• Questions/Announcements

• Python and the web
• More practice with lists
• Limits of computation
• Final Review (Q/A)
Python urllib package

open a web page for reading

returns a file-like object that you can use exactly as if you were reading a local file.
Python urllib package

```python
import urllib.request

def accessWeb(urlname):
    '''
    
    
    with urllib.request.urlopen(urlname) as myf:
        info = myf.read()
        d_info = info.decode('utf-8')
        print(d_info)
        return #None
```
Python urllib package

import urllib.request

def readTemp(airport):
    """(str) -> None
    check NOAA webpage for temperature at
    the input airport (v. API)KEUG, KCVO, KDEN, KLGA, etc.
    For example, >>> readTemp('KEUG')
    ""
    urlname = 'http://www.srh.noaa.gov/data/obhistory/'+airport+'\.html'

    weatherf = urllib.request.urlopen(urlname)
    wfinfo = weatherf.read()
    decoded_wfinfo = wfinfo.decode('utf-8')
start_index = 0
# 4th table
for _ in range(4):
    start_index = decoded_wfinfo.find('<table', start_index + 1)

# 4th row
for _ in range(4):
    start_index = decoded_wfinfo.find('<tr>', start_index + 1)

# 7th data entry
for _ in range(7):
    start_index = decoded_wfinfo.find('<td>', start_index + 1)

current_temp = decoded_wfinfo[start_index+4:start_index+6]
print('The current temperature is %s degrees.' % current_temp)
# example - updating items in a list (map)

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

for item in e:
    print(item * 2)
# example - updating items in a list (map)

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

for item in example:
    print(item * 2)

for item in example:
    item = item * 2
# example - updating items in a list (map)

def example:
    example = ['a','b','c','d']

    for item in example:
        item = item * 2

    for i in range(len(example)):
        example[i] = example[i] * 2
• Questions/Announcements

• Python and the web
• More practice with lists
• Limits of computation
• Final Review (Q/A)

Days of programming can save you hours of planning.
with open('fitbitabbrevexample.csv', 'r') as fitfile:

with urllib.request.urlopen(urlname) as weatherf:
with open('movies.txt', 'r') as movief:
    moviedata = movief.read()

with urllib.request.urlopen(urlname) as weatherf:
    webdata = weatherf.read()
example = 'Abcd ezh abbccd eezz 99 xyz abccc'
example = 'Abcd ezh abbccd eezz 99 xyz abccc'

i = 0
for ct in range(3):
    i = example.find('z', i + 1)
example = 'Abcd ezh abbccd eezz 99 xyz abccc'

i = 0
for ct in range(3):
    i = example.find('z', i + 1)

temp = example[i + 2 : i + 4]
# example - nested loops

```
outer = ['Li', 'Na', 'K']
inner = ['F', 'Cl', 'Br']

for metal in outer:
    for halogen in inner:
        print(metal + halogen)
```
# example - looping over nested lists

eelts = [['Li', 'Na', 'K'], ['F', 'Cl', 'Br']]

for item in elts:
    for elt in item:
        print(elt)
# example - looping over nested lists

global outer = ['Li', 'Na', 'K']
global inner = ['F', 'Cl', 'Br']

for metal in outer:
    for halogen in inner:
        print(metal + halogen)
def sort_and_reverse(li):
    """return li sorted and reversed""
    li = li.sort()
    li = li.reverse()
    return #None

>>> markers = ['Emb', 'Him', 'Unc', 'Lon']
>>> markers = sort_and_reverse(markers)
>>> markers
def sort_and_reverse(li):
    """return li sorted and reversed""
    li.sort()
    li.reverse()
    return None

>>> markers = ['Emb', 'Him', 'Unc', 'Lon']
>>> markers = sort_and_reverse(markers)
>>> markers
def sort_and_reverse(li):
    """return li sorted and reversed"""
    li.sort()
    li.reverse()
    return #None

>>> markers = ['Emb', 'Him', 'Unc', 'Lon']
>>> sort_and_reverse(markers)
>>> markers
The Limits of Computation

• Technological Barriers
The Limits of Computation

• Technological Barriers

• Computational Limits
  
  – Ambiguous/ill-defined problems
The Limits of Computation

• Technological Barriers

• Computational Limits
  
  – Ambiguous/ill-defined problems
  
  – Natural language
The Limits of Computation

• Technological Barriers

• Computational Limits
  – Ambiguous/ill-defined problems
  – Natural language
  – Intractable problems
The Limits of Computation

• Technological Barriers

• Computational Limits
  – Ambiguous/ill-defined problems
  – Natural language
  – Intractable problems
  – Theoretical limits, e.g., undecidability
Computational Problem Solving

TASK → Computational Thinking →
ALGORITHM → Coding
PROGRAM → Execute
DESIRED OUTCOME

Programming = Computational Thinking + Coding
Why (should you learn to) program?

(1) Gain fluency in computing which is ubiquitous in our lives today.
(2) Gain a powerful problem solving tool.
(3) Programming is a useful intellectual exercise which develops or enhances valuable transferable thinking skills, including logical, creative, and design skills.
(4) Programming is a fundamental part of computer science for minors/majors/professionals, and provides exposure to many topics in the field of computer science.
Exploring Python Further

Python help()

Additional Documentation ... Beginner’s Guide to Python ... Learning Python

python.org ... Documentation

Library Reference ... Python Standard Library

Class website – Links page
Summary

• testing and debugging
• lists – mutable, operators, functions, methods
• loops
  – indefinite, definite
  – while, for, repeat (for/range)
• files and urllib