Outline

1. Data on Files
   - data compression
   - data as text or bytes
2. File Methods in Python
   - format conversions and text encryption
   - using buffers to count and replace words

MCS 260 Lecture 17
Introduction to Computer Science
Jan Verschelde, 3 July 2023

compression and conversion using buffers

1. Data on Files
   - data compression
   - data as text or bytes
2. File Methods in Python
   - format conversions and text encryption
   - using buffers to count and replace words

data compression

Two different data compression schemes:

- lossless: no loss of information in compression,
- lossy: minor errors are tolerated (e.g. images).

Popular compression schemes use dictionary encoding.
For example: replace all `the`'s in a text file by a symbol.

The Lempel-Ziv encoding used in Gzip

- finds duplicated strings in the input data.
- The second occurrence of a string is replaced by a pointer to the previous string.

To create an archive of several files, we use `tar`.
`tar` is an archiving program to store and extract files from an archive, called a *tarfile*. The `t` in `tar` stands for "tape".

compression and conversion using buffers

1. Data on Files
   - data compression
   - data as text or bytes
2. File Methods in Python
   - format conversions and text encryption
   - using buffers to count and replace words

data as text or bytes

Two main ways to open files:

1. Open with an encoding, such as `utf-8`.
   $\Rightarrow$ characters read are strings.
2. Open as a binary file, without encoding.
   $\Rightarrow$ data on the file are bytes:
   - bytes read from file must be decoded, from bytes to strings,
   - strings written to file must be encoded, from strings to bytes.

The distinction between a text and binary file is important:

1. Text files can be processed line by line,
   provided line breaks are present in the text file.
2. Binary files can be processed byte per byte.

compression and conversion using buffers

1. Data on Files
   - data compression
   - data as text or bytes

2. File Methods in Python
   - format conversions and text encryption
   - using buffers to count and replace words

format conversions

Convert the ad hoc formatting of the file books.txt

```
1:1:Computer Science, an overview:
0:2:Python Programming in Context:
```

into one that uses lists

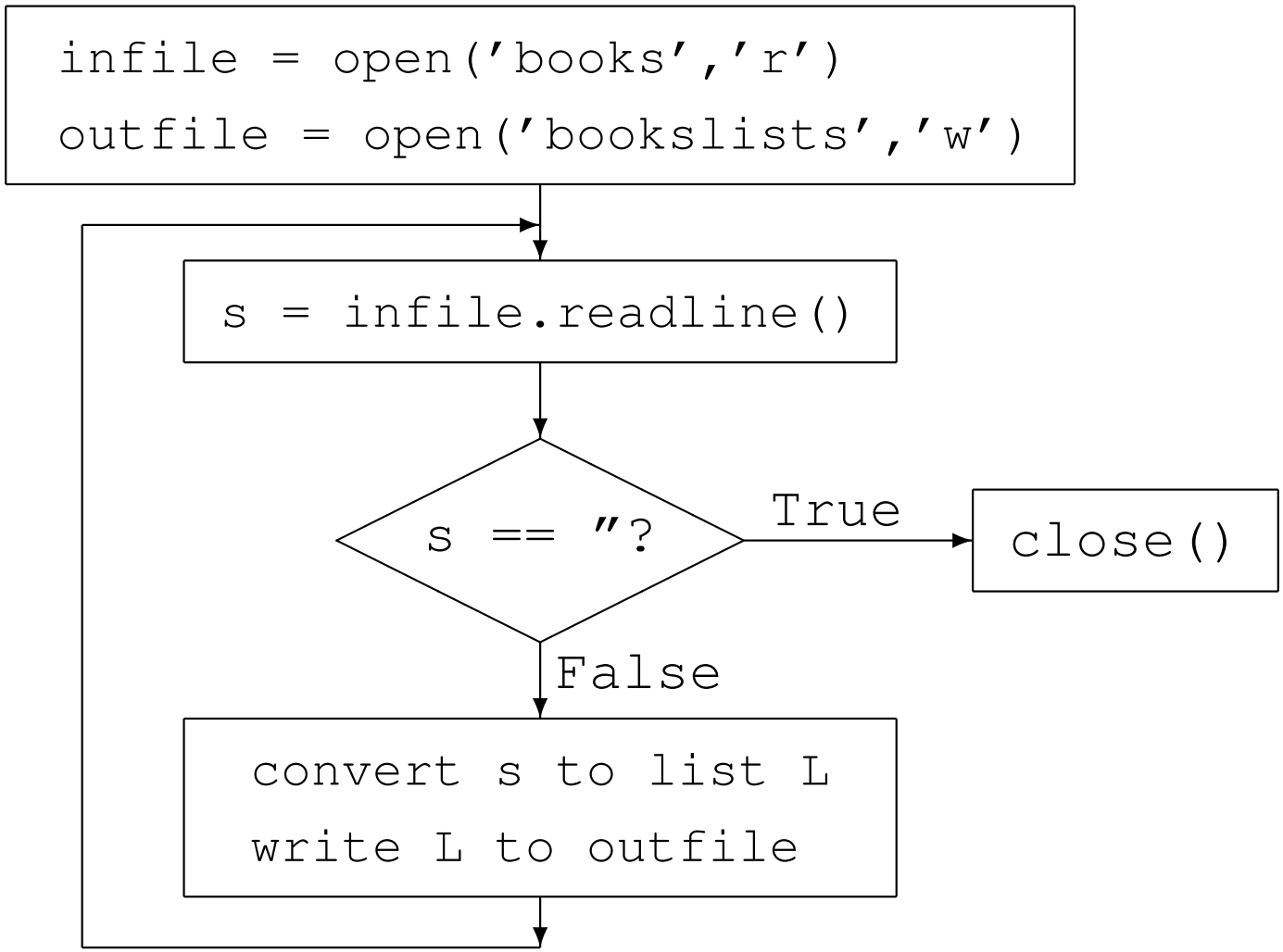
```
[True, 1, 'Computer Science, an overview']
[False, 2, 'Python Programming in Context']
```

