

Outline

Files

- organization
- managing a library

Python

- manipulating files
- library management

Maple

- writeto

Summary + Assignments

Files

- organization
- managing a library

Python

- manipulating files
- library management

Maple

- writeto

Summary + Assignments

MCS 260 Lecture 21
Introduction to Computer Science
Jan Verschelde, 15 October 2007

Outline

MCS 260 L-21

15 October 2007

Files

organization

managing a library

Files

organization

managing a library

Python

manipulating files

library management

Python

manipulating files

library management

Maple

writeto

Maple

writeto

Summary + Assignments

Summary + Assignments

Files store Data Permanently

on disk, flash drive, or tape

Random access memory is volatile: all data vanish after a program terminates.

- ▶ Files offer permanent storage for data. We distinguish between
 1. human readable text files; and
 2. binary files, only readable to the computer.
- ▶ Access to files on disk drives is slower. The operating system uses buffers:
 1. reads more than one line or character at once;
 2. writes to file, only when the buffer is full.
- ▶ Think of files as tapes, sequentially organized:
 1. one needs to read all previous items first;
 2. inserting an item in file leads to a new copy.

Files

organization

managing a library

Python

manipulating files

library management

Maple

writeln

Summary + Assignments

Files store Data Permanently

on disk, flash drive, or tape

Random access memory is volatile: all data vanish after a program terminates.

- ▶ Files offer permanent storage for data.

We distinguish between

1. human readable text files; and
2. binary files, only readable to the computer.

- ▶ Access to files on disk drives is slower.

The operating system uses buffers:

1. reads more than one line or character at once;
2. writes to file, only when the buffer is full.

- ▶ Think of files as tapes, sequentially organized:

1. one needs to read all previous items first;
2. inserting an item in file leads to a new copy.

Files

organization

managing a library

Python

manipulating files

library management

Maple

writelto

Summary + Assignments

Files store Data Permanently

on disk, flash drive, or tape

Random access memory is volatile: all data vanish after a program terminates.

- ▶ Files offer permanent storage for data. We distinguish between
 1. human readable text files; and
 2. binary files, only readable to the computer.
- ▶ Access to files on disk drives is slower. The operating system uses buffers:
 1. reads more than one line or character at once;
 2. writes to file, only when the buffer is full.
- ▶ Think of files as tapes, sequentially organized:
 1. one needs to read all previous items first;
 2. inserting an item in file leads to a new copy.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Files store Data Permanently

on disk, flash drive, or tape

Random access memory is volatile: all data vanish after a program terminates.

- ▶ Files offer permanent storage for data. We distinguish between
 1. human readable text files; and
 2. binary files, only readable to the computer.
- ▶ Access to files on disk drives is slower. The operating system uses buffers:
 1. reads more than one line or character at once;
 2. writes to file, only when the buffer is full.
- ▶ Think of files as tapes, sequentially organized:
 1. one needs to read all previous items first;
 2. inserting an item in file leads to a new copy.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Files store Data Permanently

on disk, flash drive, or tape

Random access memory is volatile: all data vanish after a program terminates.

- ▶ Files offer permanent storage for data. We distinguish between
 1. human readable text files; and
 2. binary files, only readable to the computer.
- ▶ Access to files on disk drives is slower. The operating system uses buffers:
 1. reads more than one line or character at once;
 2. writes to file, only when the buffer is full.
- ▶ Think of files as tapes, sequentially organized:
 1. one needs to read all previous items first;
 2. inserting an item in file leads to a new copy.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Files store Data Permanently

on disk, flash drive, or tape

Random access memory is volatile: all data vanish after a program terminates.

- ▶ Files offer permanent storage for data. We distinguish between
 1. human readable text files; and
 2. binary files, only readable to the computer.
- ▶ Access to files on disk drives is slower. The operating system uses buffers:
 1. reads more than one line or character at once;
 2. writes to file, only when the buffer is full.
- ▶ Think of files as tapes, sequentially organized:
 1. one needs to read all previous items first;
 2. inserting an item in file leads to a new copy.

Files

organization

managing a library

Python

manipulating files

library management

Maple

writelto

Summary + Assignments

Files store Data Permanently

on disk, flash drive, or tape

Random access memory is volatile: all data vanish after a program terminates.

- ▶ Files offer permanent storage for data. We distinguish between
 1. human readable text files; and
 2. binary files, only readable to the computer.
- ▶ Access to files on disk drives is slower. The operating system uses buffers:
 1. reads more than one line or character at once;
 2. writes to file, only when the buffer is full.
- ▶ Think of files as tapes, sequentially organized:
 1. one needs to read all previous items first;
 2. inserting an item in file leads to a new copy.

Files

organization

managing a library

Python

manipulating files

library management

Maple

writelto

Summary + Assignments

Files store Data Permanently

on disk, flash drive, or tape

Random access memory is volatile: all data vanish after a program terminates.

