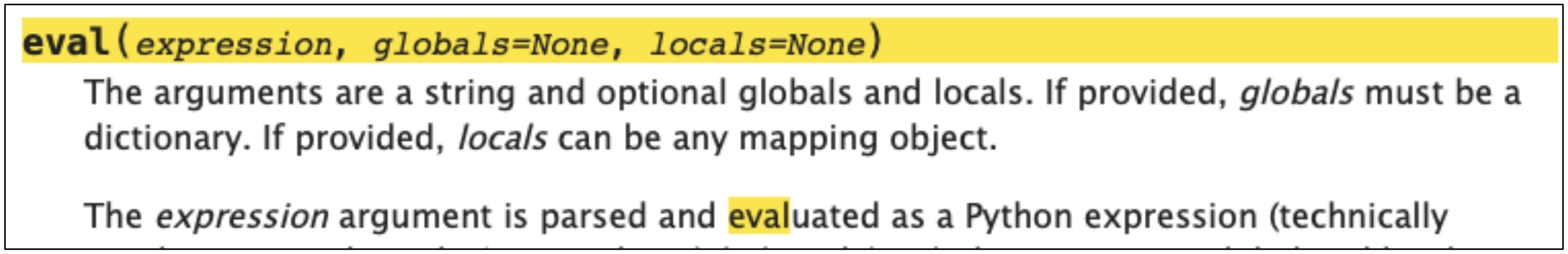
LING 388: Computers and Language

Lecture 9

Today's Topics

- Python:
  - eval()
  - sorting: sorted() and list.sort()
  - key= sort parameter
  - more on lists: stacks, queues and reversed() and .reverse()
- Homework 5

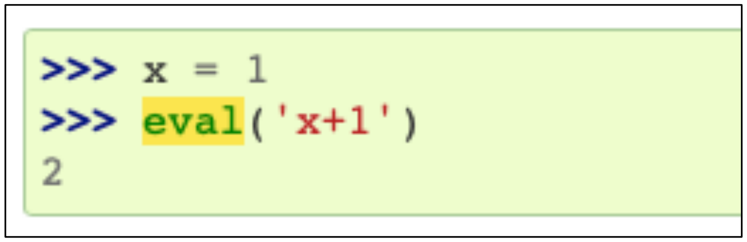
`eval()`



eval(*expression, globals=None, locals=None*)

The arguments are a string and optional globals and locals. If provided, *globals* must be a dictionary. If provided, *locals* can be any mapping object.

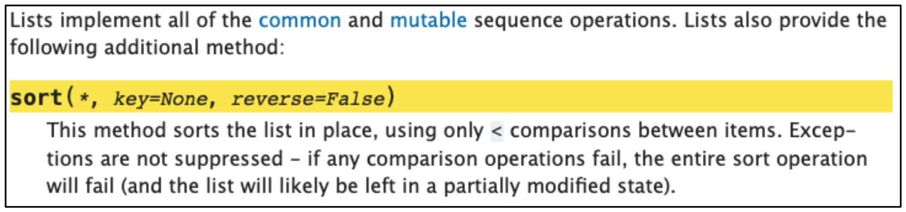
The *expression* argument is parsed and evaluated as a Python expression (technically



```
>>> x = 1
>>> eval('x+1')
2
```

`sorted(list) vs. list.sort()`

Mutable

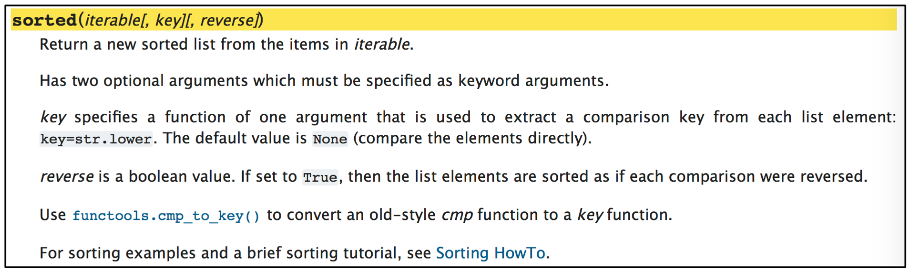


Lists implement all of the common and mutable sequence operations. Lists also provide the following additional method:

`sort(*, key=None, reverse=False)`

This method sorts the list in place, using only `<` comparisons between items. Exceptions are not suppressed – if any comparison operations fail, the entire sort operation will fail (and the list will likely be left in a partially modified state).

