This is CS50

What questions do you have?

Strings

Dot notation

e.g., `str.lower()`

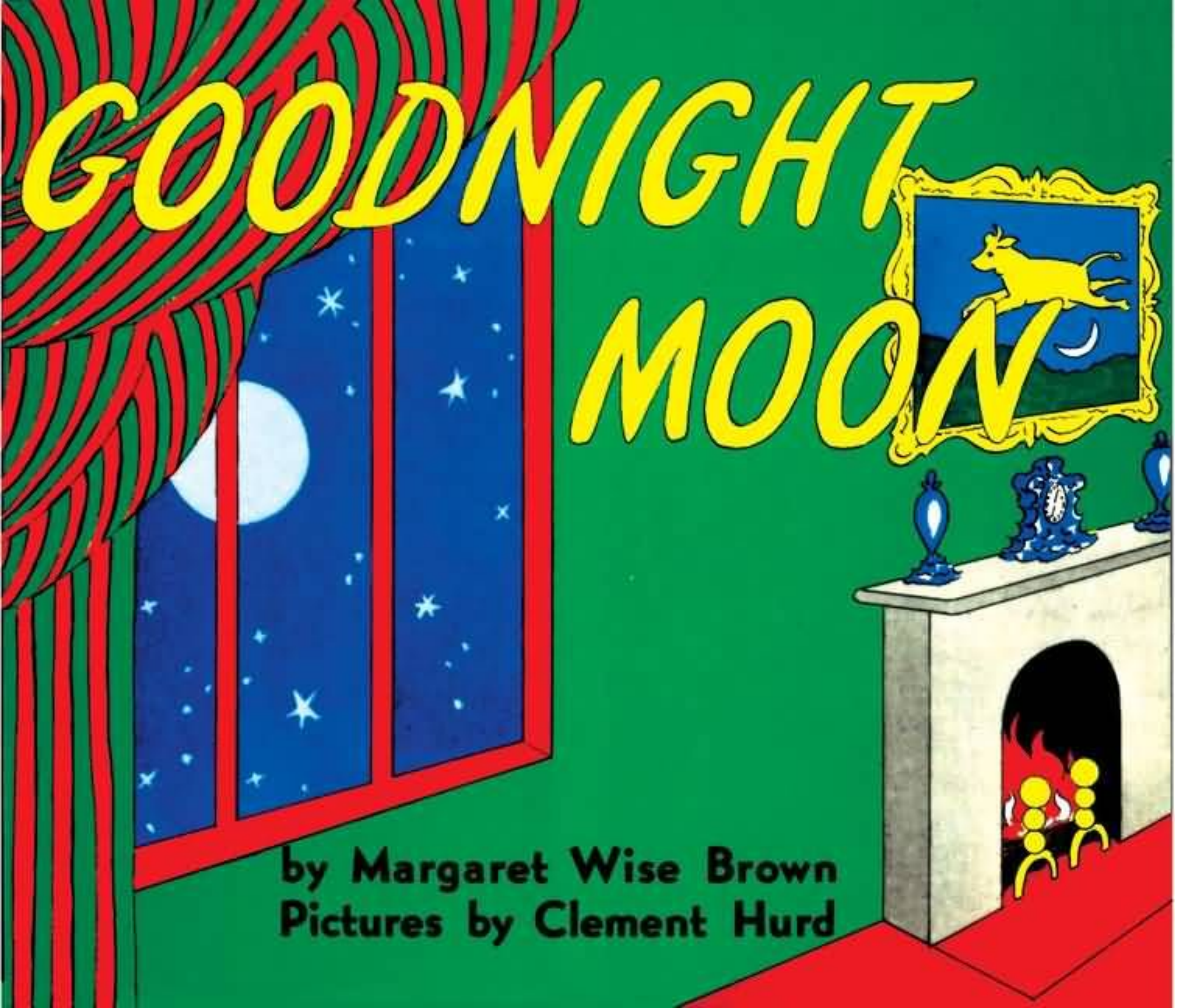
Loops

Dictionaries

Libraries

File Reading, Writing

Strings


GOODNIGHT MOON
by Margaret Wise Brown
Pictures by Clement Hurd

"In the great green room"

"there was a telephone
and a red balloon"

```
char *text = get_string("...");
```

```
text = input("...")
```

```
if (strcmp(text, "hello") == 0)
{
    ...
}
```

```
if text == "hello":
    ...
```

`text[i]`

`text[i]`

Dot Notation

```
text = input("...")
```

" In the great green room   "

```python
text = input("...")
text.strip()
```

" In the great

green room   "

```python
text = input("...")
text.strip()
```

"In the great

green room"

```
text = input("...")
```

"IN thE great green ROom"

```
text = input("...")
text.lower()
```

"IN thE great green ROom"

```
text = input("...")
text.lower()
```

"in the great green room"

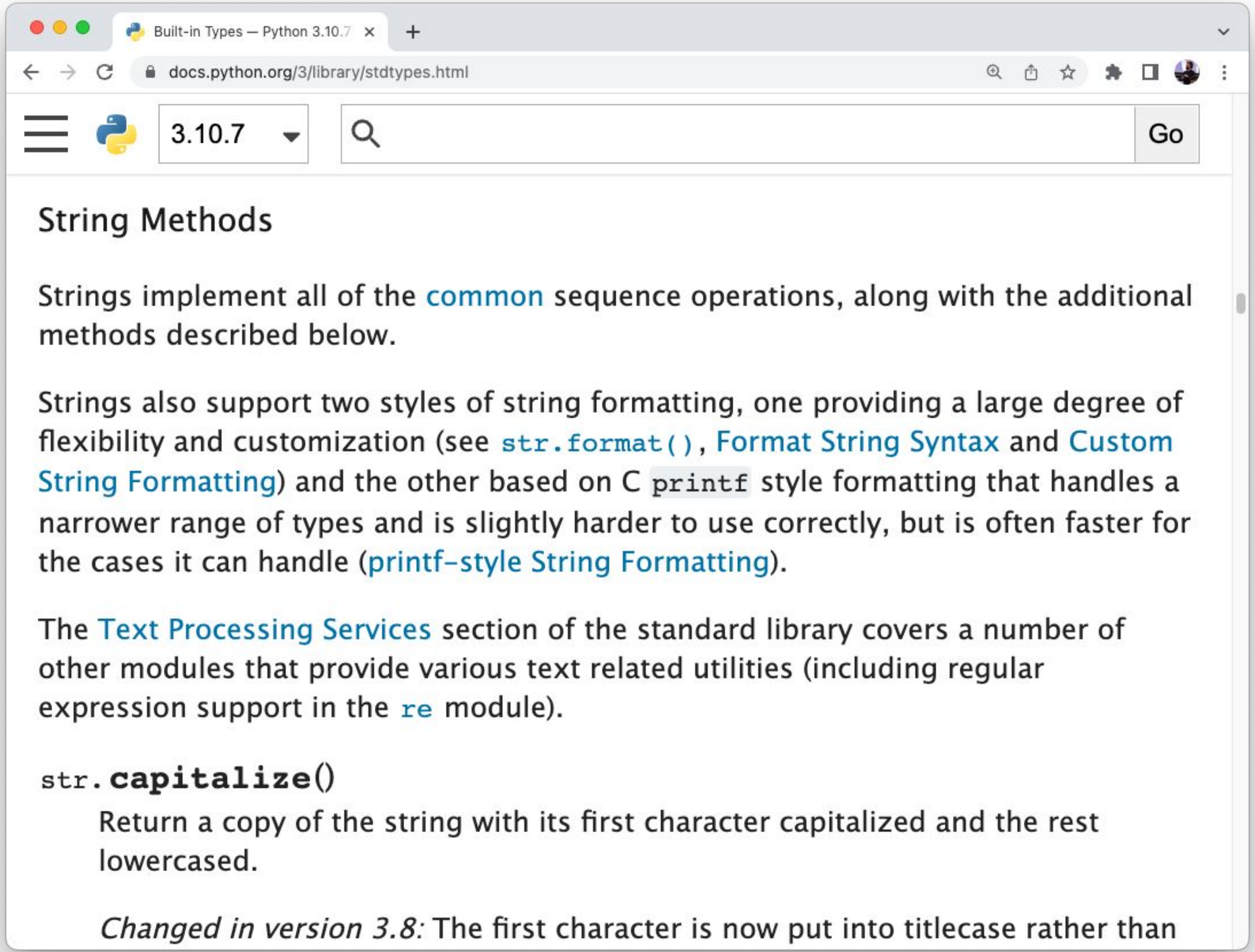
```
text = input("...")
text.capitalize()
```