- ▶ Files offer permanent storage for data. We distinguish between
 1. human readable text files; and
 2. binary files, only readable to the computer.
- ▶ Access to files on disk drives is slower. The operating system uses buffers:
 1. reads more than one line or character at once;
 2. writes to file, only when the buffer is full.
- ▶ Think of files as tapes, sequentially organized:
 1. one needs to read all previous items first;
 2. inserting an item in file leads to a new copy.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

- ▶ **Files are organized in directories.** Every file has an absolute path name, `pwd` prints the working directory, with `cd` we change the current directory.
- ▶ Access permissions: `r`: read, `w`: write, `x`: execute, organized in three user groups:
 1. the owner of the file
 2. every one in the same group
 3. every one else

For example: `-rwxr-x--x` (see via `ls -l`) means

1. the owner can read, write and execute;
2. members of the group can read and execute;
3. every one else can only execute.

`chmod 751 file` changes the file access permissions, into `-rwxr-x--x` (or `111101001`).

Files

organization

managing a library

Python

manipulating files

library management

Maple

writelto

Summary + Assignments

Files on Unix

UIC's icarus

- ▶ Files are organized in directories. Every file has an absolute path name, `pwd` prints the working directory, with `cd` we change the current directory.
- ▶ Access permissions: `r`: read, `w`: write, `x`: execute, organized in three user groups:
 1. the owner of the file
 2. every one in the same group
 3. every one else

For example: `-rwxr-x--x` (see via `ls -l`) means

1. the owner can read, write and execute;
2. members of the group can read and execute;
3. every one else can only execute.

`chmod 751 file` changes the file access permissions, into `-rwxr-x--x` (or `111101001`).

Files

organization

managing a library

Python

manipulating files

library management

Maple

writelto

Summary + Assignments

Files on Unix

UIC's icarus

- ▶ Files are organized in directories. Every file has an absolute path name, `pwd` prints the working directory, with `cd` we change the current directory.
- ▶ Access permissions: `r`: read, `w`: write, `x`: execute, organized in three user groups:
 1. the owner of the file
 2. every one in the same group
 3. every one else

For example: `-rwxr-x--x` (see via `ls -l`) means

1. the owner can read, write and execute;
2. members of the group can read and execute;
3. every one else can only execute.

`chmod 751 file` changes the file access permissions, into `-rwxr-x--x` (or `111101001`).

Files

organization

managing a library

Python

manipulating files

library management

Maple

writing

Summary +
Assignments

Files on Unix

UIC's icarus

- ▶ Files are organized in directories. Every file has an absolute path name, `pwd` prints the working directory, with `cd` we change the current directory.
- ▶ Access permissions: `r`: read, `w`: write, `x`: execute, organized in three user groups:
 1. the owner of the file
 2. every one in the same group
 3. every one else

For example: `-rwxr-x--x` (see via `ls -l`) means

1. the owner can read, write and execute;
2. members of the group can read and execute;
3. every one else can only execute.

`chmod 751 file` changes the file access permissions, into `-rwxr-x--x` (or `111101001`).

Files

organization

managing a library

Python

manipulating files

library management

Maple

writing

Summary + Assignments

- ▶ Files are organized in directories. Every file has an absolute path name, `pwd` prints the working directory, with `cd` we change the current directory.
- ▶ Access permissions: `r`: read, `w`: write, `x`: execute, organized in three user groups:
 1. the owner of the file
 2. every one in the same group
 3. every one else

For example: `-rwxr-x--x` (see via `ls -l`) means

1. the owner can read, write and execute;
2. members of the group can read and execute;
3. every one else can only execute.

`chmod 751 file` changes the file access permissions, into `-rwxr-x--x` (or `111101001`).

Files

organization

managing a library

Python

manipulating files

library management

Maple

writelto

Summary + Assignments

- ▶ Files are organized in directories. Every file has an absolute path name, `pwd` prints the working directory, with `cd` we change the current directory.
- ▶ Access permissions: `r`: read, `w`: write, `x`: execute, organized in three user groups:
 1. the owner of the file
 2. every one in the same group
 3. every one else

For example: `-rwxr-x--x` (see via `ls -l`) means

1. the owner can read, write and execute;
2. members of the group can read and execute;
3. every one else can only execute.

`chmod 751 file` changes the file access permissions, into `-rwxr-x--x` (or `111101001`).

Files

organization

managing a library

Python

manipulating files

library management

Maple

writelto

Summary + Assignments

Files on Unix

UIC's icarus

- ▶ Files are organized in directories. Every file has an absolute path name, `pwd` prints the working directory, with `cd` we change the current directory.
- ▶ Access permissions: `r`: read, `w`: write, `x`: execute, organized in three user groups:
 1. the owner of the file
 2. every one in the same group
 3. every one else

For example: `-rwxr-x--x` (see via `ls -l`) means

1. the owner can read, write and execute;
2. members of the group can read and execute;
3. every one else can only execute.

`chmod 751 file` changes the file access permissions, into `-rwxr-x--x` (or `111101001`).

