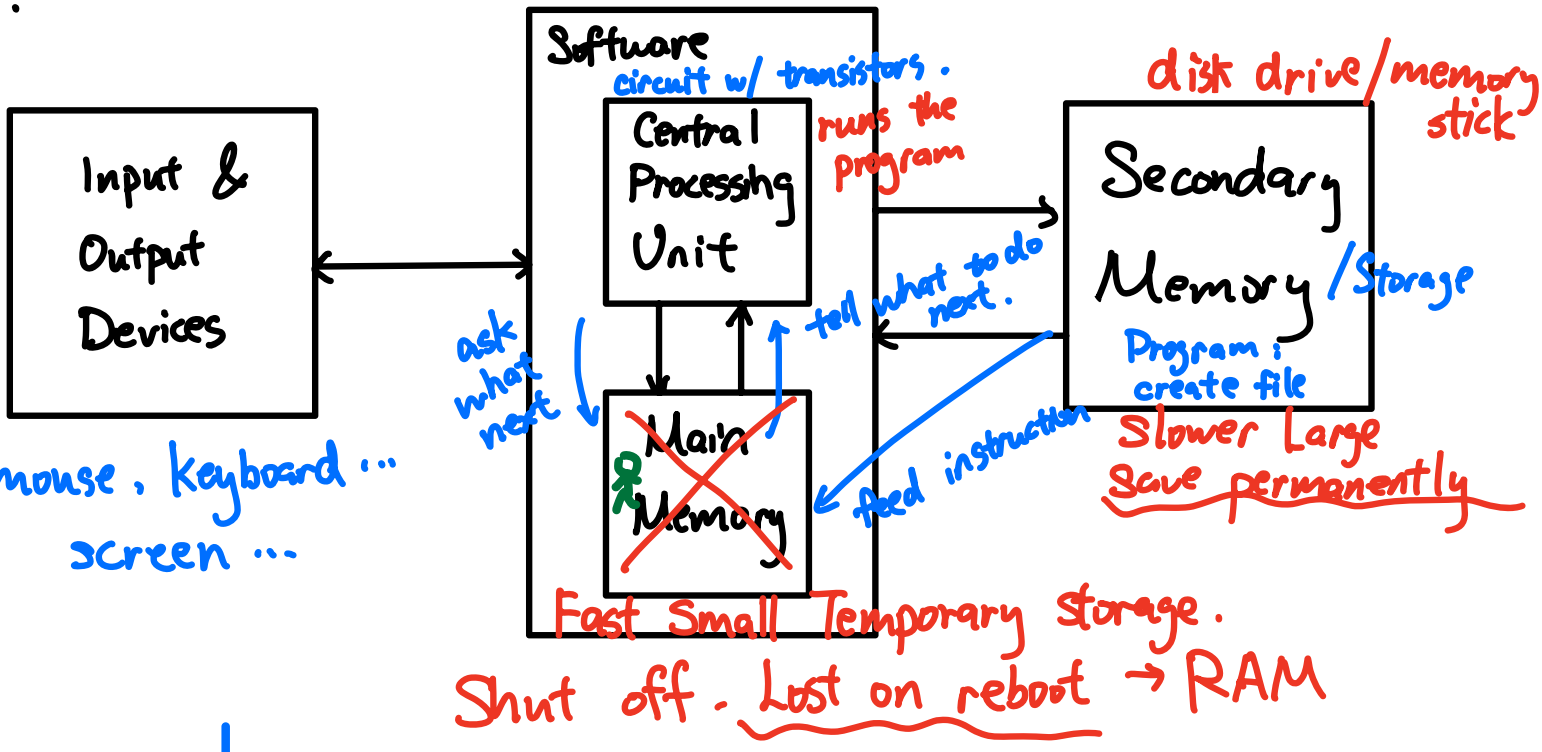
Python For everybody

1.



2. `x = x + 1`

先 evaluate 右边. 再看左边.

3. `**` power.
   `%` remainder.

operator precedance

() → ** → * / → + -
→ left to right

4. type (var). return var's type.

Type Conversions : float (99). int ('99')

* Division provides floating result. Python 3.
Could have an error if can't convert.

5. **User input** : msg = input (' ... ')

6. Try - Except : 有点 类似 if let, else
try-catch

eg.
```
try:
    sth.      isStr = int(astr)
except:
    sth.      isStr = 1.
```
给个 error detection.

7. quit() 直接退出 script

8. def : define function.
max(), min() : return max / min num/ch
return { stop the func
determine residual value

9. break : end the current loop and jumps to the statement immediately following the loop.
escape the loop.
continue : quit the current, start the next one

10. Loop : Set initial value.
for thing in data:
update & check

11. None : eg. smallest = None
if smallest is None:

12. { is. stronger ==
is not !=
type & val. 有点 类似 Nil.

Python Data Structure

1. String.
   index start w/ 0.
   python error : index out of range.
   `len(str)` return length.

   Loop through string :
   for char in str : definite loop.
   while index < len(str) : index = index + 1
   indefinite loop.

   Slicing : str[0:3]. ⇒ only show 0, 1, 2
   up to but not including.
   str[0:∞] ⇒ okay! show the string,
   || no error.
   str[:], str[0:].

2. String Modification.

   'char' in sth : likes "==". return T/F.
   str >/</== str : >/<. Upper/Lower case
   G > c.

   `str.lower()`, `str.upper()` object method.
   `dir(str)` : directions, show all methods on this type.
   `str.find(char)` : return positon index found.
   `.find(char, startPositon)` -1 NOT found.
   `str.replace(str1, str2)` : replace str1 w/ str2.
   replace all

- strip() : remove whitespaces both end
- lstrip() : left
- rstrip() : right

String - Unicode in python 3

No need for conversion. Better character set.

