8. LIBRARY

Tibor Basletić Požar

One person has decided to download all of the fiction existing in the English language and store it on a single USB stick. He expects to find or generate the respective text files, compress them, and then index them conveniently. Is this ambition realistic?
Suggest a plan to approach this goal and solve a partial problem of this plan.

One person has decided to download all of the fiction existing in the English language and store it on a single USB stick. He expects to find or generate the respective text files, compress them, and then index them conveniently. Is this ambition realistic? Suggest a plan to approach this goal and solve a partial problem of this plan.

- not so clear how to do it
- legality? free?
- where on internet?

```
# bash script
for ((a=number; a <= limit ; a++))
do
  wget https://www.gutenberg.org/ebooks/${a}.txt.utf-8
done
```

What we did:
- free books: **www.gutenberg.org**
- **'.txt'** (ASCII) format (or convert PDF/e-book to '.txt')
- no mass download -> script for automatic download -> **wget**
- script works for ~12 books
- afterwards: CAPTCHA + 24-hour ban from site
- total books: 130 (enough for testing)
- only fiction books? No.

One person has decided to download all of the fiction existing in the English language and store it on a single USB stick. He expects to find or generate the respective text files, compress them, and then index them conveniently. Is this ambition realistic? Suggest a plan to approach this goal and solve a partial problem of this plan.

- how many fiction books exist?
- USB stick memory -> typical 128 GB or 256 GB max

We estimate:
- book = 500 pages
- page = 40 rows x 60 chars = 2400 bytes (1 char = 1 byte)
- book total = 1.14 MB
- approx. 1 MB/book

- compressing (zipping) text -> ratio ~ 3:1

- total: *0.3 MB per zipped book*

768 000 zipped books
on
256 GB USB stick

(number of existing fiction books?)

One person has decided to download all of the fiction existing in the English language and store it on a single USB stick. He expects to find or generate the respective text files, compress them, and then index them conveniently. Is this ambition realistic? Suggest a plan to approach this goal and solve a partial problem of this plan.

- two non-GUI programs (unix: "*do one thing and do it well*"):

