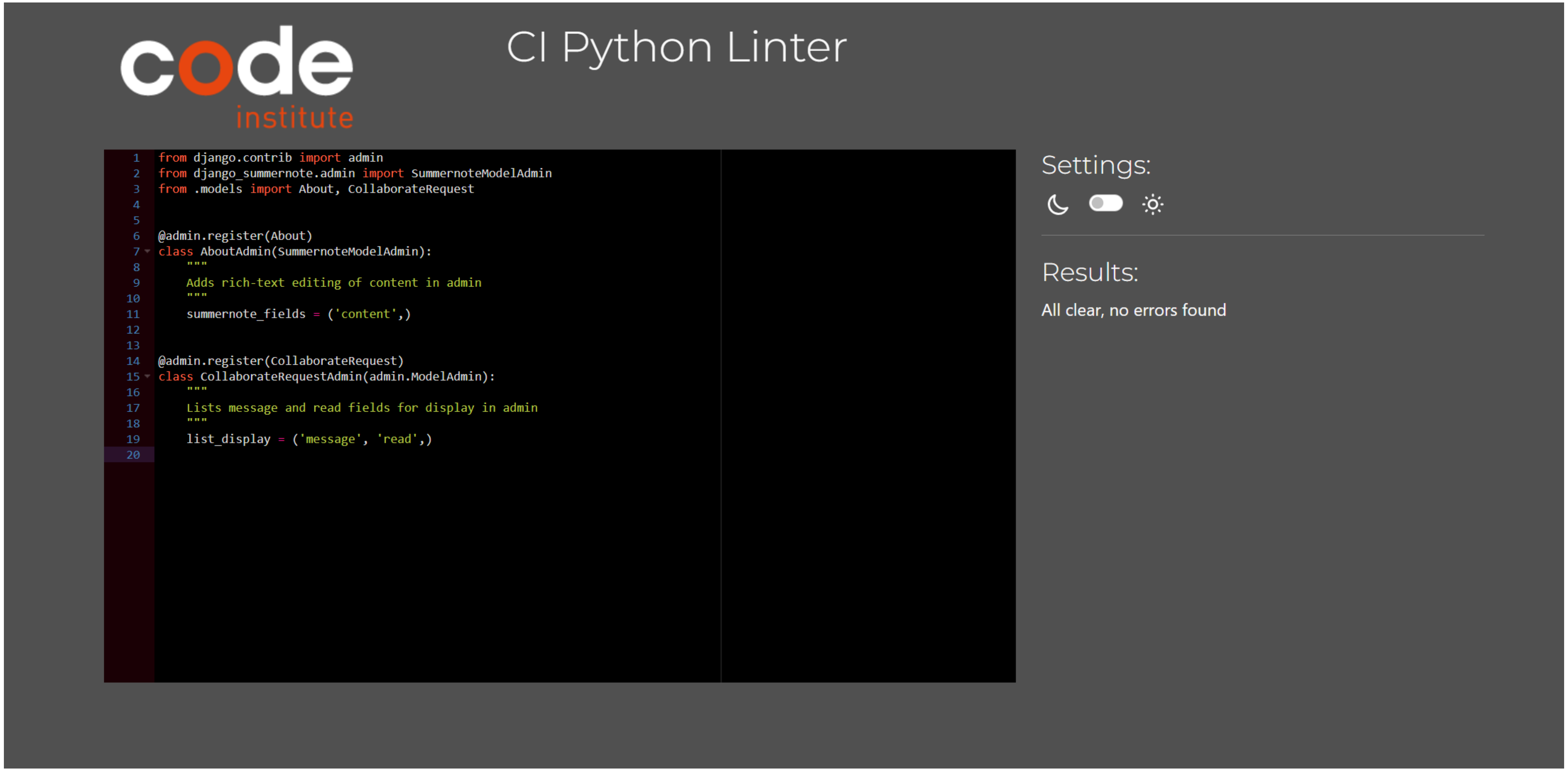
About App – admin.py



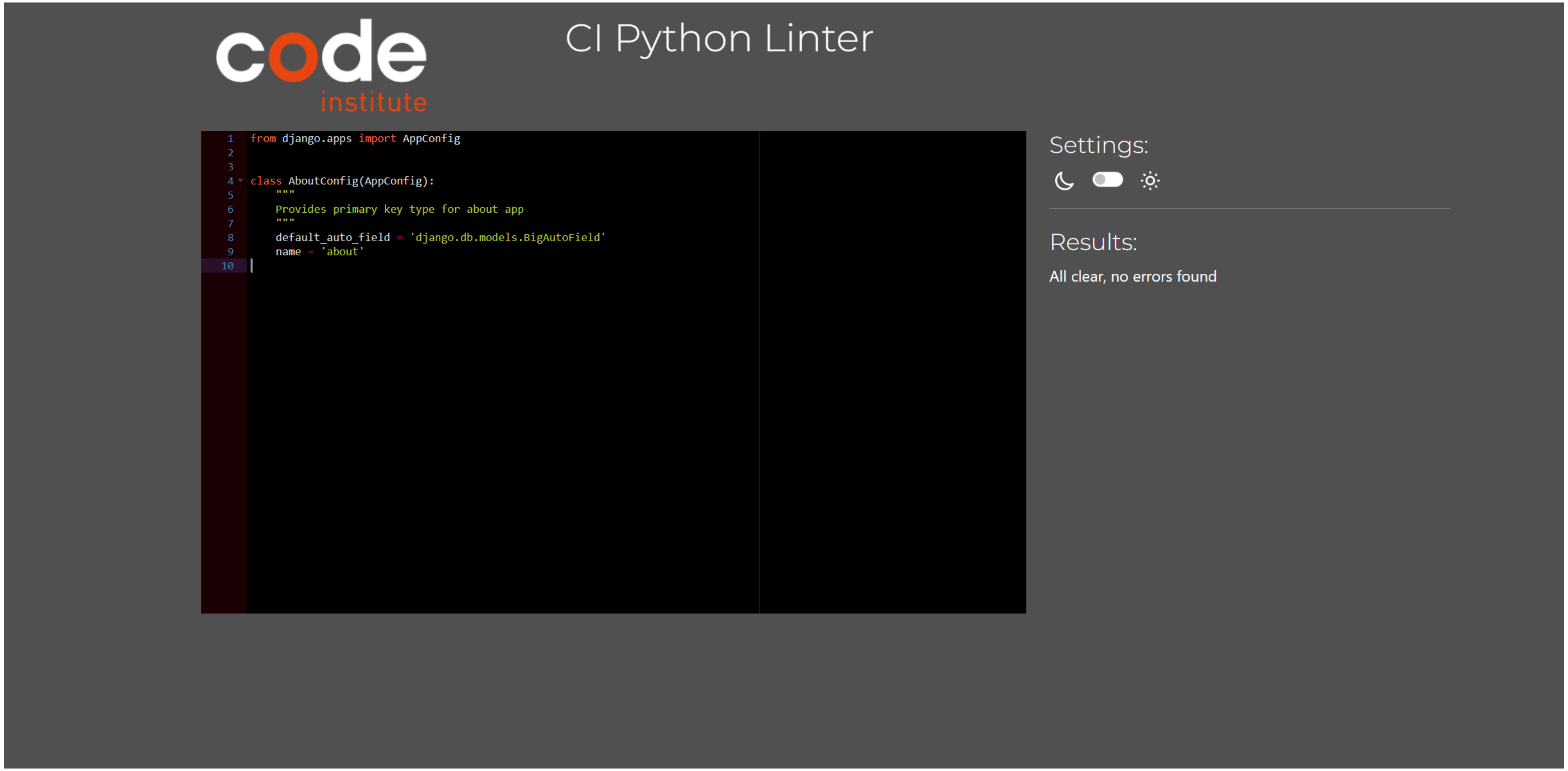
```
from django.contrib import admin
from django_summernote.admin import SummernoteModelAdmin
from .models import About, CollaborateRequest


@admin.register(About)
class AboutAdmin(SummernoteModelAdmin):
    """
    Adds rich-text editing of content in admin
    """
    summernote_fields = ('content',)


@admin.register(CollaborateRequest)
class CollaborateRequestAdmin(admin.ModelAdmin):
    """
    Lists message and read fields for display in admin
    """
    list_display = ('message', 'read',)

```

Settings:

Results:

