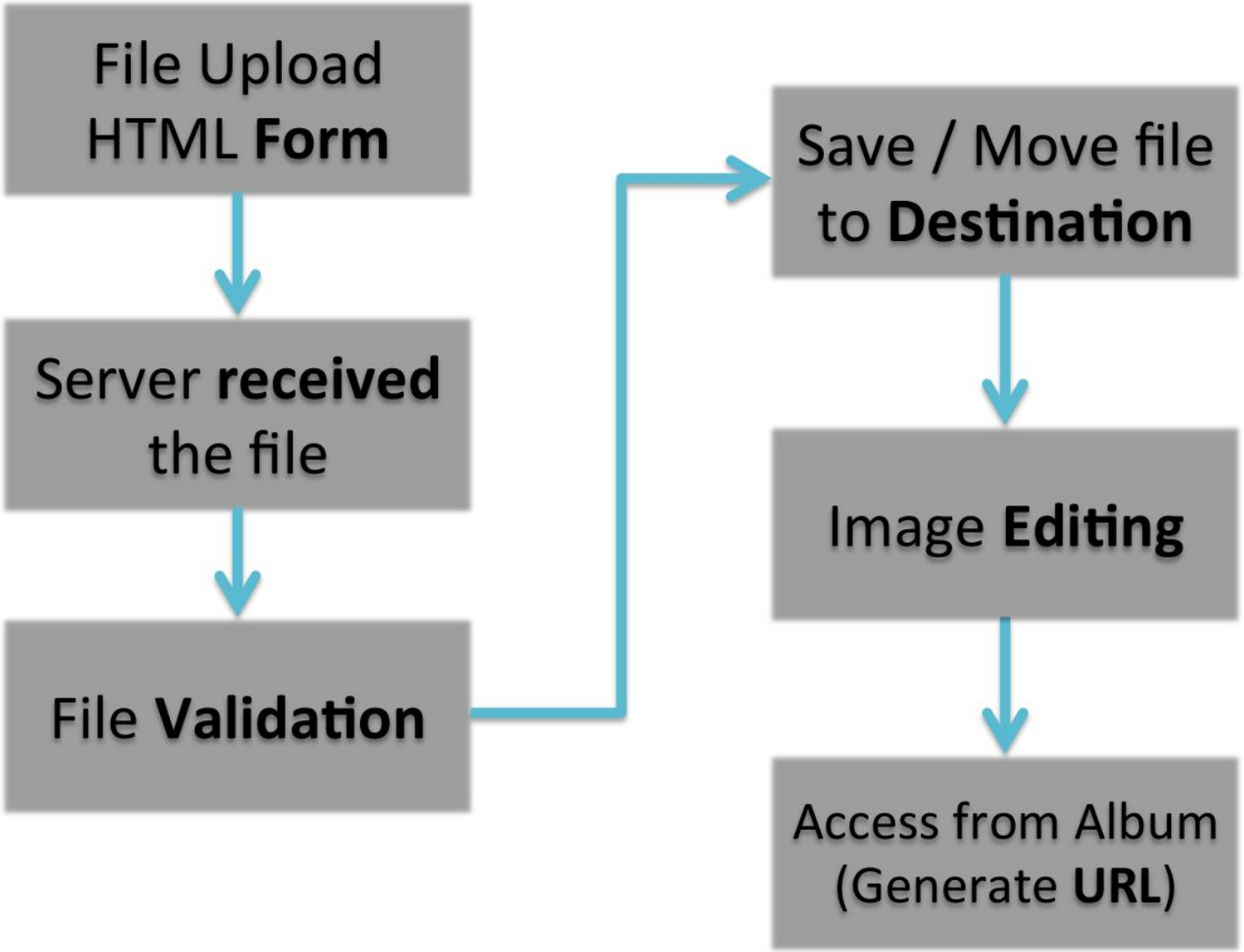
Assignment 1
ImageMagick & Database

CSCI 4140 Tutorial 4
Feb. 18
Qin Chuan

Image Editing

Image Editing



ImageMagick

- Command line tools for viewing image properties, image editing, conversion, …
- Useful for checking image format and applying filter
- Provided Docker image has ImageMagick installed
  - Available on department linux
- Install on your own computer …
  - Mac: `brew install imagemagick`
  - Ubuntu: `sudo apt-get install imagemagick`
- Execute the command from Python script
- Official Website: http://www.imagemagick.org/

