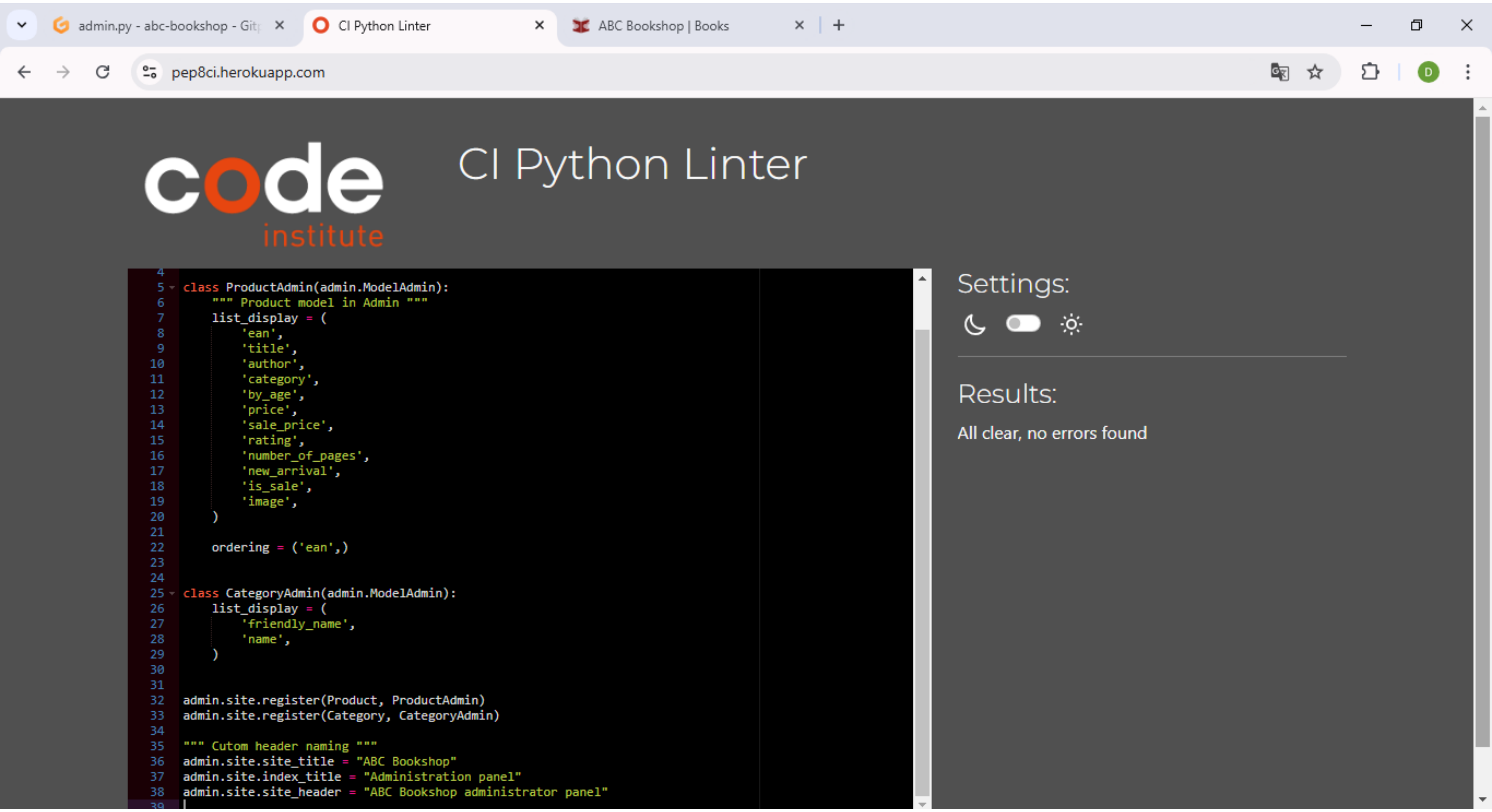
admin.py



CI Python Linter

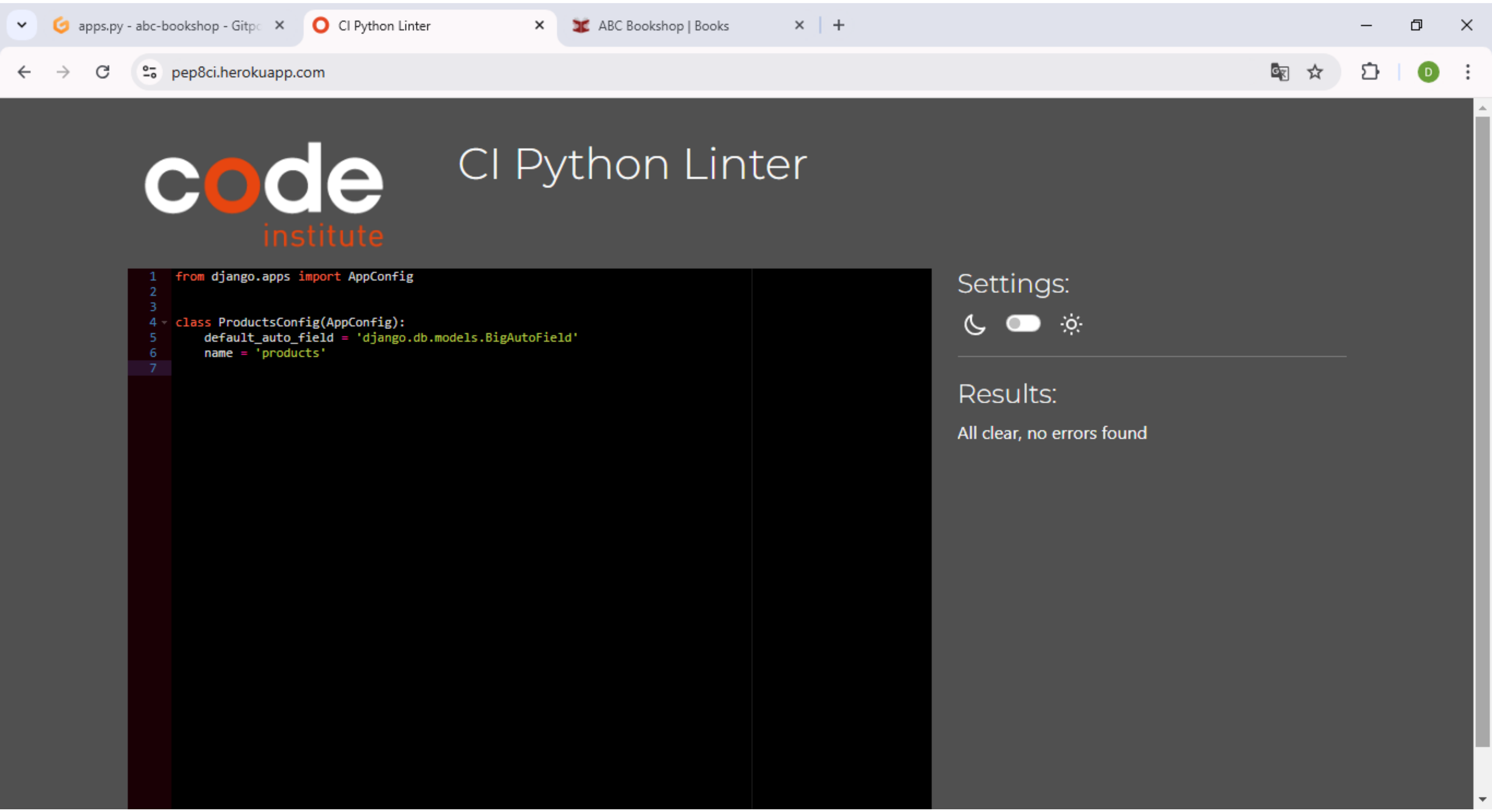
```
class ProductAdmin(admin.ModelAdmin):
    """ Product model in Admin """
    list_display = (
        'ean',
        'title',
        'author',
        'category',
        'by_age',
        'price',
        'sale_price',
        'rating',
        'number_of_pages',
        'new_arrival',
        'is_sale',
        'image',
    )

    ordering = ('ean',)


class CategoryAdmin(admin.ModelAdmin):
    list_display = (
        'friendly_name',
        'name',
    )


admin.site.register(Product, ProductAdmin)
admin.site.register(Category, CategoryAdmin)

""" Cutom header naming """
admin.site.site_title = "ABC Bookshop"
admin.site.index_title = "Administration panel"
admin.site.site_header = "ABC Bookshop administrator panel"
```