"IN thE great
green ROom"

```
text = input("...")
text.capitalize()
```

"In the great green room"

str


Built-in Types — Python 3.10.7
docs.python.org/3/library/stdtypes.html
3.10.7
Go

String Methods

Strings implement all of the common sequence operations, along with the additional methods described below.

Strings also support two styles of string formatting, one providing a large degree of flexibility and customization (see `str.format()`, Format String Syntax and Custom String Formatting) and the other based on C `printf` style formatting that handles a narrower range of types and is slightly harder to use correctly, but is often faster for the cases it can handle (printf-style String Formatting).

The Text Processing Services section of the standard library covers a number of other modules that provide various text related utilities (including regular expression support in the `re` module).

`str.`**`capitalize`**()

Return a copy of the string with its first character capitalized and the rest lowercased.

Changed in version 3.8: The first character is now put into titlecase rather than candidate

```
candidate.name
candidate.votes
```

str

```
str.capitalize()
str.lower()
```

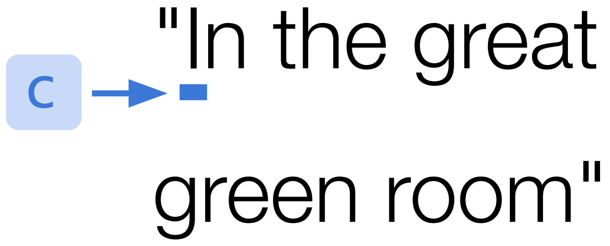
Loops

```
for c in text:
    print(c)
```

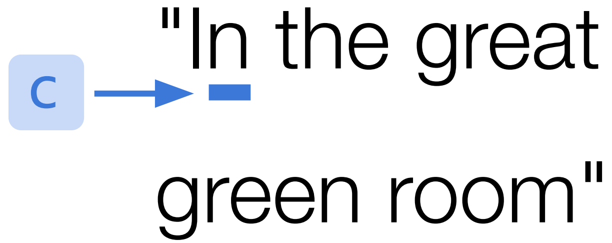
"In the great

green room"

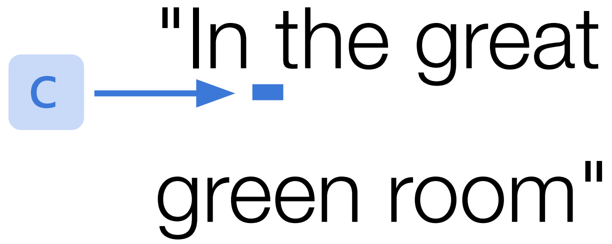
```
for c in text:
    print(c)
```



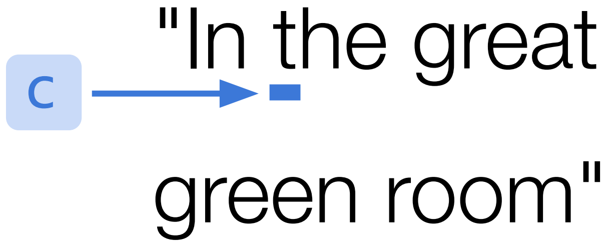
```
for c in text:
    print(c)
```



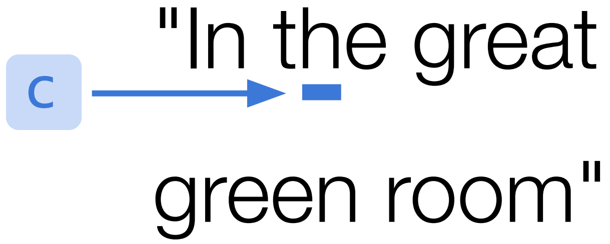
```
for c in text:
    print(c)
```



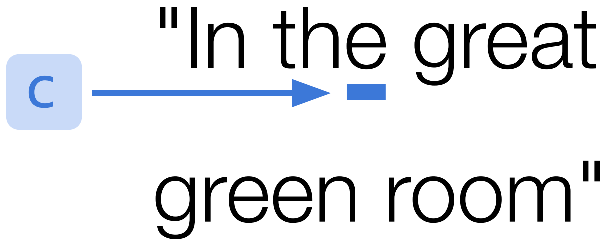
```
for c in text:
    print(c)
```



```
for c in text:
    print(c)
```



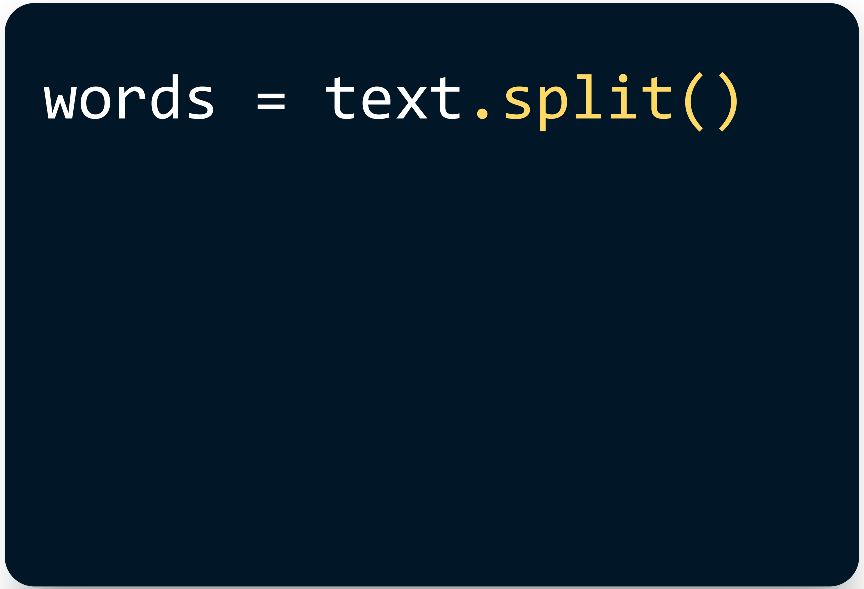
```python
for c in text:
    print(c)
```



```
words = text.split()
```

"In the great

green room"