3. Files.

handle = open(fileName, mode) — mode: whether R/W, operation.

file connection.

open, read, write, close. sequence of str.

\n : new line. single character.

4. Process Files:

```
for line in handle:    print(line).
```

line.rstrip()

add newline at the end.

```
if not line.startswith(sth): continue.
```

skip.

```
try:  # open file
except:  print("Error")
         quit()
```

Compensate for bad file

5.

Algorithm: A set of rules/steps to solve a problem

Data Structure: A particular way of organizing data in computer.

Collection: ≥ 1 value in 1 variable

- List : [...]. contain anything. mutable
- String literal immutable

range(list): return a list of numbers

range(4): [0,1,2,3] ⇒ iterator.

eg. for i in range(len(str)): counted loop.

Slicing (like String) [0:3] => 0, 1, 2
up to but NOT including.

- list() : create list
- list.append(sth) : append a new item. / remove()
- sth in list : .contains(). Bool T/F.
  - not in
- list.sort() : sort the list
- list.insert(sth)
- len(), max(), min(), sum()

6. String split into List. 定界符.
   list = str.split() whitespace as delimiter
   .split(',')

7. Dictionary. sth = dict(). key-value pair
   sth[key] = value
   Dict : { key : value, ... }. {}.

```
if keyVal in dict:
    dict[keyVal] = 1
else:
    dict[keyVal] += 1.
```

dict.get(key, defaulVal) get Value OR set default

for key in dict: go through key set.

list(dict) = dict.keys().

dict.values( ), dict.items( )

list of tuples

```
for key, value in dict.items():
```

8. Tuples. = (...). immutable.
   list(tuple) ⇒ mutable.

(x, y) = (4, 'sth') multiple assignments

tups = dict.items( ).

eg. (0, 1, 2) < (5, 1, 2) True

(0, 1, 99) < (0, 3, -1) True

ONLY Look NOT-MATCHES

sorted (dict.items( )). ⇒ sort in KEY orders

※ 可以通过 tmp.append ((value, key))
sorted (tmp) ⇒ sort in VALUE orders.

※ sorted ((v, k) for k, v in c.items( )) ≤
sorted (sth, reverse = TRUE) ≥.

From book

1. import math.
   math.log10 (sth).
2. import random
   random.random( ) # [0.0, 1.0)

random.randInt (5,10) # [5, 10].
random.choice ([1,2,3]) # choosing element.

3. delimiter.join (str_list).

4. { .append modify
   + create a new list

5. line.translate (str.maketrans (fromstr, tostr, deletestr))
   - replace the char at the same idx.
   - delete.

6. import string.
   string.punctuation # '!"#$%...'

eg. line.translate (line.maketrans ('', '', string.punc))
delete all punctuation.

7. immutable ≈ hashable.
   eg. tuple ✓ hashable.
   list ✗ hashable

Access Web Data

1. Regular Expression.

```
import re
re.search()   ≈ find()
re.findall()
```

re.search (pattern, line)

e.g. startswith() = re.search ('^start', line)

Regular Expression Quick Guide

^	Matches the beginning of a line
$	Matches the end of the line
.	Matches any character
\s	Matches whitespace
\S	Matches any non-whitespace character
*	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

dot .
matches all characters.
wild card

.* matches ALL

[^_] not underscore
[^] not whitespace

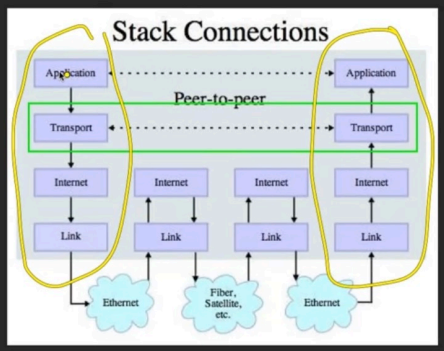
[\] escape char.
\$, \' ...

將 string list ⇒ int list.
map(int, str_list)

2. Networked Technology.

Transport Control Protocol (TCP)

- Built on top of IP (Internet Protocol)
- Assumes IP might lose some data - stores and retransmits data if it seems to be lost
- Handles "flow control" using a transmit window
- Provides a nice reliable pipe



and there's application, talks to a transport layer,

TCP Connections / Sockets

"In computer networking, an Internet socket or network socket is an endpoint of a bidirectional inter-process communication flow across an Internet Protocol-based computer network, such as the Internet."

TCP Port Numbers

- A port is an application-specific or process-specific software communications endpoint
- It allows multiple networked applications to coexist on the same server.
- There is a list of well-known TCP port numbers

Python has built-in support for TCP Sockets

```
import socket
mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect( ('data.pr4e.org', 80) )
```

Host

Port

HTTP : 80.

import socket

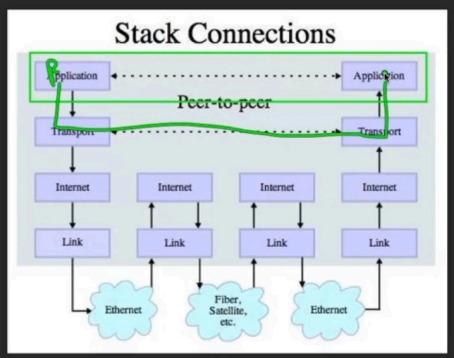
Socket : 信箱

port : 收信地址

TCPIP

Application Protocol

- Since TCP (and Python) gives us a reliable socket, what do we want to do with the socket? What problem do we want to solve?
- Application Protocols
  - Mail
  - World Wide Web



we made a connection with a socket and then connect it

HTTP - Hypertext Transfer Protocol

- The dominant Application Layer Protocol on the Internet
- Invented for the Web - to Retrieve HTML, Images, Documents, etc
- Extended to be data in addition to documents - RSS, Web Services, etc..Basic Concept - Make a Connection - Request a document - Retrieve the Document - Close the Connection

allows browsers to retrieve web docs. from internet servers.