Fresh copy (non-mutable)



`sorted(iterable[, key][, reverse])`

Return a new sorted list from the items in *iterable*.

Has two optional arguments which must be specified as keyword arguments.

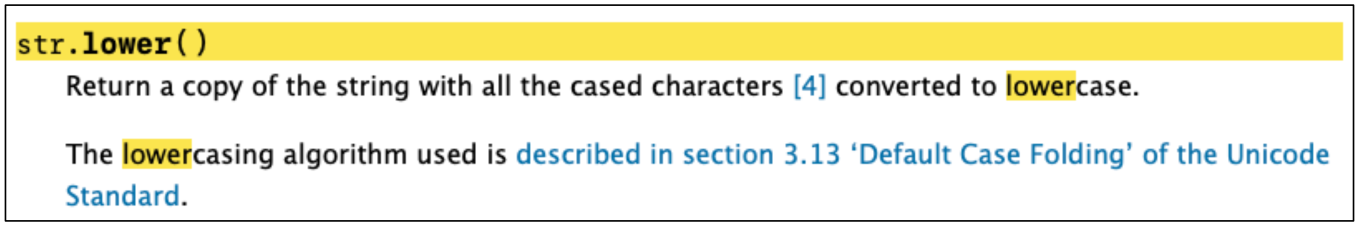
key specifies a function of one argument that is used to extract a comparison key from each list element: `key=str.lower`. The default value is `None` (compare the elements directly).

reverse is a boolean value. If set to `True`, then the list elements are sorted as if each comparison were reversed.

Use `functools.cmp_to_key()` to convert an old-style *cmp* function to a *key* function.

For sorting examples and a brief sorting tutorial, see Sorting HowTo.

- Let's talk about the key parameter!



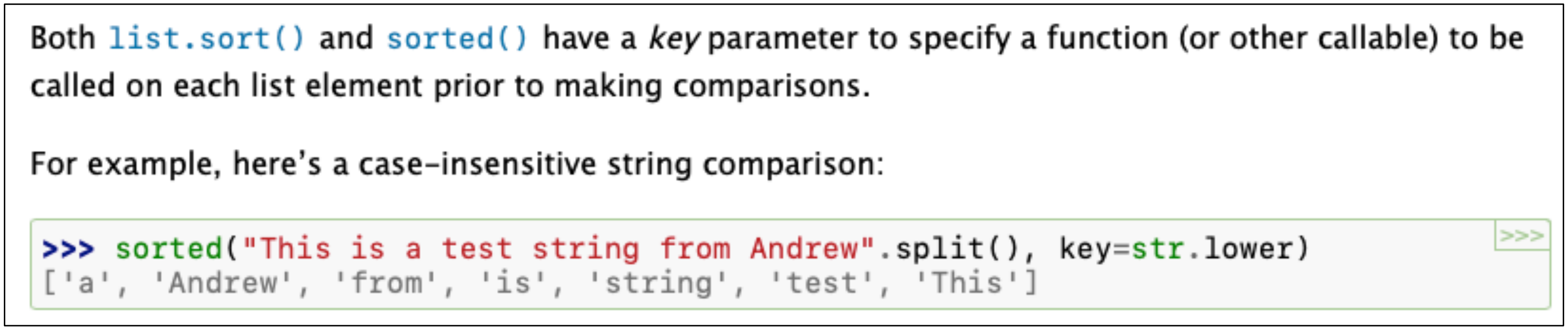
`str.lower()`

Return a copy of the string with all the cased characters [4] converted to lowercase.

The lowercasing algorithm used is described in section 3.13 ‘Default Case Folding’ of the Unicode Standard.

key= sort parameter

https://docs.python.org/3/howto/sorting.html



Both `list.sort()` and `sorted()` have a *key* parameter to specify a function (or other callable) to be called on each list element prior to making comparisons.