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

Outline

MCS 260 L-21

15 October 2007

Files

organization
managing a library

Files

organization
managing a library

Python

manipulating files
library management

Python

manipulating files
library management

Maple

writeto

Maple

writeto

Summary + Assignments

Summary + Assignments

A Simple Library Administration

a screen shot

MCS 260 L-21

15 October 2007

Executing `library.py` at the command prompt \$:

```
$ python library.py
```

```
Welcome to our library!  Choose:
```

0. leave the program
1. show all books in the collection
2. add a new book to the collection
3. check out a book of the library
4. return a book to the library

```
Type 0, 1, 2, 3, or 4 :
```

The collection of books is stored on file

→ save data about collection after program terminates.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

A Simple Library Administration

a screen shot

MCS 260 L-21

15 October 2007

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

Executing `library.py` at the command prompt \$:

```
$ python library.py
```

```
Welcome to our library!  Choose:
```

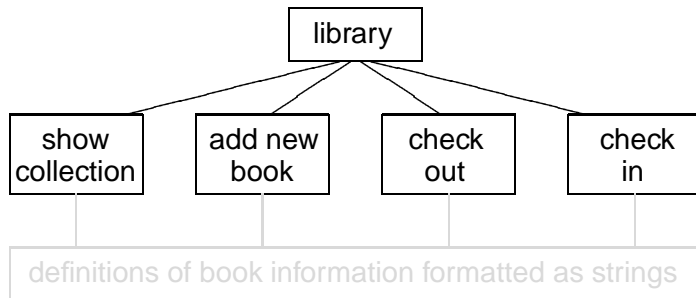
0. leave the program
1. show all books in the collection
2. add a new book to the collection
3. check out a book of the library
4. return a book to the library

```
Type 0, 1, 2, 3, or 4 :
```

The collection of books is stored on file

→ save data about collection after program terminates.

Straightforward top down functional design:



Supported by bottom up module with functions that define how we format the information of books as strings.

Files

organization
managing a library

Python

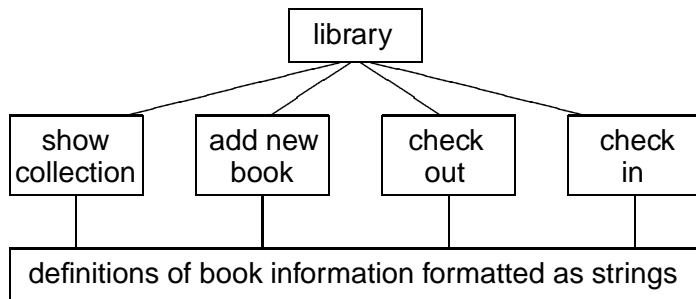
manipulating files
library management

Maple

writeln

Summary + Assignments

Straightforward top down functional design:



Supported by bottom up module with functions that define how we format the information of books as strings.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Information about Books

layout of data fields

The information stored for each book is

1. one bit: whether it is available or not;
2. a number: its key value in the collection;
3. a string: its title.

A file is just a sequence of lines.

Every line is a string.

We separate the data fields by colons:

```
1:1:The Art & Craft of Computing:
```

```
1:2:Making Use of Python:
```

→ content of file `books` (or `books.txt` on Windows)

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

Information about Books

layout of data fields

The information stored for each book is

1. one bit: whether it is available or not;
2. a number: its key value in the collection;
3. a string: its title.

A file is just a sequence of lines.

Every line is a string.

We separate the data fields by colons:

```
1:1:The Art & Craft of Computing:
```

```
1:2:Making Use of Python:
```

→ content of file `books` (or `books.txt` on Windows)

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

Information about Books

layout of data fields

The information stored for each book is

1. one bit: whether it is available or not;
2. a number: its key value in the collection;
3. a string: its title.

A file is just a sequence of lines.

Every line is a string.

We separate the data fields by colons:

```
1:1:The Art & Craft of Computing:
```

```
1:2:Making Use of Python:
```

→ content of file `books` (or `books.txt` on Windows)

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

Information about Books

layout of data fields

The information stored for each book is

1. one bit: whether it is available or not;
2. a number: its key value in the collection;
3. a string: its title.

A file is just a sequence of lines.

Every line is a string.

We separate the data fields by colons:

```
1:1:The Art & Craft of Computing:
```

```
1:2:Making Use of Python:
```

→ content of file `books` (or `books.txt` on Windows)

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary +
Assignments

Information about Books

layout of data fields

The information stored for each book is

1. one bit: whether it is available or not;
2. a number: its key value in the collection;
3. a string: its title.

A file is just a sequence of lines.

Every line is a string.

We separate the data fields by colons:

```
1:1:The Art & Craft of Computing:
```

```
1:2:Making Use of Python:
```

→ content of file [books](#) (or [books.txt](#) on Windows)

Outline

Files

- organization
- managing a library

Python

- manipulating files**
- library management

Maple

- writeto

Summary + Assignments

Files

- organization
- managing a library

Python

- manipulating files**
- library management

Maple

- writeto

Summary + Assignments

File Objects

opening files to read, write, or append

Files are objects to store data permanently on disk.

