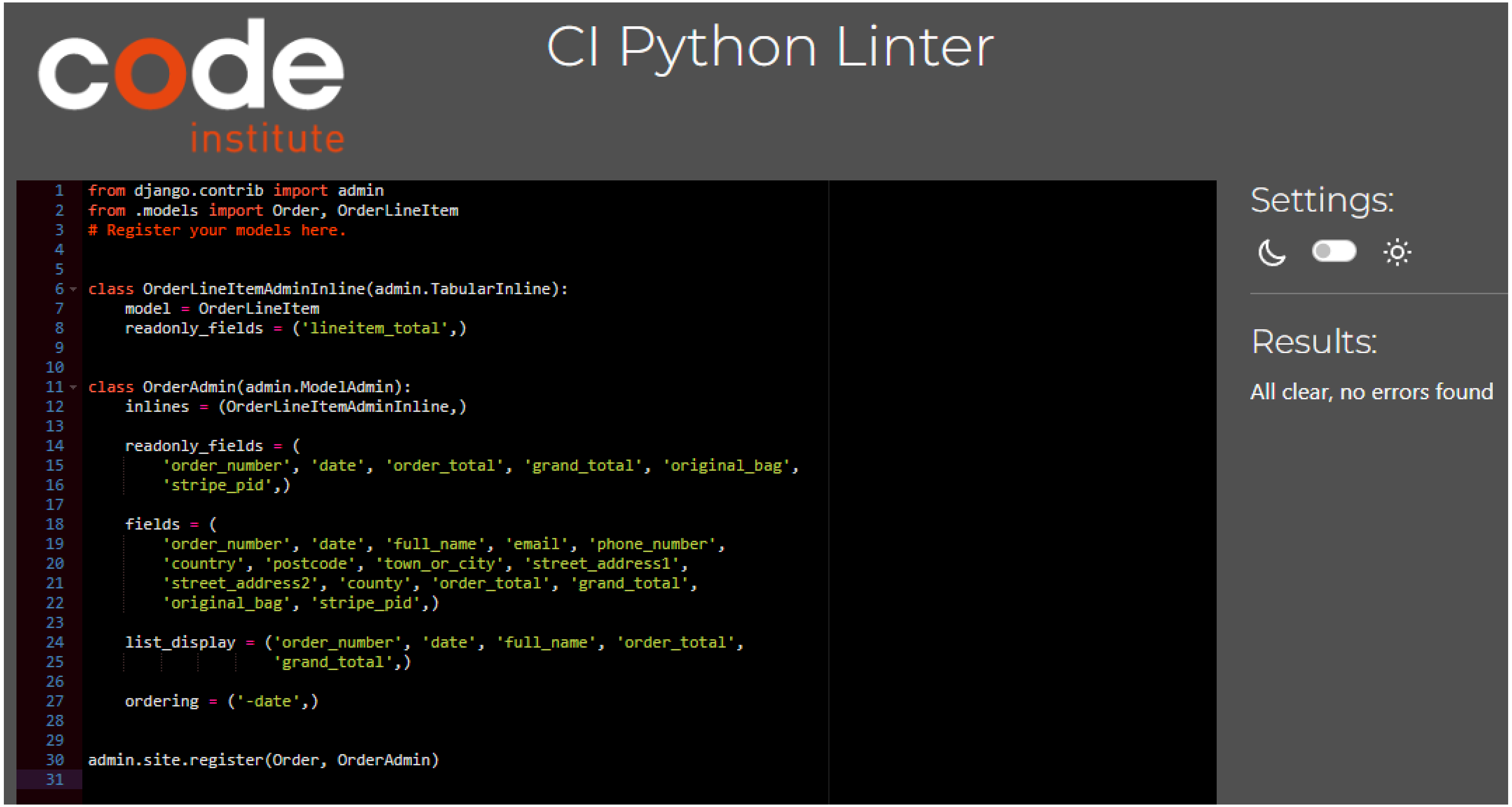
PEP8 of checkout app



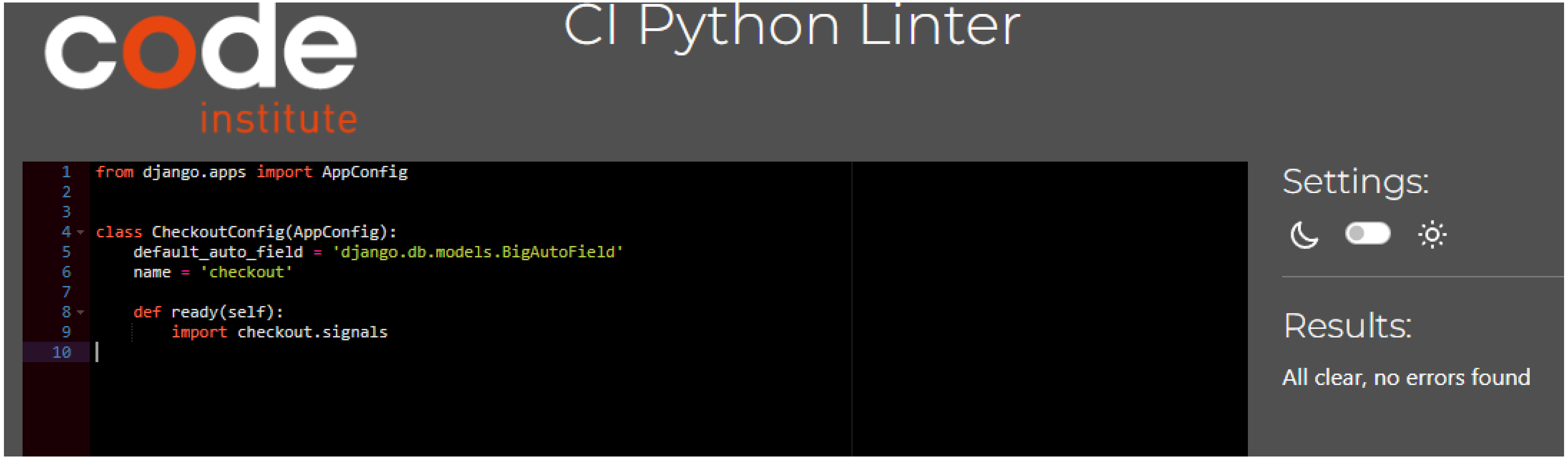
```
from django.contrib import admin
from .models import Order, OrderLineItem
# Register your models here.


class OrderLineItemAdminInline(admin.TabularInline):
    model = OrderLineItem
    readonly_fields = ('lineitem_total',)


class OrderAdmin(admin.ModelAdmin):
    inlines = (OrderLineItemAdminInline,)

    readonly_fields = (
        'order_number', 'date', 'order_total', 'grand_total', 'original_bag',
        'stripe_pid',)

    fields = (
        'order_number', 'date', 'full_name', 'email', 'phone_number',
        'country', 'postcode', 'town_or_city', 'street_address1',
        'street_address2', 'county', 'order_total', 'grand_total',
        'original_bag', 'stripe_pid',)

    list_display = ('order_number', 'date', 'full_name', 'order_total',
                    'grand_total',)

    ordering = ('-date',)


admin.site.register(Order, OrderAdmin)
```

Results:

All clear, no errors found

admin.py

PEP8 of checkout app



```
from django.apps import AppConfig


class CheckoutConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'checkout'

    def ready(self):
        import checkout.signals

