CS 111: Program Design I

Lecture 18: List methods; Starting Networks, Web, and getting text from the Web in Python

Robert H. Sloan & Richard Warner
University of Illinois at Chicago
October 19, 2019

UIC

Key list methods

- ls.append(item): add item to end of ls
- ls.pop(): remove and return element at end of ls
- ls.pop(i): remove and return element at index i in ls
- ls.remove(item): remove first occurrence of item from ls
- ls.insert(i, item): insert item into ls at position i
  - sliding elements of ls[i:] all one position right to make room
- ls.sort(): Move elements of ls so that ls is in sorted order
  - Requires all elements to be comparable

- *All those methods modify* ls
- *Only* pop *among those methods has a return value*

What is value of a after code runs?

```
a = ['a', 'b', 'c', 'd']
a.remove('b')
a.pop(2)
```

A. `['a', 'b']`
B. `['c', 'd']`
C. `['a', 'c']`
D. `['a', 'd']`
E. Nothing; code generates an error

What is value of a after code runs?

```
a = ['a', 'b', 'c', 'd']
a.remove('b')
a.pop(2)
```

A. `['a', 'b']`
B. `['c', 'd']`
C. **`['a', 'c']`**
D. `['a', 'd']`
E. Nothing; code generates an error

What is value of a after code runs?

```
a = ['a', 'b', 'c', 'd']
a.pop(2)
a.remove('b')
```

A. `['a', 'b']`
B. `['c', 'd']`
C. `['a', 'c']`
D. `['a', 'd']`
E. Nothing; code generates an error

What is value of a after code runs?

```
a = ['a', 'b', 'c', 'd']
a.pop(2)
a.remove('b')
```

A. `['a', 'b']`
B. `['c', 'd']`
C. `['a', 'c']`
D. **`['a', 'd']`**
E. Nothing; code generates an error

x == y?

```
x = ['a', 'b', 'c', 'd']
y = x.pop()
y == x
```

a. True
b. False

What is value of a after code runs?

```
a = ['a', 'b', 'c', 'd']
a.insert (4, 'e')
a.append (‘e’)
a.pop()
a.remove (‘e’)
```

A. `['a', 'b', 'c', 'd']`
B. `[ ]`
C. This code gives an error
D. `['a', 'b', 'c', 'd', 'e']`
E. `['b', 'c', 'd', 'e']`

What values of a, b, c after code runs?

- a = ['c', 'd', 'a', 'b']
- a.sort()
- b = [20, 5, 32, 1]
- b.sort()
- c = ['c', 'd', 32, 1]
- c.sort()

- Are the values :

  a = ['a', 'b', 'c', 'd']

  b = [1, 5, 20, 32]

  c = [1, 32, 'c', 'd']

a. Yes
b. No, other values
c. No, code gives error

Winter is coming

~~Winter~~ Midterm 2 is coming

- 1 week from Thursday
- Thu., Nov. 7

- Broadly similar to Midterm 1
- Comprehensive, but heavily weighted to material since last midterm
- Mildly more points on legal material than Midterm 1

COMPUTER NETWORKS, INTERNET, AND ACCESS IN PYTHON

Goals

- Learn about Internet
- Learn about how to access Internet directly from Python Program
- Use this knowledge plus all we've learned about lists to finish web crawler
- Don’t violate CFAA!

Opening a URL and reading it

- The 60 second "standing on one foot" version so you can do any basic crawler lab or project

```
import urllib.request as ur

connection = ur.urlopen("https://www.cs.uic.edu")
content = connection.read()
connection.close()
```

Networks: 2 or more computers communicating

- Networks formed when distinct computers communicate via some mechanism
  - Inside one computer: 0/1 voltages
  - Almost never on network: too hard to make work over distance
  - More commonly frequency
    - In 2019 usually not sound frequency, but modems in the dialup internet era did indeed use sound

Networks everywhere

- All non-ancient cars
  - Modern car is collection of computers–Controlling air flow, gas flow, making airbags work–that communicate, and happen to have 4 wheels attached. ☺
- Likewise planes
- (Almost?) all of you have a network in your home or dorm

Networks organized in layers

- Recurring theme throughout Computer Science: Recall hierarchical decomposition
- For networks, bottom level is physical substrate
  - What are signals being passed on? Radio? Wire?
- Middle levels: How data is encoded
  - Frequencies of sound waves or radio waves or ??? for encoding 0's and 1's
  - Transmit bit at a time? Byte at a time? Larger *packets* of bytes?

Higher levels of networks

- *Protocol* of communication
  - How do I *address* a particular computer I want to talk to? Or many computers?
  - How do I tell a computer that I want to talk to it? That I'm starting to send it data? What it's supposed to do with the data I send it? When we're done?

Internet: Collection of networks

- Internet: network of networks
- Inside your home, probably have local network so your computers can talk to one another
  - Likely uses wireless base station
  - You can probably reach printers and/or copy files between your computers
- And you connect your home network (via an *Internet Service Provider (ISP))* to global internet
- So your home network part of Internet

Internet based on agreements on encodings

- Internet built on set of agreements about
  - How computers addressed
    - Set of four 1-byte numbers (IPv4), or eight 2-byte numbers (IPv6). E.g., 128.248.155.15
    - Way of associating these numbers with *domain names*, like www.uic.edu using *domain name servers*
    - How computers will communicate
    - That data will be split into packets with various pieces in them
    - That computers will format their data and talk to one another using TCP/IP protocol
  - How packets are routed around network to find their destination

High-level protocols

- All that just lets us pass data back and forth
  - What does data *say?*
  - What does data *do?*
- One of earliest applications (involving 2 high-level protocols) was *email*
- Another: *File transfer Protocol (FTP)* allows computers to move files between each other
  - Defines what one side says to other when copying file over (e.g., "STO filename") and how file will be encoded
  - Not widely used in 2019, but big when I took CS 1

Internet vs. Web? Which is most correct?