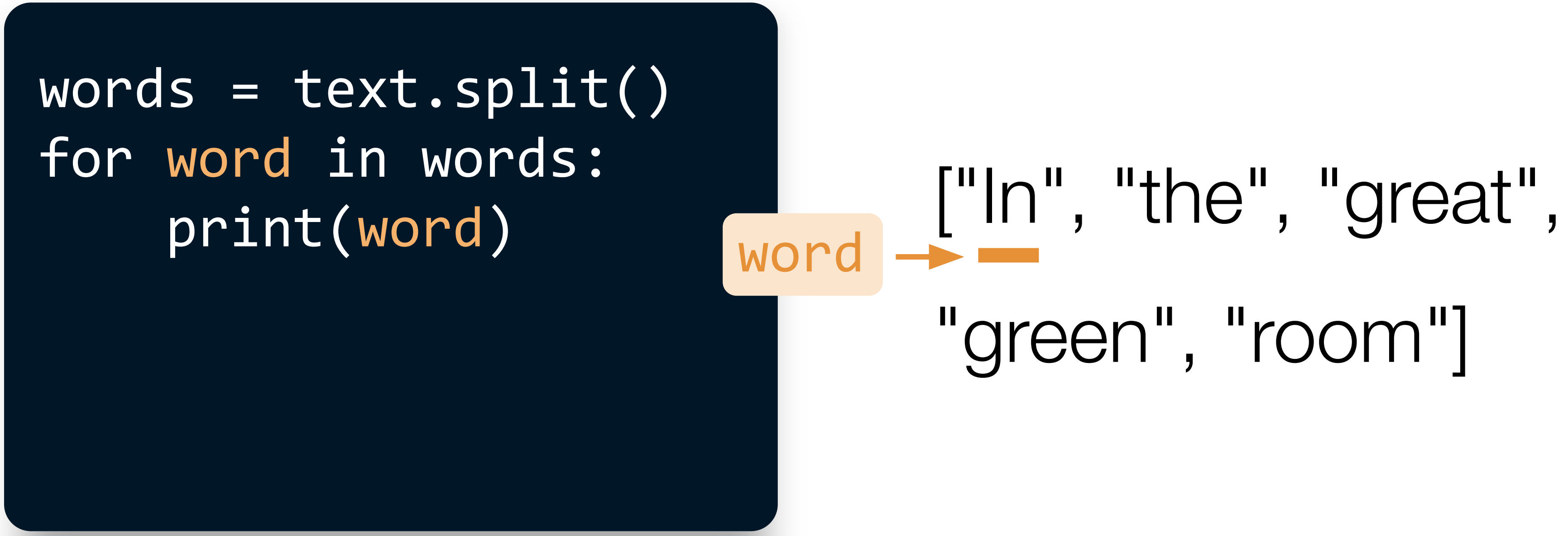


```
words = text.split()
```

["In", "the", "great", "green", "room"]

```
words = text.split()
for word in words:
    print(word)
```

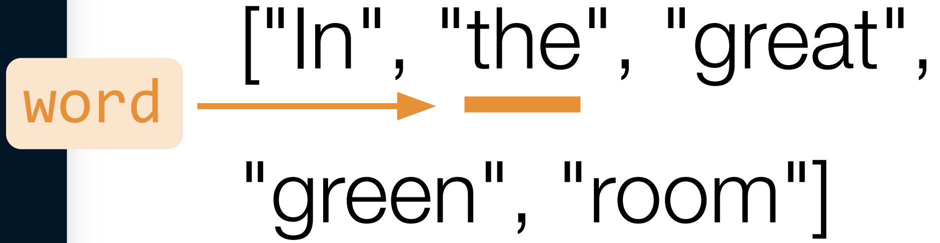
["In", "the", "great", "green", "room"]

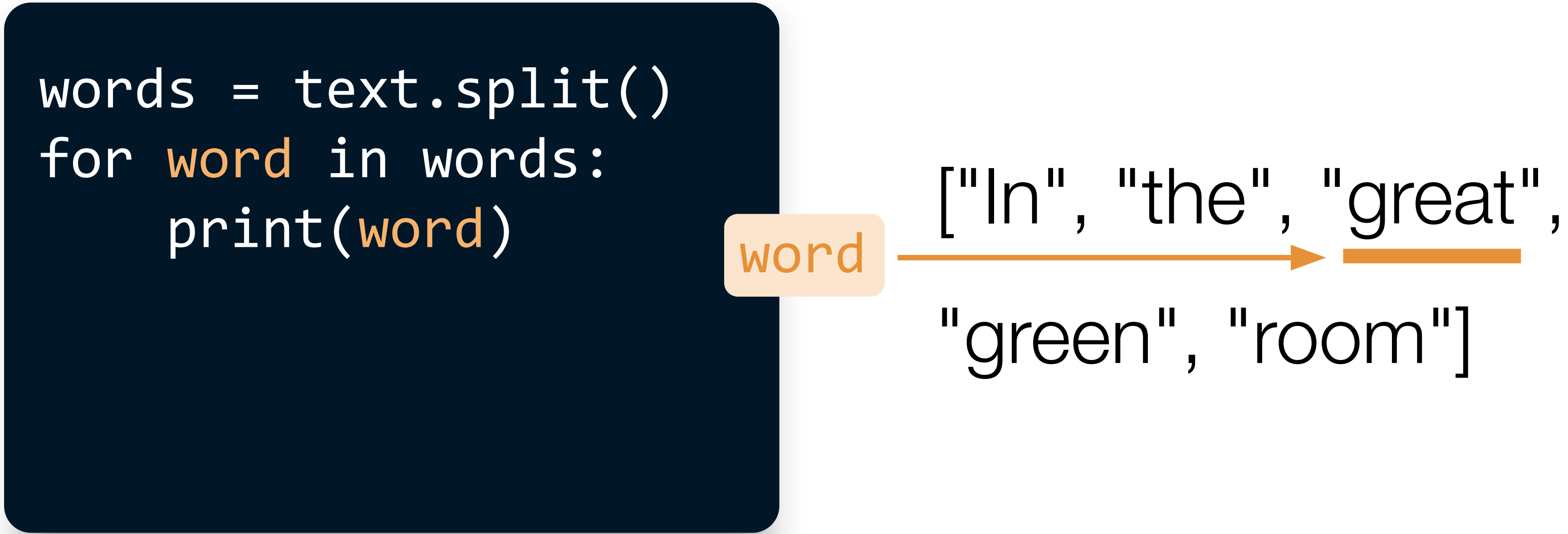


```
words = text.split()
for word in words:
    print(word)
```

["In", "the", "great",
"green", "room"]

```
words = text.split()
for word in words:
    print(word)
```





```python
words = text.split()
for word in words:
    print(word)
```

- Python's **for / in** syntax helps you iterate through components of an "iterable" while referring to them by a convenient name.

- When your *iterable* is a **list**, you'll iterate over every element of the **list**.

- When your *iterable* is a **string**, you'll iterate over every **character** of the **string**.

Text Analysis

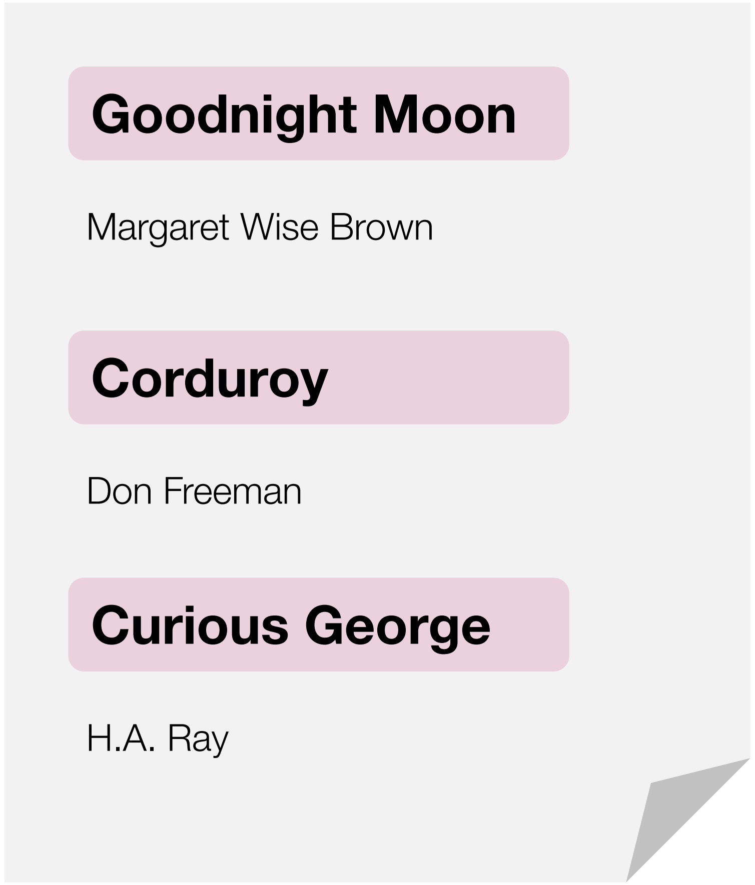
Look at **text.py**.