```

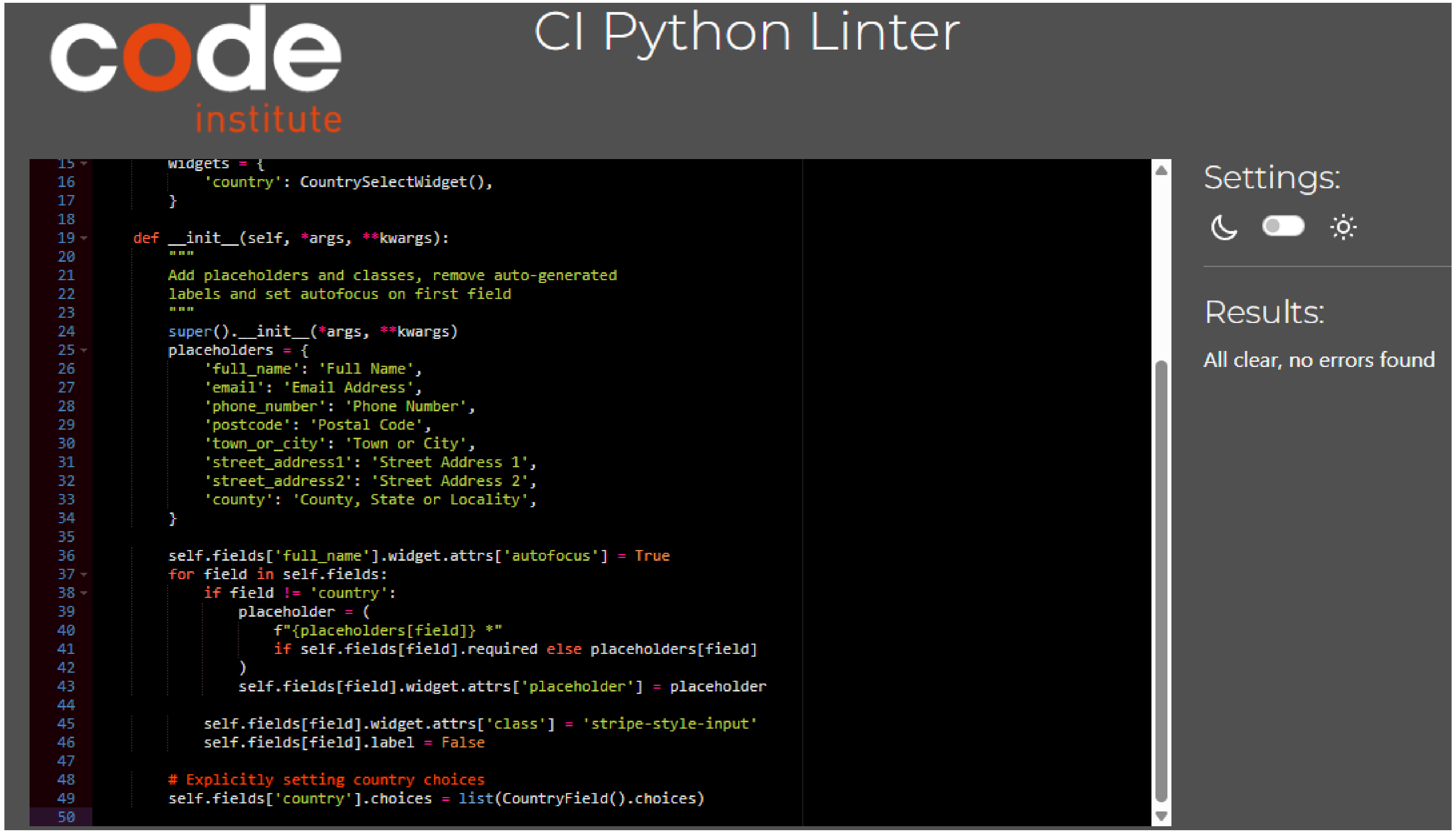
Settings:

Results:

All clear, no errors found

apps.py

PEP8 of checkout app



CI Python Linter

```
        widgets = {
            'country': CountrySelectWidget(),
        }

    def __init__(self, *args, **kwargs):
        """
        Add placeholders and classes, remove auto-generated
        labels and set autofocus on first field
        """
        super().__init__(*args, **kwargs)
        placeholders = {
            'full_name': 'Full Name',
            'email': 'Email Address',
            'phone_number': 'Phone Number',
            'postcode': 'Postal Code',
            'town_or_city': 'Town or City',
            'street_address1': 'Street Address 1',
            'street_address2': 'Street Address 2',
            'county': 'County, State or Locality',
        }

        self.fields['full_name'].widget.attrs['autofocus'] = True
        for field in self.fields:
            if field != 'country':
                placeholder = (
                    f"{placeholders[field]} *"
                    if self.fields[field].required else placeholders[field]
                )
                self.fields[field].widget.attrs['placeholder'] = placeholder

            self.fields[field].widget.attrs['class'] = 'stripe-style-input'
            self.fields[field].label = False

        # Explicitly setting country choices
        self.fields['country'].choices = list(CountryField().choices)
```