Image Format & Characteristics

```
identify <img_file>
```

- Provide format and characteristics of images
  - PNG ? JPEG? GIF?
  - Image Size

- Get `stdout` as string, then process the string to check correct image format
  - Using regular expression
  - Split the string

Convert Image Format

```
convert <in> <operations> <out>
```

- Convert image format
  - Result image format determined by extension of `out`
- Perform operations on image
  - Blur, …
- Or even join multiple images

Some Flags for Convert

```
convert <in> <operations> <out>
```

- Resize
  - Resize to best-fit: `-resize <width>x<height>`
  - Preserve aspect ratio
  - Ignoring aspect ratio: `-resize <width>x<height>\!`

Escape for shell

- Blur
  - `-blur <radius>x<sigma>`; or
  - `-gaussian-blur <radius>x<sigma>`

Some Flags for Convert

```
convert <in> <operations> <out>
```

- Annotate with Label

  `-label:<text>`

  Add following options before `-label` for more style

  - Background color of label: `-background <color>`
  - Font size: `-pointsize <size>`
  - Font: `-font <fontname>`
    - Helvetica: `Helvetica`
    - Times: `Times-Roman`
    - Courier: `Courier`

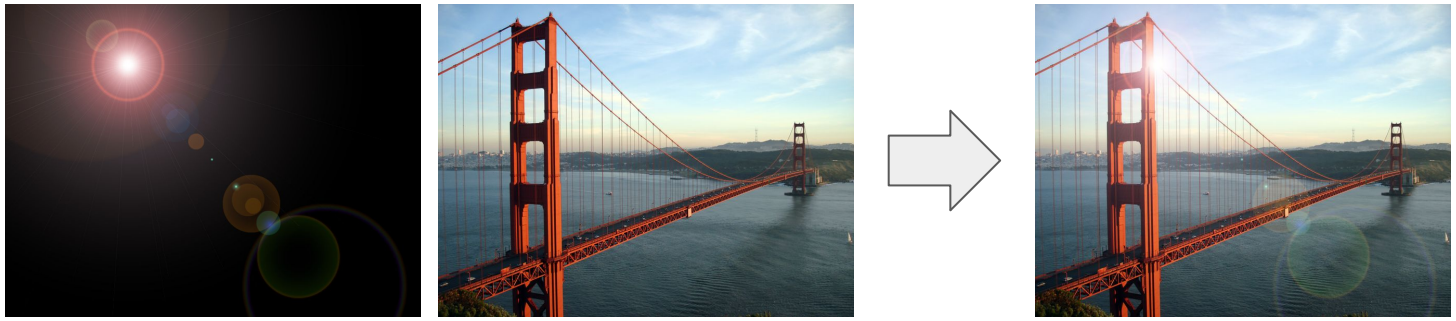
  `-append` to join label and image

Overlaying Multiple Images

```
composite <flags> <in(s)> <out>
```

- Overlay images onto another
  - Using several operations, e.g. multiply pixel value etc
- Example: Lens Flare

```
composite
    -compose screen -gravity northwest
    lensflare.png in.png out.png
```



More Usage: http://www.imagemagick.org/Usage/compose/

Executing Commands

- `subprocess` module
  - Create a subprocess to execute other commands
  - `identify`, `convert`, `composite` from ImageMagick
  - `mv`, `rm`, `ls` etc

  - Feed `stdin` and retrieve `stdout` (and `stderr`) from your python script

- Python Doc: https://docs.python.org/2/library/subprocess.html

The Simplest Call

```
retcode = subprocess.call(cmd)
```

- Blocking call (wait until it finish)
- `cmd`: list of arguments
  - E.g.: ['`ls`', '`-al`']
- Return the return code of command execution
- Optional argument:
  - `stdin=<file>`
  - Specify `stdin` from file
  - `stdout=<file>` / `stderr=<file>`
  - Write `stdout` / `stderr` to a file
  - `cwd=<path>`
  - Set working directory for execution