To use a file, we must first open it. Syntax is

```
< object > = open ( < name > , < access mode > )
```

where

`object` is the name of the object

`name` is the name of the file on disk
could be the entire path name

`access mode` defines how we use the file:

'r': read, 'w': write, 'a': append

Examples:

```
file = open('books','r') # read from 'books'  
file = open('books','w') # write to 'books'  
file = open('books','a') # append to 'books'
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

File Objects

opening files to read, write, or append

Files are objects to store data permanently on disk.

To use a file, we must first open it. Syntax is

```
< object > = open ( < name > , < access mode > )
```

where

`object` is the name of the object

`name` is the name of the file on disk
could be the entire path name

`access mode` defines how we use the file:

'r': read, 'w': write, 'a': append

Examples:

```
file = open('books','r') # read from 'books'
```

```
file = open('books','w') # write to 'books'
```

```
file = open('books','a') # append to 'books'
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

File Objects

opening files to read, write, or append

Files are objects to store data permanently on disk.

To use a file, we must first open it. Syntax is

```
< object > = open ( < name > , < access mode > )
```

where

object is the name of the object

name is the name of the file on disk
could be the entire path name

access mode defines how we use the file:

'r': read, 'w': write, 'a': append

Examples:

```
file = open('books','r') # read from 'books'
file = open('books','w') # write to 'books'
file = open('books','a') # append to 'books'
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

File Objects

opening files to read, write, or append

Files are objects to store data permanently on disk.

To use a file, we must first open it. Syntax is

```
< object > = open ( < name > , < access mode > )
```

where

object is the name of the object

name is the name of the file on disk
could be the entire path name

access mode defines how we use the file:

'r': read, 'w': write, 'a': append

Examples:

```
file = open('books','r') # read from 'books'  
file = open('books','w') # write to 'books'  
file = open('books','a') # append to 'books'
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

File Objects

opening files to read, write, or append

Files are objects to store data permanently on disk.

To use a file, we must first open it. Syntax is

```
< object > = open ( < name > , < access mode > )
```

where

object is the name of the object

name is the name of the file on disk
could be the entire path name

access mode defines how we use the file:

'r': read, 'w': write, 'a': append

Examples:

```
file = open('books','r') # read from 'books'  
file = open('books','w') # write to 'books'  
file = open('books','a') # append to 'books'
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

File Objects

opening files to read, write, or append

Files are objects to store data permanently on disk.

To use a file, we must first open it. Syntax is

```
< object > = open ( < name > , < access mode > )
```

where

object is the name of the object

name is the name of the file on disk
could be the entire path name

access mode defines how we use the file:

'r': read, 'w': write, 'a': append

Examples:

```
file = open('books','r') # read from 'books'
file = open('books','w') # write to 'books'
file = open('books','a') # append to 'books'
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Reading from File

methods `readline()` and `readlines()`

As objects, files have methods we can use to read.

1. `readline()` reads the next line from file:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b
'1:1:The Art & Craft of Computing:\n'
```

notice the end of line symbol

2. `readlines()` reads all lines of the file at once

```
>>> file = open('books', 'r')
>>> collection = file.readlines()
>>> collection
['1:1:The Art & Craft of Computing:\n',
 '1:2:Making Use of Python:\n']
```

readlines() returns a list of strings

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

Reading from File

methods `readline()` and `readlines()`

As objects, files have methods we can use to read.

1. `readline()` reads the next line from file:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b
'1:1:The Art & Craft of Computing:\n'
```

notice the end of line symbol

2. `readlines()` reads all lines of the file at once

```
>>> file = open('books', 'r')
>>> collection = file.readlines()
>>> collection
['1:1:The Art & Craft of Computing:\n',
 '1:2:Making Use of Python:\n']
```

readlines() returns a list of strings

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

Reading from File

methods `readline()` and `readlines()`

As objects, files have methods we can use to read.

1. `readline()` reads the next line from file:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b
'1:1:The Art & Craft of Computing:\n'
```

notice the end of line symbol

2. `readlines()` reads all lines of the file at once

```
>>> file = open('books', 'r')
>>> collection = file.readlines()
>>> collection
['1:1:The Art & Craft of Computing:\n',
 '1:2:Making Use of Python:\n']
```

readlines() returns a list of strings

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

Reading from File

methods `readline()` and `readlines()`

As objects, files have methods we can use to read.

1. `readline()` reads the next line from file:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b
'1:1:The Art & Craft of Computing:\n'
```

notice the end of line symbol

2. `readlines()` reads all lines of the file at once

```
>>> file = open('books', 'r')
>>> collection = file.readlines()
>>> collection
['1:1:The Art & Craft of Computing:\n',
 '1:2:Making Use of Python:\n']
```

readlines() returns a list of strings

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

Reading from File

methods `readline()` and `readlines()`

As objects, files have methods we can use to read.

1. `readline()` reads the next line from file:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b
'1:1:The Art & Craft of Computing:\n'
```

notice the end of line symbol

2. `readlines()` reads all lines of the file at once

```
>>> file = open('books', 'r')
>>> collection = file.readlines()
>>> collection
['1:1:The Art & Craft of Computing:\n',
 '1:2:Making Use of Python:\n']
```

readlines() returns a list of strings

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Reading from File

methods `readline()` and `readlines()`

As objects, files have methods we can use to read.

1. `readline()` reads the next line from file:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b
'1:1:The Art & Craft of Computing:\n'
```

notice the end of line symbol

2. `readlines()` reads all lines of the file at once

```
>>> file = open('books', 'r')
>>> collection = file.readlines()
>>> collection
['1:1:The Art & Craft of Computing:\n',
 '1:2:Making Use of Python:\n']
```

readlines() returns a list of strings

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

End of File and close()

At the end of reading ...