Using what you know about Python's syntax, guess what each round of loops will print to the terminal.

If feeling more comfortable, try writing your own loop to reverse the text given in the file.

Dictionaries

authors



Goodnight Moon

Margaret Wise Brown

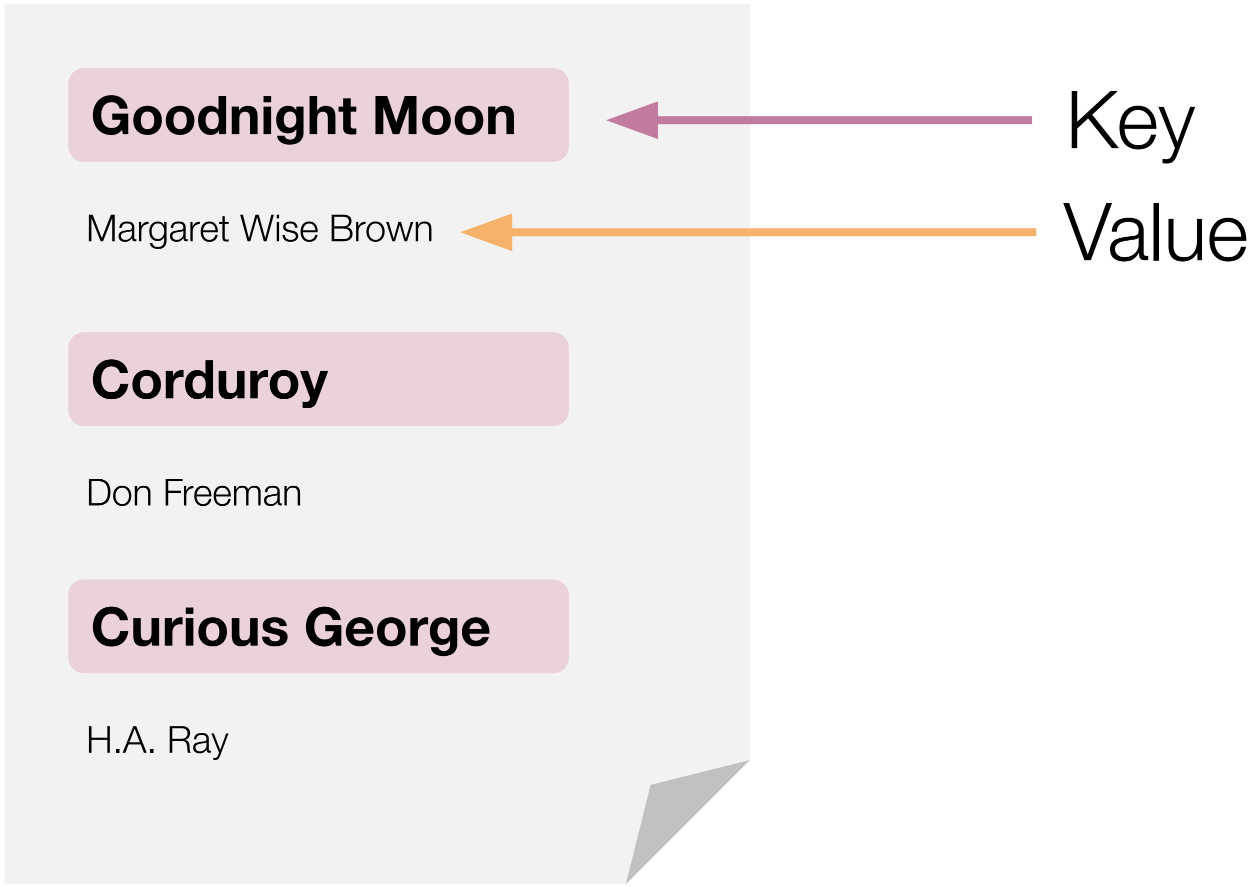
Corduroy

Don Freeman

Curious George

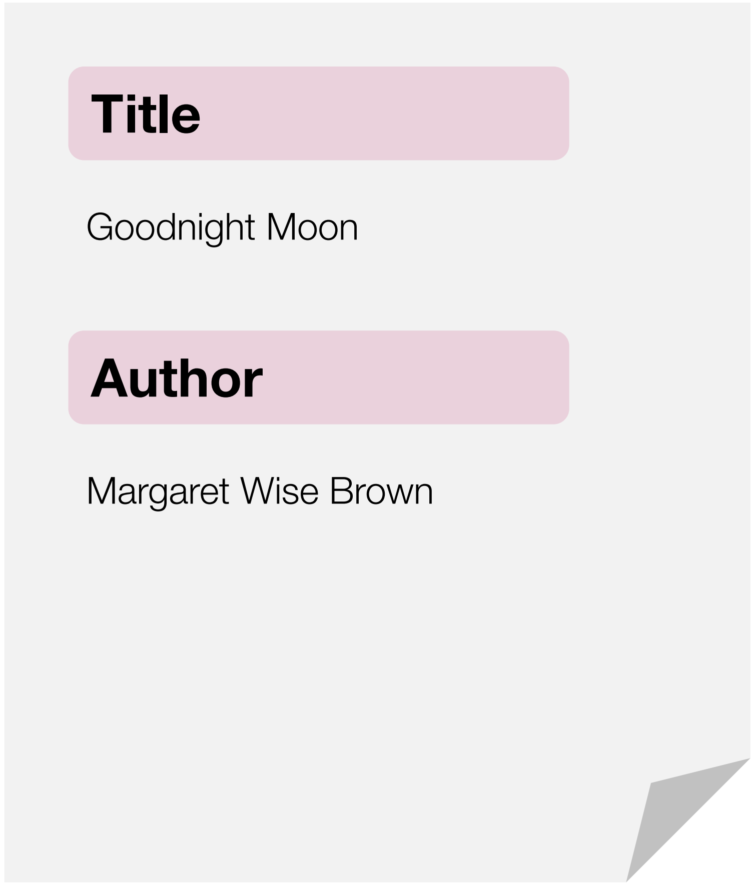
H.A. Ray

`authors`



Goodnight Moon
Margaret Wise Brown
Corduroy
Don Freeman
Curious George
H.A. Ray
Key
Value