All clear, no errors found

About App – apps.py

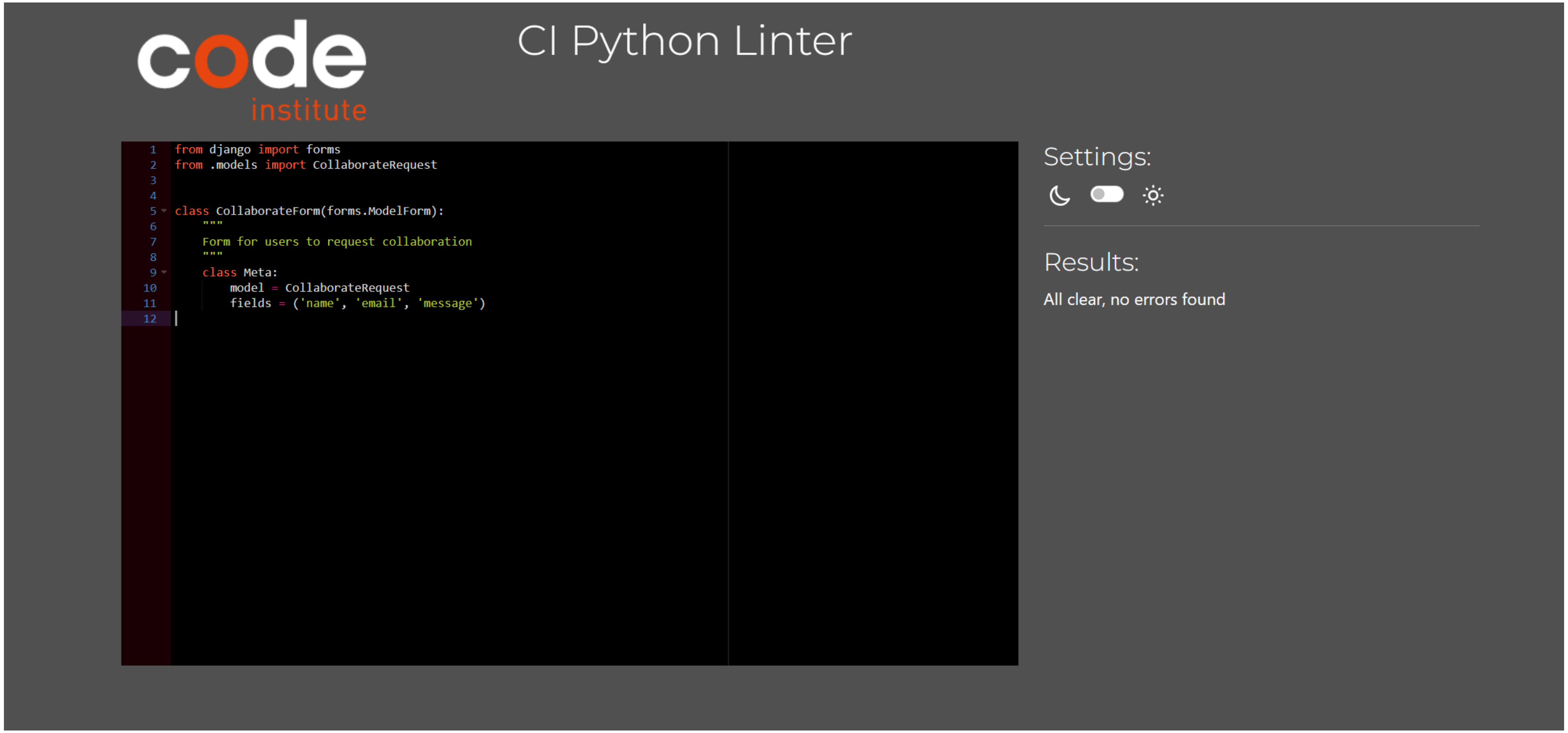


```
from django.apps import AppConfig


class AboutConfig(AppConfig):
    """
    Provides primary key type for about app
    """
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'about'

```

Results:

All clear, no errors found

About App – forms.py



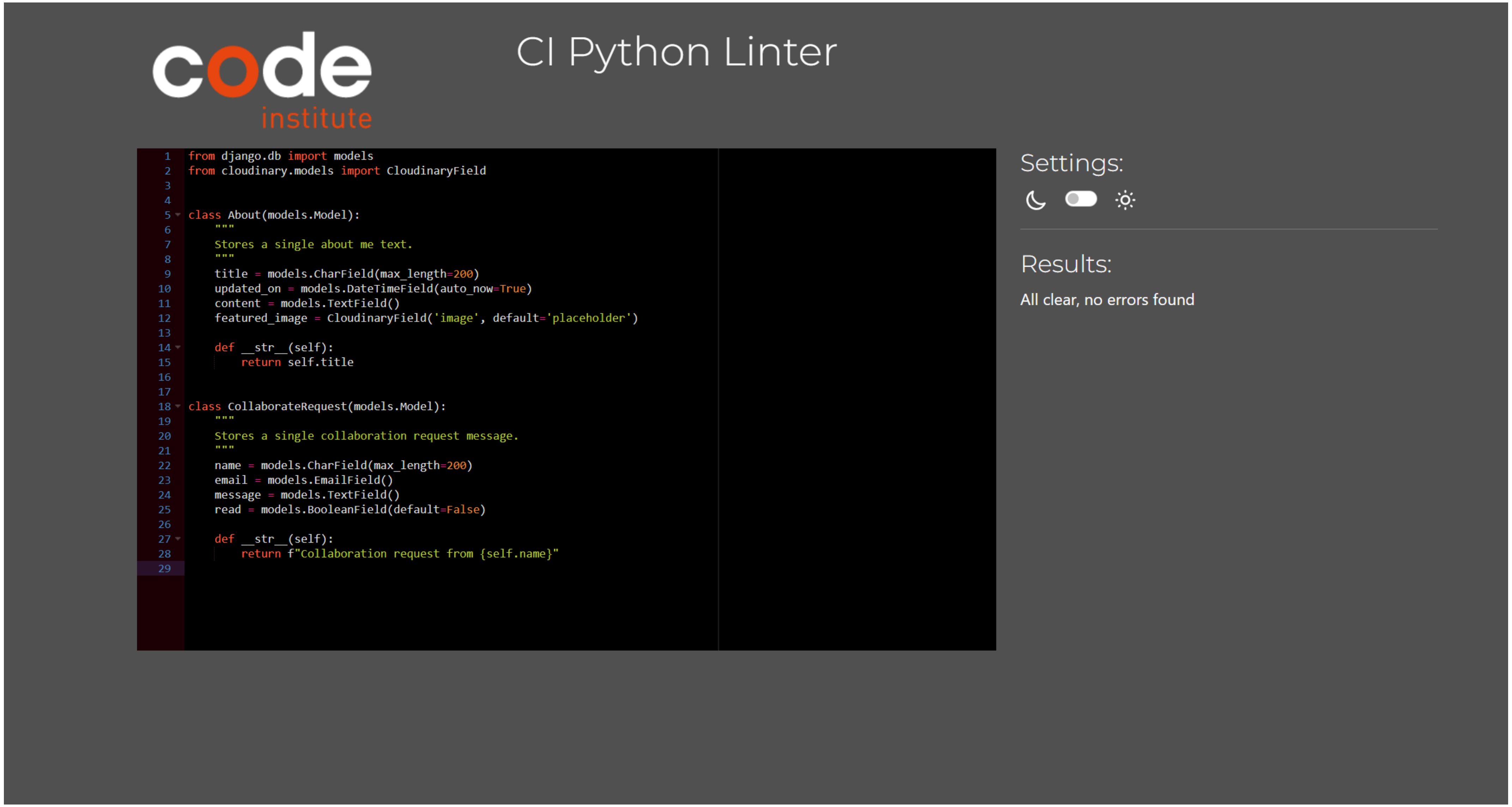
```
from django import forms
from .models import CollaborateRequest


class CollaborateForm(forms.ModelForm):
    """
    Form for users to request collaboration
    """
    class Meta:
        model = CollaborateRequest
        fields = ('name', 'email', 'message')

```

Settings:

Results:

All clear, no errors found

About App – models.py

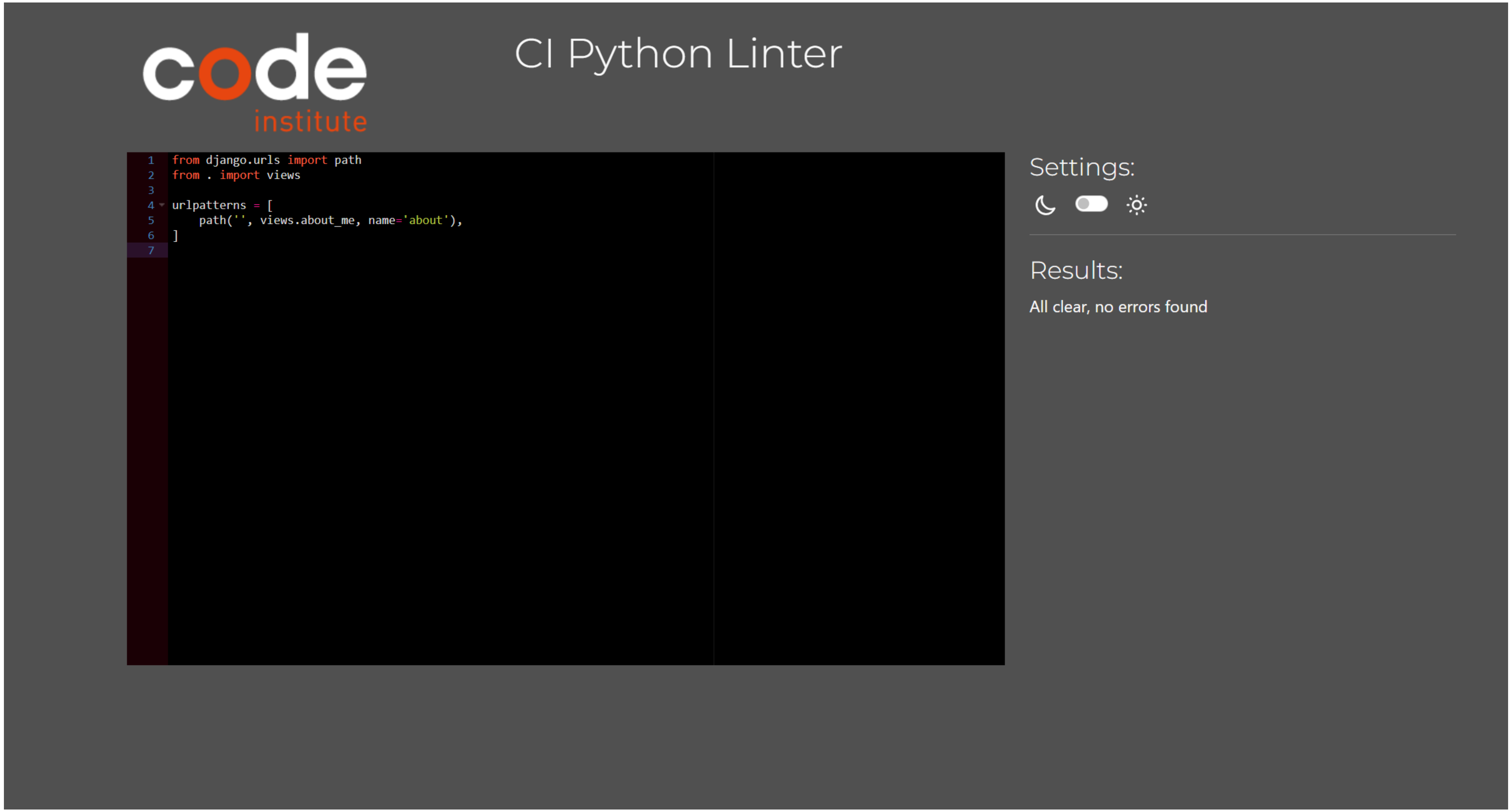


```python
from django.db import models
from cloudinary.models import CloudinaryField


class About(models.Model):
    """
    Stores a single about me text.
    """
    title = models.CharField(max_length=200)
    updated_on = models.DateTimeField(auto_now=True)
    content = models.TextField()
    featured_image = CloudinaryField('image', default='placeholder')

    def __str__(self):
        return self.title


class CollaborateRequest(models.Model):
    """
    Stores a single collaboration request message.
    """
    name = models.CharField(max_length=200)
    email = models.EmailField()
    message = models.TextField()
    read = models.BooleanField(default=False)

    def __str__(self):
        return f"Collaboration request from {self.name}"
```

