References

Google App Engine http://code.google.com/appengine/docs/
Examples in this lecture are from the Google App Engine documentation
What is Cloud Computing

Wikipedia Definition

IT-related capabilities are provided “as a service”

Services accesses anywhere via network access
Application
Web email
Software as a Service (SaaS)
SalesForce

Platform
Enables cloud applications
Google App Engine, Heroku, Mosso, Engine Yard, Joyent, force.com

Infrastructure
Enables cloud applications & platforms
Amazon’s EC2, GoGrid, RightScale, Linode

http://cloudcomputing.sys-con.com/node/609938
EC2

Linux machines

Run any linux compatible program

EBS, S3 & SimpleDB for storeage

Not web specific
Google App Engine

Dynamic web applications
  Google Web framework - webapp
  Django, CherryPy, Pylons, and web.py

Python
  Java in future

Object Database
  Queries, sorting and transactions

Automatic scaling and load balancing

Authenticating users and sending email using Google Accounts

Caching

Google Data APIs
  Calendar, Contacts, Documents, Spreadsheets, Picasa
Lock in

Using Google Web framework locks you into using App Engine
Simple Example

```
print 'Content-Type: text/plain'
print ''
print 'Hello, world!'
```

```
application: helloworld
version: 1
runtime: python
api_version: 1

handlers:
- url: /.*
  script: helloworld.py
```

```
helloworld.py

print 'Content-Type: text/plain'
print ''
print 'Hello, world!'
```
YAML Ain’t Markup Language

Human friendly data serialization format

Multilanguage support
  Python, Ruby, Java, C, Perl, JavaScript, Ocaml, Haskell

Maps
  key: value

Sequences
  each entry indicated by dash & space
  "- "

app.yaml

application: helloworld
version: 1
runtime: python
api_version: 1

handlers:
  - url: /.*
    script: helloworld.py
Using webapp Framework

helloworld.py
from google.appengine.ext import webapp
from google.appengine.ext.webapp.util import run_wsgi_app

class MainPage(webapp.RequestHandler):
    def get(self):
        self.response.headers['Content-Type'] = 'text/plain'
        self.response.out.write('Hello, webapp World!')

application = webapp.WSGIApplication(
    [('/', MainPage)],
    debug=True)

def main():
    run_wsgi_app(application)

if __name__ == "__main__":
    main()
```python
class MainPage(webapp.RequestHandler):
    def get(self):
        self.response.headers['Content-Type'] = 'text/plain'
        self.response.out.write('Hello, webapp World!')
```

- **define class**: `class MainPage(webapp.RequestHandler):`
- **define method**: `def get(self):` - self required in Python method argument list
- **parent class**: 
- **indentation required and significant**: `self.response.headers['Content-Type'] = 'text/plain'
  self.response.out.write('Hello, webapp World!')`
- **field in parent class**: 

---

**Some Python**
Variables are not explicitly declared

```python
application = webapp.WSGIApplication([('/', MainPage)], debug=True)

def main():
    run_wsgi_app(application)

if __name__ == '__main__':
    main()
```

If used as a program this will call the function main
"main" is just a convention
If called as module main is not called
Templates

Generating html in code is awkward

Supports Django html templates

```html
<html>
<body>
  {% for greeting in greetings %}
  {% if greeting.author %}
    <b>{{ greeting.author.nickname }}</b> wrote:
  {% else %}
    An anonymous person wrote:
  {% endif %}
  {% blockquote %}
    {{ greeting.content|escape }}
  {% endblockquote %}
  {% endfor %}
```
Static Files

Static files can be served directly from file

application: helloworld
version: 1
runtime: python
api_version: 1

handlers:
- url: /stylesheets
  static_dir: stylesheets

- url: /.*
  script: helloworld.py
Datastore

Distributed object database

Define a data model

class Story(db.Model):
    title = db.StringProperty()
    body = db.TextProperty()
    created = db.DateTimeProperty(auto_now_add=True)

Store object in datastore

    story = Story(title='Music of Django')
    story.body = 'There one was ...'
    story.put()
Types Supported

StringProperty  BooleanProperty  SelfReferenceProperty
IntegerProperty  BlobProperty  UserProperty
FloatProperty    TextProperty  CategoryProperty
DateTimeProperty DateProperty  LinkProperty
DateProperty     EmailProperty  GeoPtProperty
TimeProperty     IMProperty    PhoneNumberProperty
ListProperty     PostalAddressProperty  RatingProperty
StringListProperty
Queries

Django Like

stories = Story.all().filter('date >=', yesterday).order('-date')
    for story in stories:
        print story.title

GQL

Greeting.gql("WHERE author = :1 ORDER BY date DESC",
                            users.get_current_user())
Transactions

from google.appengine.ext import db

class Accumulator(db.Model):
    counter = db.IntegerProperty()

def increment_counter(key, amount):
    obj = db.get(key)
    obj.counter += amount
    obj.put()

q = db.GqlQuery("SELECT * FROM Accumulator")
acc = q.get()

db.run_in_transaction(increment_counter, acc.key(), 5)
What about CAP Theorem?

Google Appengine docs do not discuss delays in distributing data

Data does seem to be distributed world wide
Memcache

A coherent cache for data in your application
Available to all instances of your application

from google.appengine.api import memcache

# Add a value if it doesn't exist in the cache, with a cache expiration of 1 hour.
memcache.add(key="weather_USA_98105", value="raining", time=3600)

# Atomically increment an integer value.
memcache.set(key="counter", 0)
memcache.incr("counter")
memcache.incr("counter")
memcache.incr("counter")
More

mail.send_mail(sender="whitney@cs.sdsu.edu",
    to="cs683",
    subject="Cloud Computing",
    body=""""The End"""

Fetching URLS

Access to Google Apps, etc