A. Internet and (World Wide) Web are very roughly synonymous
B. The Web is one service of the Internet
C. The Internet is one service of the Web
D. There is only a very loose connection between the Internet and the Web

Internet is not new

- Internet ≠ web!
- Internet originally set up for military applications
  - Feature: Packets still reach their destination even if part is destroyed, damaged, or censored
- Internet established in 1969 as military research project with four locations, current key protocol TCP/IP dates to 1975; declared universal standard for military computing by US DoD in 1982

The Web (Web ≠ Internet!)

- Prof. Sloan heavy user of Internet 1985 onward
- Web, built on top of Internet, only came into being in December 1990, and was extremely obscure until 1993
- Web is (again) set of agreements, started by Sir Tim Berners-Lee
  - On how to refer to everything on Web: *URL (Uniform Resource Locator)*
  - On how to serve documents: *HTTP (HyperText Transfer Protocol)*
  - How documents formatted: HTML (HyperText Markup Language)

Hypertext

- Nonlinear text that links to other text and graphics via links
- You know it well!
- Some limited implementations, e.g., Apple Hypercard, a little before web, but
  - The read a little, click and read some more, then click, then …. use of hypertext didn't exist when your parents and I were in school

HyperText Transfer Protocol (HTTP)

- HTTP defines *very* simple protocol for how to exchange information between computers (on top of internet protocols)
- Defines pieces of the communication
  - Web **server** is waiting for **client** in listening state
  - What resource do you want?
  - Where is it?
  - Okay, here's the type of thing it is (HTML, JPEG, whatever), and here it is
- And words the computers say to one another:
  - Simple, e.g., "GET", "PUT", and "OK"

Uniform Resource Locators (URLs)

- URLs allow us to reference any material anywhere on Internet
  - Strictly speaking, any computer providing a protocol accessible via URL
  - Just putting your computer on Internet does not mean that all of your files are accessible to everyone on Internet
- URLs have four parts:
  1. Protocol to use to reach this resource,
  2. Domain name of the computer where the resource is,
  3. Path on the computer to the resource,
  4. And the name of the resource (optional; notion of default file).
  - (And optionally after that can be parameters introduced with a ?)

Example URL

- E.g., http://www.cs.uic.edu/CS477 3 parts:
  1. Protocol: http (usually http or https)
  2. Host name: www.cs.uic.edu
  3. Path: /CS477 (location of files on host)
- https://www.cs.uic.edu/~sloan/papers/SloanWhyCS2014.pdf
  1. Protocol: https
  2. same host name
  3. Path: /~sloan/papers
  4. Filename: SloanWhyCS2014.pdf

Path also optional

- Web servers (programs that understand HTTP protocol) typically have special directory that they serve from
  - Files in that special directory directly reachable without specifying path

Web browsers are *clients*

- Your Web browser is called a client accessing a Web server
- Programs like Chrome, Firefox, or Safari understand a lot about Internet protocols
  - They know how to interpret HTML and display it graphically
  - If HTML references other resources, like JPEG or PNG pictures, client fetches them and displays them as appropriate
  - Client also knows details of HTTP (and perhaps mailto, ftp, and some others)

Browser not only way to use Internet

- Python (and other languages) have modules that allow you to use these protocols
  - In Python, can read any URL as if it was a file
- In Python 3, key library is urllib.request
  - (Other parts of urllib useful for, e.g., parsing webpage text)

Importing urllib.request

- **Advice**: If module has dot in its name like urllib.request, matplotlib.pyplot, always import it *as* something. Else Python can get confused about multiple dots when you go to use functions inside it. Thus:

```
import urllib.request as ur
```

Opening a URL and reading it

- The 60 second "standing on one foot" version so you can write web crawler

```
import urllib.request as ur

connection = ur.urlopen('https://www.cs.uic.edu')
content = connection.read()
connection.close()
```

Status

- connection returned from urllib is a "HTTPResponse" object and has a status attribute, which should be 200 if all went well
- In crawling, if you run into difficulties, try:

```
import urllib.request as ur
connection= ur.urlopen('https://www.cs.uic.edu')
if (connection.status == 200):
    content = connection.read()
    <rest of real work here>
connection.close()
```

Reading the connection

- read() method of connection works just like read() method of files
  - Almost: this read will give you a sequence of bytes, which can in many but not all cases be treated as a string. Can force it to be 100% string instead of merely pretty string-like by
  - `content = str(connection.read())`
    - using type conversion function str()

WEB CRAWLER AGAIN

Two bits of useful Python syntax

- Don't need either one the web crawler but will make it a bit prettier:
  - break
  - list as condition for if or while

Break

- Causes immediate termination of innermost enclosing while or for loop
- Typical usage:

```
# We are inside a while or for
if <some particular case>:
    break
```

- Use carefully! Can make code very hard to read

Example

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, 'equals', x, '*', n//x)
            break
        else:
            # loop fell through without finding  factor
            print(n, 'is a prime number')
```

Which of these will exit when x is initially 9?

A

```
while (x%2 == 1 and x%3 == 0):
    x = 9
```

B

```
while True:
    if (x%2 == 1 and x%3 == 0):
        break
    x = 9
```

C. Both D. Neither E. I don't know

Continue

- Continue continues with *next* iteration of loop *instead of finishing current iteration*

Continue example

```
for num in range(2, 6):
    if num % 2 == 0:
        print("Found an even number", num)
        continue
    print("Found a number", num)
```

- Found an even number 2
- Found a number 3
- Found an even number 4
- Found a number 5