**Python - Try Django Challenges**

**Getting Started (Level 1)**

1. firstView.py

Write a simple view in an app:

* Create a function called home() that a request parameter, that returns an HttpResponse.
* Update the returned HttpResponse to include the text “Welcome Home Eaters!”.
* Copy code to \*/FoodTracker/main/views.py file.

1. firstURL.py

Create a simple URL pattern for a view:

* Create a url() object in the urlpatterns array.
* Pass in the URL you’re matching, “home/” as a parameter to the url() object.
* Pass in the corresponding views.home view as a parameter to the url() object.
* Copy code to \*/FoodTracker/FoodTracker/urls.py, and replace everything following the existing docstring.
* Navigate to <http://localhost:8000/home/> to test your code.

1. projecturls.py, mainurls.py

Refactor the existing URL Dispatchers:

* In the projecturls.py file, edit the second url() object to include(‘main.urls’).
* In the projecturls.py file, change the regex parameter to match a blank path.
* In the mainurls.py file, add a url() object whose regex parameter takes an empty path that terminates, and goes to views.home.
* Copy code from projecturls.py to \*/FoodTracker/FoodTracker/urls.py, and replace everything following the existing docstring.
* Copy code from mainurls.py to \*/FoodTracker/main/urls.py.
* Navigate to <http://localhost:8000> to test your code.

**Templates (Level 2)**

1. renderView.py, home.html

Update the existing template:

* In the renderView.py file, import render from django.shortcuts
* In the renderView.py file, update the home() function to return a render object that takes in a request object, and the home.html template
* Move home.html to \*/FoodTracker/main/templates directory.
* Update code in \*/FoodTracker/main/views.py with the code in renderView.py

1. dynamicData.py, dynamicHome.html

Render dynamic data in a template:

* In the dynamicData.py file, create a dictionary in the view that holds the “location\_name” and “flavors” values using keys with the same name.
* In the dynamicData.py file, pass the dictionary as the third argument to the render function.
* In the dynamicHome.html file, display the dictionary variables “location\_name” and “flavors” in separate paragraph tags, using the Django Language Syntax.
* Add the main app to \*/FoodTracker/FoodTracker/settings.py INSTALLED\_APPS
* Update code in \*/FoodTracker/main/views.py with the code in dynamicData.py
* Update code in \*/FoodTracker/main/templates/home.html with the code in dynamicHome.html
* Navigate to <http://localhost:8000> to test your code

1. firstClass.py

Create a simple location class:

* Add the “flavors” attribute to the existing Location class.
* Add the “num\_franchises” attribute to the existing Location class.
* Add the “image” attribute to the existing Location class.

1. classList.py

Create a list of class objects

* Add a third Location object to the locations list with the following values:
  + name = “The Hill Top, CT”, flavors = any string you want, num\_franchises = any integer you’d like, image = “http://www.gffoodservice.org/wp-content/uploads/2015/03/restaurant-e1456862749354.jpg”

Copy code in classList.py to \*/FoodTracker/main/views.py

1. renderList.py, renderList.html

Render a list in a template

* In the rednerList.py file, remove the location\_name and flavors variables, and set the context dictionary object to hold only the locations list.
* In the renderList.html file, add a Django template language for loop to loop over each location in locations.
* In the renderList.html file, add paragraphs tags in the for loop to display location.name and location.flavors.
* In the renderList.html file, add an image tag of width=80, and src=location.image in the for loop.
* Move code in renderList.py to \*/FoodTracker/main/views.py
* Move code in renderList.html to \*/FoodTracker/main/templates/home.html
* Navigate to <http://localhost:8000> to test your code

1. styledTemplate.html

Add styles to a template

* Add the Django Template Language tag to load the staticfiles directory.
* Update the first link’s href attribute to link to boostrap.min.css.
* Update the second link’s href attribute to link to styles.css.
* Add bootstrap.min.css to \*/FoodTracker/main/static.
* Add styles.css to \*/FoodTracker/main/static.
* Move code in styledTemplate.html to \*/FoodTracker/main/templates/home.html.
* Navigate to <http://localhost:8000> to test your code.

1. cycleTag.html

Add a cycle tag to a template:

* Refer to the Cycle Tag Documentation if needed (<https://docs.djangoproject.com/ja/1.9/ref/templates/builtins/#cycle>).
* On line 26, add a Django Template Language cycle tag to cycle between tracker-left and tracker-right.
* Move the /images directory to \*/FoodTracker/main/static directory
* Copy code from cycleTag.html to \*/FoodTracker/main/templates/home.html
* Navigate to <http://localhost:8000> to test your code

**Models (Level 3)**

1. firstModel.py

Create a simple model:

* Add the following fields to the model, with the correct Django field types:
  + “name” - a string, “flavors” - a string, “num\_franchises” - an integer, “image” - a string
* Set the CharField() types to have a max\_length of 75.
* Add a \_\_str\_\_ method that will return the self.name value.
* Copy code from firstModel.py into \*/FoodTracker/main/models.py

1. modelView.py

Refactor view to use model:

* Import the Location model from .models.
* Create a list object called locations, and set it equal to all() of the objects in the Location model using a QuerySet.
* Copy code from modelView.py into \*/FoodTracker/main/views.py
* Navigate to <http://localhost:8000> to test your code.

1. registerAdmin.py

Register an admin:

* Import the Location model.
* Register the Location model with the admin.
* Copy code in registerAdmin.py to \*/FoodTracker/main/admin.py.
* Navigate to <http://localhost:8000/admin> to login and test your code.