book



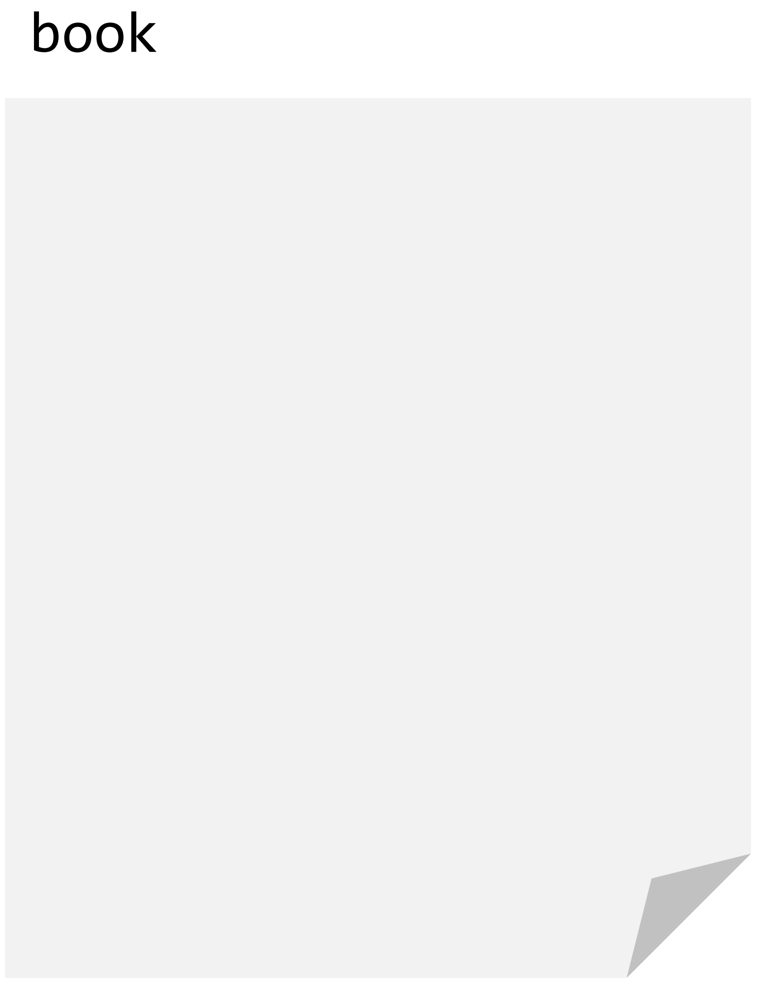
Title

Goodnight Moon

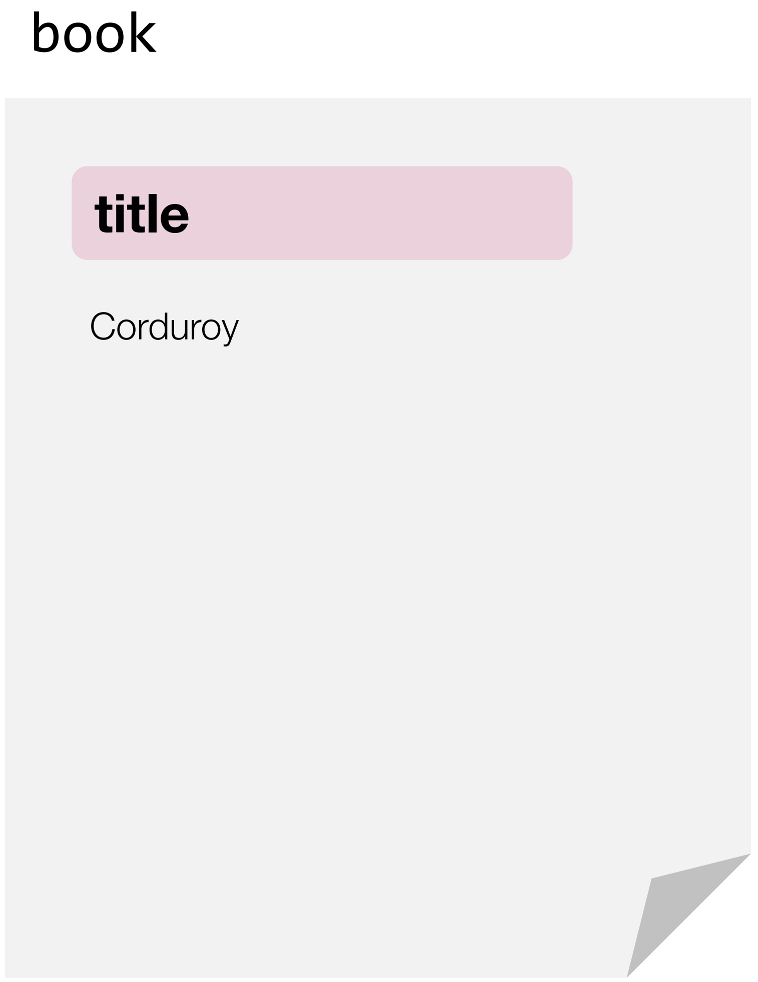
Author

Margaret Wise Brown

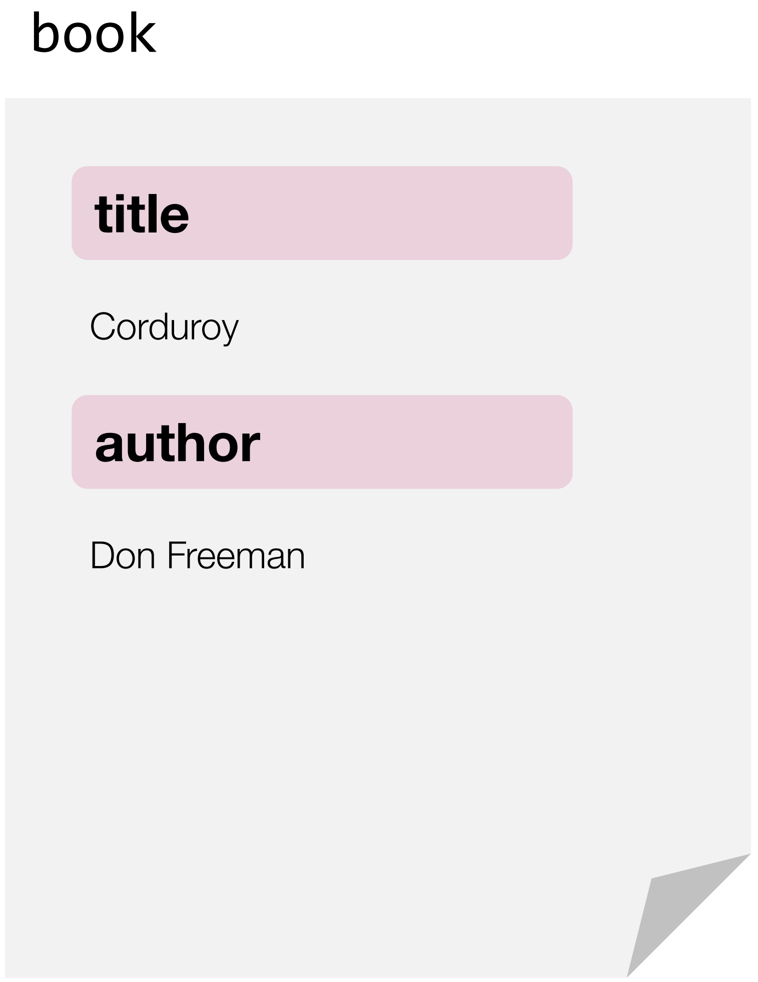
```
book = dict()
book = dict()
```



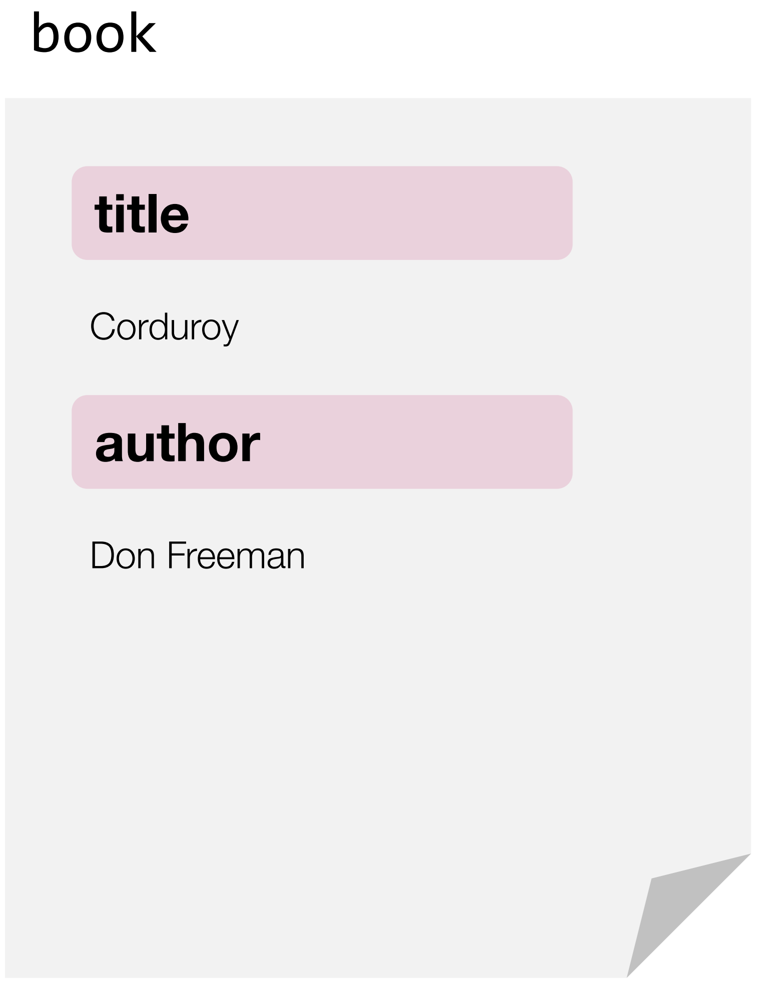
```
book = dict()
book["title"] = "Corduroy"
```



```
book = dict()
book["title"] = "Corduroy"