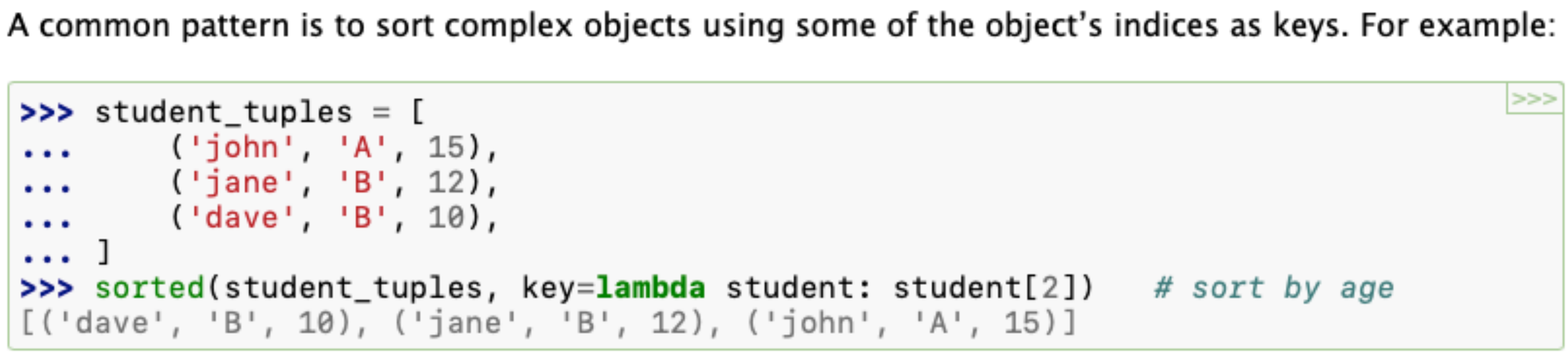
For example, here's a case-insensitive string comparison:

```
>>> sorted("This is a test string from Andrew".split(), key=str.lower)
['a', 'Andrew', 'from', 'is', 'string', 'test', 'This']
```

The value of the *key* parameter should be a function (or other callable) that takes a single argument and returns a key to use for sorting purposes. This technique is fast because the key function is called exactly once for each input record.

key= sort parameter

- Useful for sorting records



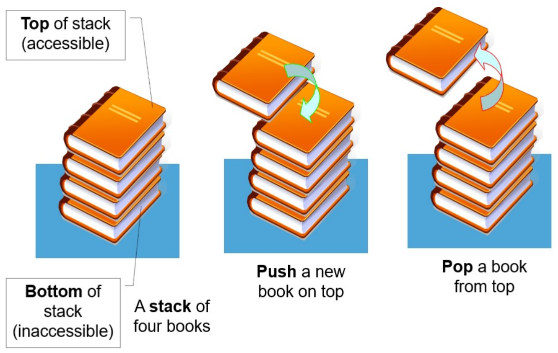
A common pattern is to sort complex objects using some of the object's indices as keys. For example:

```
>>> student_tuples = [
...     ('john', 'A', 15),
...     ('jane', 'B', 12),
...     ('dave', 'B', 10),
... ]
>>> sorted(student_tuples, key=lambda student: student[2])   # sort by age
[('dave', 'B', 10), ('jane', 'B', 12), ('john', 'A', 15)]
```

Python Lists

- Lists as stacks

https://visualgo.net/en/list?slide=4



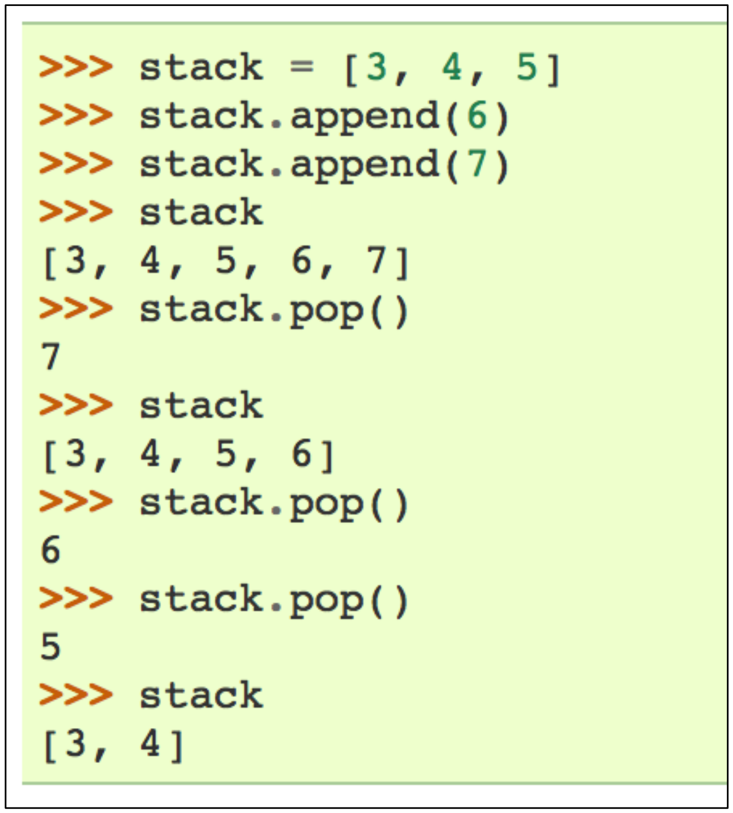
Only the top of the stack is immediately accessible