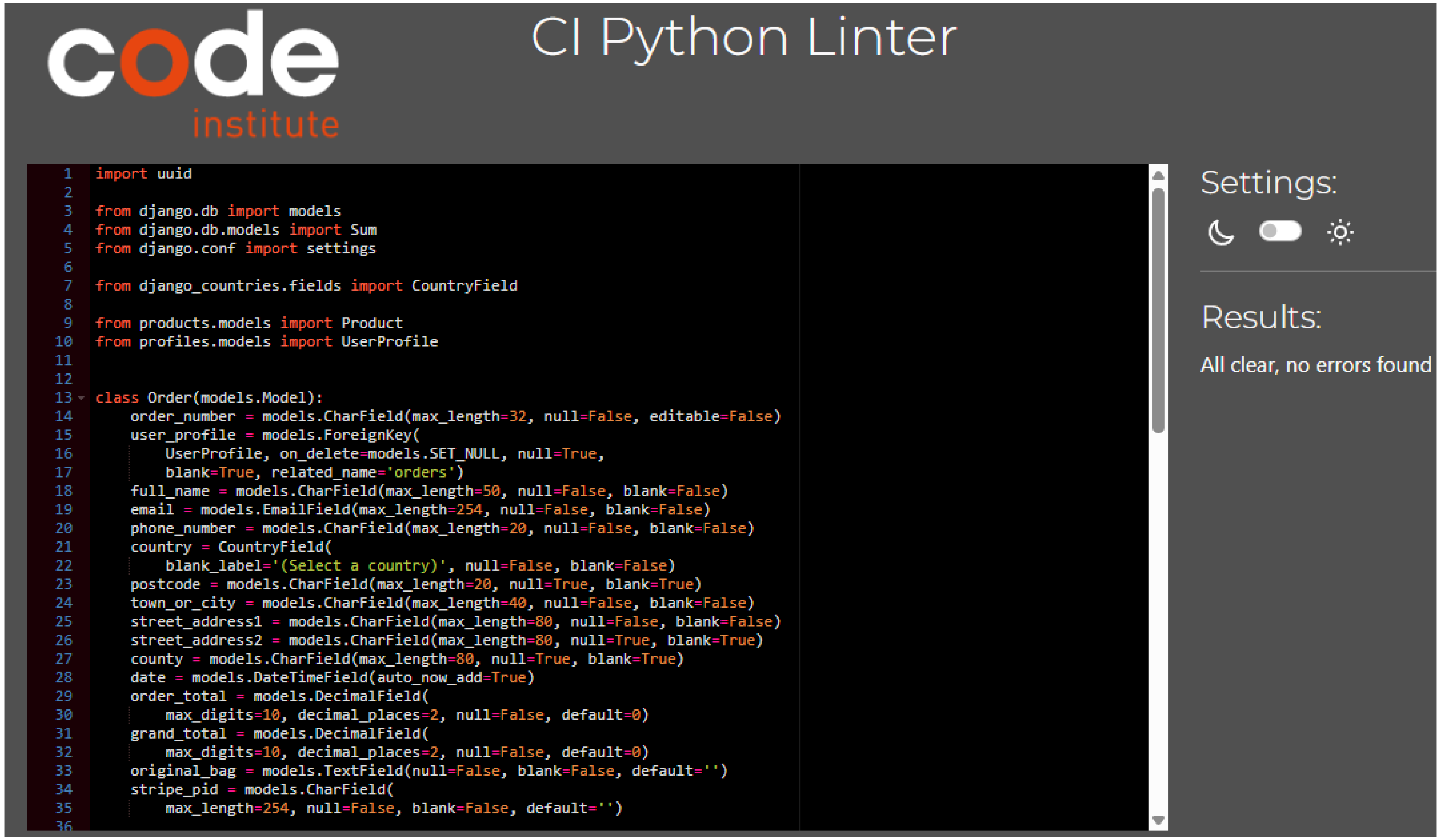
Settings:

Results:

All clear, no errors found

forms.py

PEP8 of checkout app



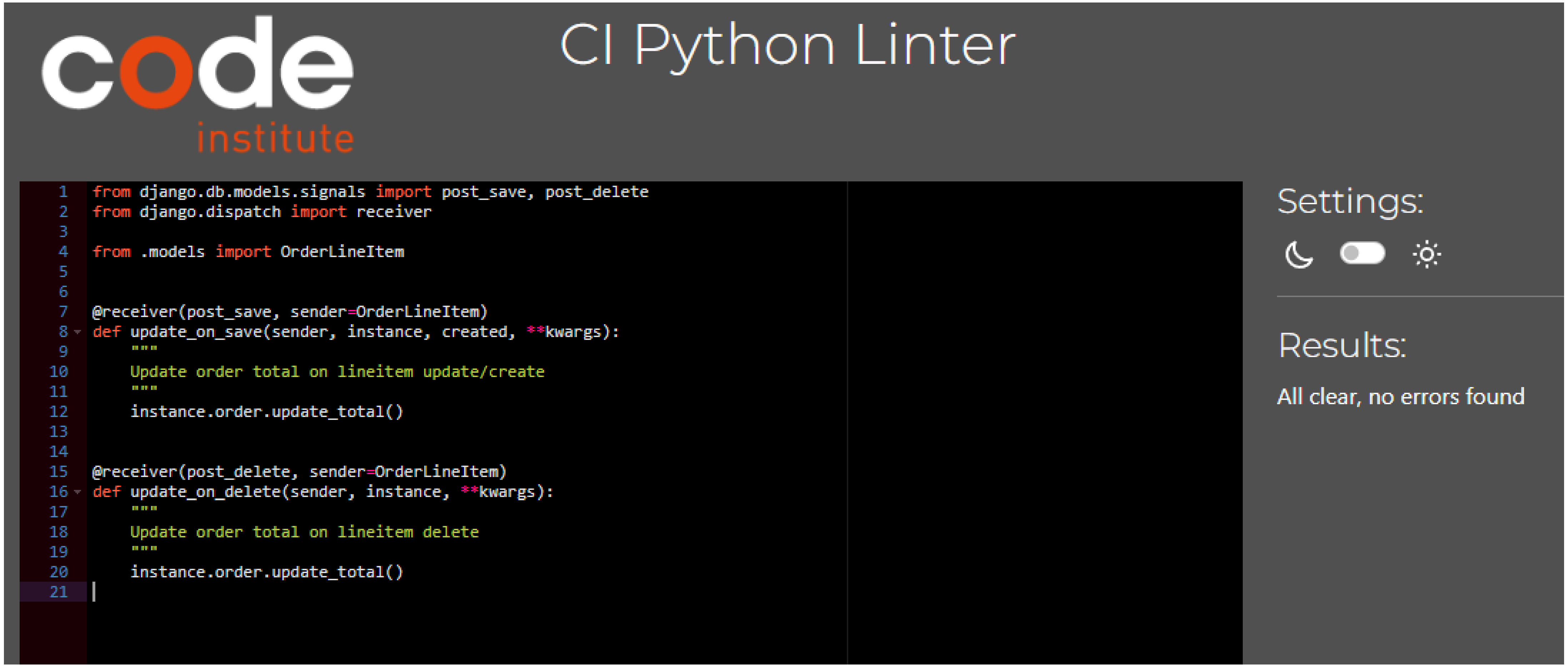
```python
import uuid

from django.db import models
from django.db.models import Sum
from django.conf import settings

from django_countries.fields import CountryField

from products.models import Product
from profiles.models import UserProfile


class Order(models.Model):
    order_number = models.CharField(max_length=32, null=False, editable=False)
    user_profile = models.ForeignKey(
        UserProfile, on_delete=models.SET_NULL, null=True,
        blank=True, related_name='orders')
    full_name = models.CharField(max_length=50, null=False, blank=False)
    email = models.EmailField(max_length=254, null=False, blank=False)
    phone_number = models.CharField(max_length=20, null=False, blank=False)
    country = CountryField(
        blank_label='(Select a country)', null=False, blank=False)
    postcode = models.CharField(max_length=20, null=True, blank=True)
    town_or_city = models.CharField(max_length=40, null=False, blank=False)
    street_address1 = models.CharField(max_length=80, null=False, blank=False)
    street_address2 = models.CharField(max_length=80, null=True, blank=True)
    county = models.CharField(max_length=80, null=True, blank=True)
    date = models.DateTimeField(auto_now_add=True)
    order_total = models.DecimalField(
        max_digits=10, decimal_places=2, null=False, default=0)
    grand_total = models.DecimalField(
        max_digits=10, decimal_places=2, null=False, default=0)
    original_bag = models.TextField(null=False, blank=False, default='')
    stripe_pid = models.CharField(
        max_length=254, null=False, blank=False, default='')
```

Results:

All clear, no errors found

models.py

PEP8 of checkout app

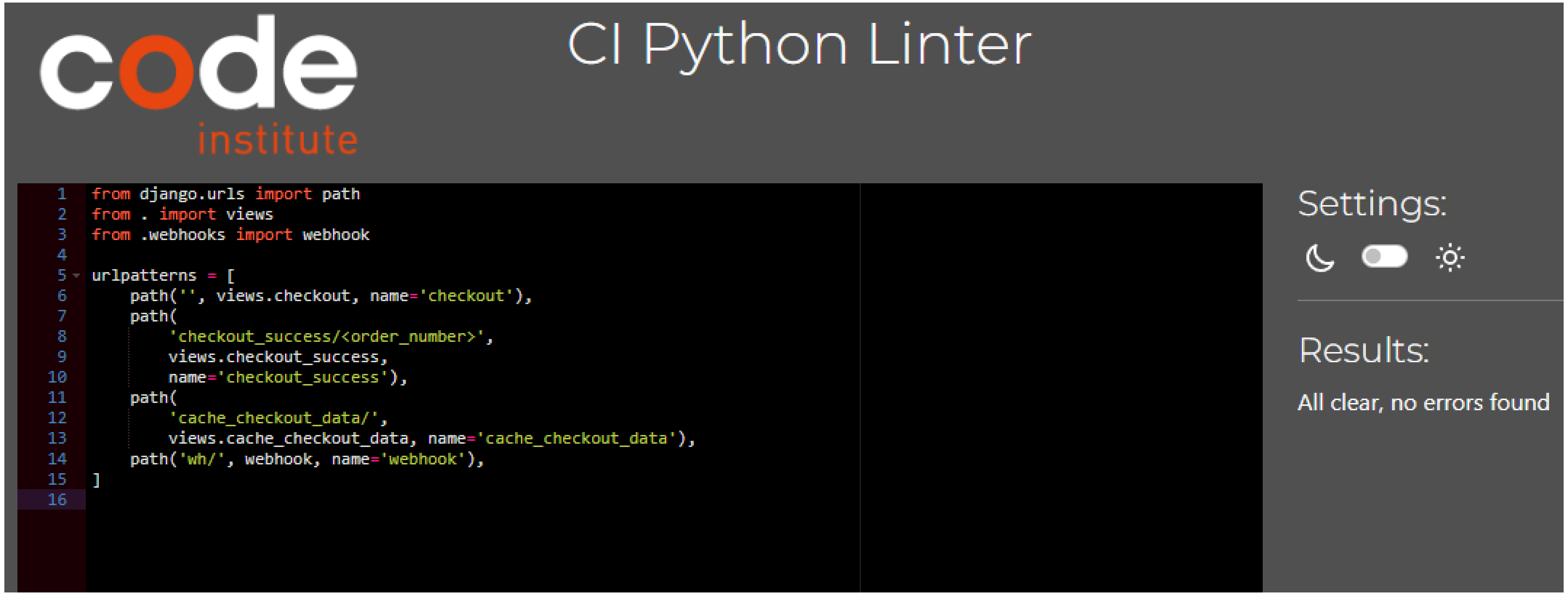


```python
from django.db.models.signals import post_save, post_delete
from django.dispatch import receiver

from .models import OrderLineItem


@receiver(post_save, sender=OrderLineItem)
def update_on_save(sender, instance, created, **kwargs):
    """
    Update order total on lineitem update/create
    """
    instance.order.update_total()


@receiver(post_delete, sender=OrderLineItem)
def update_on_delete(sender, instance, **kwargs):
    """
    Update order total on lineitem delete
    """
    instance.order.update_total()
```