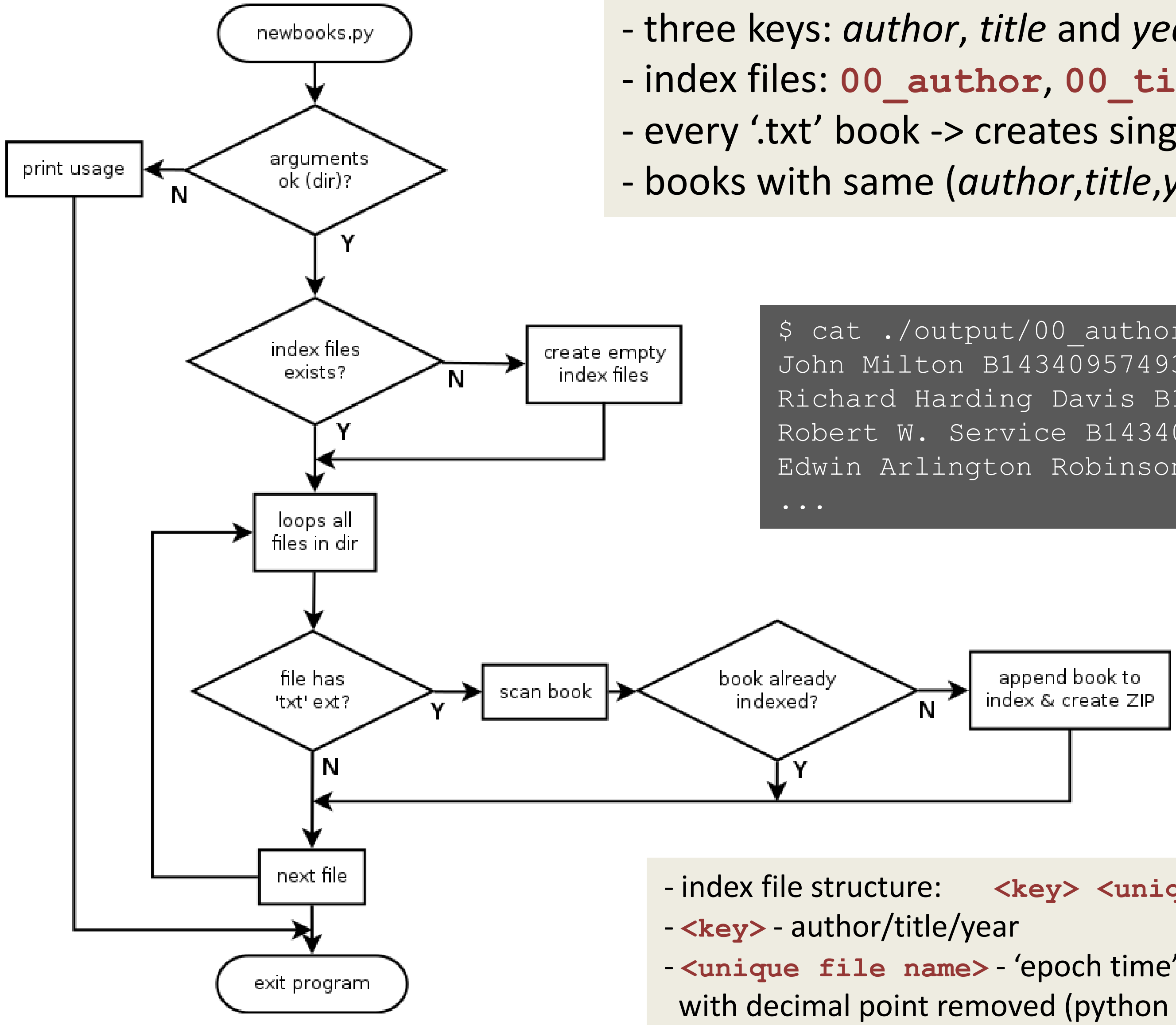
  `newbooks.py` - reads '.txt' books and populate indexes
  `searchbooks.py` - search index for books (not discussed)

- python language, 2.7 (Linux, debian)
- directory layout:

```
$ tree -d . # lists subdirs
.             # .py programs
├── input1    # downladed .txt books (anywhere)
├── input2    # downladed .txt books (anywhere)
└── output    # index (00_author, 00_title, 00_year)
              # and ZIP files
```

We assume, each book has:
- unique fields/keys **`author`**, **`title`** and **`year`** in every '.txt' file
- easily discernible fields **`author`**, **`title`** and **`year`** in every '.txt' file

`newbooks.py`



- three keys: *author*, *title* and *year*
- index files: `00_author`, `00_title` and `00_year`
- every '.txt' book -> creates single ZIP file (no duplicate!)
- books with same (*author*,*title*,*year*) -> identical books

```
$ cat ./output/00_author               # display file
John Milton B14340957493230877.zip
Richard Harding Davis B14340957493802699.zip
Robert W. Service B14340957494543322.zip
Edwin Arlington Robinson B14340957494693611.zip
...
```

- index file structure: `<key> <unique file name>.zip`
- `<key>` - author/title/year
- `<unique file name>` - 'epoch time' in seconds (0.001 ms resolution) with decimal point removed (python function `time.time()`)

Example of program run:

```
$ python newbooks.py ./input/
<...cut...>
Examining file 415.txt:
        Author: George Borrow
        Title:  The Bible in Spain
        Year:   January, 1996  [EBook #415]
        Book already exists.
Examining file 207.txt:
        Author: Robert Service
        Title:  The Spell of the Yukon
        Year:   January, 1995
        Book will be added to index.
        Zip file name: B1434388540352144.zip
Examining file 14.txt:
        Author: United States.  Central Intelligence Agency
        Title:  The 1990 CIA World Factbook
        Year:   Unknown
        Book already exists.

Books added:   49
Books skipped: 81
Total of 130 files examined
```

One person has decided to download all of the fiction existing in the English language and store it on a single USB stick. He expects to find or generate the respective text files, compress them, and then index them conveniently. Is this ambition realistic? Suggest a plan to approach this goal and solve a partial problem of this plan.