book["author"] = "Don Freeman"
```

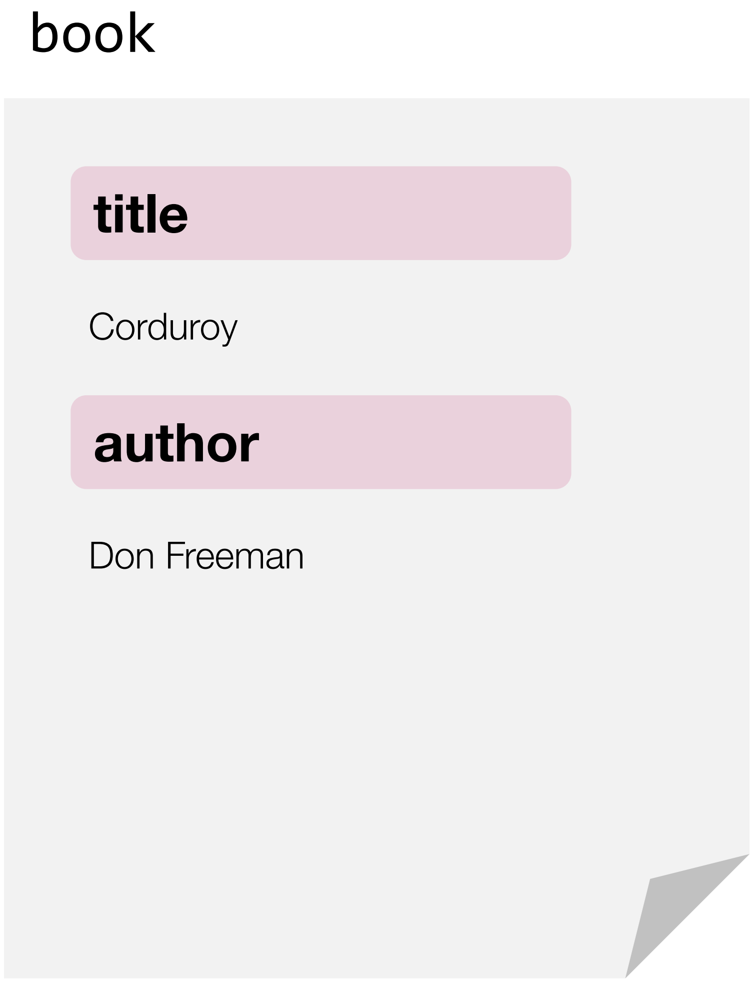


```
book = dict()
book["title"] = "Corduroy"
book["author"] = "Don Freeman"
print(book["title"])
```

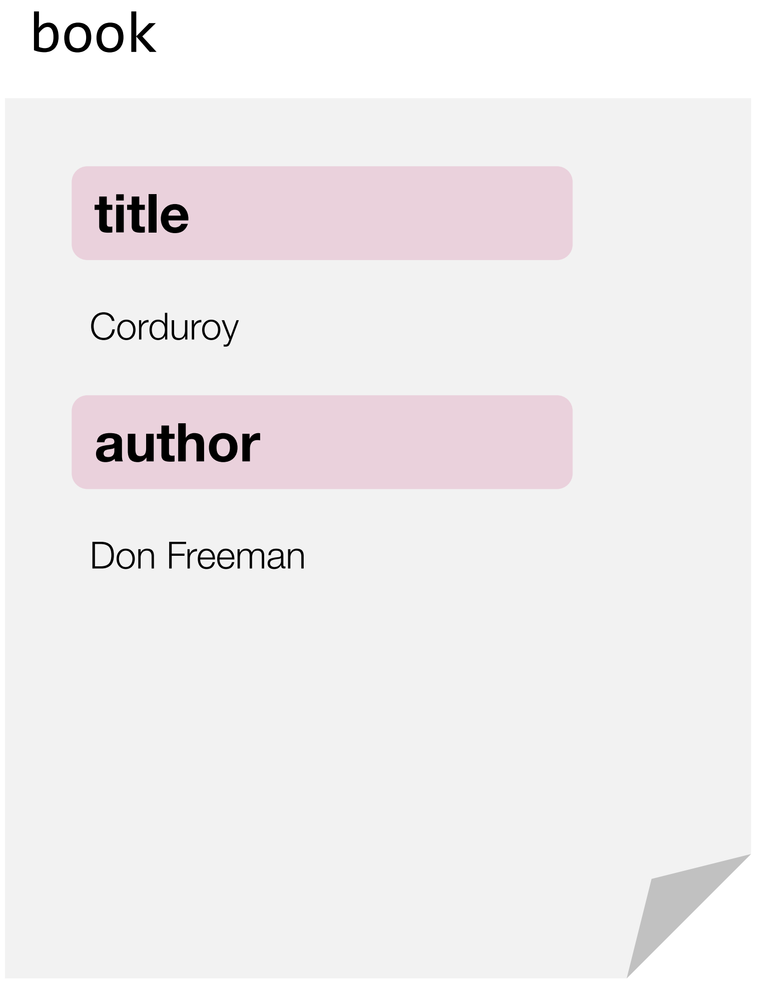


```
book = dict()
book["title"] = "Corduroy"
book["author"] = "Don Freeman"
print(book["title"])

# "Corduroy"
```