1. When do we reach the end of a file?

Suppose there are two lines in books:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b = file.readline()
>>> file.readline()
''
```

readline() returns an empty string when at the end

2. Closing access to a file:

```
>>> file.close()
```

it is good practice to use close() when done

although Python will automatically close a file when reassigning the file object, in regards to writing ...

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

End of File and close()

At the end of reading ...

1. When do we reach the end of a file?

Suppose there are two lines in books:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b = file.readline()
>>> file.readline()
''
```

readline() returns an empty string when at the end

2. Closing access to a file:

```
>>> file.close()
```

it is good practice to use close() when done

although Python will automatically close a file when reassigning the file object, in regards to writing ...

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

End of File and close()

At the end of reading ...

1. When do we reach the end of a file?

Suppose there are two lines in books:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b = file.readline()
>>> file.readline()
''
```

readline() returns an empty string when at the end

2. Closing access to a file:

```
>>> file.close()
```

it is good practice to use close() when done

although Python will automatically close a file when reassigning the file object, in regards to writing ...

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

End of File and close()

At the end of reading ...

1. When do we reach the end of a file?

Suppose there are two lines in books:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b = file.readline()
>>> file.readline()
''
```

readline() returns an empty string when at the end

2. Closing access to a file:

```
>>> file.close()
```

it is good practice to use close() when done

although Python will automatically close a file when reassigning the file object, in regards to writing ...

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

End of File and close()

At the end of reading ...

1. When do we reach the end of a file?

Suppose there are two lines in books:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b = file.readline()
>>> file.readline()
''
```

readline() returns an empty string when at the end

2. Closing access to a file:

```
>>> file.close()
```

it is good practice to use close() when done

although Python will automatically close a file when reassigning the file object, in regards to writing ...

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

End of File and close()

At the end of reading ...

1. When do we reach the end of a file?

Suppose there are two lines in books:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b = file.readline()
>>> file.readline()
''
```

readline() returns an empty string when at the end

2. Closing access to a file:

```
>>> file.close()
```

it is good practice to use close() when done

although Python will automatically close a file when reassigning the file object, in regards to writing ...

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

End of File and close()

At the end of reading ...

1. When do we reach the end of a file?

Suppose there are two lines in books:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b = file.readline()
>>> file.readline()
''
```

readline() returns an empty string when at the end

2. Closing access to a file:

```
>>> file.close()
```

it is good practice to use close() when done

although Python will automatically close a file when reassigning the file object, in regards to writing ...

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

End of File and close()

At the end of reading ...

1. When do we reach the end of a file?

Suppose there are two lines in books:

```
>>> file = open('books', 'r')
>>> b = file.readline()
>>> b = file.readline()
>>> file.readline()
''
```

readline() returns an empty string when at the end

2. Closing access to a file:

```
>>> file.close()
```

it is good practice to use close() when done

although Python will automatically close a file when reassigning the file object, in regards to writing ...

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Writing to File

methods `write()` and `writelines()`

Corresponding to `readline()` and `readlines()` are methods to write lines to file:

1. Suppose we want to append to a file:

```
>>> file = open('books', 'a')
>>> file.write('1:3:a new book:\n')
```

no `writeline()` needed: add `\n` at end

2. Writing an entire file, e.g.: to copy files

```
>>> file = open('books', 'r')
>>> L = file.readlines()
>>> newfile = open('backup', 'w')
>>> newfile.writelines(L)
```

often `newfile.close()` is needed to empty buffers

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeto

Summary + Assignments

Writing to File

methods `write()` and `writelines()`

Corresponding to `readline()` and `readlines()` are methods to write lines to file:

1. Suppose we want to append to a file:

```
>>> file = open('books', 'a')
>>> file.write('1:3:a new book:\n')
```

no `writeline()` needed: add `\n` at end

2. Writing an entire file, e.g.: to copy files

```
>>> file = open('books', 'r')
>>> L = file.readlines()
>>> newfile = open('backup', 'w')
>>> newfile.writelines(L)
```

often `newfile.close()` is needed to empty buffers

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeto

Summary + Assignments

Writing to File

methods `write()` and `writelines()`

Corresponding to `readline()` and `readlines()` are methods to write lines to file:

1. Suppose we want to append to a file:

```
>>> file = open('books', 'a')
>>> file.write('1:3:a new book:\n')
```

no `writeline()` needed: add `\n` at end

2. Writing an entire file, e.g.: to copy files

```
>>> file = open('books', 'r')
>>> L = file.readlines()
>>> newfile = open('backup', 'w')
>>> newfile.writelines(L)
```

often `newfile.close()` is needed to empty buffers

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeto

Summary + Assignments

Writing to File

methods `write()` and `writelines()`

Corresponding to `readline()` and `readlines()` are methods to write lines to file:

1. Suppose we want to append to a file:

```
>>> file = open('books', 'a')
>>> file.write('1:3:a new book:\n')
```

no `writeline()` needed: add `\n` at end

2. Writing an entire file, e.g.: to copy files

```
>>> file = open('books', 'r')
>>> L = file.readlines()
>>> newfile = open('backup', 'w')
>>> newfile.writelines(L)
```

often `newfile.close()` is needed to empty buffers

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeto

Summary + Assignments

Writing to File

methods `write()` and `writelines()`

Corresponding to `readline()` and `readlines()` are methods to write lines to file:

1. Suppose we want to append to a file:

```
>>> file = open('books', 'a')
>>> file.write('1:3:a new book:\n')
```

no `writeline()` needed: add `\n` at end

2. Writing an entire file, e.g.: to copy files

```
>>> file = open('books', 'r')
>>> L = file.readlines()
>>> newfile = open('backup', 'w')
>>> newfile.writelines(L)
```

often `newfile.close()` is needed to empty buffers

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeto

Summary + Assignments

Outline

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeto

Summary + Assignments

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeto

Summary + Assignments

The File

books or books.txt

Recall the content of the file **books**:

1:1:The Art & Craft of Computing:

1:2:Making Use of Python:

Important to know (by design):

- ▶ Every line on file represents one book.
- ▶ A line on file is a string of characters.
- ▶ For every book, we have 3 data fields:
 1. 1 or 0 for the availability of the book,
 2. the identification number of the book,
 3. the title of the book.
- ▶ Data fields are separated by colons :.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

The File

books or books.txt

Recall the content of the file **books**:

1:1:The Art & Craft of Computing:

1:2:Making Use of Python:

Important to know (by design):

- ▶ Every line on file represents one book.
- ▶ A line on file is a string of characters.
- ▶ For every book, we have 3 data fields:
 1. 1 or 0 for the availability of the book,
 2. the identification number of the book,
 3. the title of the book.
- ▶ Data fields are separated by colons :.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

The File

books or books.txt

Recall the content of the file `books`:

```
1:1:The Art & Craft of Computing:
1:2:Making Use of Python:
```

Important to know (by design):

- ▶ Every line on file represents one book.
- ▶ A line on file is a string of characters.
- ▶ For every book, we have 3 data fields:
 1. 1 or 0 for the availability of the book,
 2. the identification number of the book,
 3. the title of the book.
- ▶ Data fields are separated by colons `:`.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

The File

books or books.txt

Recall the content of the file `books`:

```
1:1:The Art & Craft of Computing:
1:2:Making Use of Python:
```

Important to know (by design):

- ▶ Every line on file represents one book.
- ▶ A line on file is a string of characters.
- ▶ For every book, we have 3 data fields:
 1. 1 or 0 for the availability of the book,
 2. the identification number of the book,
 3. the title of the book.
- ▶ Data fields are separated by colons `:`.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

The File

books or books.txt

Recall the content of the file `books`:

1:1:The Art & Craft of Computing:

1:2:Making Use of Python:

Important to know (by design):

- ▶ Every line on file represents one book.
- ▶ A line on file is a string of characters.
- ▶ For every book, we have 3 data fields:
 1. 1 or 0 for the availability of the book,
 2. the identification number of the book,
 3. the title of the book.
- ▶ Data fields are separated by colons `:`.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

The File

books or books.txt

Recall the content of the file `books`:

```
1:1:The Art & Craft of Computing:
1:2:Making Use of Python:
```

Important to know (by design):

- ▶ Every line on file represents one book.
- ▶ A line on file is a string of characters.
- ▶ For every book, we have 3 data fields:
 1. 1 or 0 for the availability of the book,
 2. the identification number of the book,
 3. the title of the book.
- ▶ Data fields are separated by colons `:`.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

The File

books or books.txt

Recall the content of the file `books`:

```
1:1:The Art & Craft of Computing:
1:2:Making Use of Python:
```

Important to know (by design):

- ▶ Every line on file represents one book.
- ▶ A line on file is a string of characters.
- ▶ For every book, we have 3 data fields:
 1. 1 or 0 for the availability of the book,
 2. the identification number of the book,
 3. the title of the book.
- ▶ Data fields are separated by colons `:`.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Module bkform

definition of data formats for book

After `import bkform`, typing `help(bkform)` shows

FUNCTIONS

`get_key(s)`

given a string with book information,
returns the book identification number

`get_status(s)`

given a string with book information,
returns True or False for the availability

`get_title(s)`

given a string with book information,
returns the title of the book

`pack(key, status, title)`

returns the string with book information
where `key` is the book identification number,
`status` is True or False for availability,
`title` is the title of the book

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

pack() and get_status()

of the module bkform

```
def pack(key,status,title):
    "returns the string with book information\n\
    where key is the book identification number,\n\
    status is True or False for availability,\n\
    title is the title of the book"
    if status: s = '1:'
    else: s = '0:'
    s += '%d:' % key
    s += title + ':\n'
    return s
```

```
def get_status(s):
    "given a string with book information,\n\
    returns True or False for the availability"
    L = s.split(':')
    return int(L[0]) == 1
```

get_key and get_title are similar

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

pack() and get_status()

of the module bkform