Results:

All clear, no errors found

About App – urls.py

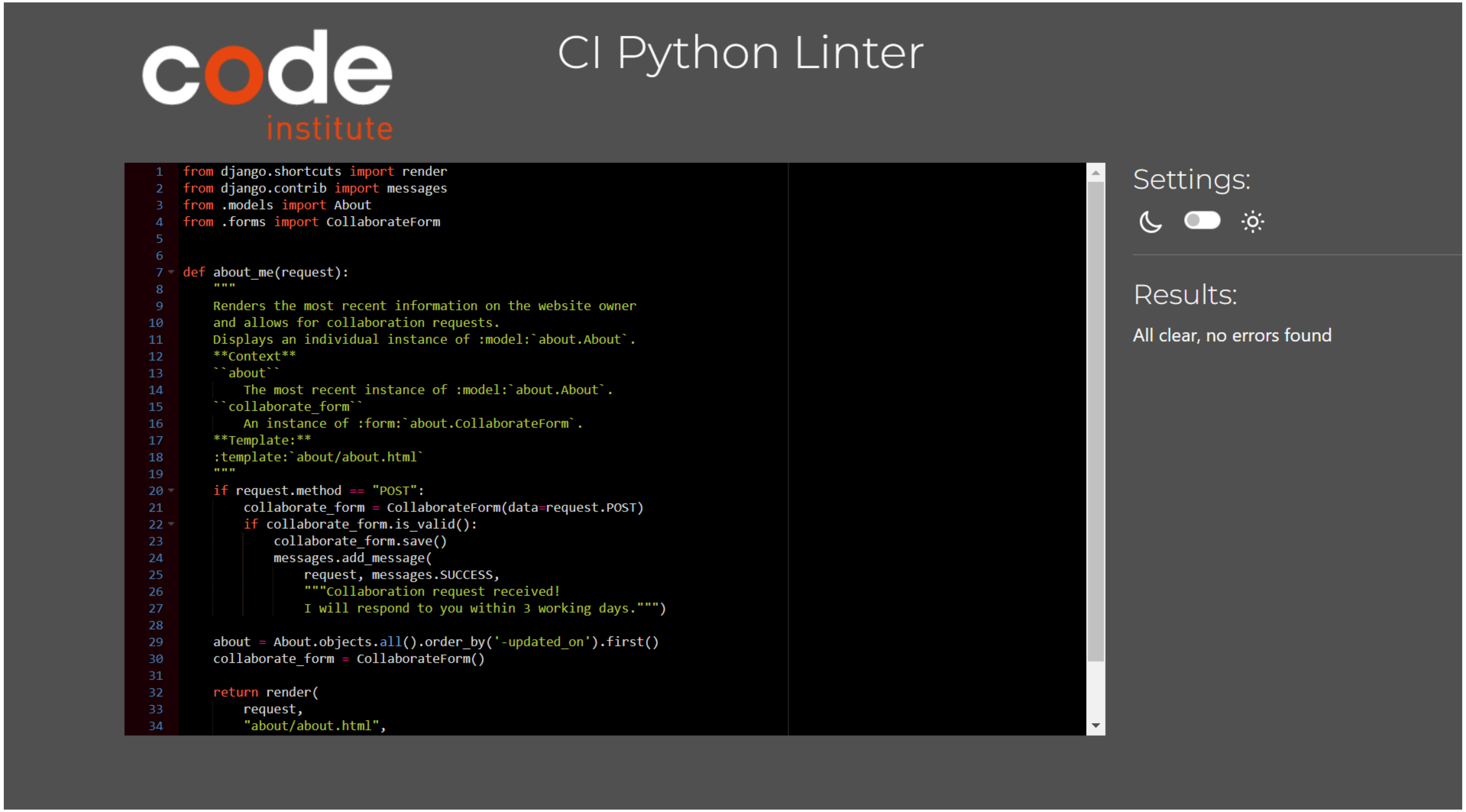


```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.about_me, name='about'),
]
```

Results:

All clear, no errors found

About App – views.py



CI Python Linter

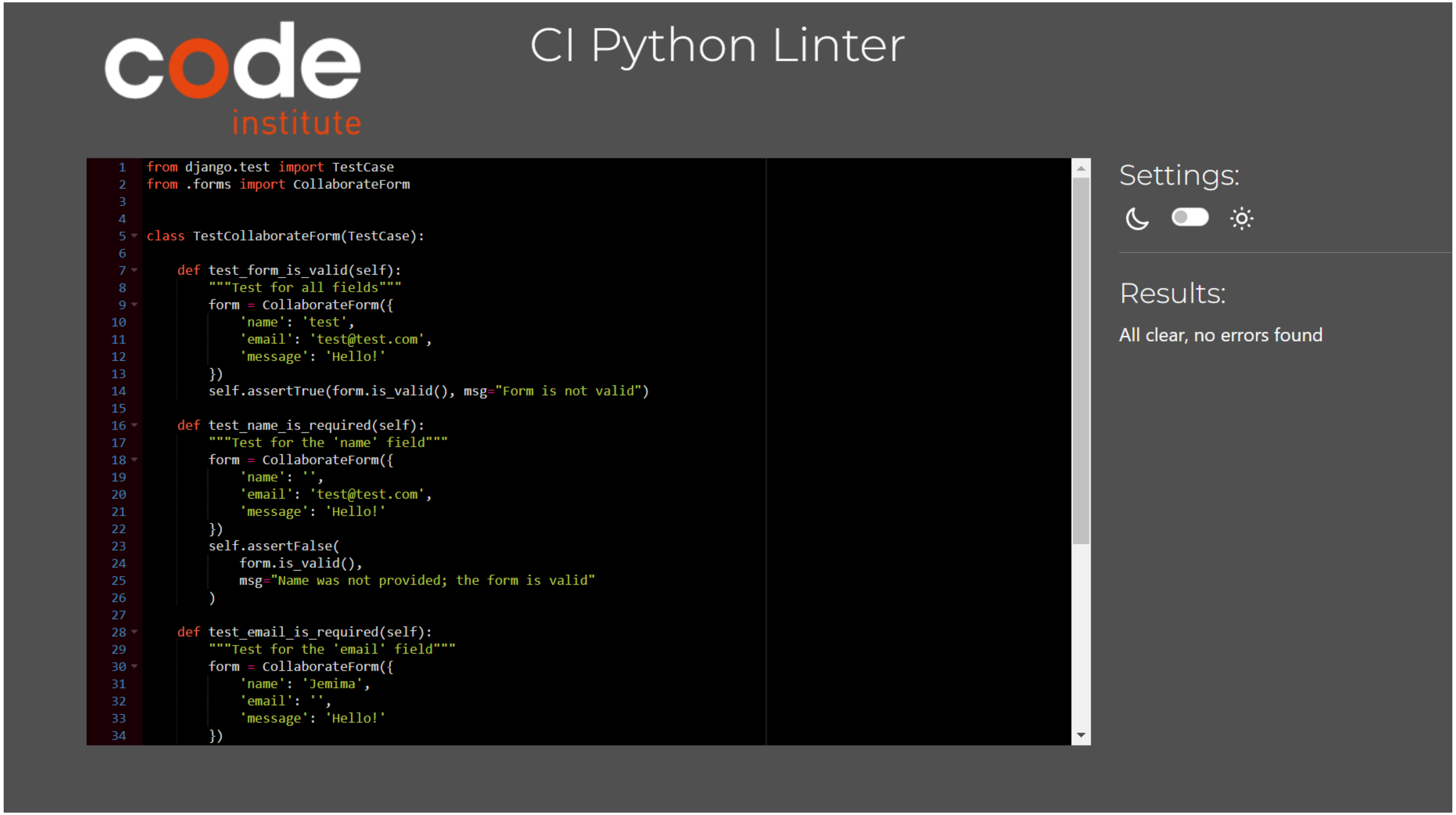
```python
from django.shortcuts import render
from django.contrib import messages
from .models import About
from .forms import CollaborateForm


def about_me(request):
    """
    Renders the most recent information on the website owner
    and allows for collaboration requests.
    Displays an individual instance of :model:`about.About`.
    **Context**
    ``about``
        The most recent instance of :model:`about.About`.
    ``collaborate_form``
        An instance of :form:`about.CollaborateForm`.
    **Template:**
    :template:`about/about.html`
    """
    if request.method == "POST":
        collaborate_form = CollaborateForm(data=request.POST)
        if collaborate_form.is_valid():
            collaborate_form.save()
            messages.add_message(
                request, messages.SUCCESS,
                """Collaboration request received!
                I will respond to you within 3 working days.""")

    about = About.objects.all().order_by('-updated_on').first()
    collaborate_form = CollaborateForm()

    return render(
        request,
        "about/about.html",
```

