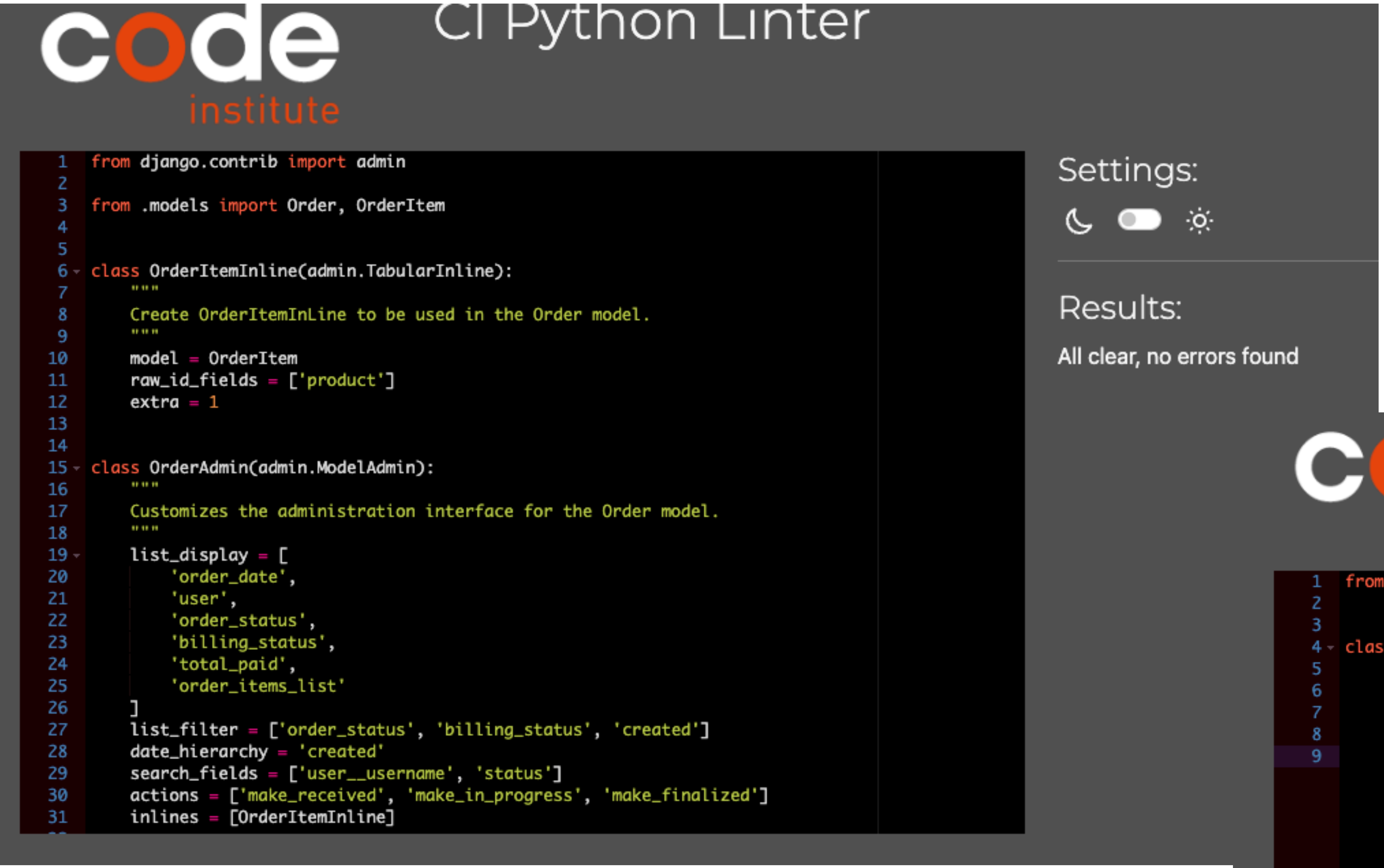
Orders

admin.py


CI Python Linter

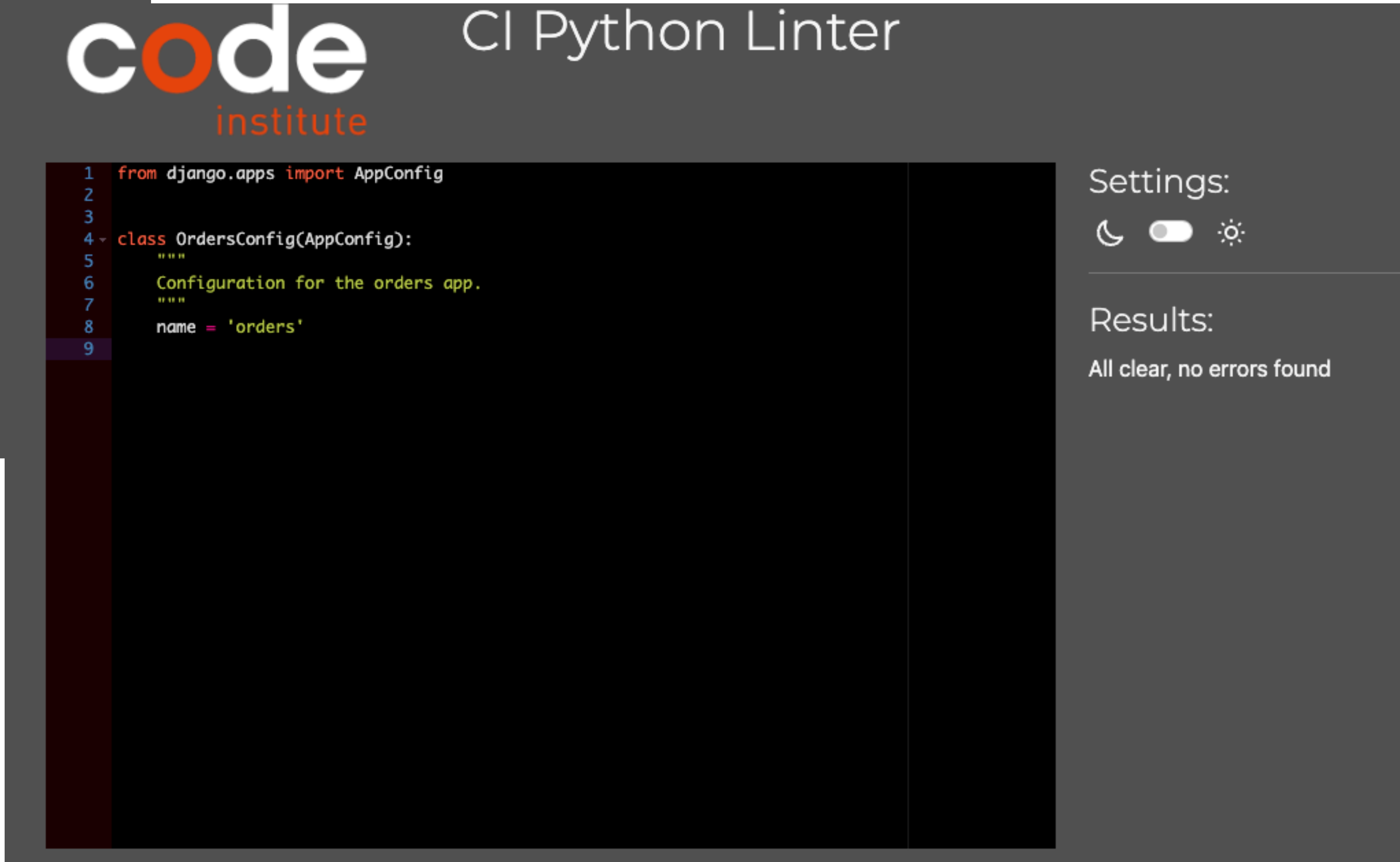
```
from django.contrib import admin

from .models import Order, OrderItem


class OrderItemInline(admin.TabularInline):
    """
    Create OrderItemInLine to be used in the Order model.
    """
    model = OrderItem
    raw_id_fields = ['product']
    extra = 1


class OrderAdmin(admin.ModelAdmin):
    """
    Customizes the administration interface for the Order model.
    """
    list_display = [
        'order_date',
        'user',
        'order_status',
        'billing_status',
        'total_paid',
        'order_items_list'
    ]
    list_filter = ['order_status', 'billing_status', 'created']
    date_hierarchy = 'created'
    search_fields = ['user__username', 'status']
    actions = ['make_received', 'make_in_progress', 'make_finalized']
    inlines = [OrderItemInline]
```