```
def pack(key,status,title):
    "returns the string with book information\n\
    where key is the book identification number,\n\
    status is True or False for availability,\n\
    title is the title of the book"
    if status: s = '1:'
    else: s = '0:'
    s += '%d:' % key
    s += title + ':\n'
    return s

def get_status(s):
    "given a string with book information,\n\
    returns True or False for the availability"
    L = s.split(':')
    return int(L[0]) == 1
```

get_key and get_title are similar

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

pack() and get_status()

of the module bkform

```
def pack(key,status,title):
    "returns the string with book information\n\
    where key is the book identification number,\n\
    status is True or False for availability,\n\
    title is the title of the book"
    if status: s = '1:'
    else: s = '0:'
    s += '%d:' % key
    s += title + ':\n'
    return s
```

```
def get_status(s):
    "given a string with book information,\n\
    returns True or False for the availability"
    L = s.split(':')
    return int(L[0]) == 1
```

get_key and get_title are similar

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

The Main Program

library.py

```
from bkform import *

def show_menu():
    "shows the menu and prompts for a choice"

def main():
    while True:
        choice = show_menu()
        if choice == '0': break
        if choice == '1': show_books()
        if choice == '2': add_book()
        if choice == '3': checkout()
        if choice == '4': checkin()

main()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

The Main Program

library.py

```
from bkform import *

def show_menu():
    "shows the menu and prompts for a choice"

def main():
    while True:
        choice = show_menu()
        if choice == '0': break
        if choice == '1': show_books()
        if choice == '2': add_book()
        if choice == '3': checkout()
        if choice == '4': checkin()

main()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

The Main Program

library.py

```
from bkform import *

def show_menu():
    "shows the menu and prompts for a choice"

def main():
    while True:
        choice = show_menu()
        if choice == '0': break
        if choice == '1': show_books()
        if choice == '2': add_book()
        if choice == '3': checkout()
        if choice == '4': checkin()

main()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

Showing the Collection

the function `show_books()`

`show_books()` reads the entire collection from file and shows for every book its key, title, and availability.

```
def show_books():
    "shows the books currently in the collection"
    file = open('books', 'r')
    collection = file.readlines()
    for book in collection:
        s = ' ' + str(get_key(book))
        s += ' ' + get_title(book)
        if get_status(book):
            print s + ' available'
        else:
            print s + ' checked out'
    file.close()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Showing the Collection

the function `show_books()`

`show_books()` reads the entire collection from file and shows for every book its key, title, and availability.

```
def show_books():
    "shows the books currently in the collection"
    file = open('books', 'r')
    collection = file.readlines()
    for book in collection:
        s = ' ' + str(get_key(book))
        s += ' ' + get_title(book)
        if get_status(book):
            print s + ' available'
        else:
            print s + ' checked out'
    file.close()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Showing the Collection

the function `show_books()`

`show_books()` reads the entire collection from file and shows for every book its key, title, and availability.

```
def show_books():
    "shows the books currently in the collection"
    file = open('books', 'r')
    collection = file.readlines()
    for book in collection:
        s = ' ' + str(get_key(book))
        s += ' ' + get_title(book)
        if get_status(book):
            print s + ' available'
        else:
            print s + ' checked out'
    file.close()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Adding a New Book

the function `add_book()` needs to know the `#books`

```
def number_of_books():
    "returns the number of books in the collection"
    file = open('books', 'r')
    collection = file.readlines()
    N = len(collection)
    file.close()
    return N

def add_book():
    "adds a book to the collection"
    N = number_of_books()
    title = raw_input('Give title : ')
    file = open('books', 'a')
    file.write(pack(N+1, True, title))
    file.close()
```

Files

organization
managing a library

Python

"manipulating files
library management

Maple

writeln

Summary + Assignments

Adding a New Book

the function `add_book()` needs to know the `#books`

```
def number_of_books():
    "returns the number of books in the collection"
    file = open('books', 'r')
    collection = file.readlines()
    N = len(collection)
    file.close()
    return N

def add_book():
    "adds a book to the collection"
    N = number_of_books()
    title = raw_input('Give title : ')
    file = open('books', 'a')
    file.write(pack(N+1, True, title))
    file.close()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Borrowing a Book

the function checkout()

```
def checkout():
    "checks out a book"
    show_books()
    k = input('give book number : ')
    file = open('books', 'r')
    collection = file.readlines()
    file.close()
    file = open('books', 'w')
    for book in collection:
        if get_key(book) == k:
            if not get_status(book):
                print get_title(book) + ' is unavailable'
                file.write(book)
            else:
                file.write(pack(k, False, get_title(book)))
        else:
            file.write(book)
    file.close()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Borrowing a Book

the function checkout()