Non-blocking Execution

```
p = subprocess.Popen(cmd)
```

- Non-blocking call
- Return: Popen object
- To wait until execution

Optional:
Omit=None

```
(out, err) = p.communicate(in)
```

- Optional arguments
  - `stdin=<file>` / `stdout=<file>` / `stderr=<file>`
  - Specify `stdin` / `stdout` / `stderr` from file
  - Use `subprocess.PIPE` if you need to communicate from script
  - `cwd=<path>`
  - Set working directory for execution

tutorial4/subprocess.cgi

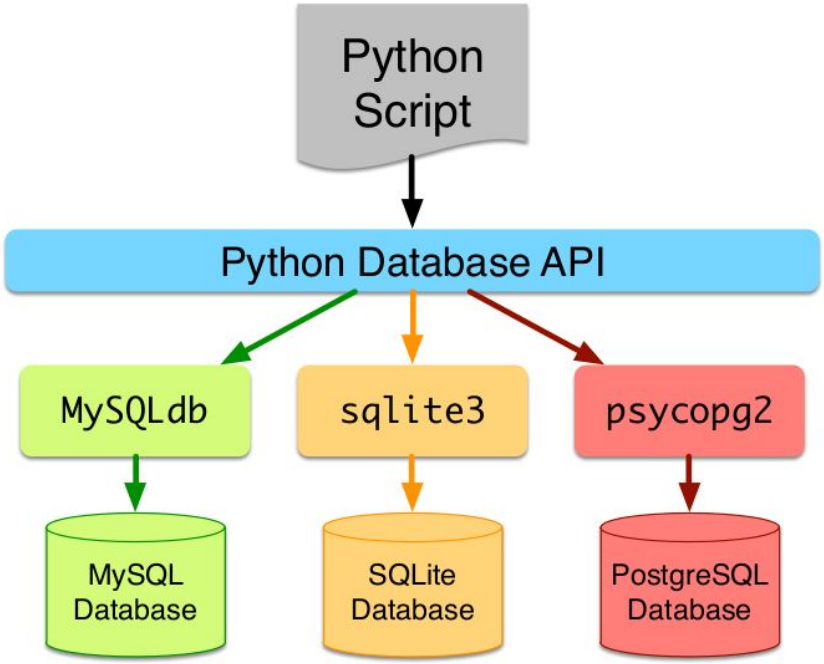
ImageMagick & Subprocess

- Use subprocess to execute ImageMagick
  - Use `identify` to **check** whether uploaded file is image / match with extension
  - Get **image dimension** of `identify`
  - Use `convert` / `composite` to perform image filters
  - Detail commands are provided in specification
- Use `cwd` argument to specify where is working directory
  - Location of image files
- You can also use `subprocess` to execute other shell commands
  - `rm`, `mv`, …

Database

Python Database APIs

- Interface for different database system
  - Same API for MySQL, SQLite, etc.
  - Just need different module (driver) & different construction of connection object (generic class)

