JSON(Javascript obj. notion)
-web serv, and APIs use JSON
-key/val pairs
-json.loads=returns python obj, from JSONstring
-json.dumps= returns JSONstring from pyth. obj.(dict/list usually) (usually list of dicts)
ex.
```
{
"tweet": [{
   "created_at": "2001",
   "id_str":"89",
   "text": "1√ Today we",}]
}
```
Steps:
1.import json
2. open file/read it
3. use json.loads()
4.use json.dumps()

Git

(version control system for tracking changes in files)
3 stages: committed, modified, staged
3 sections: Git direct, working tree, staging area
git init – initializes repository
git remote add (alias/name) (url) - est connection between local file and git rep.
git add . –adds all files in folder – stages commits
git commit –m 'messge' – commit new content to repo
git push – publishes changes
git pull <remote><branch>- download changes and merge
git clone –clones existing repo
git status –changed files in local directory
git diff – changes to tracked files
git log- shows all commits- starting with newest
git remote –v –list all currently configured remotes
branching- work on diff versions of repo @ 1 time
git fork – create a copy of a lab under own account
git commit –am "message" – commit syntax for all changes
-when clone repo on local workstation, can't contribute back to upstream unless "contributor"
github=website
git=VCS
ex.
code for new rep:
git init
git remote add (alias) (url)
git add .
git commit -m "comment"
git push origin master

RegEx
import re
f=open(textfile.txt)
x=f.read()
z=findall("insert regex", x(reference variable that stores text file))
Search for string format in string
^ - Matches the beginning of a line
[^]- Not in
$ - Matches the end of the line
\-need this exact char.
. - Matches any character
\s - Matches whitespace
\S - Matches any non-whitespace character
\w-letters +numbers
\W +neither
\Z- end of string
\d- digit
\D- non-digit
\A-start of string
***** - Repeats a character zero or more times
***?** - Repeats a character zero or more times (non-greedy)
+ - Repeats a character one or more times
+? - Repeats a character one or more times (non-greedy)
[aeiou] - Matches a single character in the listed set
[^XYZ] - Matches a single character not in the listed set
[a-z0-9] - The set of characters can include a range
(- Indicates where string extraction is to start
) - Indicates where string extraction is to end
{m}- exactly m occurrences
{m, n}- from m to n, m defaults to 0, n to infinity
-- Use re.search() to see if a string matches regular expression- returns a True/False
-- Use re.findall() to **extract portions** of a string to a regular expression- returns a list of substrings that match regex
ex.
```
f=open('mbox.txt')
for line in f:
   line=line.rstrip()
   if re.search('^From:', line):
            print(line)
```
email: [a-zA-Z0-9]\S*@\S*[a-zA-Z]
or \S+@\S+
website:
http[s]?://[a-z0-9]+[.].[\S]+
or ^\S+[.]\S+[.]\S+

Working w File Basics

-sequence of lines
-break file into lines using \n
file=open("file.txt", "r")
-open returns file "obj"
-read returns as string
readlines(): returns entire contents as a list of strings, each item in list=1line in file
readline(): reads one line from file and returns string
.rstrip(): gets rid of white at front and end
.find(): looks for occurrence of string w in another string, returns position of string or -1 if not found
"w" mode:if file exists, clears out old data
-if doesn't exist, new one created
-must use file.close()

HTTP(hypertext transport protocol)
-rules that allow browsers to retrieve web docs from server over internet
-used to transfer hypertext pgs.from server to browser (google)
HTML(hyper text markup language)
-used to structure doc. and provide content to be converted into hypertext
-uses set of tags
Attributes: provide additional info about element
(always specified in start tag, come in name/value pairs)
ex. <img src= '**filename**'>
<h1>: largest heading

:line break
\:end tag
<a href= www.com a>
Socket: like file except single socket provides 2 way connection
http: protocol
www.com: host
page1.htm: document
Spider: search engines scrape web pages
OAuth: authorization protocol/set of rules that allow 3rdparty website or app. to access user's data w out user need to share login credentials

API Requests

1 program makes set of services available to other apps. and publishes API (the rules) that must be followed
-Call data from internet
1. make request by calling requests.get(baseurl, params={})
2. extract content from response obj. by accessing the .text attribute and calling json.loads if string is in json format
3. process data extracted, may required nested

web scraping: program pretends to be browser and retrieve web
uses: monitor site for new info, get your own data out of sites, data(social data)

BeautifulSoup

pulls data out of HTML/ XML files
1. use urlib to read pgs.
2. use BS to extract

add to this

Get Method for {}

used to check if key in dict, assumes value if key is not
ex.
```
if x in counts:
  x= counts[name]
else:
  x=0
x=counts.get(name, 0)
```
Finding which occurs most
Ex.
```
count= {}
names = [list of names]
for x in names:
   if x not in counts:
      counts[x]=1
  else:
 counts[x]= counts[x]+1
```
Counting words in text
```
count={}
words=line.split()
for word in words:
count[word]=count.get(word,0) +1
```

List Methods

.append() – adds a new item to the end of a list
list.append(item)
.insert() – inserts item at position given
.reverse –reverses list order
.sort – modifies list to be sorted
.index () – returns the position of first occurrence of item
.count() – returns number of occurrences of item
.remove() – removes first occurrence of item
.join() – joins items of a list to a string
del listname[ ] – removes item from list by position
sorted(), sum(), max(), len()

String Methods

.upper() – returns all uppercase
.lower() – returns all lowercase
.strip() – removes whitespace
.count() – returns number of occurrences of item
.find() – leftmost index where item is found; if item is not in string, returns -1
.index() – same function as find; returns runtime error if item is not found
.split() – splits a string to a list of strings (cant split list)

