---------- DJANGO TUTORIAL 14 ----------

In this video we will add restrictions on our site based off of whether someone is logged in or not. Creating and editing articles will be based on the logged in user. We'll require everyone to log in and more.

1. I want to fix items/models.py by indenting __str__ and get_absolute_url

```
    seller = models.ForeignKey(
    get_user_model(),
    on_delete=models.CASCADE,)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('item_detail', args=[str(self.id)])
```

2. Set create view so that the author is defined by who is logged in in items/views.py

```
class ItemCreateView(CreateView):
  model = Item
  template_name = 'item_new.html'
  fields = ('title', 'description', 'seller', 'price', 'photo')
  success_url = reverse_lazy('item_list')

  # Verify that the seller is the user when creating an item post
  def form_valid(self, form):
    form.instance.seller = self.request.user
    return super().form_valid(form)
```

3. Restrict access to only logged in users with the LoginRequired mixin in items/views.py
Now if a user tries to access an article and they aren't logged in they are redirected to the login page

```
from django.contrib.auth.mixins import LoginRequiredMixin
```

4. Add LoginRequiredMixin to all article views in items/views.py

```
class ItemListView(LoginRequiredMixin, ListView):
  model = Item
  template_name = 'item_list.html'
  login_url = 'login'

class ItemDetailView(LoginRequiredMixin, DetailView):
  model = Item
  template_name = 'item_detail.html'
  login_url = 'login'

class ItemUpdateView(LoginRequiredMixin, UpdateView):
  model = Item
  fields = ('title', 'description', 'price')
  template_name = 'item_edit.html'
```

```
  login_url = 'login'

class ItemDeleteView(LoginRequiredMixin, DeleteView):
  model = Item
  template_name = 'item_delete.html'
  success_url = reverse_lazy('item_list')
  login_url = 'login'
```

5. Restrict access to change items only by the item poster by adding the mixin UserPassesTestMixin to the update and delete views that verify that the current logged in user is the poster. items/views.py

```
from django.contrib.auth.mixins import (
LoginRequiredMixin, UserPassesTestMixin)

class ItemUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
  model = Item
  fields = ('title', 'description', 'price')
  template_name = 'item_edit.html'
  login_url = 'login'

  # Verifies author is equal to logged in user
  def test_func(self):
      obj = self.get_object()
      return obj.seller == self.request.user

class ItemDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
  model = Item
  template_name = 'item_delete.html'
  success_url = reverse_lazy('item_list')
  login_url = 'login'

  # Verifies author is equal to logged in user
  def test_func(self):
      obj = self.get_object()
      return obj.seller == self.request.user
```

6. Add the ability to Edit and Delete item posts. Also add a link to show the whole list of bikes.

```
<article>
    <p>offered by {{ object.seller }} | {{ object.date }}</p>
    <img src="{{ item.photo.url }}" alt="{{ item.title }}" class="art-img">
    <p>{{ object.description|linebreaks }}</p>
    <p>Price : {{ object.price }}</p>
  </article>
  </div>
  <p><a href="{% url 'item_edit' item.pk %}">Edit</a> | <a href="{% url 'item_delete' item.pk
%}">Delete</a>
  </p>
  <p>Back to <a href="{% url 'item_list' %}">All Items</a>.</p>
{% endblock content %}
```

And, that's it! It is amazing how much functionality we can add to our site because Django has so many features built in.