https : // www . baidu . com / sth . html

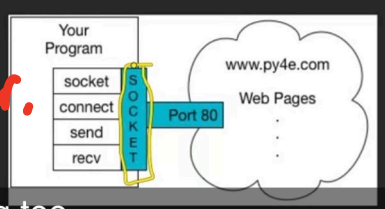
protocal host document

An HTTP Request in Python

```
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))
cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n'.encode()
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if (len(data) < 1):
        break
    print(data.decode())
mysock.close()
```

bytes

receive up to 512 char.



That's like a matrix thing too,

string

Status code :
302 : post. provide location header. redirect immed
200 : give a doc.
404 : Not found.

In the metadata in the header, it tells content-type. knows how to interpret .

The Request / Response Cycle .

Which application talks first ? Client or Server ?

3. Retrieve Web Page.

ASCII : ord (char) ⇔ chr (Int) ⇒ char

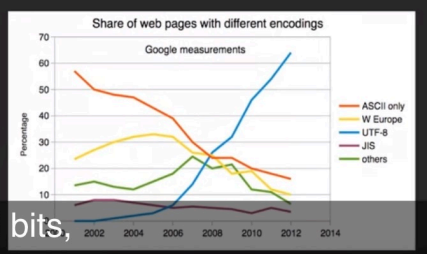
Unicode for diff. lang.

provide ASCII ⇒ Int { ordinal 序数. character

Multi-Byte Characters

- To represent the wide range of characters computers must handle we represent characters with more than one byte
  - UTF-16 – Fixed length - Two bytes
  - UTF-32 – Fixed Length - Four Bytes
  - UTF-8 – 1-4 bytes
    - *Upwards compatible with ASCII*
    - *Automatic detection between ASCII and UTF-8*
    - *UTF-8 is recommended practice for encoding data to be exchanged between systems*

https://en.wikipedia.org/wiki/UTF-8

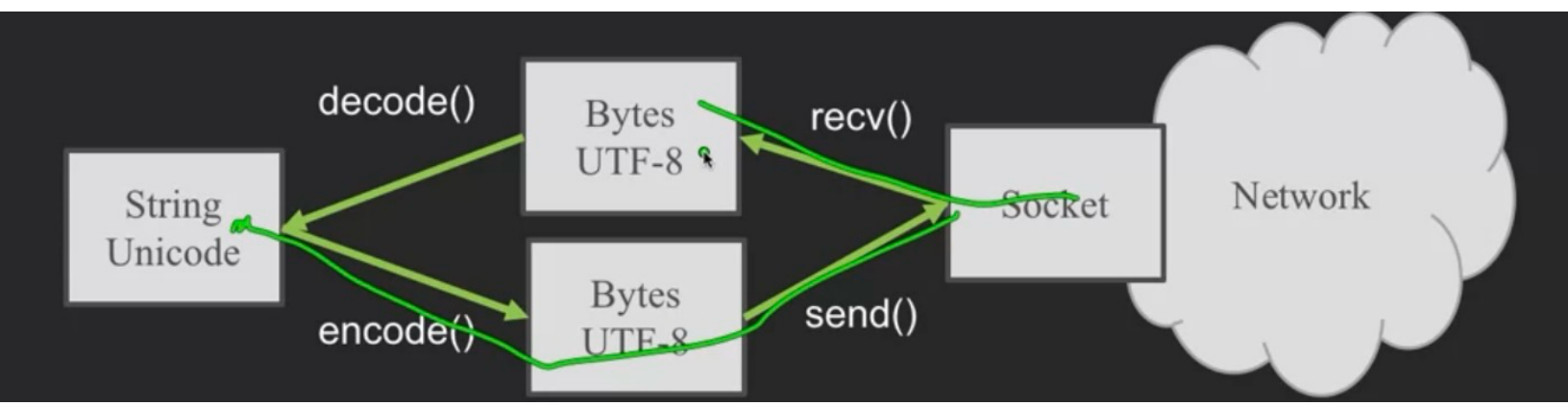


Now the gigantic one is 32 bits, which is 4 bytes.

Python 3 :
All str ⇒ Unicode
type = String.

※ { Python 2 : byte str = str.
Python 3 : str = unicode.

※ str.encode() ⇒ byte array
data.decode() ⇒ string
(data: bytes)



4. handle web request

Using urllib in Python

Since HTTP is so common, we have a library that does all the socket work for us and makes web pages look like a file

```
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
for line in fhand:
    print(line.decode().strip())
```

handler

=) body data

some library bits and then we say,
okay, urllib.request.urlopen.