- Lists as queues

https://www.appcoda.com/ios-concurrency/

Front and rear are accessible

Python List as a Stack


```
>>> stack = [3, 4, 5]
>>> stack.append(6)
>>> stack.append(7)
>>> stack
[3, 4, 5, 6, 7]
>>> stack.pop()
7
>>> stack
[3, 4, 5, 6]
>>> stack.pop()
6
>>> stack.pop()
5
>>> stack
[3, 4]
```

- Note that .pop() removes from the right end and .append() adds to the right end.
- How to save the element of a list popped off the stack? Use a variable, e.g. x:
  - `x = list.pop()`
- Stacking operations are:
  - `list.append()`
  - `list.pop()`

Python List as a Stack

- Suppose we have

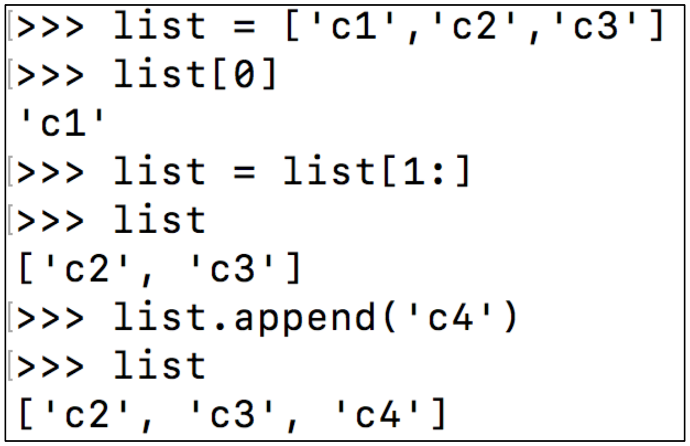
```
list = ['a', 'c', 'b']
```

- How do we flip the order of b and c using stack operations?
- **Answer**:

```
>>> x1 = list.pop()
>>> x2 = list.pop()
>>> list.append(x1)
>>> list.append(x2)
```

Python List as a Queue

EXAMPLE:



```
>>> list = ['c1','c2','c3']
>>> list[0]
'c1'
>>> list = list[1:]
>>> list
['c2', 'c3']
>>> list.append('c4')
>>> list
['c2', 'c3', 'c4']
```

- Method append() to add to right end of the queue
- `list[0]` gives us the head, i.e. left end, of the queue
- Note: `x = list[0]` saves the head of the queue into variable x
- `list = list[1:]` deletes the head of the queue from the queue
- Also can use `del list[0]`

Python List as a Queue

- Queuing operations are:
  - `list.append(newitem)`
  - `del list[0]`
  - `first = list[0]`

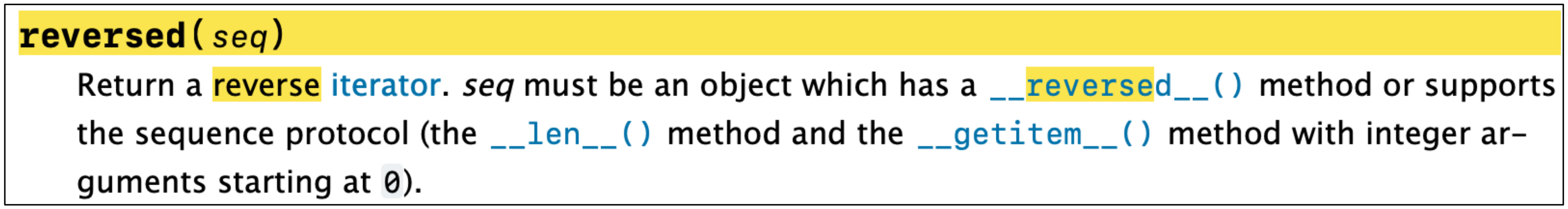
  - `first` = first in queue
- Note:

  `list`, `first` and newitem are variable names; *you can use any name you like*

- Recall stacking operations are:
  - `list.append(newitem)`
  - `top = list.pop()`

  - `top` = top of stack

reversed()



reversed(*seq*)

Return a reverse iterator. *seq* must be an object which has a `__reversed__()` method or supports the sequence protocol (the `__len__()` method and the `__getitem__()` method with integer arguments starting at `0`).

```
>>> reversed(['a','b','c'])
<list_reverseiterator object at 0x10f0fa850>
>>> list(reversed(['a','b','c']))
['c', 'b', 'a']
>>> for x in reversed(['a','b','c']):
...     print(x)
...
c
b
a
```

.reverse()

Similar to `.sort()` vs. `sorted()`:

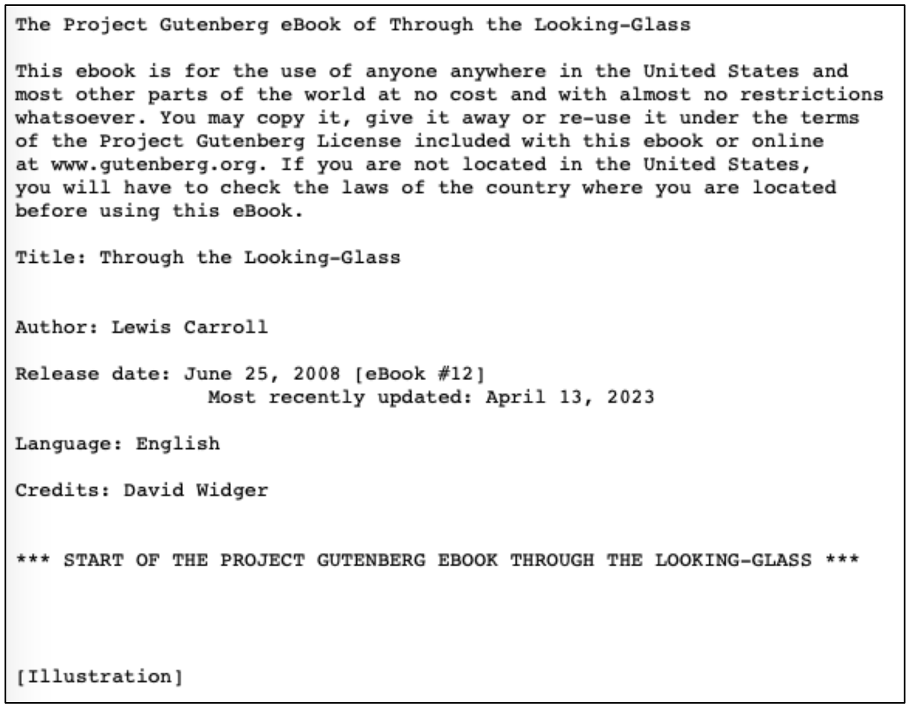
```
>>> ['a','b','c'].reverse()
>>> x = ['a','b','c']
>>> x.reverse()
>>> x
['c', 'b', 'a']
>>>
```

Homework 5

- *Through the Looking-Glass* (1872), by Charles Dodgson, AKA Lewis Carroll is a sequel to *Alice's Adventures in Wonderland* (1865).
- Step1:
  - Go to Project Gutenberg (www.gutenberg.org)
  - Find it and download the Plain Text (UTF-8 format) file
  - You might want to rename it to something memorable, e.g. `looking–glass.txt`
  - Put it in the same directory as where you run your Python