Settings:

Results:

All clear, no errors found

apps.py



CI Python Linter

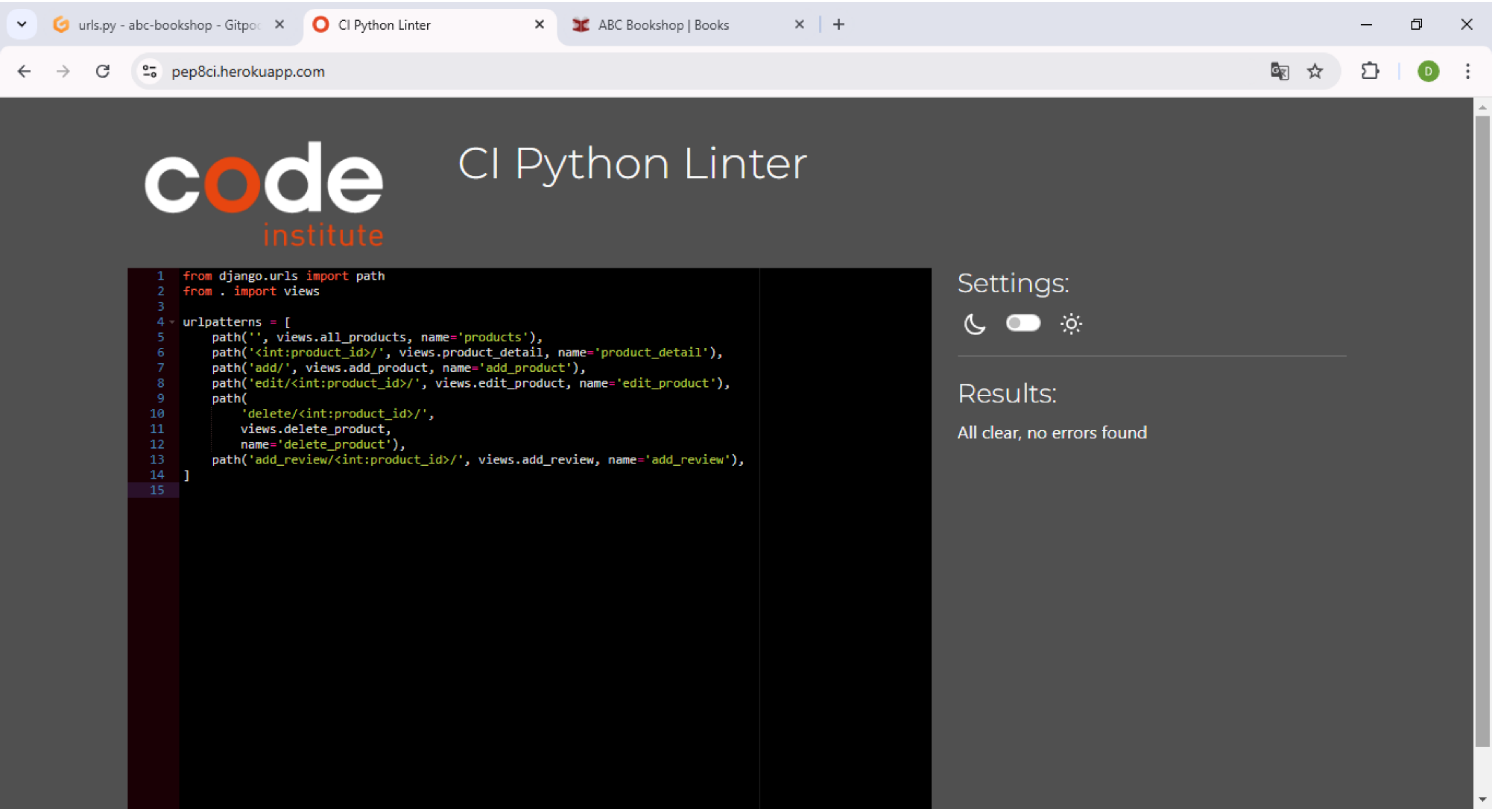
```
from django.apps import AppConfig


class ProductsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'products'
```