urllib1.py

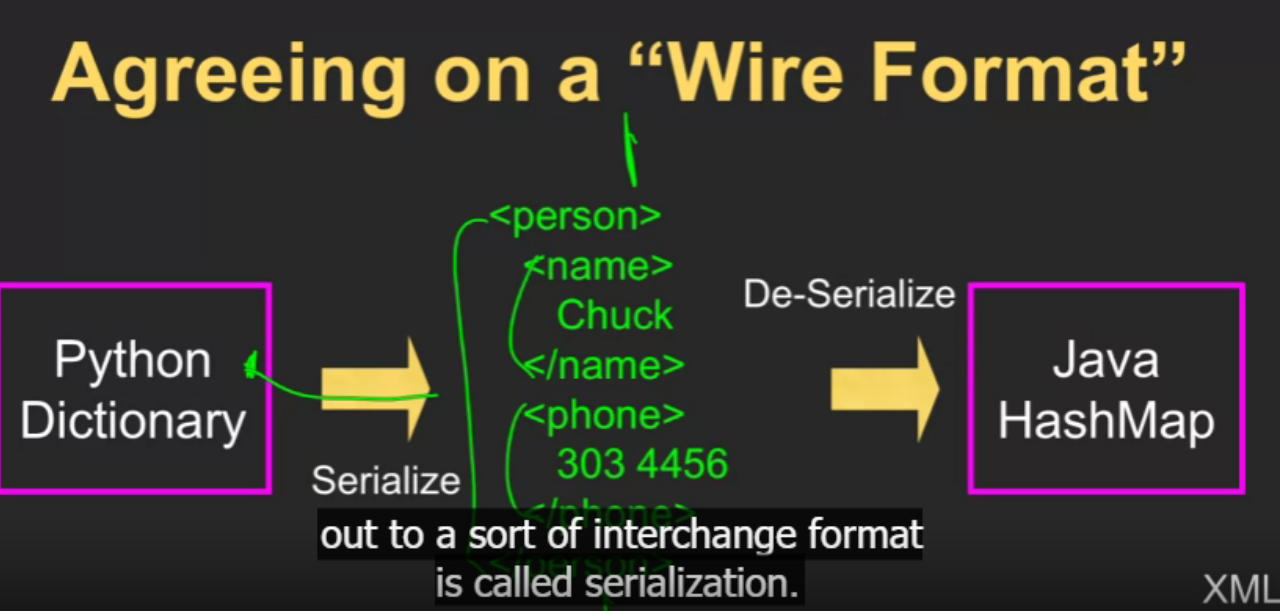
* Beautiful Soup - search string.
from bs4 import BeautifulSoup

```
import urllib.request, urllib.parse, urllib.error
from bs4 import BeautifulSoup

url = input('Enter - ')
html = urllib.request.urlopen(url).read()
soup = BeautifulSoup(html, 'html.parser')

# Retrieve all of the anchor tags
tags = soup('a')
for tag in tags:
    print(tag.get('href', None))
```

python urllinks.py
Enter - http://www.dr-chuck.com/page1.htm
http://www.dr-chuck.com/page2.htm

That's what this parsing is doing.

* The TCP/IP provides pipes/sockets between applications.



5. eXtensible Markup Language (XML)
可扩展 标记 语言.

{ HTML: 显示网页数据 ⇒ 预定义标签.
XML: 交换、传输数据. ⇒ 自定义标签.

↳ { Simple Element
Complex Element: has child node

Share structured data.

XML Basics

- Start Tag
- End Tag
- Text Content
- Attribute
- Self Closing Tag

```
<person>
  <name>Chuck</name>
  <phone type="intl">
    +1 734 303 4456
  </phone>
  <email hide="yes" />
</person>
```

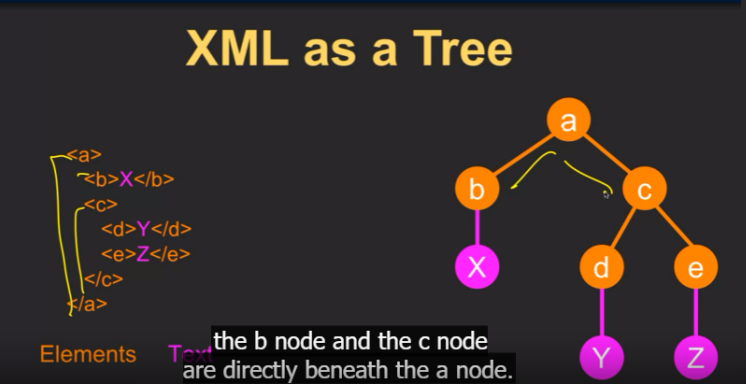
XML Terminology

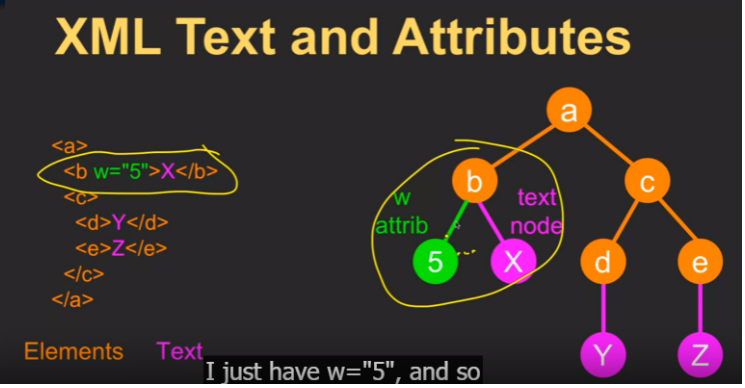
- Tags indicate the beginning and ending of elements
- Attributes - Keyword/value pairs on the opening tag of XML
- Serialize / De-Serialize - Convert data in one program into a common format that can be stored and/or transmitted between systems in a programming language-independent manner

Serialization and deserialization is the act of taking from an internal structure

