***CostOut*** ([www.cbcsecosttoolkit.org](http://www.cbcsecosttoolkit.org)) is hosted by Pythonanywhere, an online Integrated Development Environment and Web hosting service based on the Python programming language. The domain name for the website is provided by HostCentric.

**Username & passwords**:

[www.pythonanywhere.com](http://www.pythonanywhere.com/) Login amrithany@gmail.com password is c0sttool

[www.hostcentric.com](http://www.hostcentric.com/).  Username **cbcsecosttoolkitorg** Password **Mar2015$**

**Structure on Pythonanywhere:** In the Dashboard in Pythonanywhere there are five tabs. In order to change the code, start the Bash console in the Consoles tab and navigate to the file you wish to change. After you have made your changes, you can reload your website by using the Reload [www.cbcsecosttoolkit.org](http://www.cbcsecosttoolkit.org) button in the Web tab.

You can upload any files you have changed on your local machine in the Files tab. You will have to reload your website after you do that.

**Code Structure:** There is a directory called costtool in //home/amritha. Within this, you should see two items:

* another directory with the same name as your project costtool and
* a Python script called manage.py.

Within the directory //home/amritha/costtool/costtool, you will find some Python scripts.

* settings.py - the place to store all your Django project’s settings; this is where you define the database, static and media location.
* urls.py - a Python script to store URL patterns for your project; every time you add a new template to your project, you must add a line to this file – it links the template file to the relevant function in views.py . An example is

**url(r'^project/add\_project.html$','costtool.views.add\_project','add\_project')**

* models.py - A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you’re storing. Generally, each model maps to a single database table. If you wish to create a new database table, add a new model, run the below line and the table will be created by django.

**python manage.py syncdb**

An example of a model is

**class Projects(m.Model):**

**projectname = m.CharField(max\_length=256)**

**typeanalysis = m.CharField(max\_length=256)**

**typeofcost = m.CharField(max\_length=256)**

**created\_at = m.DateTimeField(default=datetime.datetime.now)**

**updated\_at = m.DateTimeField()**

**user = m.CharField(max\_length=200,null=True, blank=True)**

**def \_\_unicode\_\_(self):**

**return self.projectname**

* forms.py - is where the django documentation recommends you place all your forms code; to keep your code easily maintainable. Also, since its a convention mentioned in the documentation, it helps when you are collaborating with others because that is where others will expect to look for your code dealing with forms. An example is

**class ProjectsForm(forms.ModelForm):**

**projectname = forms.CharField(label="Project Name:",error\_messages = {'required': "The Project Name is required"})**

**typeanalysis = forms.ChoiceField(required=False, choices=(('Cost Analysis','Cost Analysis'), ('Cost-Effectiveness Analysis', 'Cost-Effectiveness Analysis')), label="Type of Analysis:")**

**typeofcost = forms.ChoiceField(required=False, choices=(('Total Costs', 'Total Costs'), ('Incremental Costs', 'Incremental Costs')),label="Are you considering?")**

**class Meta:**

**model = Projects**

**fields = ('id','projectname','typeanalysis','typeofcost')**

* views.py - A view function, or *view* for short, is simply a Python function that takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image . . . or anything, really. The view itself contains whatever arbitrary logic is necessary to return that response. This code can live anywhere you want, as long as it’s on your Python path. There’s no other requirement–no “magic,” so to speak. For the sake of putting the code *somewhere*, the convention is to put views in a file called **views.py**. This is the file most used in Django.