Settings:

Results:

All clear, no errors found

apps.py


CI Python Linter

```
from django.apps import AppConfig


class OrdersConfig(AppConfig):
    """
    Configuration for the orders app.
    """
    name = 'orders'
```

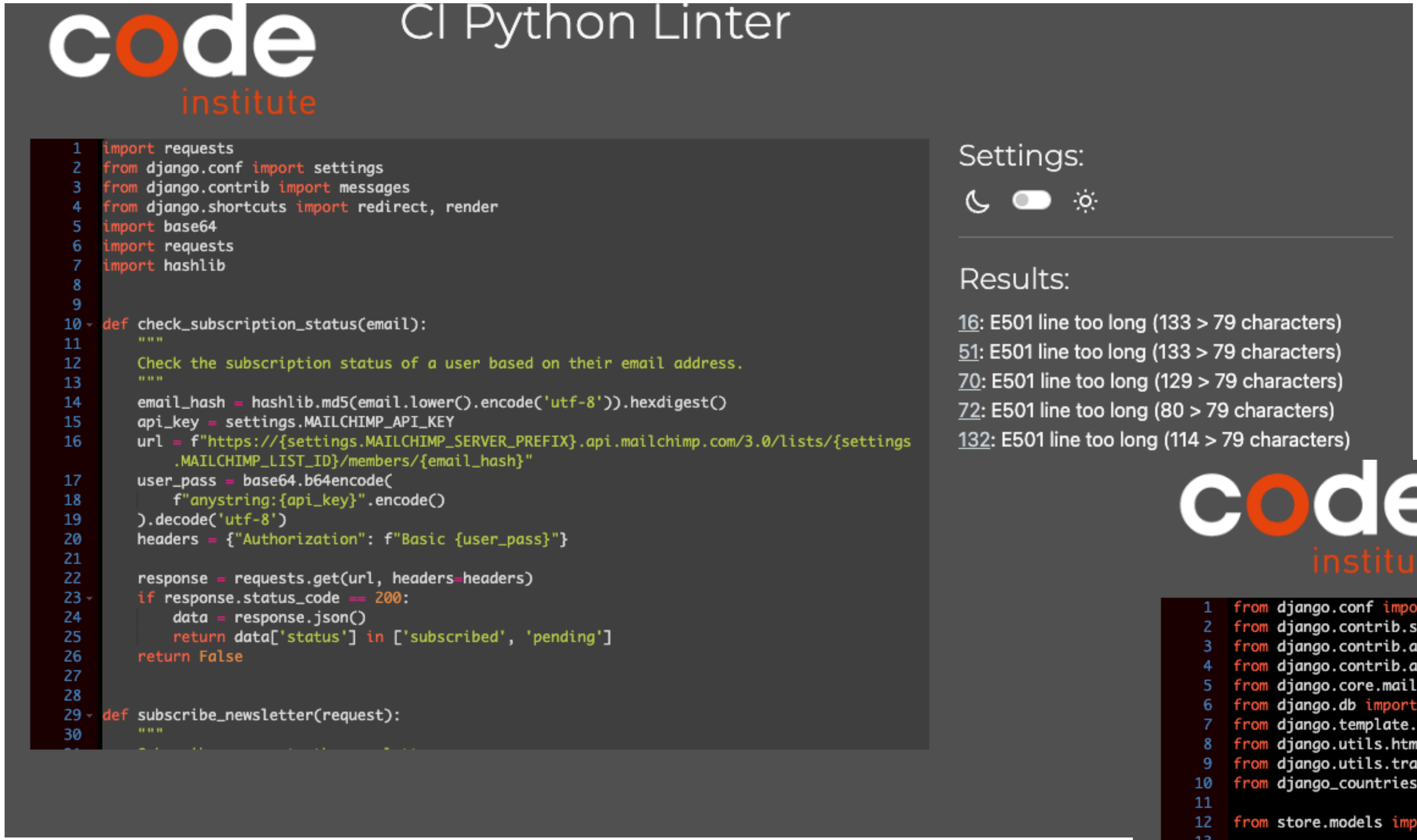
Settings:

Results:

All clear, no errors found

Orders

views.py



CI Python Linter

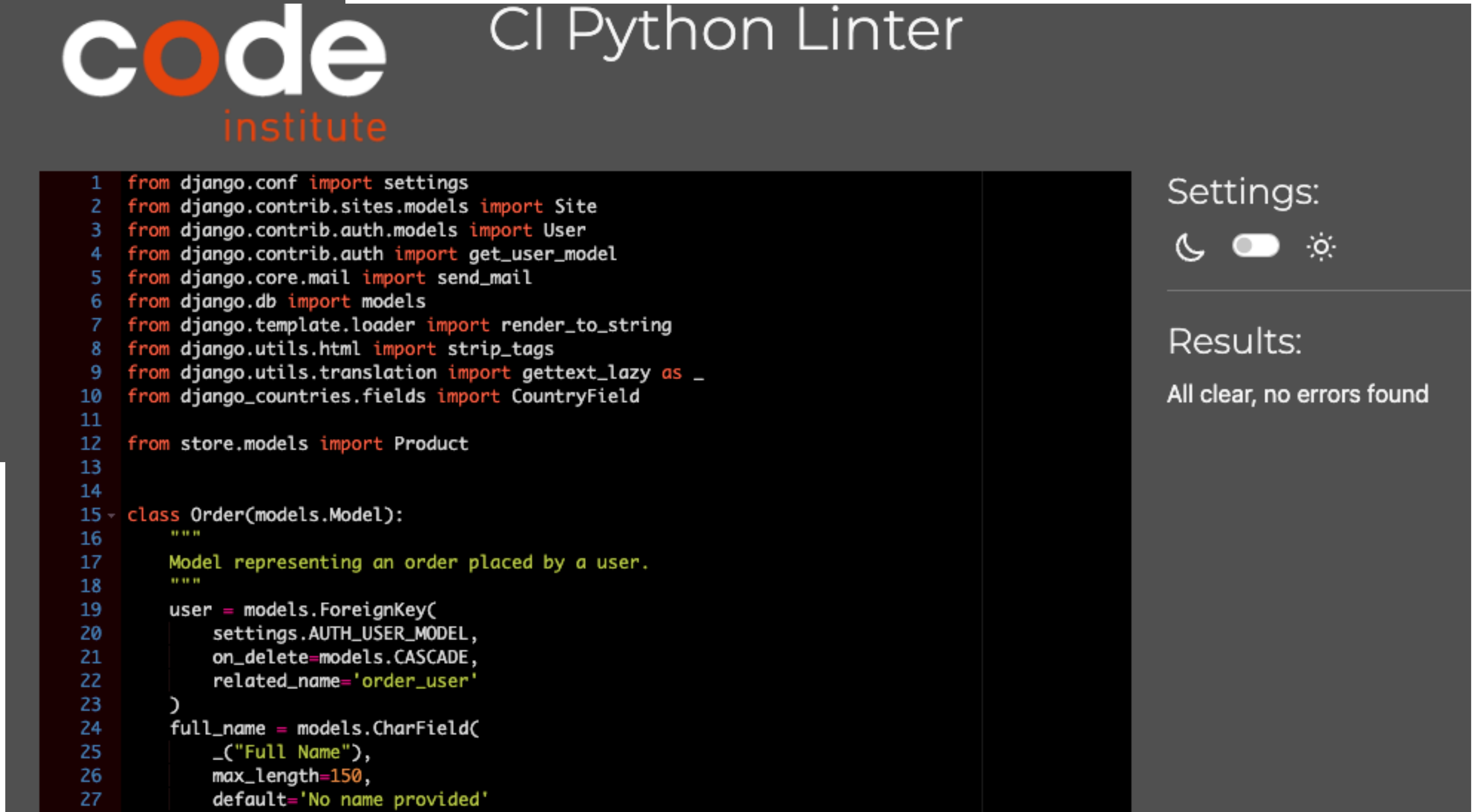
```
 1  import requests
 2  from django.conf import settings
 3  from django.contrib import messages
 4  from django.shortcuts import redirect, render
 5  import base64
 6  import requests
 7  import hashlib
 8
 9
10  def check_subscription_status(email):
11      """
12      Check the subscription status of a user based on their email address.
13      """
14      email_hash = hashlib.md5(email.lower().encode('utf-8')).hexdigest()
15      api_key = settings.MAILCHIMP_API_KEY
16      url = f"https://{settings.MAILCHIMP_SERVER_PREFIX}.api.mailchimp.com/3.0/lists/{settings
            .MAILCHIMP_LIST_ID}/members/{email_hash}"
17      user_pass = base64.b64encode(
18          f"anystring:{api_key}".encode()
19      ).decode('utf-8')
20      headers = {"Authorization": f"Basic {user_pass}"}
21
22      response = requests.get(url, headers=headers)
23      if response.status_code == 200:
24          data = response.json()
25          return data['status'] in ['subscribed', 'pending']
26      return False
27
28
29  def subscribe_newsletter(request):
30      """
```

