In the file page.py in the class main handler and inside the def get, make a variable and make it equal to FormPage(). The variable to the FormPage will allow us to make a reference to the HTML in the form. Then type self response write in parenthesis p print_out(). This will print out the HTML page. Make a new class called Page in parenthesis object. This is the beginning of the HTML page. Next def __init__self. Self_open =’’’
<!DOCTYPE html><html><head><title></head>
<body>’’’ inside type self_content equal to a string then we need a self_close equal to ‘’’</body></html>’’’
Then type in self all equal to a black string. This variable will hold all the HTML elements. Next make a def called print_out in parenthesis self. Inside that put self update() and then return self all. The update will be a function that will take all the elements from the HTML and display it. Make a new def and call it update in parenthesis self. Inside it put self all equal to self _open plus self _content plus self _close and self all equal to self all format in parenthesis **locals(). This will take all the privet HTML elements and makes them unprivet. Make a new class and call it FormPage in parenthesis Page. This will make it possible for the FormPage to talk to Page. Inside that make a def__init__self. Inside that put Page.__init__self. Below that put self double underscore form_open and make it equal to ‘<form method=”Get” action=””>’, below that put self double underscore inputs and make it equal to ‘’’<input type=”text” name=”code” placeholder=”placeholder” /> <input type=”submit” name=”submit” />’’’. This will be the form that the user will interact with. Next type self double underscore form close and make it equal to ‘</form>’ now type self form_header equal to a string. Next is self page content equal to a blank string. Finally is self_content equal to self form_header plus self double underscore form_open plus self double underscore inputs plus self double underscore form_close. Just like we did with page, we need to do with the form page. We need to take the HTML elements in in the form page. We now have to make a new def and call this one update in parenthesis self. Inside it there will be self all is equal to self_open plus self form_header plus self double underscore form_open plus self double underscore inputs plus self double underscore form_close plus self page_content plus self_close. Next we type self all and make it equal to self all format in parenthesis **locals(). This will take all the privet HTML elements and display it publicly. That concludes the page.py file make sure that the app thing at the bottom isn’t in the page.py file only in the main file. We are now in the main.py file. At the top, make sure to add and import what we need. We need from page import FormPage, from xml.dom import minidom and import urllib2. We need to import the form page and the xml bits to make the api work. Inside the main handler and inside the def get, type view and make it equal to FormPage(). This will make a reference to our form page. Next view.form_header and make it equal to a string. Make a if statement with self request GET inside type code is equal to self request GET in brackets ‘code’. This will look for code in the input field. Next w_model is equal to Model(), next w_model.code is equal to self request GET in brackets ‘code’, next type in w_model.send_req(), next w_view is equal to View(), next w_view.wdo is equal to w_model.wdo, next type w_view.update(), next view.page_content is equal to w_view.content. Finally outside the if statement type in self response write in

parenthesis view.print_out(). All of that will take the information from the form field and take those results and puts them into future classes to spit the information right back. Now make a new class called View and object in the parenthesis. Inside type def __init__self, inside that self wdo is equal to DataObject(). Next line is self content is equal to blank string. Now make a new def called update and self is in parenthesis. Inside update type self content is equal to '''<h1>{self.wdo.location}<h1> {self.wdo.temp} {self.wdo.condition} '''. This will be where information from the api will be held. Below that type self content is equal to self content format and **locals() in parenthesis. Next make a new class called Model and object in parenthesis. Inside it type in def __init__self. Inside that type in self url is equal to the url in a string. Next line is self full_url is equal to a blank sting and below that goes self double underscore code is equal to a blank sting. This is where the url for the api will be and the variable, code and place it in the url. Make a new def called send_req and self in parenthesis. Inside that type self full_url is equal to self url plus self double underscore code. Next line, req is equal to urllib2.Request in parenthesis self full_url. This will make sense of the url and make a request for the information. Next line opener is equal to urllib2.builder_opener(). Next line data is equal to opener.open(req). Next line xmldoc is equal to minidom.parse(data). Next line is self double underscore wdo is equal to DataObject(). All of this will take and display the information and push it to the data object. Next line condition is equal to xmldoc.getElementsByTagName('yweather:conditon'). This is the tag in the xml where the data is located. Next line is self double underscore wdo.condition is equal to condition[0].attributes['text'].value, next is self double underscore wdo.temp is equal to condition[0].attributes['temp'].value, next is self double underscore wdo.code is equal to condition[0].attributes['code'].value. All of this is drilling down to the information inside of the tag and putting in the variable code the receive the request. Next is loc is equal to xmldoc.getElementsByTagName('yweather:location'), next line is self double underscore wdo.location is equal to loc[0].attributes['city'].value. Just as before, we are getting the information in the xml tag and drilling down to the information inside of the tag. Now make a getter @property, right below it make a def called wdo and self in parenthesis. Inside it type return self double underscore wdo. This will make a reference to the view. Now make a new getter @property. Right below that make a new def called code and self is in the parenthesis. Inside it type return self double underscore code. This will make the variable private. Now we need to make a @code.setter. Right underneath it make a def called code, in parenthesis self, c. inside it type in self double underscore code is equal to c. This will make the variable unprivate. Make your last class called DataObject and object is in the parenthesis. Inside it type def __init__self. Inside that type self location is equal to a blank string, self temp is equal to a blank sting, self condition is equal to a blank sting and self code is equal to zero. This is where the blank data objects are kept to hold the list of information.