Range

range() – function that produces a list of numbers
**Starts at 0
range(len()) – provide bound for range function; pattern for traversing sequence by position

Slicing

up to but not including
(:5) or (5:) -1st= start to 4th
x[::-1]-returns in reverse

Dictionaries

Key: data item mapped to a value in dict
Value: value associated to a key
Items are key-value pairs
.keys () – returns list of keys
Adding or modifying a key:
mydict['key'] = value
Deleting a key:
del mydict['key']
.items returns a tuple key-value
sorted(dic)returns list sorted keys
-order=unpredictable, "bag"
indexing in dict w values that are dicts: dict["e"]["f"]-indexes to inner value
-doesnt matter what type of obj value is, follow indexing

Dictionary Accumulation*

Use to accumulate maximum / minimum value, or common key
*Accumulate values within a dictionary that already exists
1. *Initialize* accumulator variable
2. *Iterate* items in sequence
3. *Update* accumulator variable
Dictionary accumulation has multiple accumulator variables; stored as values in a dictionary

Test Cases:expresses

requirement for program
-test asserts something about state of program@part. point
unittest: built in model for writing/ running test cases

Tuples

-immutable
-can be any type indexed by #
-comparable/washable, can sort lists of them and use tuples as key values in dicts.
-() values separated by ,
-can put on left side x,y,z=1,2,3
-can't use: .sort(), .append(), .reverse()

Tuples & Dicts:

.item(): gives list of key+value pairs(tuples)
-converting dict. to tuples is a way to output contents of dict. sorted by keys(key, val)
```
d={}
t=list(d.items())
t.sort
OR for key, val in list(d.items()):
        print(key,val)
```
-can also use val, key to sort by val
-if more than 1 tuple same value, look at 2nd element(val)

Sort Dict by Values:

```
c={}
tmp=[]
for k,v in c.items():
  tmp.append(v,k)
tmp=sorted(tmp, reverse=true)
```
-must use sorted bc immutable

List Comprehension

General Syntax:
[<expression> for <item> in <sequence> if <condition>]
if clause is optional
ex:
```
def keep_evens(nums):
 new_list=[num for num in nums if num %2 ==0]
  return new_list
print keep_evens([3,4,6,7,0,1])
```
#prints [4,6,0]
*above statement done using list comp.:
```
c={}
print (sorted([(v,k) for k,v in c.items()])
```

Sorting

Generally use the function sorted rather than the method .sort()
.sort () method mutates original list and returns None
Sorted does not change the original list; it returns a new list
Sorted can accept optional parameters
- Key: specifies what the ordering should be
- Reverse: prints reverse order

Ex.
```
L = ['A', 'C', 'B']
    print sorted(L, reverse = True)
#prints ['C', 'B', 'A']
L1= [1,7,4,-2,3]
L2= sorted(L1, key = lambda x: abs(x))
print L2 #prints [1, -2, 3, 4, 7]
```

While Loops

Can take place of range in for loop.
1.Evaluate the condition yielding False or True
2.If the condition is false, exit the while statement and continue execution to the next statement
3.If the condition is true execute each of the statements in the body and then go back to step 1.
Ex: n=5
```
    while n> 0:
        print(n)
        n=n-1
    print("Blastoff!")
```
#prints 5 4 3 2 1 Blastoff! on different lines
Break statement jumps out of loop
Continue statement skip to next iteration without finishing the loop for the current iteration

Zip

Pairs two lists together and creates a tuple or list of tuples
The lists must have same number of items
Ex:
```
L1= [3,4,5]
L2=[1,2,3]
L4=zip(L1, L2)
Print L4
```
#print [(3,1), (4,2), (5,3)]

Map

Takes two arguments, a function, and a sequence. The function is the mapper that transforms items.
Ex:
```
print map((lambda value: 5*value), [1,2,3])
```

TestCase: class, use this class by creating subclasses
-name of each method begins with test_
-self.assertEqual

TestCase Example:

```
Import Unittest
class TestStringMEthods(unittest.TestCase):
#make subclass
   def test_upper(self): #define method
       self.assertEqual('foo'.upper(), 'FOO') #use
an assert method to test if something
                                    is true
unittest.main(verbosity=2) #invoke the main()
function from unites module, runs all tests
```

-define functions to get the content of instance
-requires self parameter so that
function knows which instance of pet to operate
on

Classes:

-specifies which content should exist
-instance:unique info that specifies what content
actually is
format:

```
class__:
def __init__(self):
   self.data = []
```

Class Vars: shared by all instances
Instance Vars: unique to each instance
ex. self.name=name

Cache:

Cache

```
CACHE_FNAME = 'cache.json'

try:
    cache_file = open(CACHE_FNAME, 'r')
    cache_contents = cache_file.read()
    cache_file.close()
    CACHE_DICTION =
json.loads(cache_contents)
except:
    CACHE_DICTION = {}
```

Sqlite

Relation- table contains tuples and attributes
Tuple- row
Attribute- column
Insert- inserts rows into table
  INSERT INTO Users(name, email) VALUES
('Kristen', 'kf@umich.edu')
Delete- deletes row from table
  DELETE FROM Users WHERE email =
'ted@umich.edu
Update- updates a field with where clause
  UPDATE Users SET name= 'Charles' WHERE
email= 'csev@umich.edu'
Retrieving Records- gets group of records with
where clause
  SELECT * FROM Users WHERE
email='csev@umich.edu'
Sorting- can sort info in ascending or descending
order
  SELECT*FROM Users ORDER BY name DESC
(*note* ascending is default)