```
def checkout():
    "checks out a book"
    show_books()
    k = input('give book number : ')
    file = open('books', 'r')
    collection = file.readlines()
    file.close()
    file = open('books', 'w')
    for book in collection:
        if get_key(book) == k:
            if not get_status(book):
                print get_title(book) + ' is unavailable'
                file.write(book)
            else:
                file.write(pack(k, False, get_title(book)))
        else:
            file.write(book)
    file.close()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Borrowing a Book

the function checkout()

```
def checkout():
    "checks out a book"
    show_books()
    k = input('give book number : ')
    file = open('books', 'r')
    collection = file.readlines()
    file.close()
    file = open('books', 'w')
    for book in collection:
        if get_key(book) == k:
            if not get_status(book):
                print get_title(book) + ' is unavailable'
                file.write(book)
            else:
                file.write(pack(k, False, get_title(book)))
        else:
            file.write(book)
    file.close()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Returning a Book

the function checkin()

```
def checkin():
    "return a book to the library"
    k = input('give book number : ')
    file = open('books', 'r')
    collection = file.readlines()
    file.close()
    file = open('books', 'w')
    for book in collection:
        if get_key(book) == k:
            d = pack(k, True, get_title(book))
            file.write(d)
        else:
            file.write(book)
    file.close()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Returning a Book

the function checkin()

```
def checkin():
    "return a book to the library"
    k = input('give book number : ')
    file = open('books', 'r')
    collection = file.readlines()
    file.close()
    file = open('books', 'w')
    for book in collection:
        if get_key(book) == k:
            d = pack(k, True, get_title(book))
            file.write(d)
        else:
            file.write(book)
    file.close()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Returning a Book

the function checkin()

```
def checkin():
    "return a book to the library"
    k = input('give book number : ')
    file = open('books', 'r')
    collection = file.readlines()
    file.close()
    file = open('books', 'w')
    for book in collection:
        if get_key(book) == k:
            d = pack(k, True, get_title(book))
            file.write(d)
        else:
            file.write(book)
    file.close()
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Outline

MCS 260 L-21

15 October 2007

Files

organization
managing a library

Files

organization
managing a library

Python

manipulating files
library management

Python

manipulating files
library management

Maple

writeto

Maple

writeto

Summary + Assignments

Summary + Assignments

Maple's writeto

With `writeto` we can *redirect* the output:
what normally goes to screen, is written to file.

```
> p := randpoly(x, degree=3):
```

To write `p` to the file `/tmp/mypoly`:

```
> writeto("/tmp/mypoly")  
> p;
```

The last command produced no output to screen,
it has put `p` in the buffer,
and then the buffer is written after

```
> writeto(terminal);
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

`writeto`

Summary + Assignments

With `writeto` we can *redirect* the output:
what normally goes to screen, is written to file.

```
> p := randpoly(x, degree=3):
```

To write `p` to the file `/tmp/mypoly`:

```
> writeto("/tmp/mypoly")  
> p;
```

The last command produced no output to screen,
it has put `p` in the buffer,
and then the buffer is written after

```
> writeto(terminal);
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

`writeto`

Summary + Assignments

Maple's writeto

With `writeto` we can *redirect* the output: what normally goes to screen, is written to file.

```
> p := randpoly(x, degree=3):
```

To write `p` to the file `/tmp/mypoly`:

```
> writeto("/tmp/mypoly")  
> p;
```

The last command produced no output to screen, it has put `p` in the buffer, and then the buffer is written after

```
> writeto(terminal);
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

`writeto`

Summary + Assignments

Maple's writeto

With `writeto` we can *redirect* the output: what normally goes to screen, is written to file.

```
> p := randpoly(x, degree=3):
```

To write `p` to the file `/tmp/mypoly`:

```
> writeto("/tmp/mypoly")  
> p;
```

The last command produced no output to screen, it has put `p` in the buffer, and then the buffer is written after

```
> writeto(terminal);
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

`writeto`

Summary + Assignments

Maple's writeto

With `writeto` we can *redirect* the output: what normally goes to screen, is written to file.

```
> p := randpoly(x, degree=3):
```

To write `p` to the file `/tmp/mypoly`:

```
> writeto("/tmp/mypoly")  
> p;
```

The last command produced no output to screen, it has put `p` in the buffer, and then the buffer is written after

```
> writeto(terminal);
```

Files

organization
managing a library

Python

manipulating files
library management

Maple

`writeto`

Summary + Assignments

Summary + Assignments

We started chapter 7 in *Making Use of Python*;
for Unix, see §15.6.2 in *The Art & Craft of Computing*.

Assignments:

1. Extend the program to also contain the year of publication and author(s) for each book.
2. Write a function `empty_file` that takes on input a file name and returns `True` or `False` depending whether the file is empty or not.
3. Use the design of the library program as a model to stores the identification number, name, and address of every library patron. How will you format your file?
4. With a catalog of books and a file of patrons (from the previous exercise), design a third file that connects the borrowed books to the library patrons.

Homework will be collected on Monday 22 October.

Files

organization
managing a library

Python

manipulating files
library management

Maple

writeln

Summary + Assignments

Some extra Assignments

Files

organization
managing a library

Python

manipulating files
library management

Maple

writelto

Summary + Assignments

5. Write a Python function `search_book()` that prompts the user to enter the title of a book. This title is then used to search for the book with the same title in the file `books`.
6. Write a Python function `delete_book()` that prompts the user for an identification number and removes the corresponding book from the file `books`. When deleting a book, change also the identification numbers so that they match the line numbers.

Homework will be collected on Monday 22 October.