SOME STATISTICS

Memory:

- 130 ASCII books = 67 MB → 1 book ~ 0.5 MB
- 130 ZIP books = 25 MB → 1 ZIP book ~ 0.2 MB
- similar to our initial estimate (0.3 MB per ziped book)

Speed:

- adding 130 books ~ 3.8 sec
- adding 130 duplicate books ~ 0.008 sec
- for 768 000 books (256 GB USB): ~ 22500 sec = 6.25 h

→ **Yes, this ambition is realistic!**

(SPECS: 64 bit processor, 2.8 GHz, SSD)

CONCLUSION

- One could collect (download) large number of '.txt' books, and index them and store as compressed files on USB stick
- Number of books ~ 768 000 (256 GB USB)
- Simple python programs for indexing and compressing (ZIP)
- Required time: approx 6 hours
- ***This ambition is realistic***

Open questions:
- how many fiction books really exist?
- where to find all books?
- total download time (for almost TB of data)?
- '.txt' format -> we can convert PDF/e-book to ASCII
- extracting key fields (author/title/year) from '.txt' book?

References:
Python language – http://www.python.com/
Free books – https://www.gutenberg.org/
USB stick – https://en.wikipedia.org/wiki/USB_flash_drive

T H A N K Y O U

FOR YOUR ATTENTION

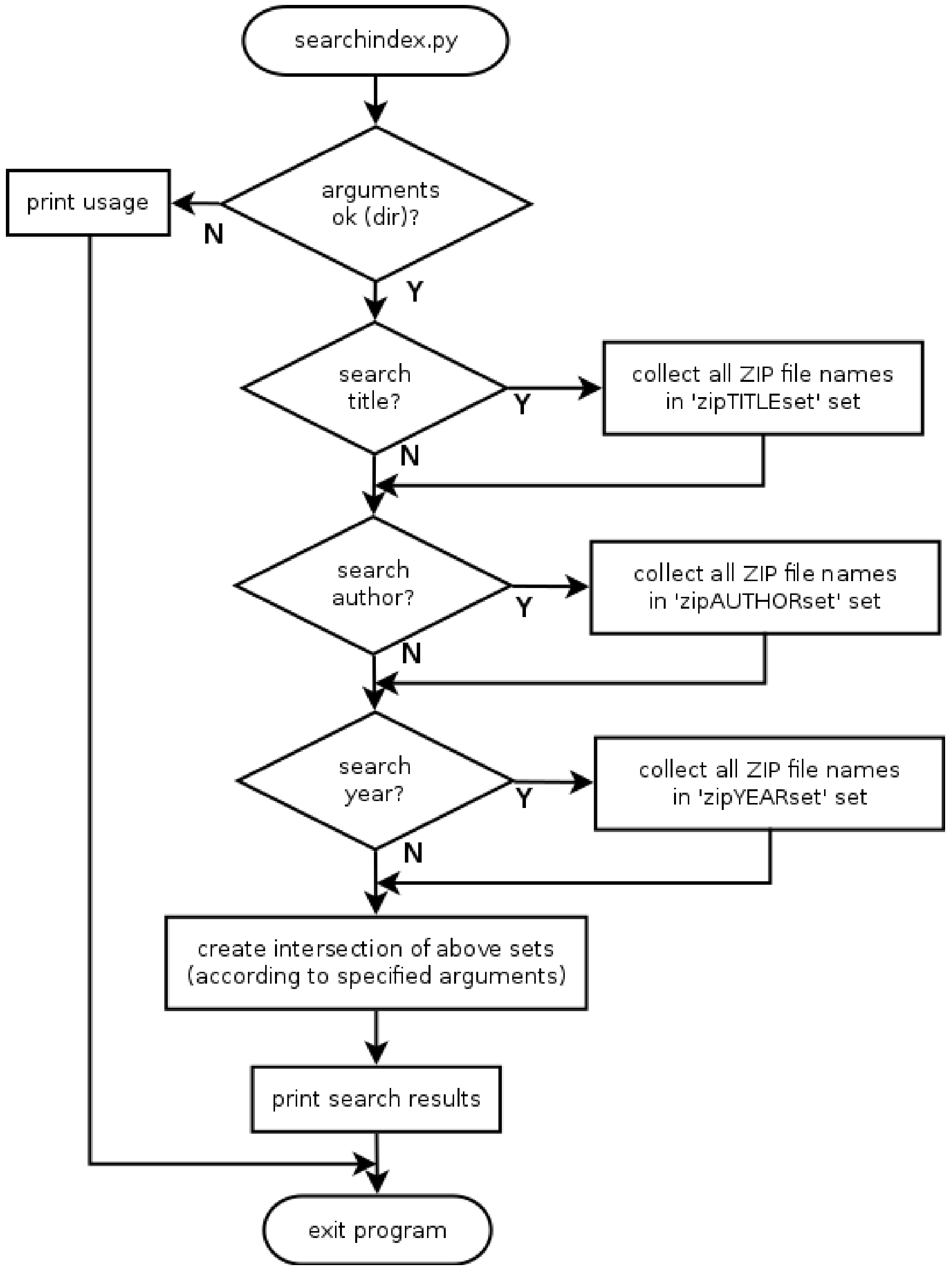
```
$ python searchindex.py
This program will search index for books by one or more keys.
Usage:
         searchindex.py -t TITLE | -a AUTHOR | -y YEAR

If search term contains space, put it in double quotes.
Multiple search conditions (keys) are possible, e.g.:
         searchindex.py -t title -a author -y 1982
```

Note:
- searches index files `00_author`, `00_title` and `00_year`
- single book has unique ZIP filename
- uses 'set' python language construct with set *union* and set *intersect*, for multi-key search
- case sensitive search
- outputs list of ZIP files

Improvement:
- case insensitive search
- more identical keys -> `-t "War and Peace" -t "Part 2"`
- search by first/last name
- more keys? (number of pages, ISBN/ISSN, ...)

```
$ python searchindex.py -a am
We found 11 books:
         ...
         B143409575299569.zip
         B1434388539853344.zip
         B1434388539095902.zip
         B1434388539021432.zip
         B1434095749934556.zip
         ...
$ python searchindex.py -y 1991
We found 7 books:
         ...
         B1434095752856314.zip
         B143409575299569.zip
         B1434095751078819.zip
         B1434095749934556.zip
$ python searchindex.py -a am -y 1991
We found 2 books:
         B143409575299569.zip
         B1434095749934556.zip
```


searchindex.py
arguments ok (dir)?
N
print usage
Y
search title?
Y
collect all ZIP file names in 'zipTITLEset' set
N
search author?
Y
collect all ZIP file names in 'zipAUTHORset' set
N
search year?
Y
collect all ZIP file names in 'zipYEARset' set
N
create intersection of above sets (according to specified arguments)
print search results
exit program

```
#<...cut...>
Xtitle = Xauthor = Xyear = False
zipset = set()
if '-t' in sys.argv:
    # search for title
    Xtitle = sys.argv[ sys.argv.index('-t')+1 ]
    ziptitleset = set(findtitle(Xtitle))
    zipset = zipset | ziptitleset
if '-a' in sys.argv:
    # search for author
    Xauthor = sys.argv[ sys.argv.index('-a')+1 ]
    zipauthorset = set(findauthor(Xauthor))
    zipset = zipset | zipauthorset
if '-y' in sys.argv:
    # search for year
    Xyear = sys.argv[ sys.argv.index('-y')+1 ]
    zipyearset = set(findyear(Xyear))
    zipset = zipset | zipyearset

if Xtitle:
    zipset = zipset & ziptitleset
if Xauthor:
    zipset = zipset & zipauthorset
if Xyear:
    zipset = zipset & zipyearset
#<...cut...>
```

