorts a list of books by publishing year.

:param book\_list: a list of books, that is sorted.

:param desc: if 1, sorts books by publishing year increasing, if -1 - by

publishing year decreasing; otherwise sorting is not performed.

:returns a sorted list of books.

In next versions function will be private: usage outside this module is

not recommended.

FILE

c:\users\jakov\pycharmprojects\labtest\books.py

NAME

booksupfunc

FUNCTIONS

dump\_into\_library()

Dump current book catalogue into inputted library file

get\_author\_name()

Get book author name value;

:returns standard python function input

get\_book()

Create new dictionary element by using get\_xxx functions;

:return dictionary element that contains book info

get\_book\_name()

Get book name value;

:returns standard python function input

get\_description()

Get book description value;

:returns standard python function input

get\_genre()

Get genre name

:returns: None: if genre is not containing

any symbol or containing only whitespaces

OR genre name if genre was inputted

get\_key()

Get key name for filtering function; :returns standard python function input

get\_key\_value()

Get filter key value value;

:returns standard python function input

get\_optional\_values()

Get optional values for filtering function;

if input value of sort order cannot be converted to int turns desc to 0

else sort order is converted value

:returns: year\_f,year\_t: integer values of years

desc: sort order

get\_path()

Get book path value;

:returns standard python function input

get\_publication()

Get book publication value;

:returns standard python function input

get\_year()

Get book year value;

:returns: None: if year value cannot be converted

OR integer value of year

load\_library()

Load books from inputted library file

:return: function load\_library from books module

mod\_chosen()

Choose format of library file

:returns: 'w' if chosen formats are json or yaml

or 'wb' if chosen format is pickle

pprint(catalogue)

Prints values of all elements in book catalogue

:param catalogue: a list of books.

print\_authors(authors)

Prints names of all authors

:param: authors: a list of authors

print\_menu()

Print simple console menu

DATA

book\_catalogue = []

FILE

c:\users\jakov\pycharmprojects\labtest\booksupfunc.py