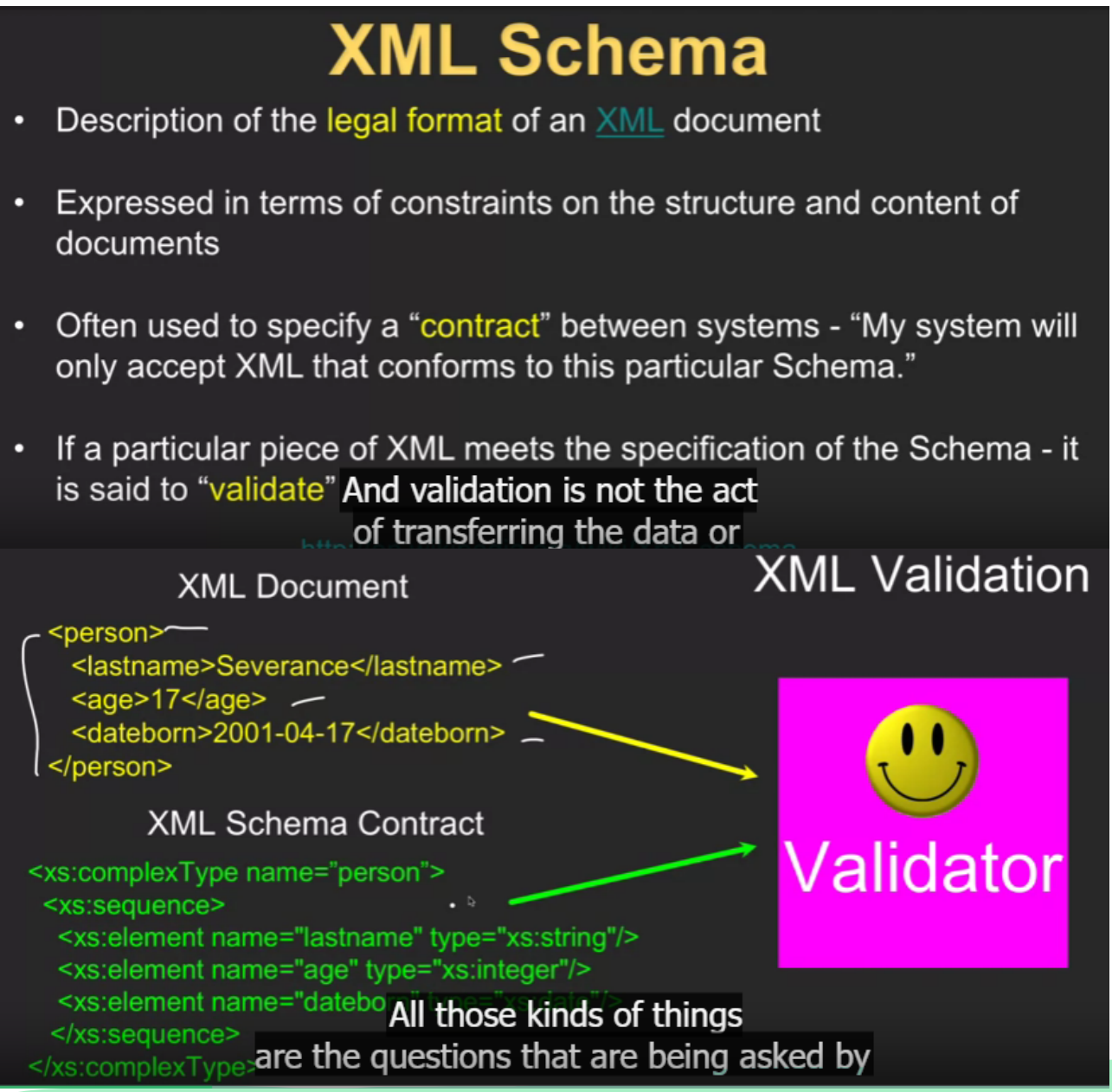
Serialization
有序
program-indep.
deserialization
无序
program-dep

{ /a/b x
/a/c/d y
/a/c/e z.



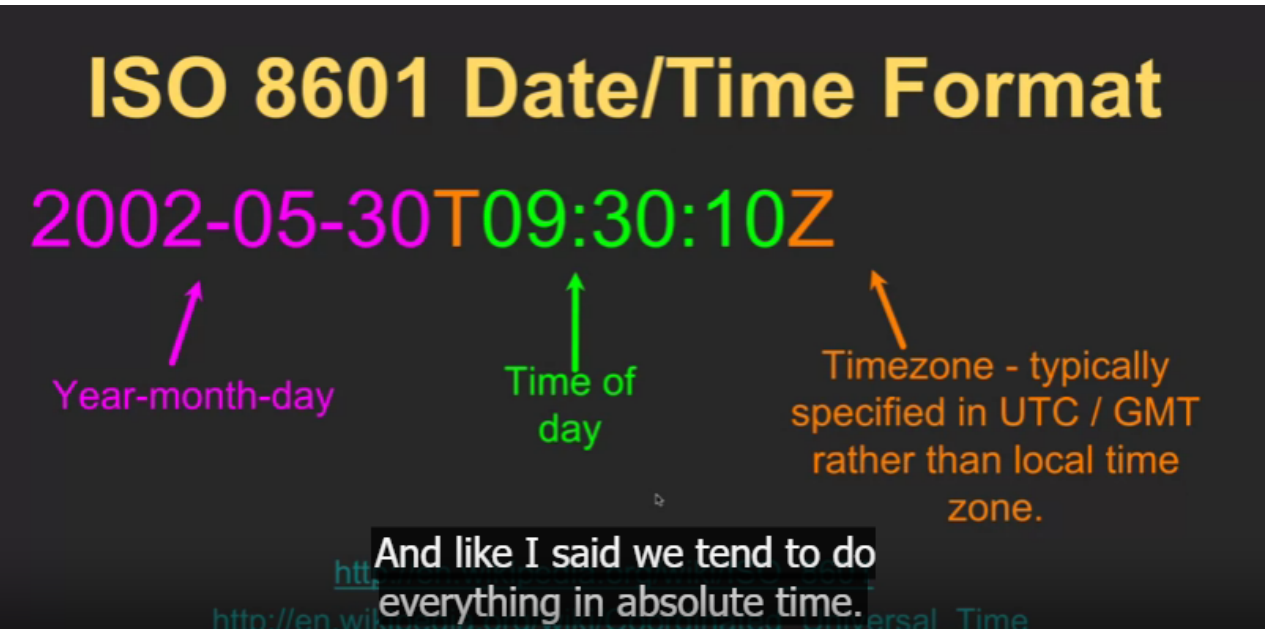


※ XML Schema : a way to establish outside of program. and the separately check on send/receive ends. resolve disagreements.