Settings:

Results:

All clear, no errors found

About App – test_forms.py

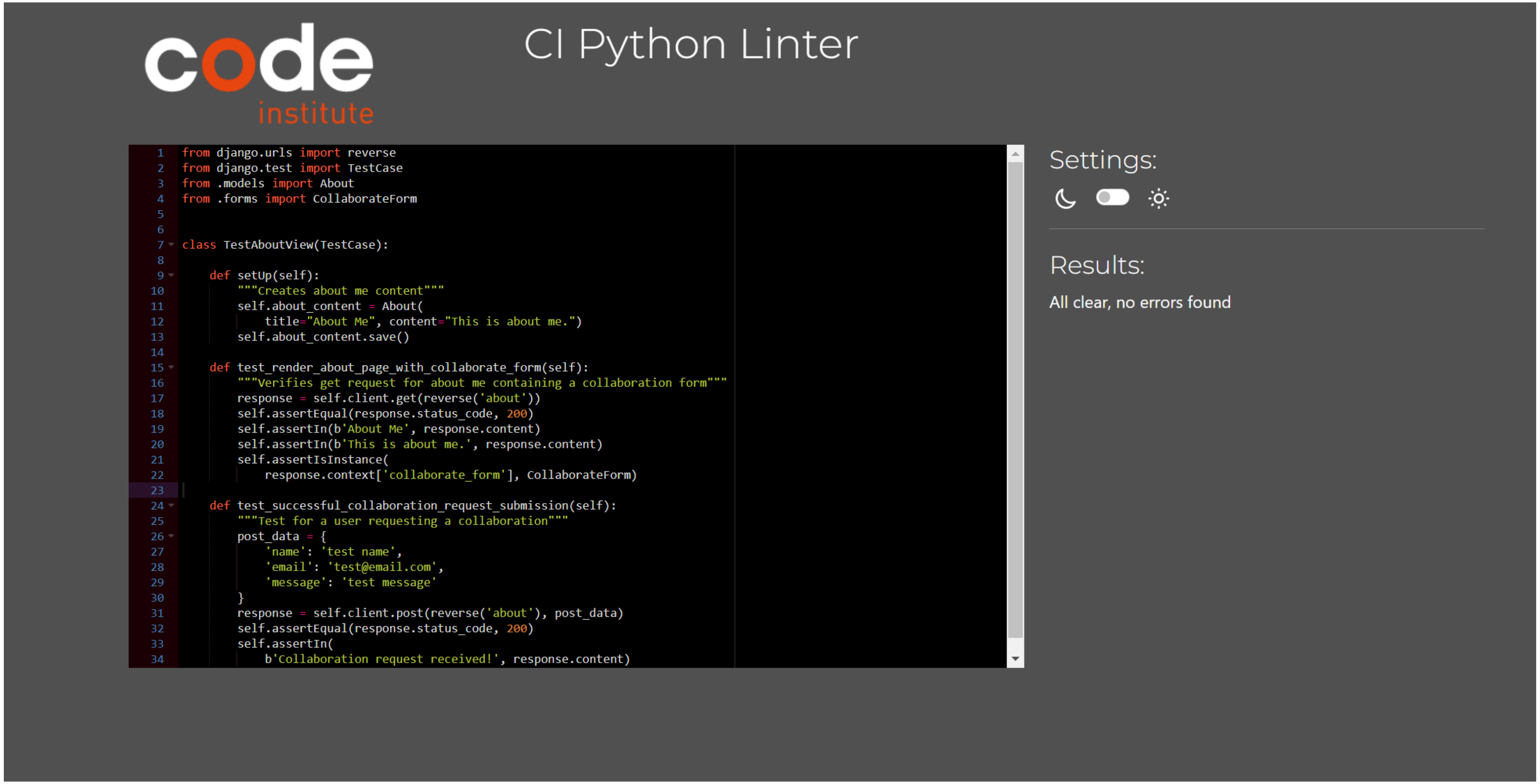


```python
from django.test import TestCase
from .forms import CollaborateForm


class TestCollaborateForm(TestCase):

    def test_form_is_valid(self):
        """Test for all fields"""
        form = CollaborateForm({
            'name': 'test',
            'email': 'test@test.com',
            'message': 'Hello!'
        })
        self.assertTrue(form.is_valid(), msg="Form is not valid")

    def test_name_is_required(self):
        """Test for the 'name' field"""
        form = CollaborateForm({
            'name': '',
            'email': 'test@test.com',
            'message': 'Hello!'
        })
        self.assertFalse(
            form.is_valid(),
            msg="Name was not provided; the form is valid"
        )

    def test_email_is_required(self):
        """Test for the 'email' field"""
        form = CollaborateForm({
            'name': 'Jemima',
            'email': '',
            'message': 'Hello!'
        })
```

Results:

All clear, no errors found

About App – test_views.py



CI Python Linter

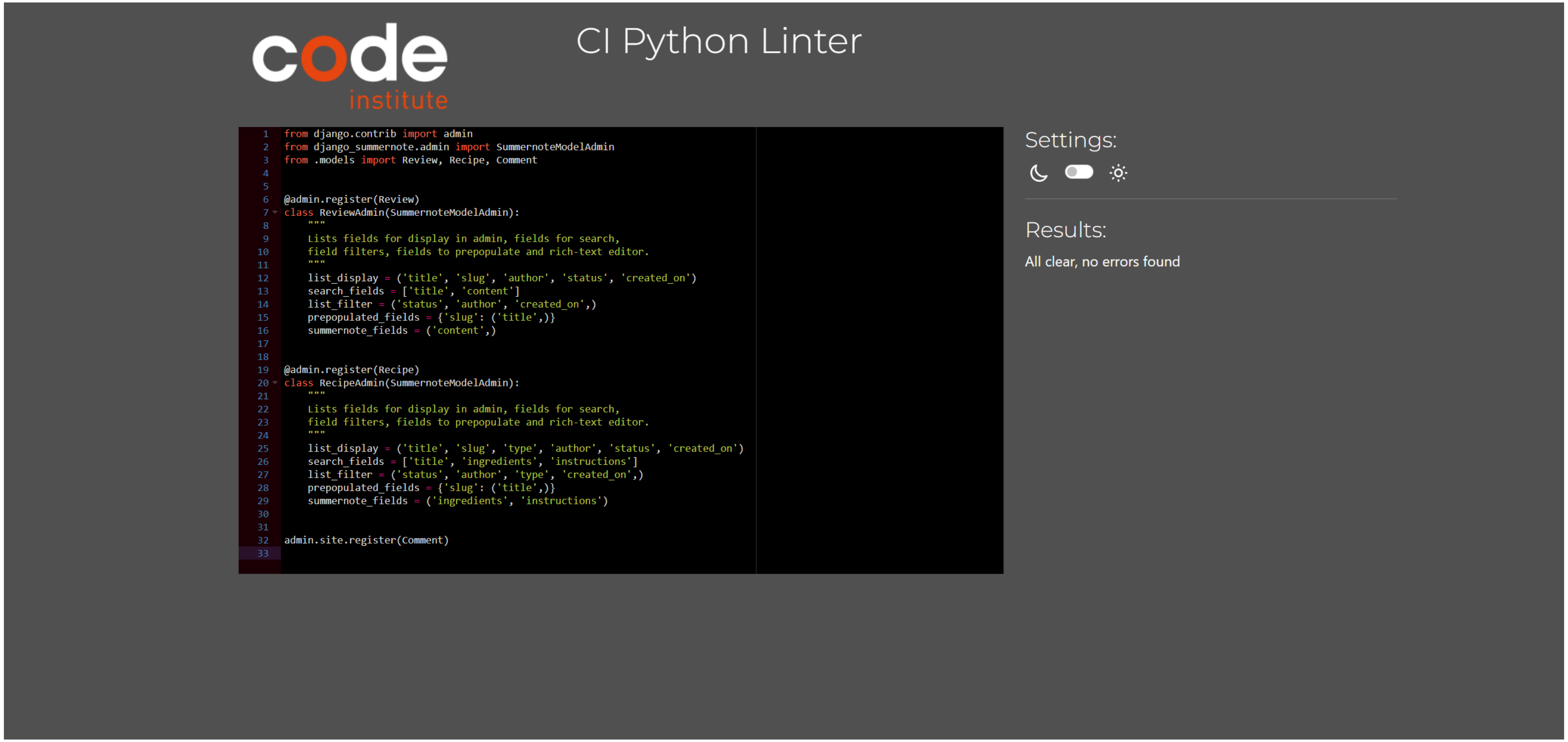
```
from django.urls import reverse
from django.test import TestCase
from .models import About
from .forms import CollaborateForm


class TestAboutView(TestCase):

    def setUp(self):
        """Creates about me content"""
        self.about_content = About(
            title="About Me", content="This is about me.")
        self.about_content.save()

    def test_render_about_page_with_collaborate_form(self):
        """Verifies get request for about me containing a collaboration form"""
        response = self.client.get(reverse('about'))
        self.assertEqual(response.status_code, 200)
        self.assertIn(b'About Me', response.content)
        self.assertIn(b'This is about me.', response.content)
        self.assertIsInstance(
            response.context['collaborate_form'], CollaborateForm)

    def test_successful_collaboration_request_submission(self):
        """Test for a user requesting a collaboration"""
        post_data = {
            'name': 'test name',
            'email': 'test@email.com',
            'message': 'test message'
        }
        response = self.client.post(reverse('about'), post_data)
        self.assertEqual(response.status_code, 200)
        self.assertIn(
            b'Collaboration request received!', response.content)
```