Settings:

Results:

All clear, no errors found

urls.py

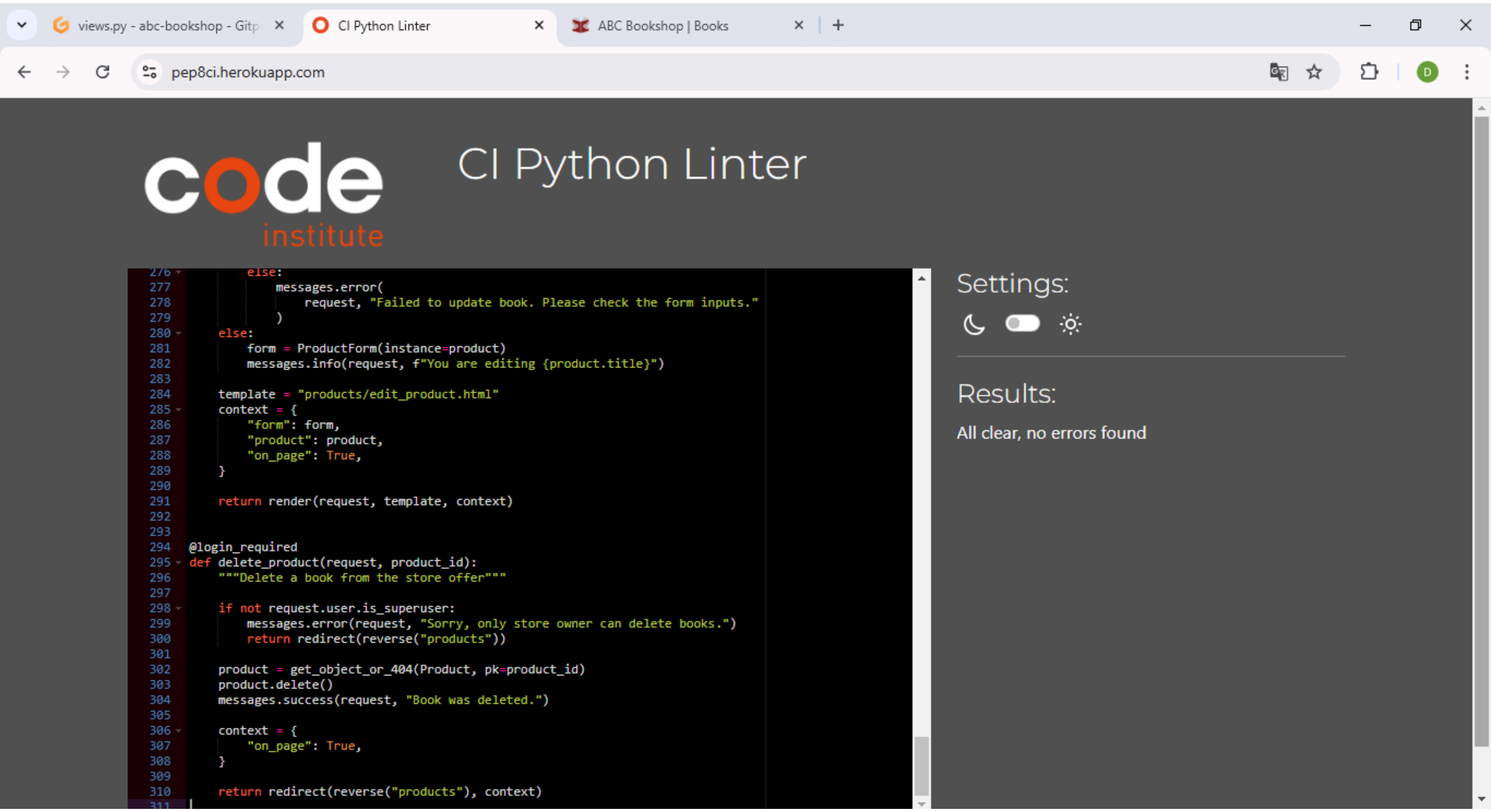


```python
from django.urls import path
from . import views

urlpatterns = [
    path('', views.all_products, name='products'),
    path('<int:product_id>/', views.product_detail, name='product_detail'),
    path('add/', views.add_product, name='add_product'),
    path('edit/<int:product_id>/', views.edit_product, name='edit_product'),
    path(
        'delete/<int:product_id>/',
        views.delete_product,
        name='delete_product'),
    path('add_review/<int:product_id>/', views.add_review, name='add_review'),
]
```