Settings:

Results:

16: E501 line too long (133 > 79 characters)
51: E501 line too long (133 > 79 characters)
70: E501 line too long (129 > 79 characters)
72: E501 line too long (80 > 79 characters)
132: E501 line too long (114 > 79 characters)

models.py



CI Python Linter

```
from django.conf import settings
from django.contrib.sites.models import Site
from django.contrib.auth.models import User
from django.contrib.auth import get_user_model
from django.core.mail import send_mail
from django.db import models
from django.template.loader import render_to_string
from django.utils.html import strip_tags
from django.utils.translation import gettext_lazy as _
from django_countries.fields import CountryField

from store.models import Product


class Order(models.Model):
    """
    Model representing an order placed by a user.
    """
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name='order_user'
    )
    full_name = models.CharField(
        _("Full Name"),
        max_length=150,
        default='No name provided'
```

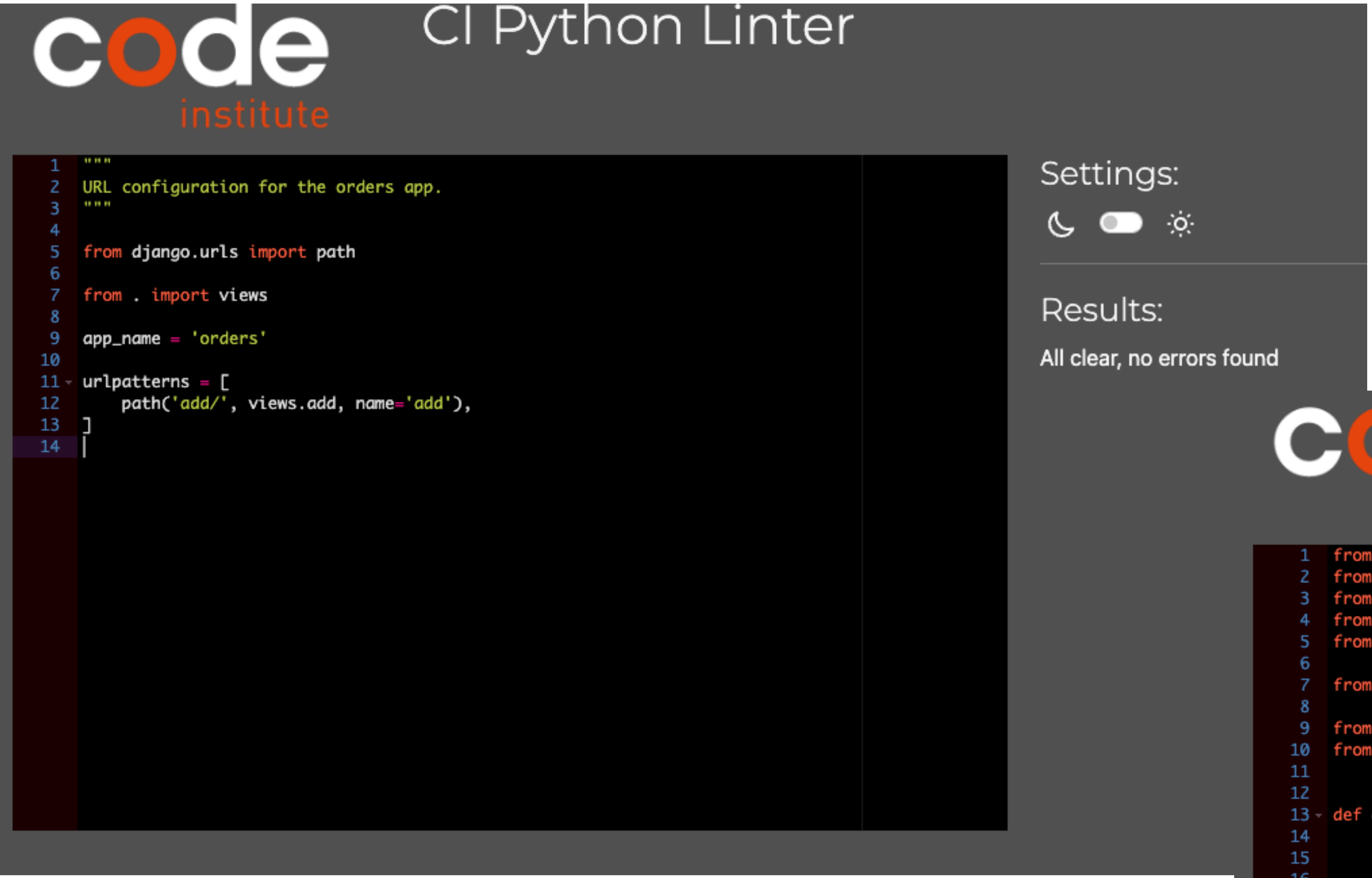
Settings:

Results:

All clear, no errors found

Orders

Urls.py



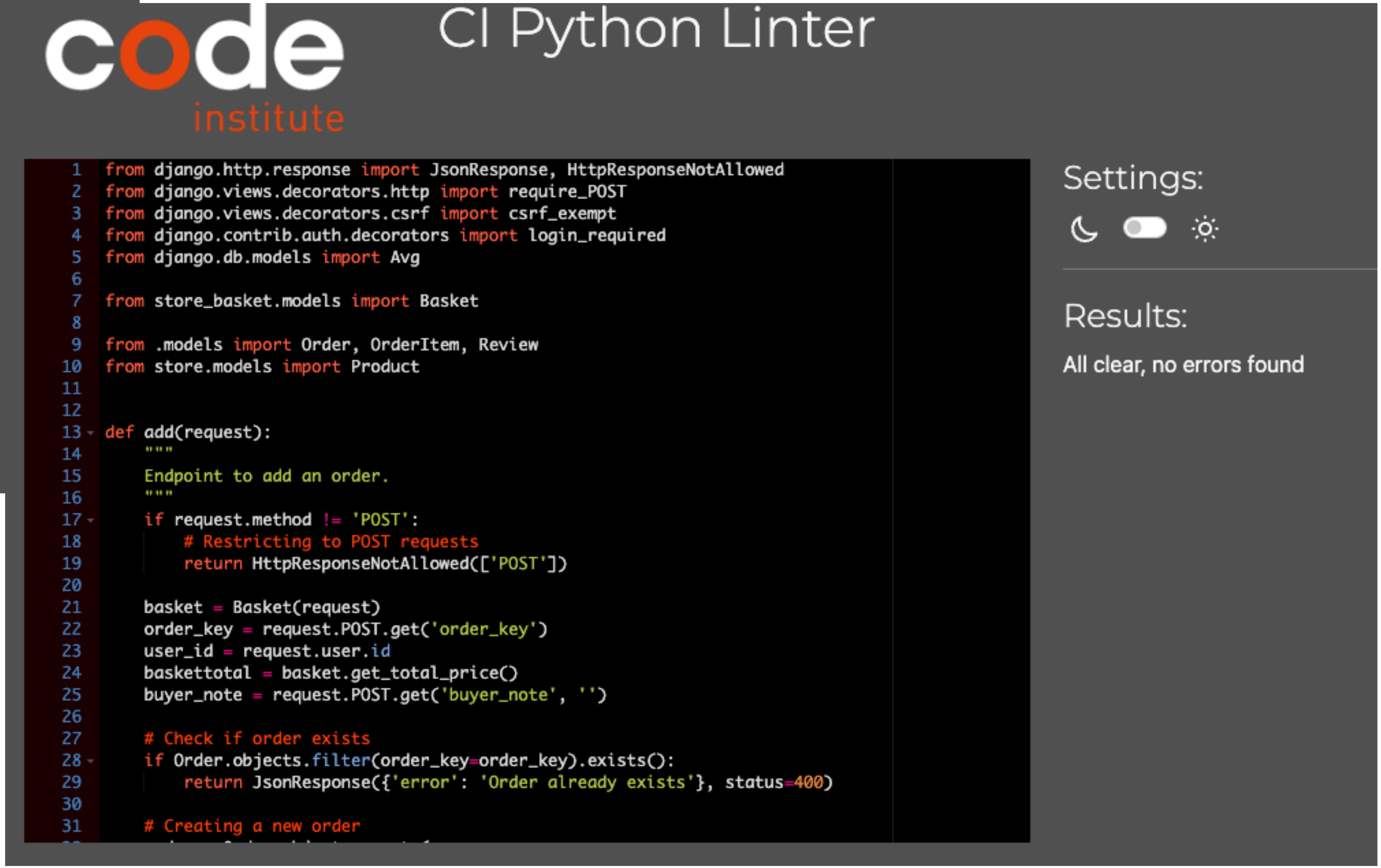
```python
"""
URL configuration for the orders app.
"""

from django.urls import path

from . import views

app_name = 'orders'

urlpatterns = [
    path('add/', views.add, name='add'),
]
```

Settings:

Results:

All clear, no errors found

views.py



```python
from django.http.response import JsonResponse, HttpResponseNotAllowed
from django.views.decorators.http import require_POST
from django.views.decorators.csrf import csrf_exempt
from django.contrib.auth.decorators import login_required
from django.db.models import Avg

from store_basket.models import Basket

from .models import Order, OrderItem, Review
from store.models import Product


def add(request):
    """
    Endpoint to add an order.
    """
    if request.method != 'POST':
        # Restricting to POST requests
        return HttpResponseNotAllowed(['POST'])

    basket = Basket(request)
    order_key = request.POST.get('order_key')
    user_id = request.user.id
    baskettotal = basket.get_total_price()
    buyer_note = request.POST.get('buyer_note', '')

    # Check if order exists
    if Order.objects.filter(order_key=order_key).exists():
        return JsonResponse({'error': 'Order already exists'}, status=400)

    # Creating a new order
```

Settings:

Results:

All clear, no errors found