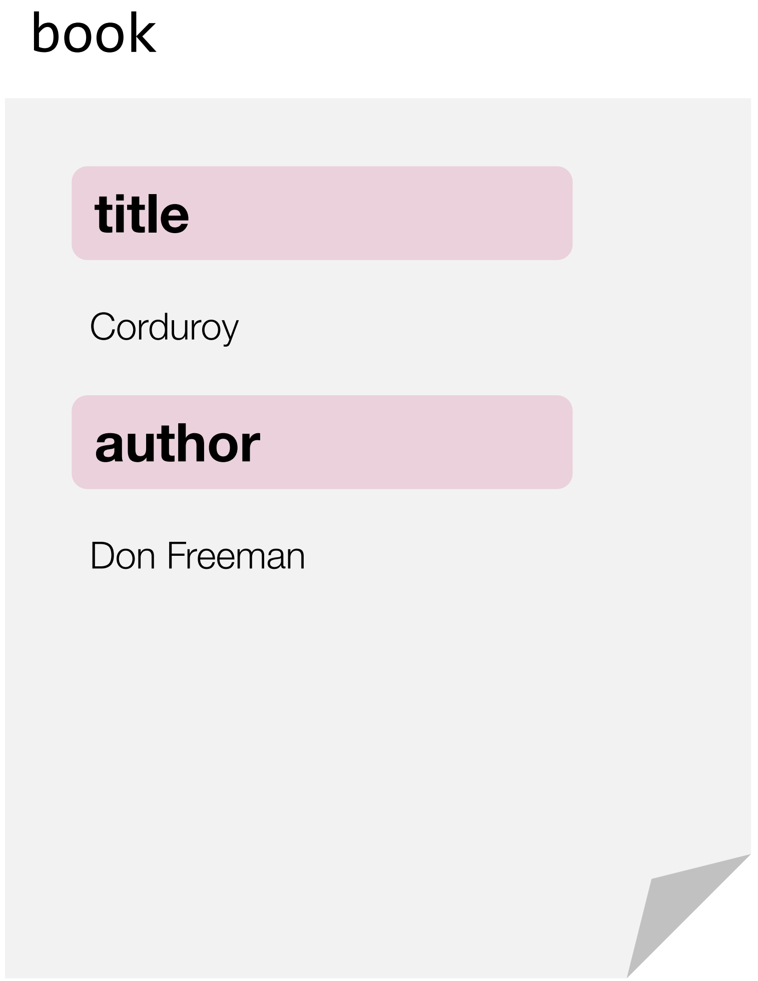


```
book = dict()
book["title"] = "Corduroy"
book["author"] = "Don Freeman"
print(book["Corduroy"])
```



```
book = dict()
book["title"] = "Corduroy"
book["author"] = "Don Freeman"
print(book["Corduroy"])

KeyError: 'Corduroy'
```



```
book = {
    "title": "Goodnight Moon",
    "author": "Margaret Wise Brown"
}
```

```
[{"title": "Goodnight Moon", ...},
 {"title": "Corduroy", ...},
 {"title": "Curious George", ...}]
```

```
[{"title": "Goodnight Moon", ...},
 {"title": "Corduroy", ...},
 {"title": "Curious George", ...}]
```

```
[{"title": "Goodnight Moon", ...},
 {"title": "Corduroy", ...},
 {"title": "Curious George", ...}]
```

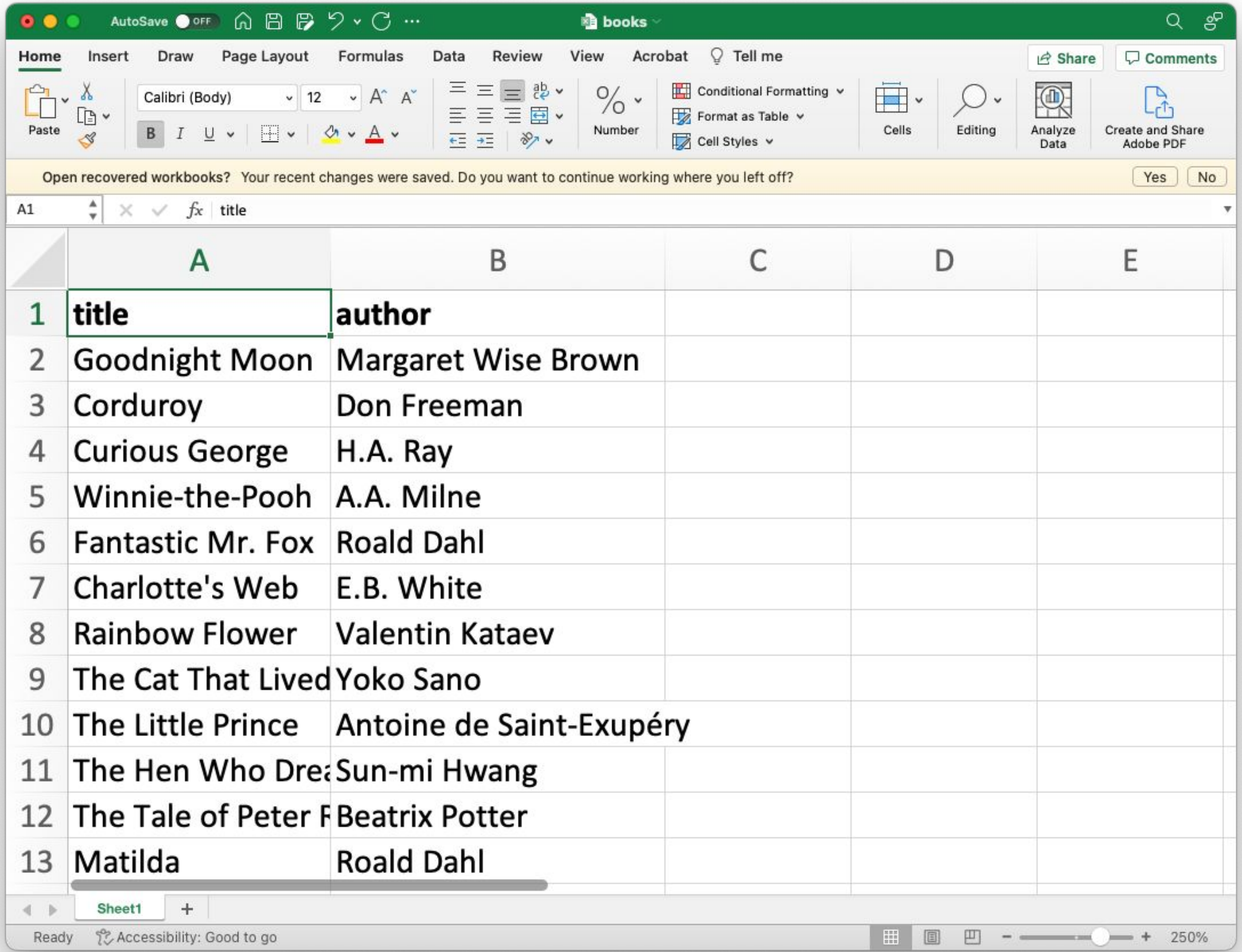
Shelf of Books

Download **books.py**.

Complete **books.py** so that a user is prompted to continue adding books (with a title and author) to a list of books.

If feeling more comfortable, try "sanitizing" the user's input by stripping whitespace and capitalizing each word in the title.