DTD → SGML → XML
Doc. Type. Def → Stand. General. ML → XML

W3C (World Wide Web Consortium 联盟) = xsd



6. import xml.

```
import xml.etree.ElementTree as ET
data = '''<person>
  <name>Chuck</name>
  <phone type="intl">
     +1 734 303 4456
   </phone>
   <email hide="yes"/>
</person>'''

tree = ET.fromstring(data)
print('Name:',tree.find('name').text)
print('Attr:',tree.find('email').get('hide'))
```

ACIAC

content

attr

But just to make these simple on one screen I've kept it simple.

toplevel = ET.fromstring(data)

```
import xml.etree.ElementTree as ET

input = '''
<stuff>
  <users>
    <user x="2">
      <id>001</id>
      <name>Chuck</name>
    </user>
    <user x="7">
      <id>009</id>
      <name>Brent</name>
    </user>
  </users>
</stuff>'''

stuff = ET.fromstring(input)
lst = stuff.findall('users/user')
print('User count:', len(lst))

for item in lst:
    print('Name', item.find('name').text)
    print('Id', item.find('id').text)
    print('Attribute', item.get('x'))
```

list of tags = tree.findall('class/item')

7. JSON (JavaScript Object Notation)

import json

json.loads(data) ⇒ info["key"]

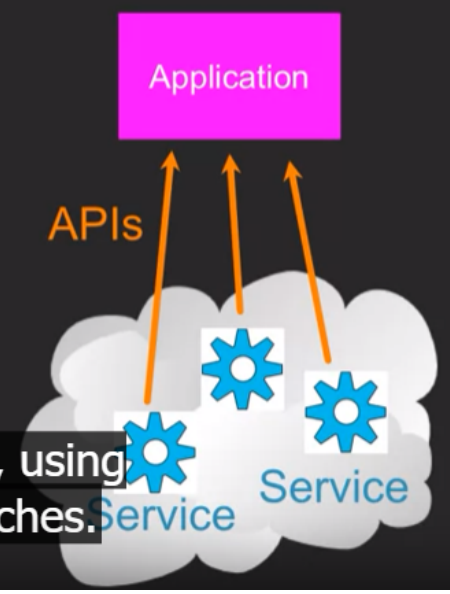
python dict

8. Service-oriented architecture.

Service Oriented Approach

- Most non-trivial web applications use services
- They use services from other applications
  - Credit Card Charge
  - Hotel Reservation systems
- Services publish the "rules" applications must follow to make use of the service (API)

protocols to access data on systems, using well-defined and structured approaches.



customize

Sender ↔ Service layer ○

common

Service layer ○ ↔ customize Receiver □

9. Application Programming Interface (API)

```
import urllib.request, urllib.parse, urllib.error
import json

serviceurl = 'http://maps.googleapis.com/maps/api/geocode/json?'

while True:
    address = input('Enter location: ')
    if len(address) < 1: break

    url = serviceurl + urllib.parse.urlencode({'address': address})

    print('Retrieving', url)
    uh = urllib.request.urlopen(url)
    data = uh.read().decode()
    print('Retrieved', len(data), 'characters')

    try:
        js = json.loads(data)
    except:
        js = None

    if not js or 'status' not in js or js['status'] != 'OK':
        print('==== Failure To Retrieve ====')
        print(data)
        continue

    lat = js["results"][0]["geometry"]["location"]["lat"]
    lng = js["results"][0]["geometry"]["location"]["lng"]
    print('lat', lat, 'lng', lng)
    location = js['results'][0]['formatted_address']
    print(location)
```

```
Enter location: Ann Arbor, MI
Retrieving http://maps.googleapis.com/...
Retrieved 1669 characters
lat 42.2808256 lng -83.7430378
Ann Arbor, MI, USA
Enter location:
```

that goes into the string address.

geojson.py

```
import urllib.request, urllib.parse, urllib.error
import twurl
import json

TWITTER_URL = 'https://api.twitter.com/1.1/friends/list.json'

while True:
    print('')
    acct = input('Enter Twitter Account:')
    if (len(acct) < 1): break
    url = twurl.augment(TWITTER_URL,
                        {'screen_name': acct, 'count': '5'})
    print('Retrieving', url)
    connection = urllib.request.urlopen(url)
    data = connection.read().decode()
    headers = dict(connection.getheaders())
    print('Remaining', headers['x-rate-limit-remaining'])
    js = json.loads(data)
    print(json.dumps(js, indent=4))

    for u in js['users']:
        print(u['screen_name'])
        s = u['status']['tex
        print('  ', s[:50])
```

security stuff.

headers.

limit

Of course we need JSON, and so

```
import urllib
import oauth
import hidden

def augment(url, parameters) :
    secrets = hidden.oauth()
    consumer = oauth.OAuthConsumer(secrets['consumer_key'], secrets['consumer_secret'])
    token = oauth.OAuthToken(secrets['token_key'],secrets['token_secret'])
    oauth_request = oauth.OAuthRequest.from_consumer_and_token(consumer,
        token=token, http_method='GET', http_url=url, parameters=parameters)
    oauth_request.sign_request(oauth.OAuthSignatureMethod_HMAC_SHA1(), consumer, token)
    return oauth_request.to_url()
```

twurl.py

4 strings

```
https://api.twitter.com/1.1/statuses/user_timeline.json?count=2
&oauth_version=1.0&oauth_token=101...SGI&screen_name=drchuck&oa
uth_nonce=09239679&oaut...5644&oauth_signature=r
LK...BoD&oauth_cons...uth_signature_method=H
MAC-SHA1
```

You set all that, and
then I call this oauth library,

JSON = json.dump(str)

more succinct.

less self-describing

.urlopen(...) return byteString.