Settings:

Results:

All clear, no errors found

Blog App – admin.py



CI Python Linter

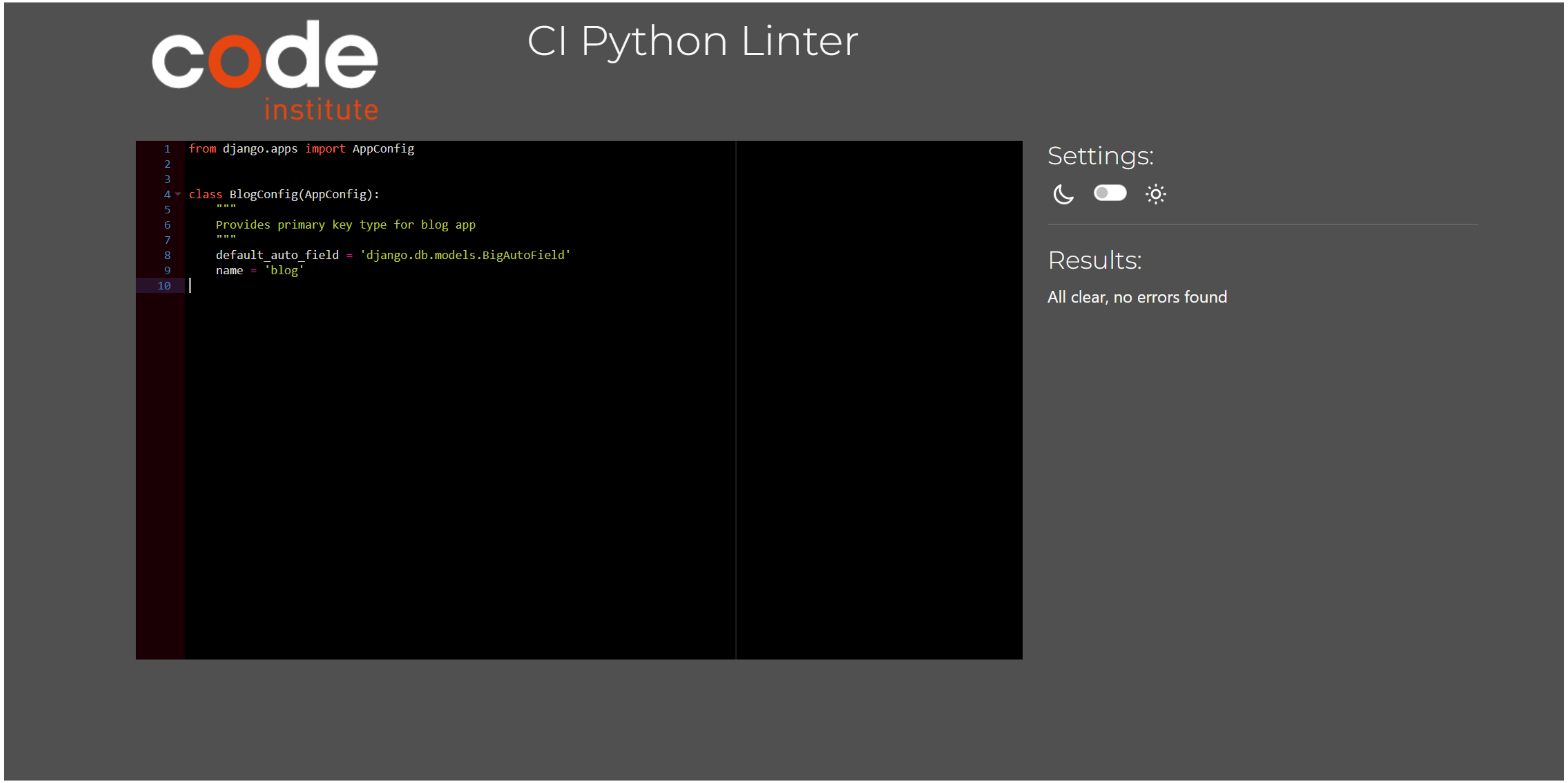
```python
from django.contrib import admin
from django_summernote.admin import SummernoteModelAdmin
from .models import Review, Recipe, Comment


@admin.register(Review)
class ReviewAdmin(SummernoteModelAdmin):
    """
    Lists fields for display in admin, fields for search,
    field filters, fields to prepopulate and rich-text editor.
    """
    list_display = ('title', 'slug', 'author', 'status', 'created_on')
    search_fields = ['title', 'content']
    list_filter = ('status', 'author', 'created_on',)
    prepopulated_fields = {'slug': ('title',)}
    summernote_fields = ('content',)


@admin.register(Recipe)
class RecipeAdmin(SummernoteModelAdmin):
    """
    Lists fields for display in admin, fields for search,
    field filters, fields to prepopulate and rich-text editor.
    """
    list_display = ('title', 'slug', 'type', 'author', 'status', 'created_on')
    search_fields = ['title', 'ingredients', 'instructions']
    list_filter = ('status', 'author', 'type', 'created_on',)
    prepopulated_fields = {'slug': ('title',)}
    summernote_fields = ('ingredients', 'instructions')


admin.site.register(Comment)
```

Settings:

Results:

All clear, no errors found

Blog App – apps.py

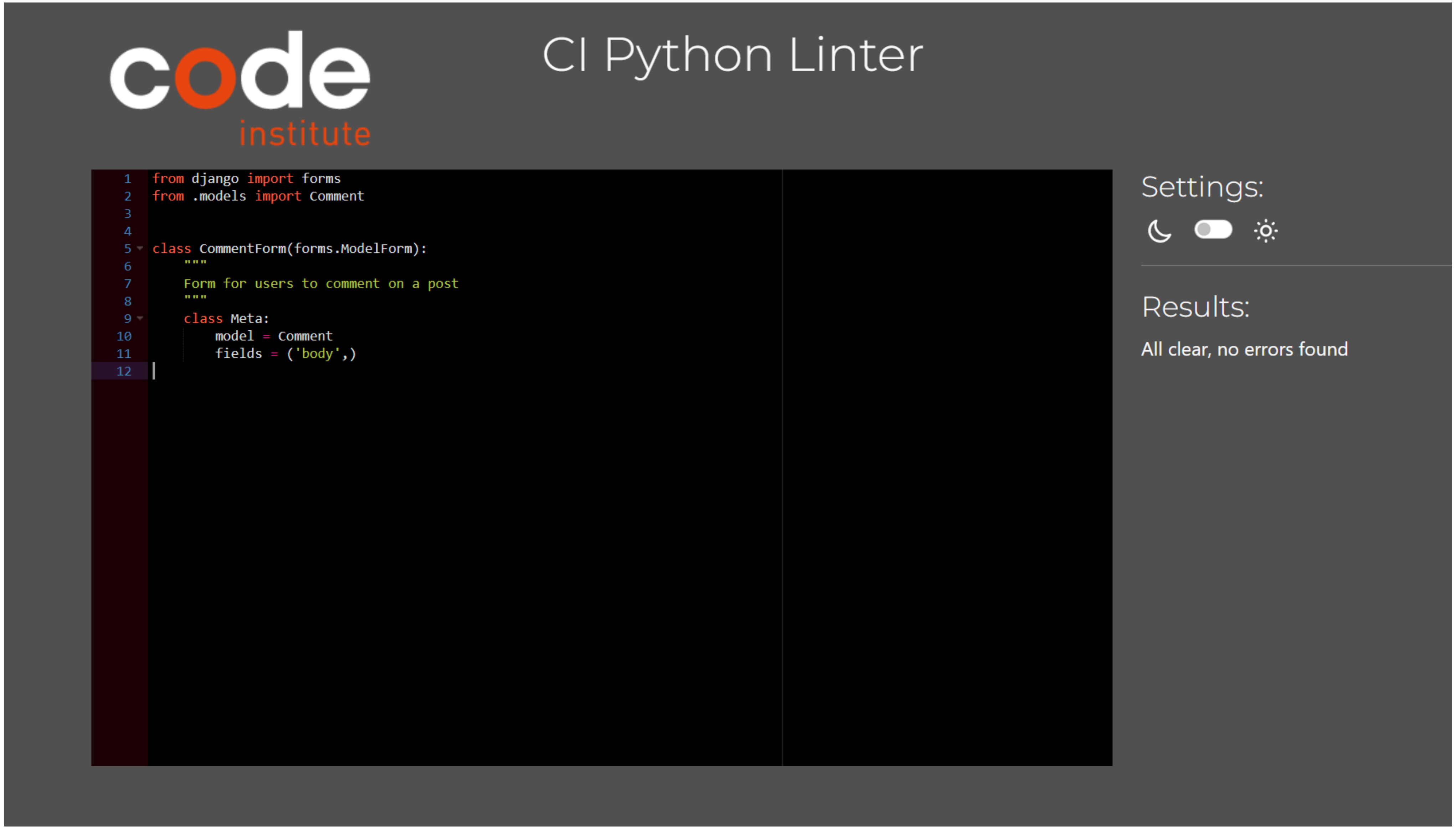


```python
from django.apps import AppConfig


class BlogConfig(AppConfig):
    """
    Provides primary key type for blog app
    """
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'blog'

```

Results:

All clear, no errors found

Blog App – forms.py

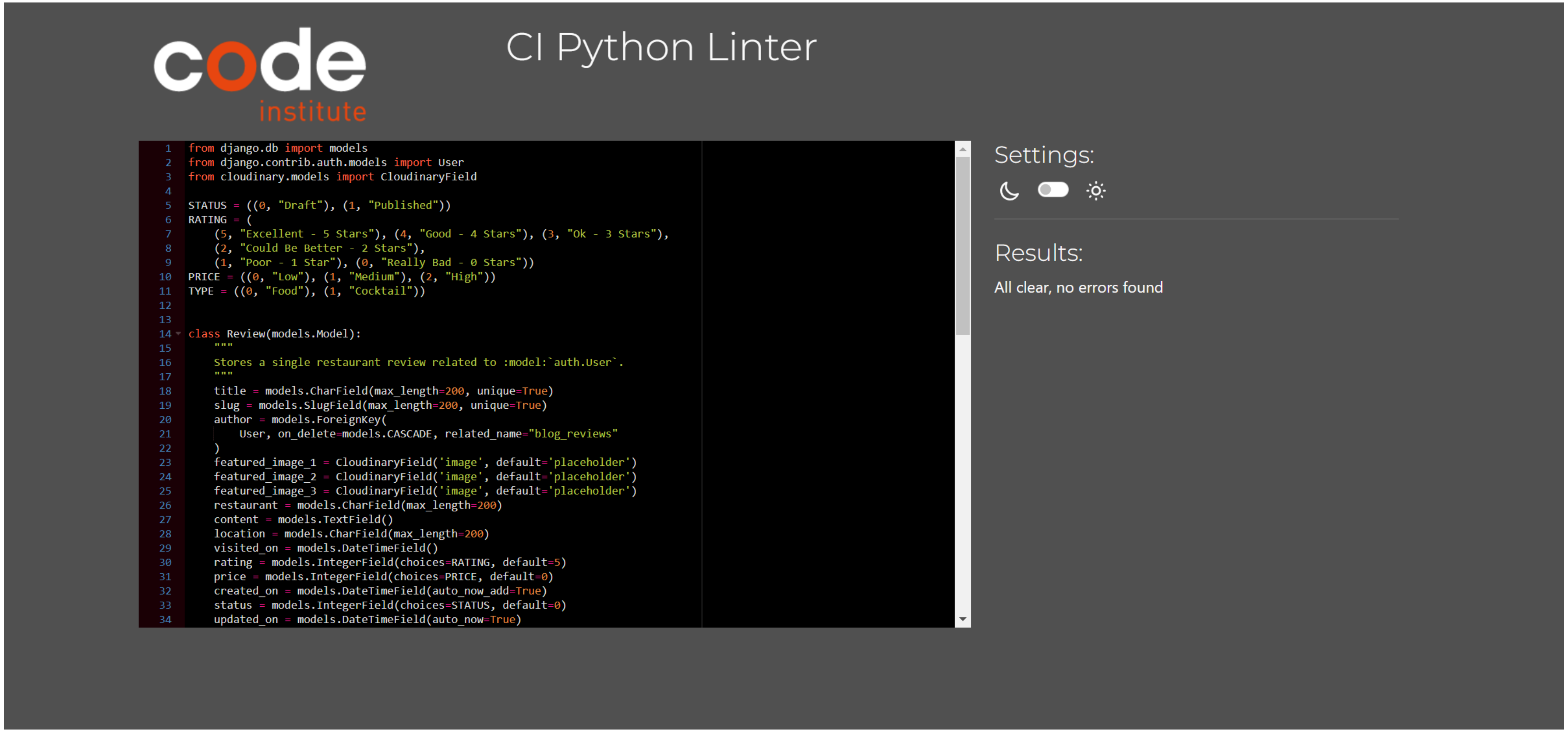


```python
from django import forms
from .models import Comment


class CommentForm(forms.ModelForm):
    """
    Form for users to comment on a post
    """
    class Meta:
        model = Comment
        fields = ('body',)
```

Results:

All clear, no errors found

Blog App – models.py



CI Python Linter

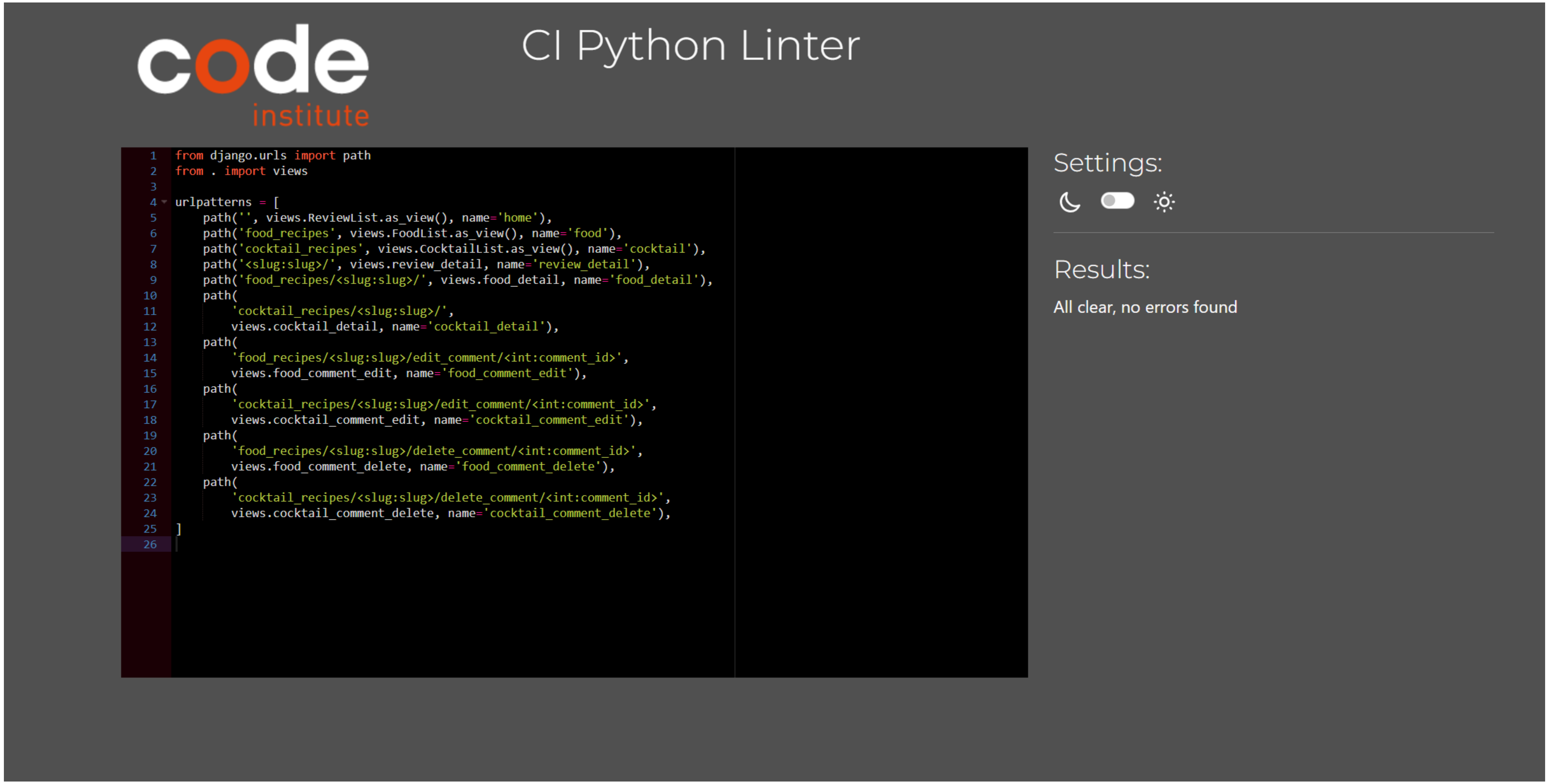
```python
from django.db import models
from django.contrib.auth.models import User
from cloudinary.models import CloudinaryField

STATUS = ((0, "Draft"), (1, "Published"))
RATING = (
    (5, "Excellent - 5 Stars"), (4, "Good - 4 Stars"), (3, "Ok - 3 Stars"),
    (2, "Could Be Better - 2 Stars"),
    (1, "Poor - 1 Star"), (0, "Really Bad - 0 Stars"))
PRICE = ((0, "Low"), (1, "Medium"), (2, "High"))
TYPE = ((0, "Food"), (1, "Cocktail"))


class Review(models.Model):
    """
    Stores a single restaurant review related to :model:`auth.User`.
    """
    title = models.CharField(max_length=200, unique=True)
    slug = models.SlugField(max_length=200, unique=True)
    author = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name="blog_reviews"
    )
    featured_image_1 = CloudinaryField('image', default='placeholder')
    featured_image_2 = CloudinaryField('image', default='placeholder')
    featured_image_3 = CloudinaryField('image', default='placeholder')
    restaurant = models.CharField(max_length=200)
    content = models.TextField()
    location = models.CharField(max_length=200)
    visited_on = models.DateTimeField()
    rating = models.IntegerField(choices=RATING, default=5)
    price = models.IntegerField(choices=PRICE, default=0)
    created_on = models.DateTimeField(auto_now_add=True)
    status = models.IntegerField(choices=STATUS, default=0)
    updated_on = models.DateTimeField(auto_now=True)
```

Settings:

Results:

All clear, no errors found

Blog App – urls.py



CI Python Linter

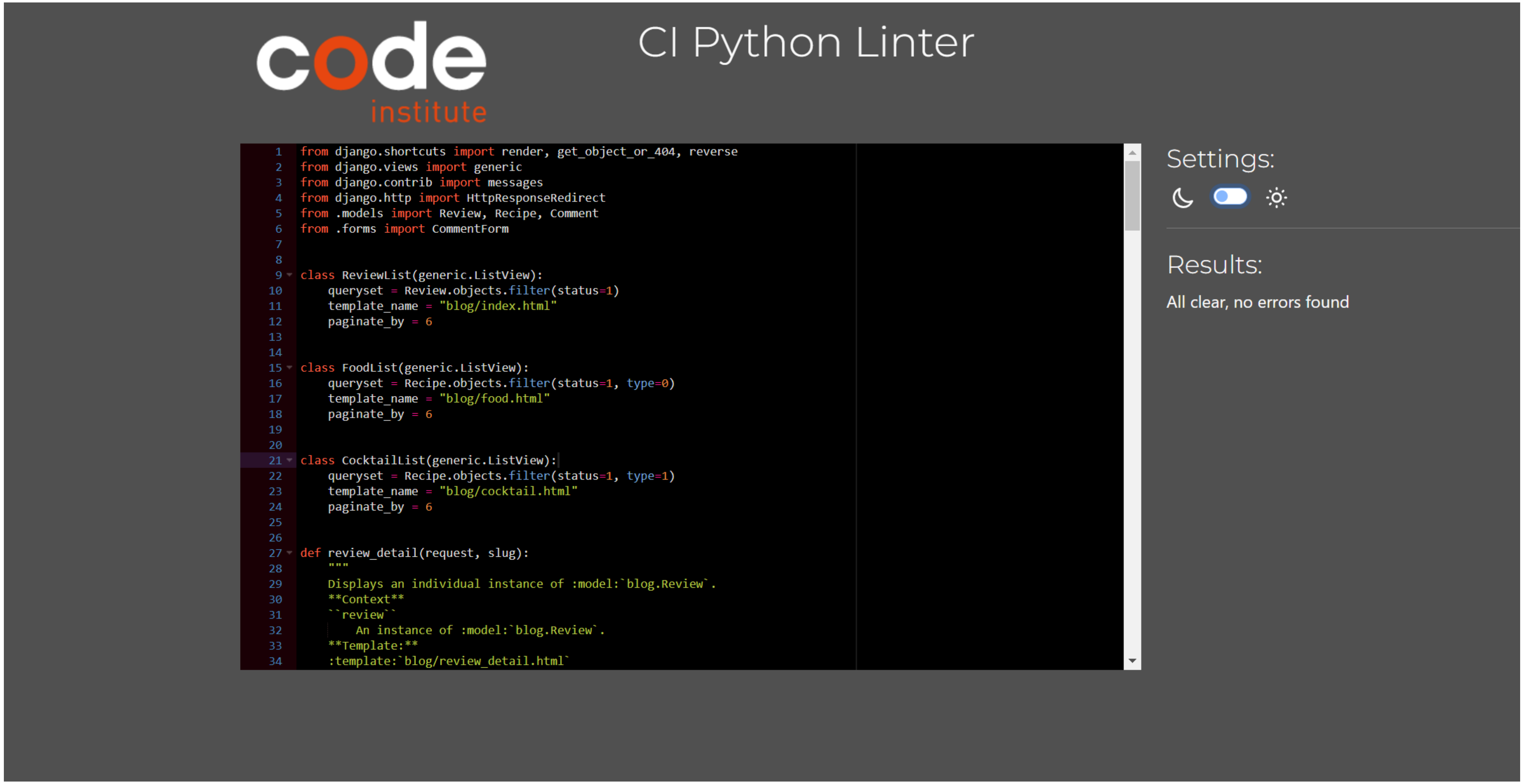
```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.ReviewList.as_view(), name='home'),
    path('food_recipes', views.FoodList.as_view(), name='food'),
    path('cocktail_recipes', views.CocktailList.as_view(), name='cocktail'),
    path('<slug:slug>/', views.review_detail, name='review_detail'),
    path('food_recipes/<slug:slug>/', views.food_detail, name='food_detail'),
    path(
        'cocktail_recipes/<slug:slug>/',
        views.cocktail_detail, name='cocktail_detail'),
    path(
        'food_recipes/<slug:slug>/edit_comment/<int:comment_id>',
        views.food_comment_edit, name='food_comment_edit'),
    path(
        'cocktail_recipes/<slug:slug>/edit_comment/<int:comment_id>',
        views.cocktail_comment_edit, name='cocktail_comment_edit'),
    path(
        'food_recipes/<slug:slug>/delete_comment/<int:comment_id>',
        views.food_comment_delete, name='food_comment_delete'),
    path(
        'cocktail_recipes/<slug:slug>/delete_comment/<int:comment_id>',
        views.cocktail_comment_delete, name='cocktail_comment_delete'),
]
```

Settings:

Results:

All clear, no errors found

Blog App – views.py



CI Python Linter

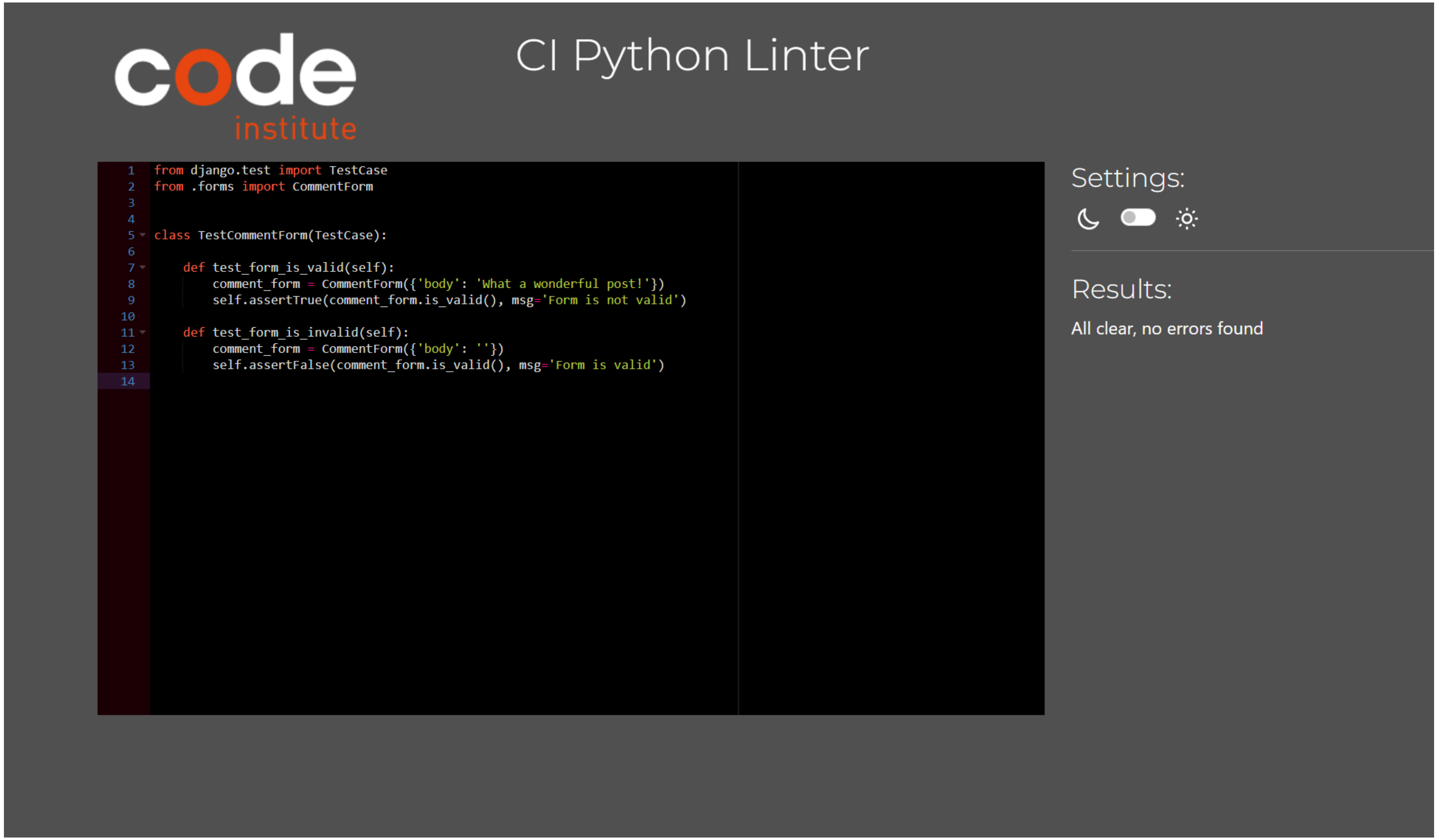
```
from django.shortcuts import render, get_object_or_404, reverse
from django.views import generic
from django.contrib import messages
from django.http import HttpResponseRedirect
from .models import Review, Recipe, Comment
from .forms import CommentForm


class ReviewList(generic.ListView):
    queryset = Review.objects.filter(status=1)
    template_name = "blog/index.html"
    paginate_by = 6


class FoodList(generic.ListView):
    queryset = Recipe.objects.filter(status=1, type=0)
    template_name = "blog/food.html"
    paginate_by = 6


class CocktailList(generic.ListView):
    queryset = Recipe.objects.filter(status=1, type=1)
    template_name = "blog/cocktail.html"
    paginate_by = 6


def review_detail(request, slug):
    """
    Displays an individual instance of :model:`blog.Review`.
    **Context**
    ``review``
        An instance of :model:`blog.Review`.
    **Template:**
    :template:`blog/review_detail.html`
```

Settings:

Results:

All clear, no errors found

Blog App – test_forms.py

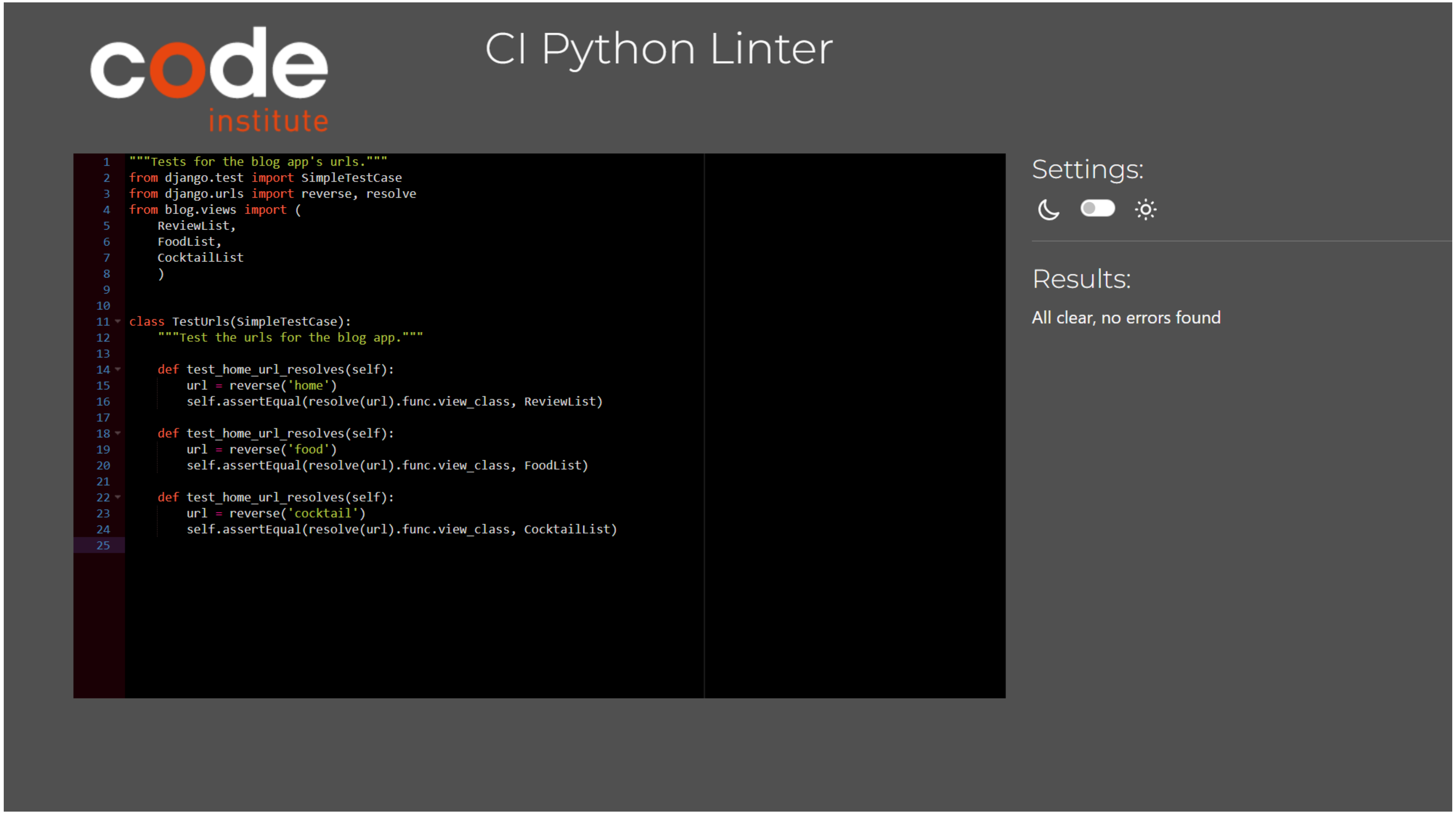


```python
from django.test import TestCase
from .forms import CommentForm


class TestCommentForm(TestCase):

    def test_form_is_valid(self):
        comment_form = CommentForm({'body': 'What a wonderful post!'})
        self.assertTrue(comment_form.is_valid(), msg='Form is not valid')

    def test_form_is_invalid(self):
        comment_form = CommentForm({'body': ''})
        self.assertFalse(comment_form.is_valid(), msg='Form is valid')
```

Results:

All clear, no errors found

Blog App – test_urls.py


CI Python Linter

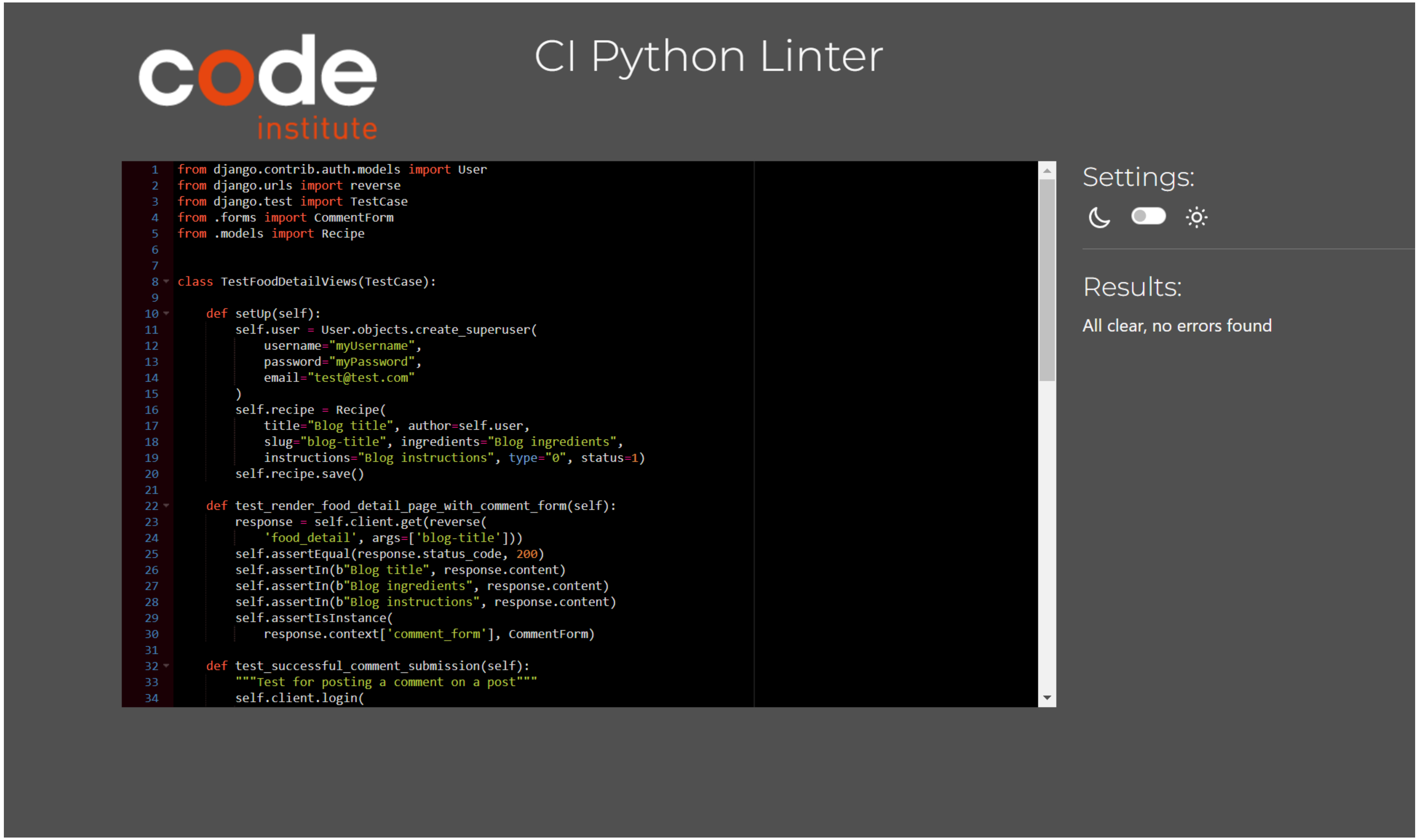
```python
"""Tests for the blog app's urls."""
from django.test import SimpleTestCase
from django.urls import reverse, resolve
from blog.views import (
    ReviewList,
    FoodList,
    CocktailList
    )


class TestUrls(SimpleTestCase):
    """Test the urls for the blog app."""

    def test_home_url_resolves(self):
        url = reverse('home')
        self.assertEqual(resolve(url).func.view_class, ReviewList)

    def test_home_url_resolves(self):
        url = reverse('food')
        self.assertEqual(resolve(url).func.view_class, FoodList)

    def test_home_url_resolves(self):
        url = reverse('cocktail')
        self.assertEqual(resolve(url).func.view_class, CocktailList)
```

Settings:

Results:

All clear, no errors found

Blog App – test_views.py



```python
from django.contrib.auth.models import User
from django.urls import reverse
from django.test import TestCase
from .forms import CommentForm
from .models import Recipe


class TestFoodDetailViews(TestCase):

    def setUp(self):
        self.user = User.objects.create_superuser(
            username="myUsername",
            password="myPassword",
            email="test@test.com"
        )
        self.recipe = Recipe(
            title="Blog title", author=self.user,
            slug="blog-title", ingredients="Blog ingredients",
            instructions="Blog instructions", type="0", status=1)
        self.recipe.save()

    def test_render_food_detail_page_with_comment_form(self):
        response = self.client.get(reverse(
            'food_detail', args=['blog-title']))
        self.assertEqual(response.status_code, 200)
        self.assertIn(b"Blog title", response.content)
        self.assertIn(b"Blog ingredients", response.content)
        self.assertIn(b"Blog instructions", response.content)
        self.assertIsInstance(
            response.context['comment_form'], CommentForm)

    def test_successful_comment_submission(self):
        """Test for posting a comment on a post"""
        self.client.login(
```

Settings:

Results:

All clear, no errors found