Libraries and Modules



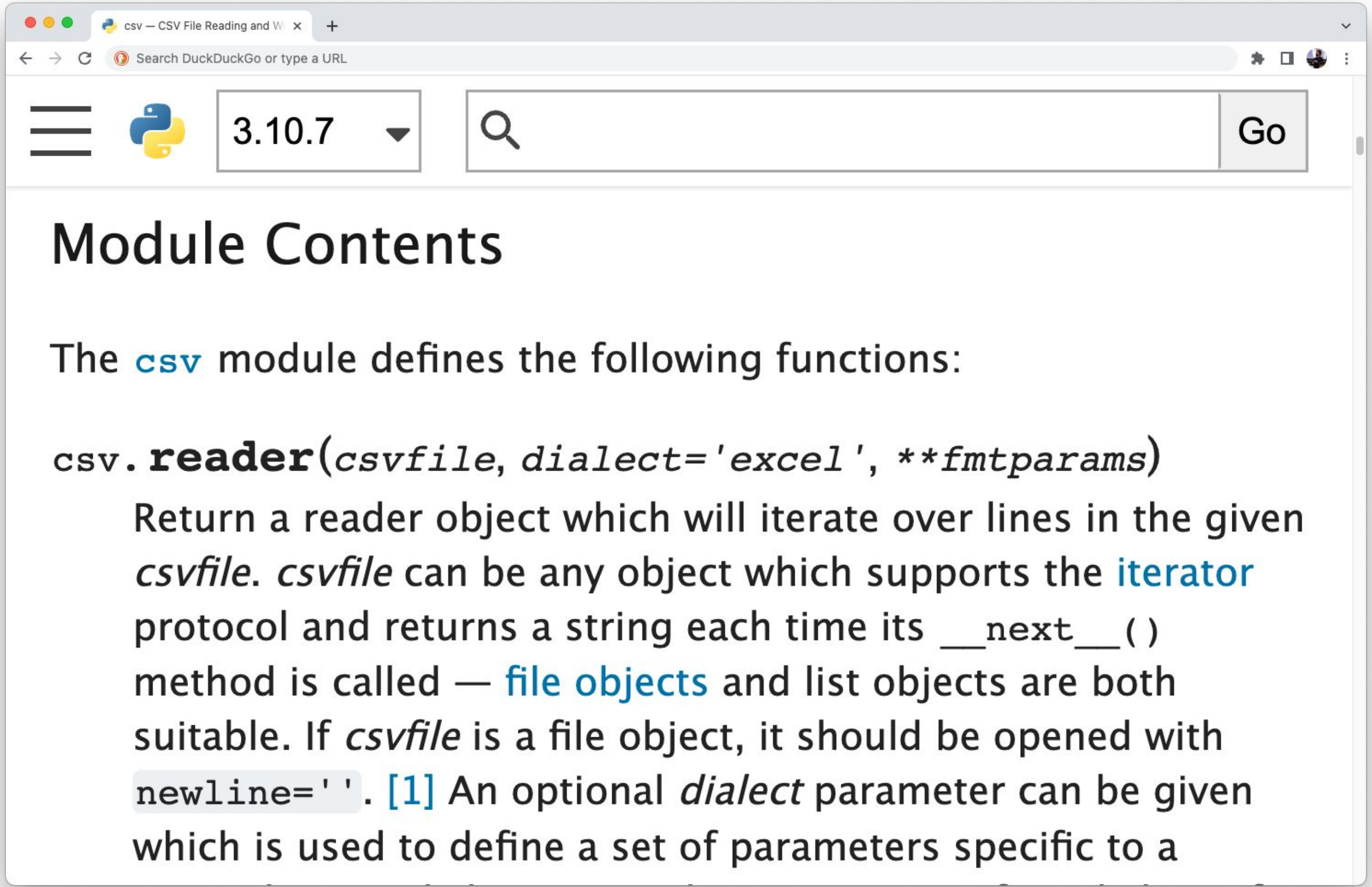
Open recovered workbooks? Your recent changes were saved. Do you want to continue working where you left off?

A1 title

	A	B	C	D	E
1	**title**	**author**			
2	Goodnight Moon	Margaret Wise Brown			
3	Corduroy	Don Freeman			
4	Curious George	H.A. Ray			
5	Winnie-the-Pooh	A.A. Milne			
6	Fantastic Mr. Fox	Roald Dahl			
7	Charlotte's Web	E.B. White			
8	Rainbow Flower	Valentin Kataev			
9	The Cat That Lived	Yoko Sano			
10	The Little Prince	Antoine de Saint-Exupéry			
11	The Hen Who Drea	Sun-mi Hwang			
12	The Tale of Peter F	Beatrix Potter			
13	Matilda	Roald Dahl			

books.csv

```
title,author
Goodnight Moon,Margaret Wise Brown
Corduroy,Don Freeman
Curious George,H.A. Ray
Winnie-the-Pooh,A.A. Milne
Fantastic Mr. Fox,Roald Dahl
...
```



Module Contents

The `csv` module defines the following functions:

`csv.`**`reader`**`(`*`csvfile`*`, `*`dialect='excel'`*`, `*`**fmtparams`*`)`

Return a reader object which will iterate over lines in the given *csvfile*. *csvfile* can be any object which supports the iterator protocol and returns a string each time its `__next__()` method is called — file objects and list objects are both suitable. If *csvfile* is a file object, it should be opened with `newline=''`. [1] An optional *dialect* parameter can be given which is used to define a set of parameters specific to a

```
import csv
import csv
```

csv

```
import csv
```

DictReader

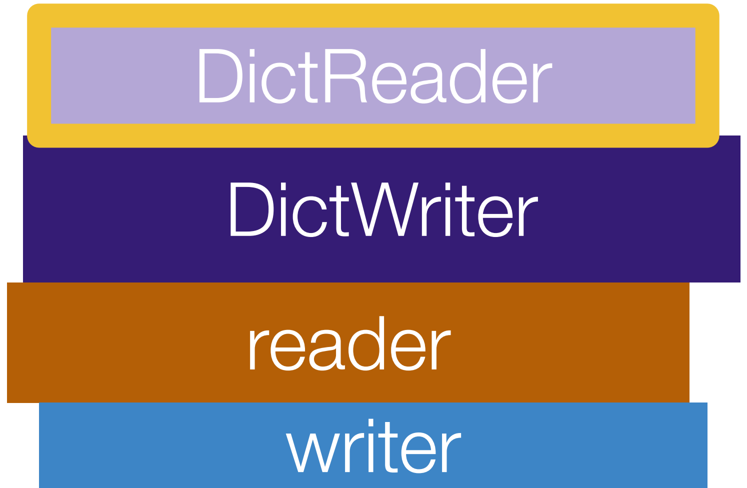
DictWriter

reader

writer

```
import csv

csv.DictReader(...)
```



```
import csv

csv.DictReader(...)
csv.reader(...)
```

DictReader
DictWriter
reader
writer

```
import csv
```

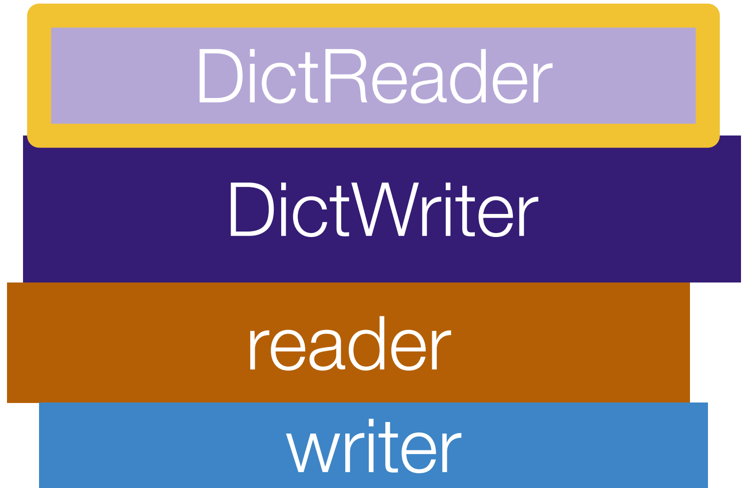
DictReader

DictWriter

reader

writer

```
from csv import DictReader
```



```
from csv import DictReader

DictReader(...)
```

DictReader
DictWriter
reader
writer

File Reading and Writing

```
with open(FILENAME) as file:
```

```
with open(FILENAME) as file:
    text = file.read()
```

```
with open(FILENAME) as file:
    file_reader = csv.DictReader(file)
```

```
with open(FILENAME) as file:
    file_reader = csv.DictReader(file)
    for row in file_reader:
```

```
with open(FILENAME) as file:
    file_reader = csv.DictReader(file)
    for row in file_reader:
        ...
```

Good reads

Download **reads.py** and **books.csv**.

Complete **reads.py** so that a user can build a list of children's books.

If feeling more comfortable, try allowing the user to eliminate books by certain authors (e.g., "Roald Dahl").

This was CS50