and into using dictionaries

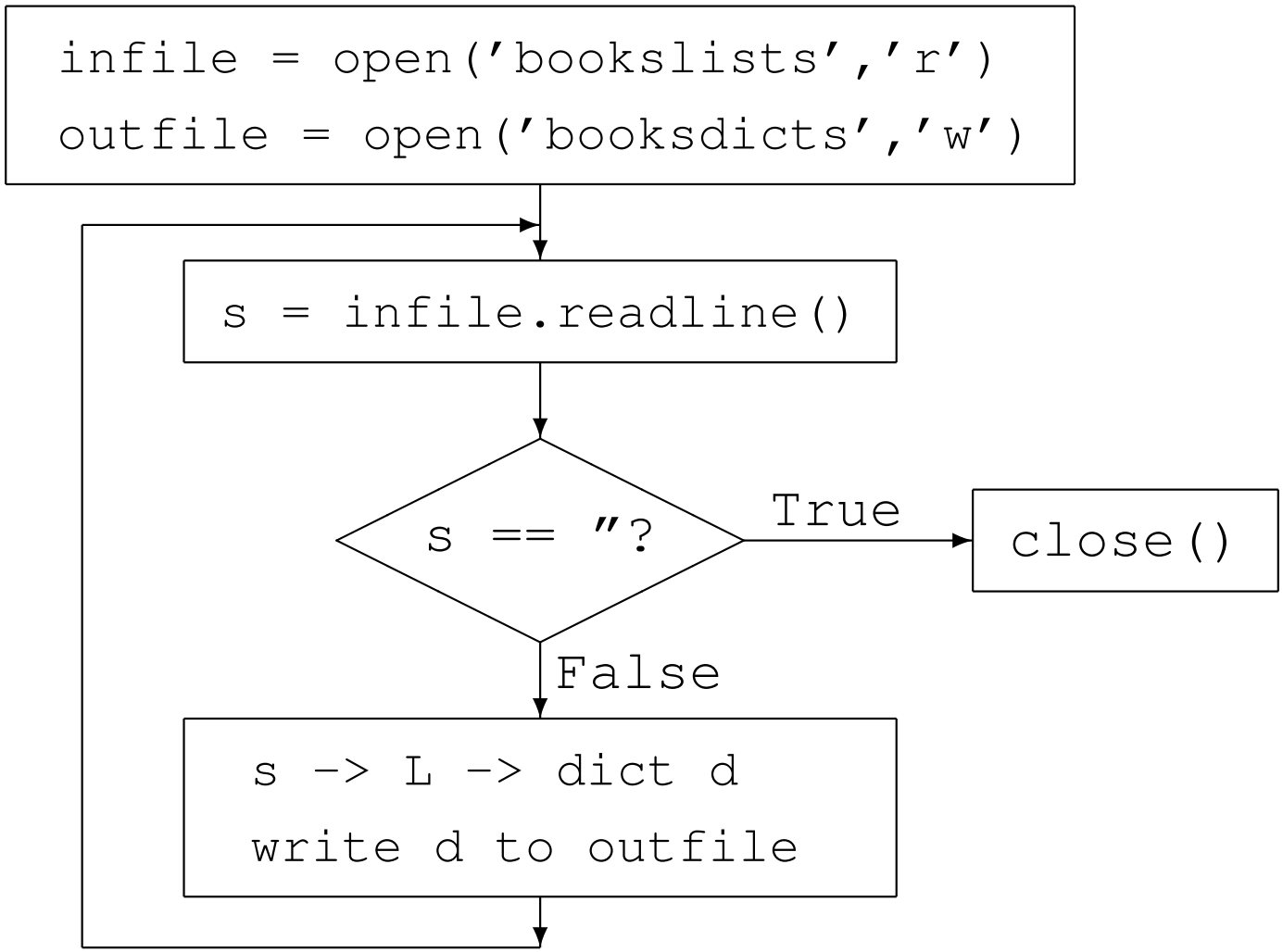
```
{'available': True, 'key': 1, \
'title': 'Computer Science, an overview}
{'available': False, 'key': 2, \
'title': 'Python Programming in Context'}
```