Results:

All clear, no errors found

views.py



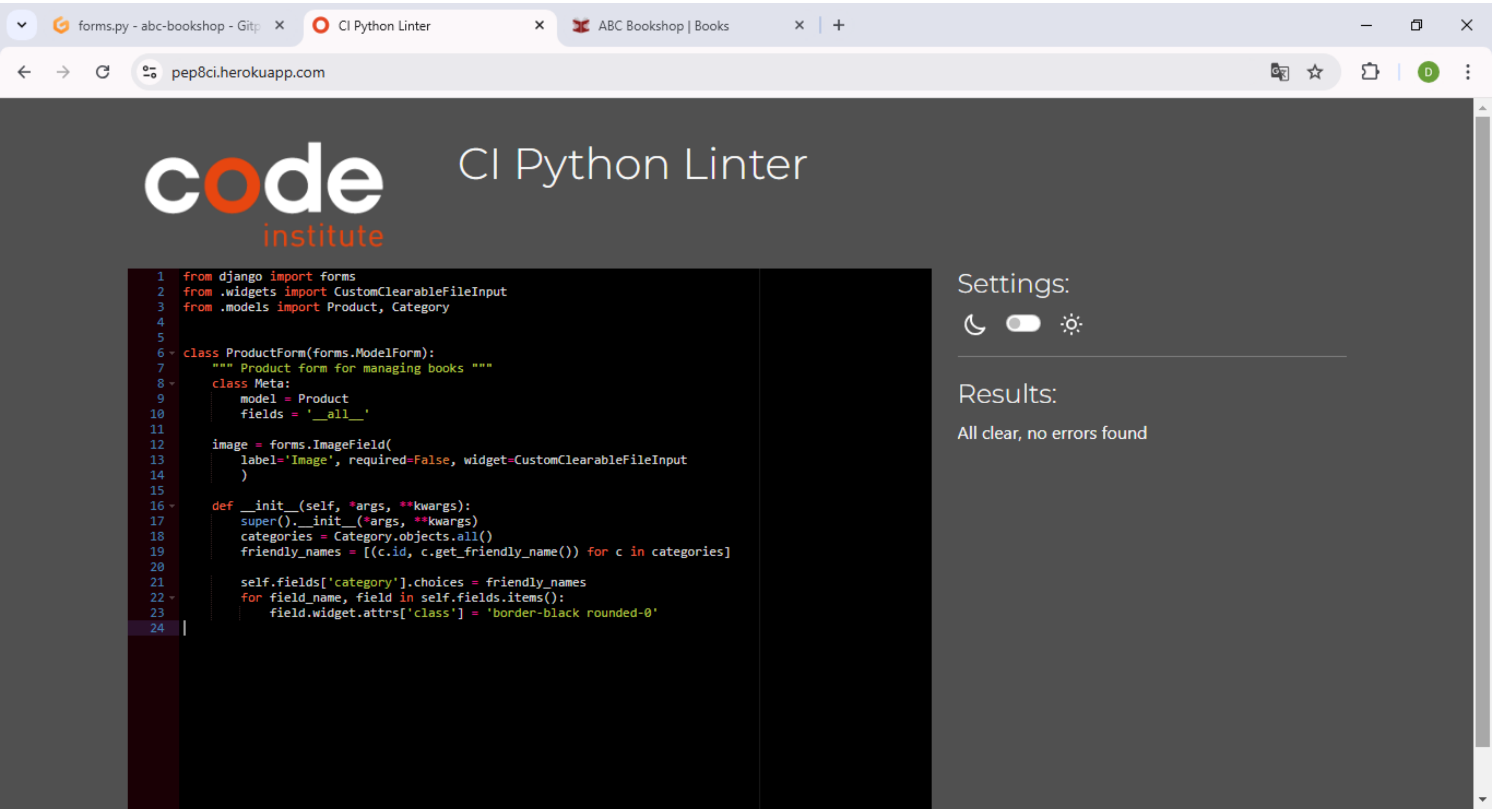
```
            else:
                messages.error(
                    request, "Failed to update book. Please check the form inputs."
                )
        else:
            form = ProductForm(instance=product)
            messages.info(request, f"You are editing {product.title}")

        template = "products/edit_product.html"
        context = {
            "form": form,
            "product": product,
            "on_page": True,
        }

        return render(request, template, context)


@login_required
def delete_product(request, product_id):
    """Delete a book from the store offer"""

    if not request.user.is_superuser:
        messages.error(request, "Sorry, only store owner can delete books.")
        return redirect(reverse("products"))

    product = get_object_or_404(Product, pk=product_id)
    product.delete()
    messages.success(request, "Book was deleted.")

    context = {
        "on_page": True,
    }

    return redirect(reverse("products"), context)
```