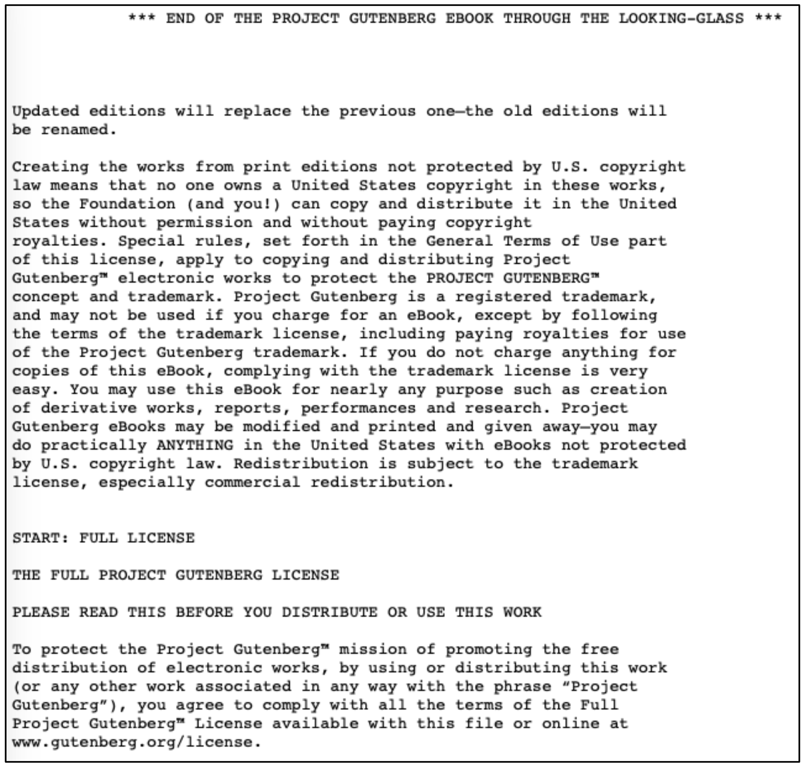
Homework 5

- Step 2:
  - Open the file in a text editor, e.g. NotePad (Windows) or TextEdit (macOS) etc.
  - Delete the lines that are NOT part of the book
  - Save the file as Plain Text

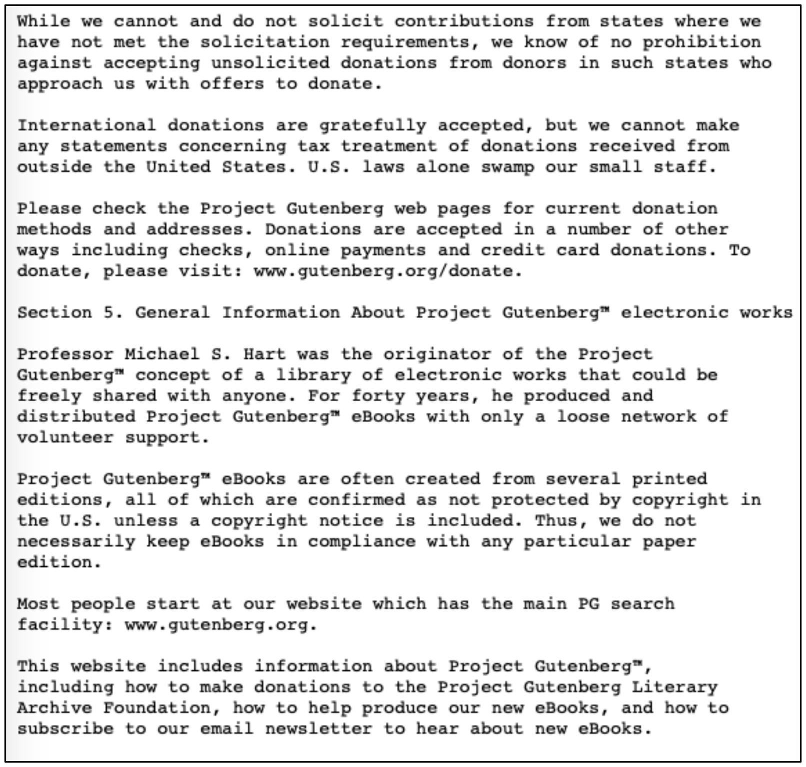


```
The Project Gutenberg eBook of Through the Looking-Glass

This ebook is for the use of anyone anywhere in the United States and
most other parts of the world at no cost and with almost no restrictions
whatsoever. You may copy it, give it away or re-use it under the terms
of the Project Gutenberg License included with this ebook or online
at www.gutenberg.org. If you are not located in the United States,
you will have to check the laws of the country where you are located
before using this eBook.

Title: Through the Looking-Glass


Author: Lewis Carroll

Release date: June 25, 2008 [eBook #12]
                Most recently updated: April 13, 2023

Language: English

Credits: David Widger


*** START OF THE PROJECT GUTENBERG EBOOK THROUGH THE LOOKING-GLASS ***



[Illustration]
```