Assumptions and requirements:
- programming language: python 2.7.10 (Linux, Debian)
- books are in '.txt' (ASCII) format -> not really a problem (PDF->ASCII, ...)
- has unique fields/keys **`author`**, **`title`** and **`year`** in every '.txt' file (might be problematic?)
- has easily discernible fields **`author`**, **`title`** and **`year`** in every '.txt' file (might be problematic?)

- two non-GUI programs (unix: "*do one thing and do it well*"):

`newbooks.py` - reads books and populate indexes
`searchbooks.py` - search index for books

```
$ tree -d . # lists subdirs
.             # .py programs
├── input1    # .txt books (anywhere)
├── input2    # .txt books (anywhere)
└── output    # index and ZIP files
```

- 130 '.txt' books *downloaded* from **`www.gutenberg.org`**

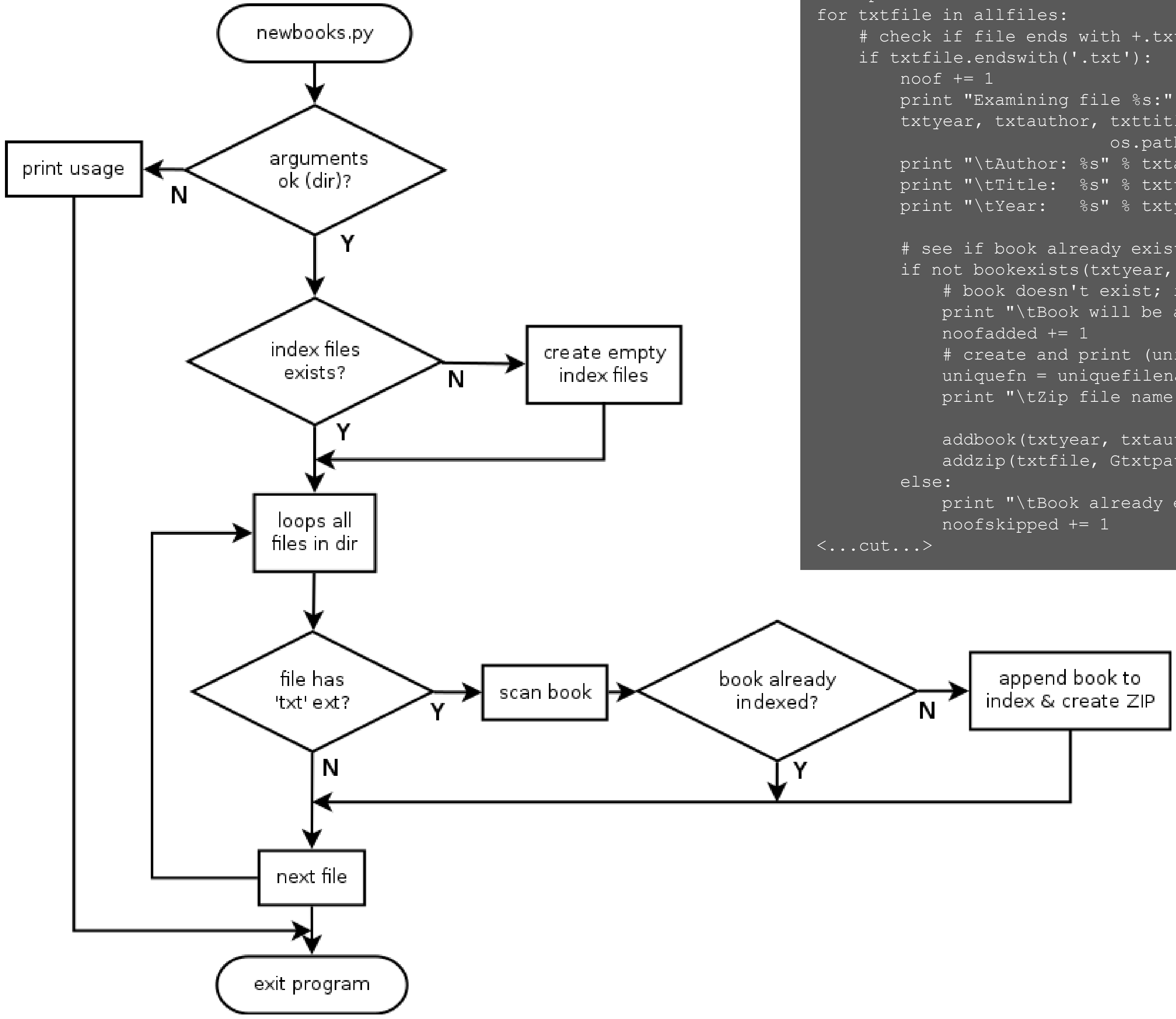
```
$ python newbooks.py
This program will scann and index books in '.txt.' format.
Usage:
        newbooks.py DIR
where DIR is directory with '.txt' books
```