需要先 .decode => string.

再进行处理. json.loads(byte/str)

xml ET.fromstring(str).

也可以byte.

看情况.

Databases

1. SQL (Structured Query Language)
2. Object-Oriented
3. Inheritance — subclasses.

e.g. class SubClass(ParentClass):

extends Parent Class

Definitions

- Class - a template
- Attribute – A variable within a class
- Method - A function within a class
- Object - A particular instance of a class
- Constructor – Code that runs when an object is created
- Inheritance - The ability to extend a class to make a new class.

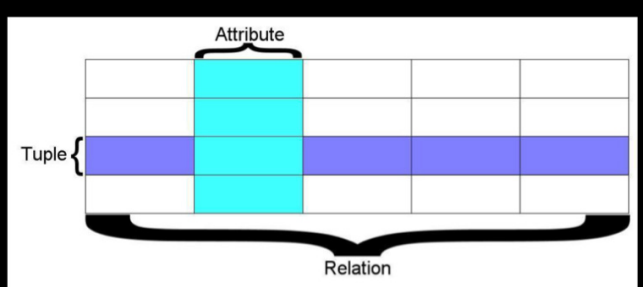
Summary

- Object Oriented programming is a very structured approach to code reuse
- We can group data and functionality together and create many independent instances of a class

4. Relational Database: store rows & cols. in tables.

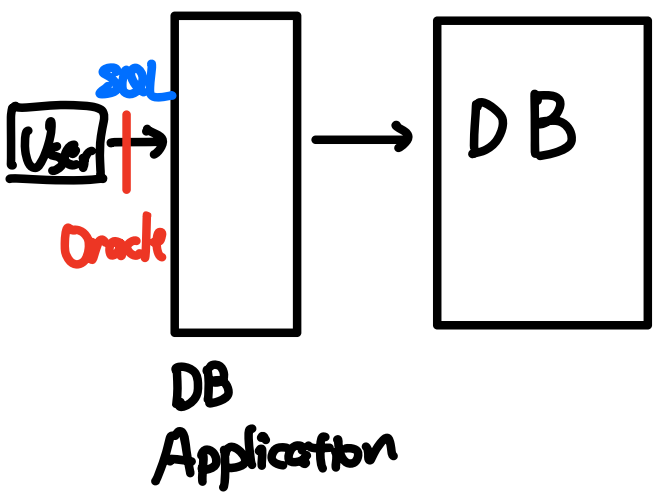
Terminology

- Database - contains many tables
- Relation (or table) - contains tuples and attributes
- Tuple (or row) - a set of fields that generally represents an "object" like a person or a music track
- Attribute (also column or field) - one of possibly many elements of data corresponding to the object represented by the row



A relation is defined as a set of tuples that have the same attributes. A tuple usually represents an object and information about that object. Objects are typically physical objects or concepts. A relation is usually described as a table, which is organized into rows and columns. All the data referenced by an attribute are in the same domain and conform to the same constraints. (Wikipedia)

※ First row: Schema metadata



CRUD

Database Model

A database model or database schema is the structure or format of a database, described in a formal language supported by the database management system. In other words, a "database model" is the application of a data model when used in conjunction with a database management system.

http://en.wikipedia.org/wiki/Database_model

※ Create:

```
CREATE TABLE name(
  attr cond
       eg. VARCHAR(128)
)
```

5. SQLite - embedded db

※ schema ≈ contract . agreement

C: INSERT INTO Table (Attr) VALUES (data, data)
R: SELECT * FROM Table WHERE cond.
U: UPDATE Table SET NewAttr. WHERE cond.
D: DELETE FROM Table WHERE cond.

SELECT * FROM Table ORDER BY attr. DESC.

SQL Summary

```
INSERT INTO Users (name, email) VALUES ('Kristin', 'kf@umich.edu')

DELETE FROM Users WHERE email='ted@umich.edu'

UPDATE Users SET name="Charles" WHERE email='csev@umich.edu'

SELECT * FROM Users

SELECT * FROM Users WHERE email='csev@umich.edu'

SELECT * FROM Users ORDER BY email
```

6. Python SQLite.

```python
import sqlite3
conn = sqlite3.connect('emaildb.sqlite')
cur = conn.cursor()

cur.execute('DROP TABLE IF EXISTS Counts')

cur.execute('''
CREATE TABLE Counts (email TEXT, count INTEGER)''')

fname = input('Enter file name: ')
if (len(fname) < 1): fname = 'mbox-short.txt'
fh = open(fname)
for line in fh:
    if not line.startswith('From: '): continue
    pieces = line.split()
    email = pieces[1]
    cur.execute('SELECT count FROM Counts WHERE email = ? ', (email,))
    row = cur.fetchone()
    if row is None:
        cur.execute('''INSERT INTO Counts (email, count)
                VALUES (?, 1)''', (email,))
    else:
        cur.execute('UPDATE Counts SET count = count + 1 WHERE email = ?',
                (email,))
    conn.commit()

# https://www.sqlite.org/lang_select.html
sqlstr = 'SELECT email, count FROM Counts ORDER BY count DESC LIMIT 10'

for row in cur.execute(sqlstr):
    print(str(row[0]), row[1])

cur.close()
```

handle

? placeholder

fetch one row.

commit the change. slowest.