Homework 5



```
          *** END OF THE PROJECT GUTENBERG EBOOK THROUGH THE LOOKING-GLASS ***



Updated editions will replace the previous one—the old editions will
be renamed.

Creating the works from print editions not protected by U.S. copyright
law means that no one owns a United States copyright in these works,
so the Foundation (and you!) can copy and distribute it in the United
States without permission and without paying copyright
royalties. Special rules, set forth in the General Terms of Use part
of this license, apply to copying and distributing Project
Gutenberg™ electronic works to protect the PROJECT GUTENBERG™
concept and trademark. Project Gutenberg is a registered trademark,
and may not be used if you charge for an eBook, except by following
the terms of the trademark license, including paying royalties for use
of the Project Gutenberg trademark. If you do not charge anything for
copies of this eBook, complying with the trademark license is very
easy. You may use this eBook for nearly any purpose such as creation
of derivative works, reports, performances and research. Project
Gutenberg eBooks may be modified and printed and given away—you may
do practically ANYTHING in the United States with eBooks not protected
by U.S. copyright law. Redistribution is subject to the trademark
license, especially commercial redistribution.


START: FULL LICENSE

THE FULL PROJECT GUTENBERG LICENSE

PLEASE READ THIS BEFORE YOU DISTRIBUTE OR USE THIS WORK

To protect the Project Gutenberg™ mission of promoting the free
distribution of electronic works, by using or distributing this work
(or any other work associated in any way with the phrase "Project
Gutenberg"), you agree to comply with all the terms of the Full
Project Gutenberg™ License available with this file or online at
www.gutenberg.org/license.
```



```
While we cannot and do not solicit contributions from states where we
have not met the solicitation requirements, we know of no prohibition
against accepting unsolicited donations from donors in such states who
approach us with offers to donate.

International donations are gratefully accepted, but we cannot make
any statements concerning tax treatment of donations received from
outside the United States. U.S. laws alone swamp our small staff.

Please check the Project Gutenberg web pages for current donation
methods and addresses. Donations are accepted in a number of other
ways including checks, online payments and credit card donations. To
donate, please visit: www.gutenberg.org/donate.

Section 5. General Information About Project Gutenberg™ electronic works

Professor Michael S. Hart was the originator of the Project
Gutenberg™ concept of a library of electronic works that could be
freely shared with anyone. For forty years, he produced and
distributed Project Gutenberg™ eBooks with only a loose network of
volunteer support.

Project Gutenberg™ eBooks are often created from several printed
editions, all of which are confirmed as not protected by copyright in
the U.S. unless a copyright notice is included. Thus, we do not
necessarily keep eBooks in compliance with any particular paper
edition.

Most people start at our website which has the main PG search
facility: www.gutenberg.org.

This website includes information about Project Gutenberg™,
including how to make donations to the Project Gutenberg Literary
Archive Foundation, how to help produce our new eBooks, and how to
subscribe to our email newsletter to hear about new eBooks.
```

Homework 5

- Step 3: load the file as a String into Python
  - *String stored as a variable* raw *below.*
  - `fh = open(`*`filename`*`)`
  - `raw = fh.read()`
  - what does `len(raw)` report?

Homework 5

- Step 4:
  - nltk has a `nltk.word_tokenize(string)` function to convert an English language String into a list of words.
  - `import nltk`
  - `words = nltk.word_tokenize(raw)`
  - what does `len(words)` report?

Homework 5

- Step 5:
  - Calculate the average number of characters per word of the entire book.
  - compare your answer to that for `carroll-alice.txt`
  - (See Exercise 2 from previous lecture.)

Homework 5

- Step 6:
  - Compute the word length distribution for `looking_glass.txt`.
  - do it both for the book AND the vocabulary of the book
  - see Exercises last lecture and nltk.FreqDist()
  - Compare the graphs with `carroll-alice.txt` (*same author*)
  - Are they similar or different?

Homework 5

- Submit to sandiway@arizona.edu
- SUBJECT: 388 Homework 5 *YOUR NAME*
- One PDF file only
    - include Python terminal and graph screenshots in your answer
- Deadline:
    - midnight Monday
    - we will review the homework on Tuesday