Note:
- output into 3 ASCII files (indexes): `00_author`, `00_title` and `00_year` (into directory `./output/`)
- for every book, creates single ZIP file (no duplicate!)
- books with same (author,title,year) -> identical books
- index file structure: `<key> <unique file name>.zip`
- `<key>` - author/title/year
- `<unique file name>` - ‘epoch time’ in seconds (0.001 ms resolution) with decimal point removed (python function `time.time()`)
- example:

```
$ cat ./output/00_author           # display file
John Milton B1434095749323087.zip
Richard Harding Davis B1434095749380269.zip
Robert W. Service B1434095749454332.zip
Edwin Arlington Robinson B1434095749469361.zip
...
```

```
$ python newbooks.py ./input/
<...cut...>
Examining file 415.txt:
        Author: George Borrow
        Title:  The Bible in Spain
        Year:   January, 1996  [EBook #415]
        Book already exists.
Examining file 207.txt:
        Author: Robert Service
        Title:  The Spell of the Yukon
        Year:   January, 1995
        Book will be added to index.
        Zip file name: B1434388540352144.zip
Examining file 14.txt:
        Author: United States.  Central Intelligence Agency
        Title:  The 1990 CIA World Factbook
        Year:   Unknown
        Book already exists.

Books added:   49
Books skipped: 81
Total of 130 files examined
```



```
<...cut...>
# loop all files
for txtfile in allfiles:
    # check if file ends with +.txt'
    if txtfile.endswith('.txt'):
        noof += 1
        print "Examining file %s:" % txtfile
        txtyear, txtauthor, txttitle = discover(
                            os.path.join(Gtxtpath, txtfile))
        print "\tAuthor: %s" % txtauthor
        print "\tTitle:  %s" % txttitle
        print "\tYear:   %s" % txtyear

        # see if book already exists in index
        if not bookexists(txtyear, txtauthor, txttitle):
            # book doesn't exist; index book and create zip
            print "\tBook will be added to index."
            noofadded += 1
            # create and print (unique) file name
            uniquefn = uniquefilename(Gindexpath)
            print "\tZip file name: %s" % uniquefn

            addbook(txtyear, txtauthor, txttitle, uniquefn)
            addzip(txtfile, Gtxtpath, uniquefn)
        else:
            print "\tBook already exists."
            noofskipped += 1
<...cut...>
```