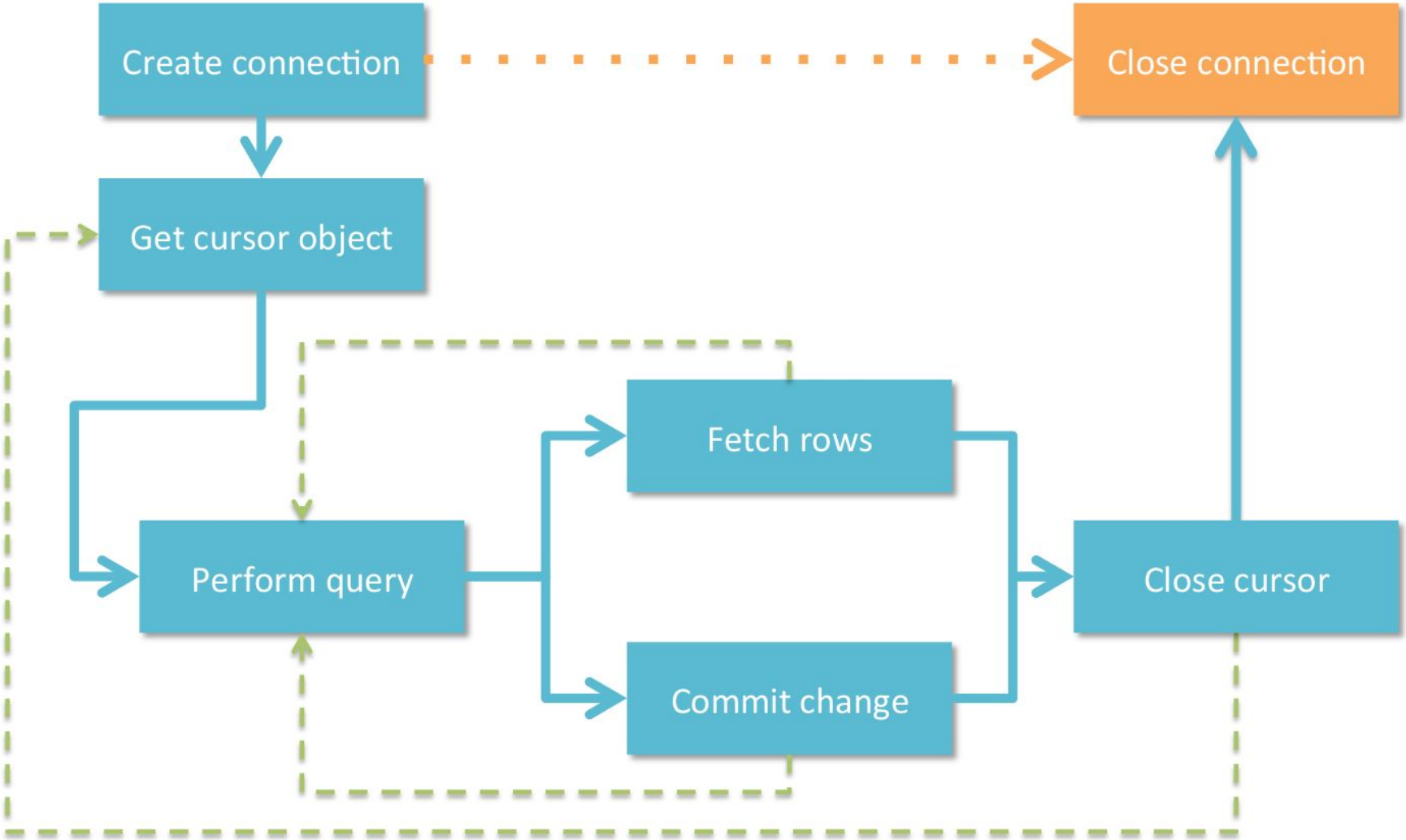


Class of Python Database API

- Connection
  - Connection created by constructor from different modules (for different database)
  - To manage connection and changes to database

- Cursor
  - Created from connection (`connection.cursor()`)
  - For traverse records in database (perform query)

Python Database API



Methods of Python Database API

- Create connection object
  - Depends on database module / driver
  - Assume we have created the connection object …

```
conn = <constructor_from_module>
```

- Get cursor object from connection

```
cursor = conn.cursor()
```

Using Cursor to Execute Query

- Write the query
  - As string
  - Parameters can be hard coded inside string
  - By concatenation or string formatting
  - Or parameter leaved to be filled

```
query = 'INSERT INTO images (filename, imagetype) VALUES (%s, %s)'
```

  - Query can be reuse / execute with different parameters

- Perform query

```
cursor.execute(query, [filename, imgtype])
```

Parameter to be substituted
If no parameter, just omit it

tutorial4/db_{query,insert}.cgi

Using Cursor to Retrieve Records

- Retrieve the records (rows) selected / affected
- Fetch row by row

```
while True:
    result = cursor.fetchone()
    if (not result):
        break;
    # Perform your jobs
```