Benefit: the module `bkform.py` is no longer needed.

format conversion algorithm



converting lists into dictionaries



scrambling text – encrypting information

The content of the file sometext.txt:

```
This is a sample text, used as an example
for a message whose vowels will be scrambled.
```

After scrambling vowels, we obtain codetext.txt:

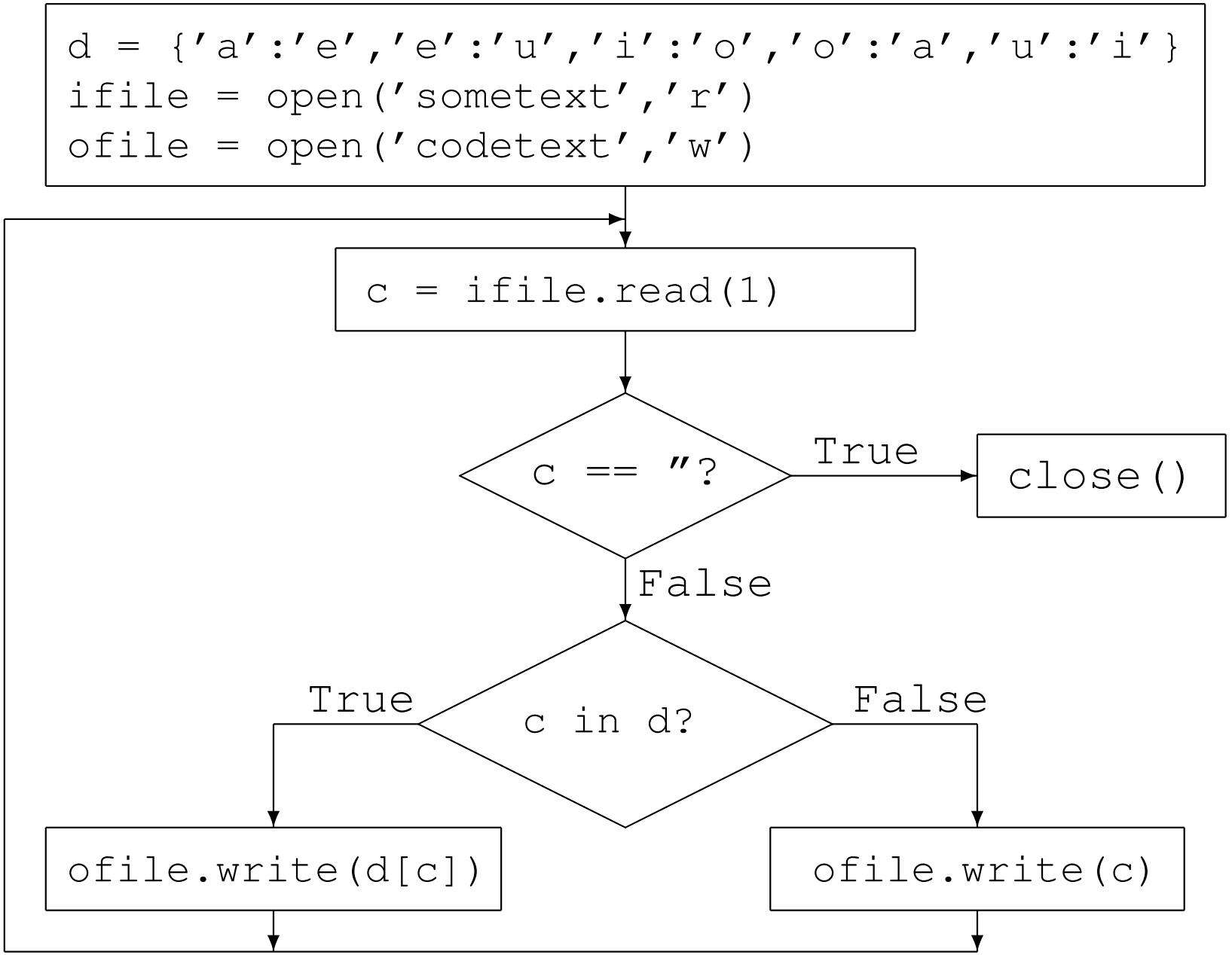
```
Thos os e semplu tuxt, isud es en uxemplu
far e mussegu whasu vawuls woll bu scremblud.
```

The cipher used:

```
['a','i','e','u','o'] -> ['e','o','u','i','a']
```

Python: file.read(1) returns one byte read from file.

the scrambling algorithm



compression and conversion using buffers

1. Data on Files
   - data compression
   - data as text or bytes
2. File Methods in Python
   - format conversions and text encryption
   - using buffers to count and replace words

counting words in a text

Problem: count number of times `the` occurs.

For example: if the file sometext2.txt has content

```
At the end of the lecture we go home,
taking the bus or the train.
```

then the number of times we count `the` is 4.

Algorithm:

1. maintain a buffer of 3 consecutive letters on file;
2. after each new byte read, compare with `the`.

Using lists and tuple assignments:

1. the buffer is stored as a list,
2. a tuple assignment shifts the characters in the buffer.

Moving the Cursor

The *cursor* indicates the current position in a file.

Let `F` be a file object.

- `F.tell()` returns the current position in the file.
- `F.seek(nbr)` moves the cursor with `nbr` positions.
  1. The file must be opened in binary mode.
  2. Characters read must be decoded.
  3. Strings written to file must be encoded.

Exercises

1. Write a program `bkform2dict.py` to convert the formatting in the file books directly to booksdicts.
2. Rewrite the program `library.py` of the previous lecture, using dictionaries as formats for the file books.
3. Modify the `wordcount.py` so that it considers a word as separated from others by spaces or commas and other punctuation symbols, i.e.: the `the` in `there` or `then` does not count as an occurrence of `the`.
4. Design a permutation of the vowels so that after applying `scramblevowels.py` twice we get the original message.

more exercises

5. Download “Macbeth” of William Shakespeare as plain UTF-8 text from `http://www.gutenberg.org`
   and write a script to count the number of occurrences of the strings `"LADY MACBETH"` and `"MACBETH"`.
6. From `https://finance.yahoo.com`, download the historical values of the IBM stock as a file in csv (comma separated value) format. Write a script to find the days at which IBM stock was valued at its highest and lowest price.