* Introduction
  + WEB2PY: web development with python
    - Integrate numerous technologies
    - Python
    - Bootstrap
    - SQLite
    - Rocket web server
    - Browser base IDE
  + Full Stack Web Development
    - Front end
    - Server code
    - Database
* Course Overview
  + Download WEB2PY
  + MVW, SQLFORMS
  + Database Admin
* Download and Start WEB2PY
  + Download WEB2PY
  + Start web server
  + Model-View-Controller
  + Go to web2py website
  + In the application folder
    - Actual source code will be in there
  + To start the web server, click on the web2py executable
  + It will open a default welcome app
  + Admin on the welcome app
    - Browser based ide that allows us to access and edit code
* Model-View-Controller
  + controller
    - python code
  + view
    - html files
  + MVC Convention
    - software architecture
    - naming convention also
    - def index(), returns message back to the view
  + view name
    - [controller]/[method name].html
* Build Our First Web Page
  + create a controller
  + inside controller create a method
  + ex) create controller name basics
    - def helloworld():
    - msg = “Hello from the Controller!”
    - return locals()
  + return locals()
    - take all local variables defined in method and return it to view
  + then create a view
    - [controllername]/[methodname]
  + ex)
    - basics/helloworld
  + inside the view, you can mix python with html
  + ex)
    - {{extend ‘layout.html’}}
    - <h1>Our First Web Page</h1>
    - <h2> {{=msg}}</h2>
  + {{}} double curly bracket means python code
  + to go to webpage use url [local address]/[application name]/[controller name]/[view name]
* Request Object
  + web application use http protocol
  + request object and response object
  + request object: anything passed from client to web server
  + arguments: part of url
    - request.args(0), request.args(1), …
  + variables: part of html forms
    - form
    - associated with post
    - request.vars.form\_name
  + in side controller
    - def request\_args():
    - arg1 = float(request.args(0))
    - arg2 = float(request.args(1))
  + have to convert arguments to float because they come across as strings
  + then create a view
    - basics/request\_args
  + url is then [local address]/basics/request\_args/[parameter 1]/[parameter 2]
  + instead of passing in parameters via url, can pass variables in html form using request.vars
  + ex)
    - def request\_vars():
    - num1 = 0
    - num2 = 0
    - if request.post\_vars:
    - num1 = float(request.post\_vars.num1)
    - num2 = float(request.post\_vars.num2)
    - total = num1 + num2
    - return locals()
  + then create a view
    - <form method=’post’>
    - <input type=’text’ name=’num1’/>
    - <input type=’text’ name=’num2’/>
    - <input type=’submit’ value=’Add’/>
    - </form>
* Response Object
  + Serve sends response back to application
  + Ex)
    - response.flash = T(“The total is “ + str(total))
    - T(): indicates translation
    - Web2py has translation feature that translate information into multiple languages based on the client
  + response.flash displays the data into pop up
* Business Rules & Libraries
* Deploying Our App
  + Pythonanyhwere
    - Simply and easy to use to host, run and code in python
    - Create a beginner account
  + Click ‘Web’ Tab
    - Click ‘Add a new web app’
    - Click ‘web2py’
    - Create Admin password
    - This will launch a web server with a web2py welcome application
    - Make sure the python version of web server is same for the code you wrote
  + Take our existing code in local host and publish to website
    - In the local host go to the administrative interface
    - Click on ‘Manage’ -> ‘Pack all’
    - It will package it into simple folder with .w2p extension
    - On the web server login
    - In the administrative interface, go to the ‘Upload and install packed application’ section
    - Name the application and upload the files(or get from git repo)
    - Click ‘Overwrite installed app’
    - Click ‘Install’
* Build a Blog App
  + SQLForms: inserting record and updating in database
* Design the Application and Da..
  + Full stack development is essentially connecting a database to a web application
    - Application design
    - Database design
  + Application design
    - Post
    - View
  + Database design
    - Data design
* Use Models and Validators to …
  + Create a model
    - Click edit on the db.py file
    - Scroll down to the defining tables section
  + By default web2py is a sql lite database
  + Define the table with fields
  + Ex)
    - db.define\_table(‘blog’, Field(‘blog\_title’), Field(‘blog\_details’), Field(‘blog\_image’), Field(‘blog\_url’), Field(‘blog\_category’), Field(‘blog\_date\_posted’))
  + Validators
    - Validate data before they go into database
    - Generate JS(jquery) code to validate data
  + Ex)
    - db.define\_table(‘blog’, Field(‘blog\_title’, requires=IS\_NOT\_EMPTY()), (Field(‘blog\_details’, type=’text’), …. Field(‘blog\_category’, requires=IS\_IN\_SET([‘News’, ‘Events’])), Field(‘blog\_date\_posted’, type=’date’, requires=IS\_DATE() )
* Build the Post Page with WEB…
  + Create a new controller
  + Method post to take form information an post it to database
  + Ex)
    - def post():
    - form = SQLFORM(db.blog).process()
    - return locals
  + db.blog references the blog table name from the model
  + define the view method to display return data
    - def view():
    - rows = db(db.blog).select(orderby=~db.blog.id)
  + Then build a view(blog/post)
    - {{extend ‘layout.html’}}
    - <h1>Post Blog</h1>
    - {{=form}}
* Update Records with SQLFOR…