7. DB Normalization (3NF)

Don't replicate data. Add keys.

Three Kinds of Keys

- Primary key - generally an integer auto-increment field
- Logical key - What the outside world uses for lookup
- Foreign key - generally an integer key pointing to a row in another table

Album
id
title
artist_id
...

Key Rules

Best practices

- Never use your logical key as the primary key
- Logical keys can and do change, albeit slowly
- Relationships that are based on matching string fields are less efficient than integers

User
id
login
password
name
email
created_at
modified_at
login_at

* Logic Key - Lookup

Primary key:
eg. Integer NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE.

8. JOIN link across multiple tables.

id	artist_id	title
Filter	Filter	Filter
1	2	Who Made Who
2	1	IV

Album

Artist

	title	name
1	Who Made Who	AC/DC
2	IV	Led Zepplin

id	name
Filter	Filter
1	Led Zepplin
2	AC/DC

```
select Album.title, Artist.name from Album join Artist on Album.artist_id = Artist.id
```

What we want to see

The tables that hold the data

How the tables are linked

	title	genre_id	id	name
1	Black Dog	1	1	Rock
2	Black Dog	1	2	Metal
3	Stairway	1	1	Rock
4	Stairway	1	2	Metal
5	About to Rock	2	1	Rock
6	About to Rock	2	2	Metal
7	Who Made Who	2	1	Rock
8	Who Made Who	2	2	Metal

```
SELECT Track.title,
    Track.genre_id,
    Genre.id, Genre.name
FROM Track JOIN Genre
```

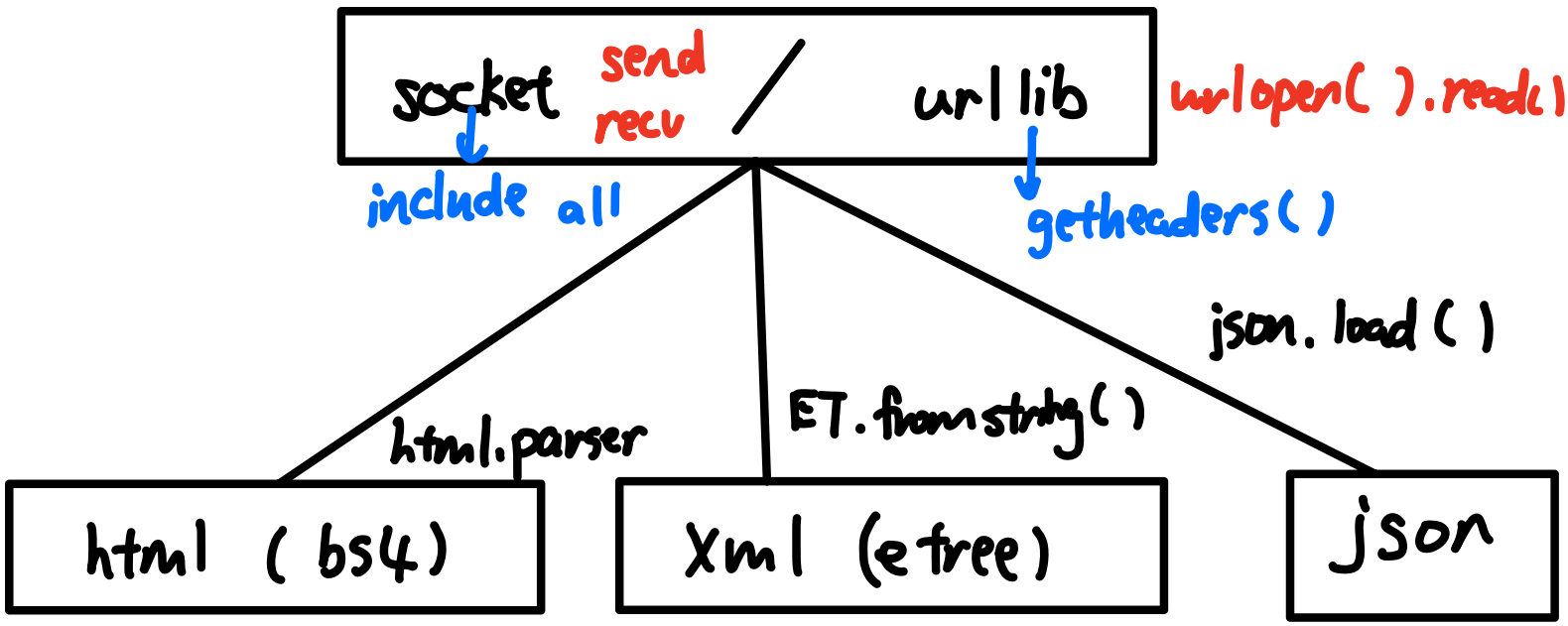
排列組合.

Joining two tables without an ON clause gives all possible combinations of rows.

※ INSERT OR IGNORE
REPLACE

※ ∫. executescript (…) ⇒ multiple SQL stmts.

. execute (one) ⇒ ONE stmt.



DB : Con . connect( )
Cur . cursor( )
Cur . execute /executescript( )
Con . commit( )
Cur . close( ) 按顺序关.
Con . close( )
※ import time. time.sleep(5)

Cur . fetchone( ), .fetchall( )
.fetchmany( )