Settings:

Results:

All clear, no errors found

forms.py



CI Python Linter

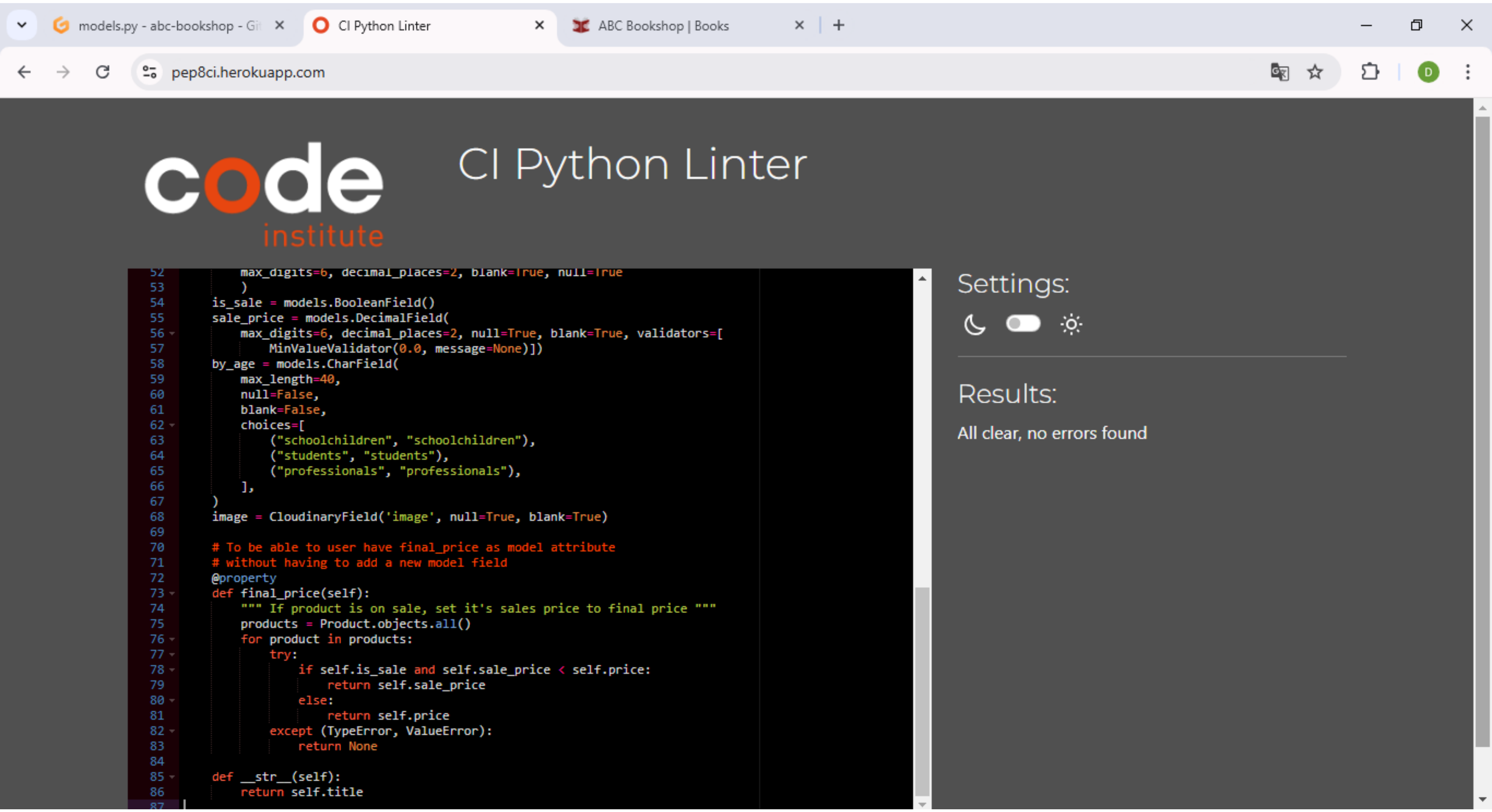
```
from django import forms
from .widgets import CustomClearableFileInput
from .models import Product, Category


class ProductForm(forms.ModelForm):
    """ Product form for managing books """
    class Meta:
        model = Product
        fields = '__all__'

    image = forms.ImageField(
        label='Image', required=False, widget=CustomClearableFileInput
        )

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        categories = Category.objects.all()
        friendly_names = [(c.id, c.get_friendly_name()) for c in categories]

        self.fields['category'].choices = friendly_names
        for field_name, field in self.fields.items():
            field.widget.attrs['class'] = 'border-black rounded-0'
```

Settings:

Results:

All clear, no errors found

models.py



CI Python Linter

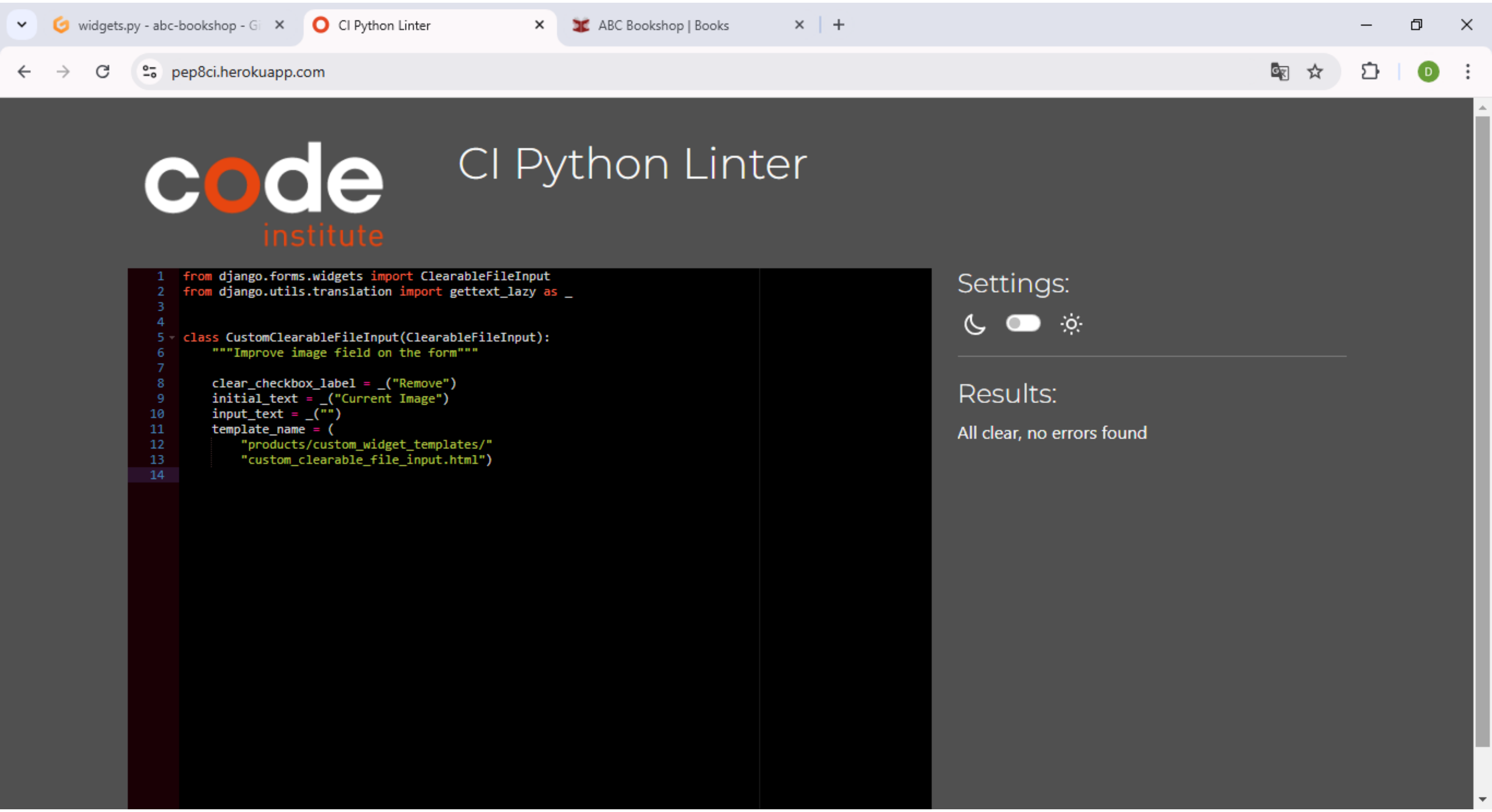
```
            max_digits=6, decimal_places=2, blank=True, null=True
        )
    is_sale = models.BooleanField()
    sale_price = models.DecimalField(
        max_digits=6, decimal_places=2, null=True, blank=True, validators=[
            MinValueValidator(0.0, message=None)])
    by_age = models.CharField(
        max_length=40,
        null=False,
        blank=False,
        choices=[
            ("schoolchildren", "schoolchildren"),
            ("students", "students"),
            ("professionals", "professionals"),
        ],
    )
    image = CloudinaryField('image', null=True, blank=True)

    # To be able to user have final_price as model attribute
    # without having to add a new model field
    @property
    def final_price(self):
        """ If product is on sale, set it's sales price to final price """
        products = Product.objects.all()
        for product in products:
            try:
                if self.is_sale and self.sale_price < self.price:
                    return self.sale_price
                else:
                    return self.price
            except (TypeError, ValueError):
                return None

    def __str__(self):
        return self.title
```

Settings:

Results:

All clear, no errors found

widgets.py



CI Python Linter

```
from django.forms.widgets import ClearableFileInput
from django.utils.translation import gettext_lazy as _


class CustomClearableFileInput(ClearableFileInput):
    """Improve image field on the form"""

    clear_checkbox_label = _("Remove")
    initial_text = _("Current Image")
    input_text = _("")
    template_name = (
        "products/custom_widget_templates/"
        "custom_clearable_file_input.html")
```

Settings:

Results:

All clear, no errors found