  - Return a list
- Or fetch all rows by

```
results = cursor.fetchall()
```

  - Return list of row (list)

tutorial4/db_query.cgi

Commit Change & Close Connection

- If your query involve change to database
  - INSERT, DELETE, …

- Commit the change to database

```
conn.commit()
```

  - Without commit it, change will be rollback when connection close
- Finally, close cursor and connection

```
cursor.close()
conn.close()
```

tutorial4/db_insert.cgi

MySQLdb

- Module for connecting MySQL database from Python
  - Import by `import MySQLdb`
- Create Connection object

```
conn = MySQLdb.connect(host = dbHost,
                       user = dbUser,
                       passwd = dbPass
                       db = dbName)
```

  - You will need database host, login, password, database name
  - Please according to specification

Python Doc: http://mysql-python.sourceforge.net/MySQLdb.html

tutorial4/db_conn.cgi

Debugging

Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator, root@localhost and inform them of the time the error occurred, and anything you might have done that may have caused the error.

More information about this error may be available in the server error log.

Apache/2.2.22 (Red Hat Enterprise Web Server) Server at asg1-csci4140ltsinn.rhcloud.com Port 80

- Server did not give a proper response
- Maybe due to **bug** in script, fail to **execute**, or even **syntax** error
  - No error detail is exposed to user
- Apache log may have hints …

Apache Log

- Access log
  - All received requests
  - Not our main concern
- Error log
  - Our main focus
  - Log all error with timestamp
  - Time when executing script (visiting page)
  - Check current time with date command
- You will find this command very useful

```
grep error <log_file>
```

Apache Log

- Permission Denied

```
[Sat Jan 24 07:38:25 2015] [error] [client 127.7.67.129] (13)Permission denied: exec of '
/var/lib/openshift/54b574cde0b8cd2b340000a3/app-root/runtime/repo/gg.cgi' failed
```

  - Did you enable execute of script ? (`chmod a+x`)
  - If you use Windows, try `git update-index --chmod a+x <filename>`
- Premature end of script header / Malformed header

```
[Sat Jan 24 07:38:25 2015] [error] [client 127.7.67.129] Premature end of script headers: gg.cgi
```

```
[Sat Jan 24 07:42:11 2015] [error] [client 127.7.67.129] malformed header from script. Bad header=hihi
: gg.cgi
```

  - Did you print HTTP header ?
- Traceback (`stderr`) also printed to error log

```
[Sat Jan 24 06:59:52 2015] [error] [client 127.7.67.129] Traceback (most recent call last):
[Sat Jan 24 06:59:52 2015] [error] [client 127.7.67.129]   File "/var/lib/openshift/54b574cde0b8cd2b34
0000a3/app-root/runtime/repo/db_insert.cgi", line 11, in <module>
[Sat Jan 24 06:59:52 2015] [error] [client 127.7.67.129]     sys.stderr = std.stdout
[Sat Jan 24 06:59:52 2015] [error] [client 127.7.67.129] NameError: name 'std' is not defined
[Sat Jan 24 06:59:56 2015] [error] [client 127.7.67.129] Traceback (most recent call last):
[Sat Jan 24 06:59:56 2015] [error] [client 127.7.67.129]   File "/var/lib/openshift/54b574cde0b8cd2b34
0000a3/app-root/runtime/repo/db_insert.cgi", line 11, in <module>
[Sat Jan 24 06:59:56 2015] [error] [client 127.7.67.129]     sys.stderr = std.stdout
[Sat Jan 24 06:59:56 2015] [error] [client 127.7.67.129] NameError: name 'std' is not defined
```

CGI Traceback

- Sometime we see blank page
  - Unhandled exception raised in script
  - Trackback printed to stderr (apache error log)
- `cgitb` module
  - Print the Traceback to browser (stdout)
  - In formatted HTML
- How to use ?

```
import cgitb
cgitb.enable()
```

  - If exception raised **before** HTTP header, still Internal Server Error

Summary

- You have the knowledge to almost finish the assignment
  - Next tutorial: cookies and session for resume
- Play with ImageMagick on your own machine first
- Add more interesting filters by ImageMagick
  - But make sure you have completed the requirement
  - And no bonus will be given … For your fun only

More filters: http://www.fmwconcepts.com/imagemagick/

Thank You