An example is

**def edit\_project(request, project\_id):**

**proj = m.Projects.objects.get(pk=project\_id)**

**context = RequestContext(request)**

**if request.method == 'POST':**

**projectform = ProjectsForm(request.POST,instance=proj)**

**if projectform.is\_valid():**

**projectname = projectform.save(commit=False)**

**try:**

**p = m.Projects.objects.filter(projectname = projectname.projectname).count()**

**if p > 1:**

**return render\_to\_response('project/edit\_project.html',{'projectform':projectform,'err':'This project name is already taken. Please enter a unique name.'}, context**

**except ObjectDoesNotExist:**

**print projectname.projectname**

**projectname.updated\_at = datetime.datetime.now()**

**projectname.save(update\_fields=['projectname', 'updated\_at'])**

**return HttpResponseRedirect('/project/project\_list.html')**

**else:**

**print projectform.errors**

**else:**

**projectform = ProjectsForm(instance=proj)**

**return render\_to\_response(**

**'project/edit\_project.html',**

**{'projectform': projectform}, context)**

**Templates** - Django provides templates to make it easier for developers to achieve their design goal, as well as separating application logic from presentational concerns.

A template is simply a text file. It can generate any text-based format (HTML, XML, CSV, etc.). A template contains **variables**, which get replaced with values when the template is evaluated, and **tags**, which control the logic of the template.

//home/amritha/costtool/costtool/templates is where we are storing templates associated with our application. The home page of the website is //home/amritha/costtool/costtool/templates/about.html.

The content of the website is divided into different areas and the template files for each of them are stored in different directories.

1. Projects - all the screens pertaining to a project, including settings, inflation indices and geographical indices can be found in //home/amritha /costtool/costtool/templates/project directory.
2. Programs - all the screens pertaining to a program can be found in //home/amritha /costtool/costtool/templates/project/programs directory.
3. Tabbed screen- the main tabbed screen with program description, effectiveness, cost information, distribution of costs and transfers and subsidies is in the //home/amritha /costtool/costtool/templates/project/programs/effect directory. The files pertaining only to cost information can be found in //home/amritha /costtool/costtool/templates/project/programs/costs directory. The files pertaining only to distribution of costs can be found in //home/amritha /costtool/costtool/templates/project/programs/dist directory. The files pertaining only to transfers and subsidies can be found in //home/amritha /costtool/costtool/templates/project/programs/transfer directory.
4. Prices - all the screens pertaining to prices can be found in //home/amritha /costtool/costtool/templates/prices directory.
5. Registration and Login - all the screens pertaining to new registration can be found in //home/amritha /costtool/costtool/templates/register directory and to login in //home/amritha /costtool/costtool/templates/login directory.
6. Reports - all the screens pertaining to reports can be found in //home/amritha /costtool/costtool/templates/reports directory.
7. Admin - all the admin screens can be found in //home/amritha /costtool/costtool/templates/admin directory.

**Database:** To access the MySql database, type the below line in your Bash console. The password is pass.

**mysql -u amritha -h mysql.server –p**

Type **use amritha$costtool;** at the SQL prompt and you are good to make any changes to your database.

PYTHONPATH on Pythonanywhere:

**export PYTHONPATH=/home/amritha/.virtualenvs/costtool/lib/python2.7/site-packages:$PYTHONPATH**

Documents: All the documents on the website are in the //home/amritha/costtool/documents directory.

**Changes on your local machine:**

The costtool directory is in /users/amritha/downloads

In order to run the changes you have made on your local machine, navigate to /users/amritha/downloads/costtool and run this command

**python manage.py runserver**

Type <http://127.0.0.1:8000> to view your webpage. If you want to sync your database, use

**python manage.py syncdb**

If you are running the above commands for the first time, you have to start the mysql server by typing the below command

**mysql.server start**

In order to navigate to the database, use the below commands

**cd /usr/local/mysql/bin/**

**./mysql -u root -h localhost -p**

PYTHONPATH on your local machine:

**export PYTHONPATH=/usr/local/lib/python2.7/site-packages:/usr/local/mysql/bin:$PYTHONPATH**

**A few other things:**

* All the files in pythonanywhere and your local machine should be exactly the same except for settings.py and views.py.
* tabbedlayout in views.py is a long and complicated function. It has POST and GET code for all the tabs in the tabbed screen – program description, effectiveness, cost information, distribution of costs and transfers and subsidies. If you change “render” or HttpResponse in one part of the function, make sure you change it everywhere else in the function.
* Two unix commands useful while pasting the code into the bash console in Pythonanywhere are **set nonumber** and **set paste**.