All clear, no errors found

signals.py

PEP8 of checkout app



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

urlpatterns = [
    path('', views.checkout, name='checkout'),
    path(
        'checkout_success/<order_number>',
        views.checkout_success,
        name='checkout_success'),
    path(
        'cache_checkout_data/',
        views.cache_checkout_data, name='cache_checkout_data'),
    path('wh/', webhook, name='webhook'),
]
```

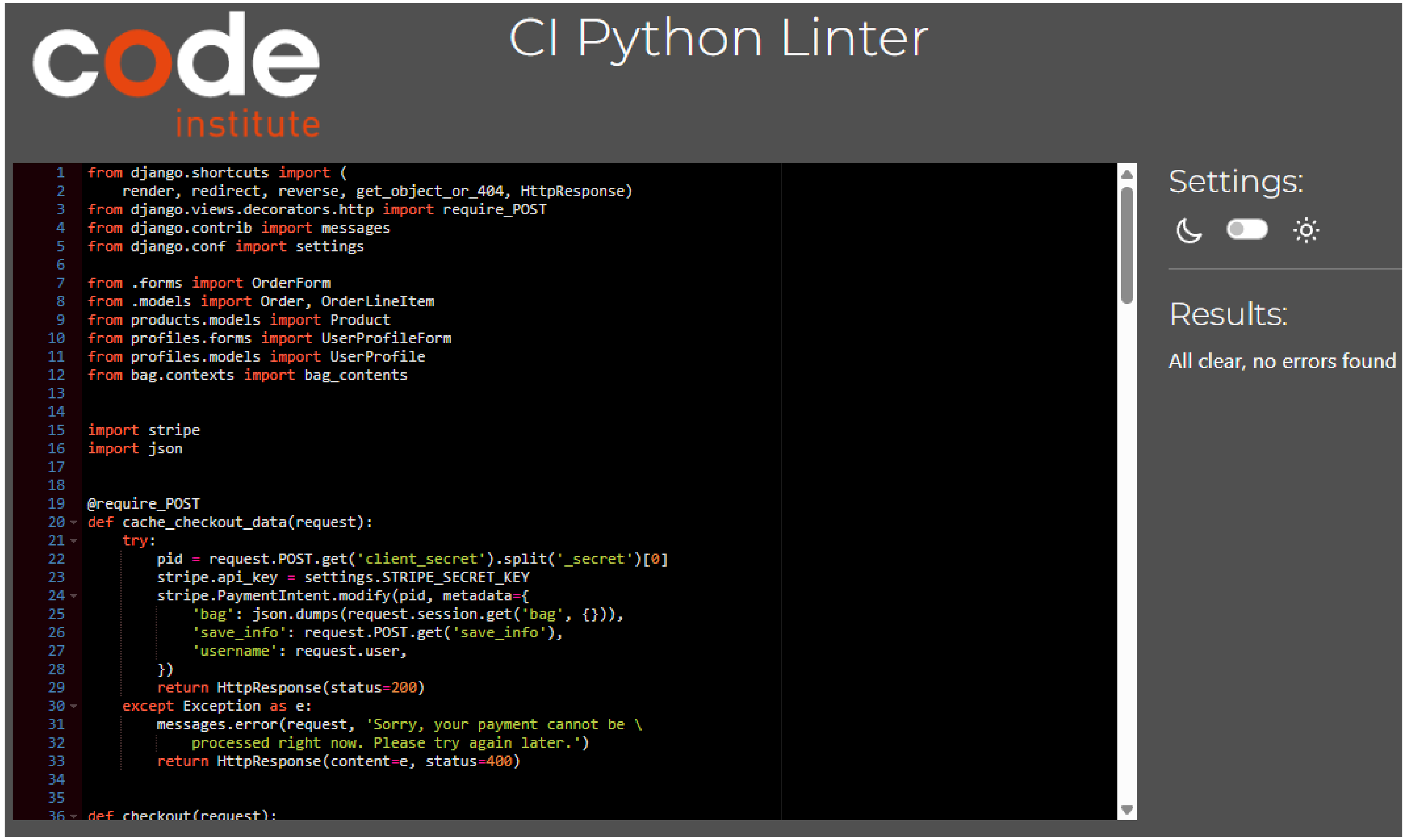
Settings:

Results:

All clear, no errors found

urls.py

PEP8 of checkout app

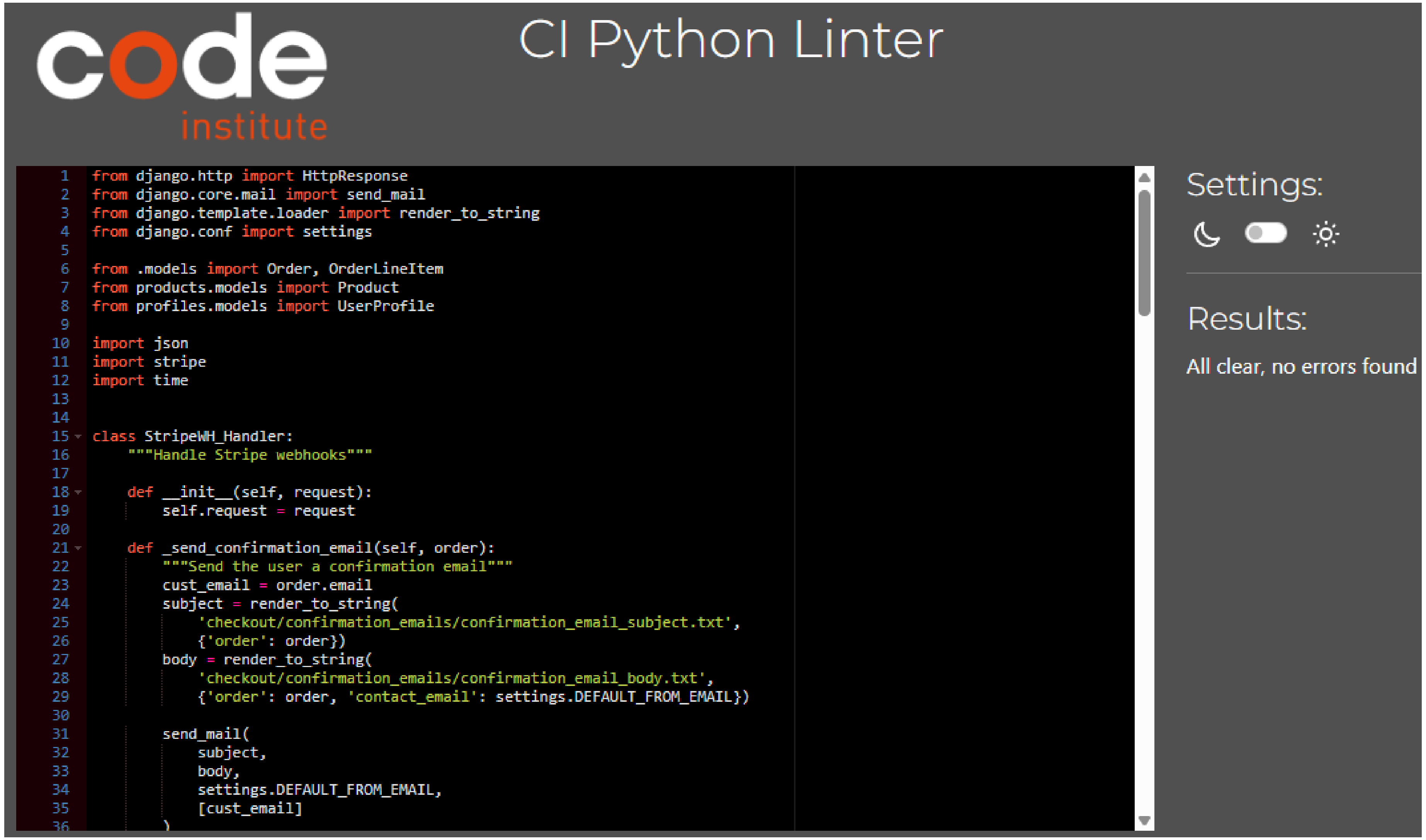


```
from django.shortcuts import (
    render, redirect, reverse, get_object_or_404, HttpResponse)
from django.views.decorators.http import require_POST
from django.contrib import messages
from django.conf import settings

from .forms import OrderForm
from .models import Order, OrderLineItem
from products.models import Product
from profiles.forms import UserProfileForm
from profiles.models import UserProfile
from bag.contexts import bag_contents


import stripe
import json


@require_POST
def cache_checkout_data(request):
    try:
        pid = request.POST.get('client_secret').split('_secret')[0]
        stripe.api_key = settings.STRIPE_SECRET_KEY
        stripe.PaymentIntent.modify(pid, metadata={
            'bag': json.dumps(request.session.get('bag', {})),
            'save_info': request.POST.get('save_info'),
            'username': request.user,
        })
        return HttpResponse(status=200)
    except Exception as e:
        messages.error(request, 'Sorry, your payment cannot be \
            processed right now. Please try again later.')
        return HttpResponse(content=e, status=400)


def checkout(request):
```

All clear, no errors found

views.py

PEP8 of checkout app



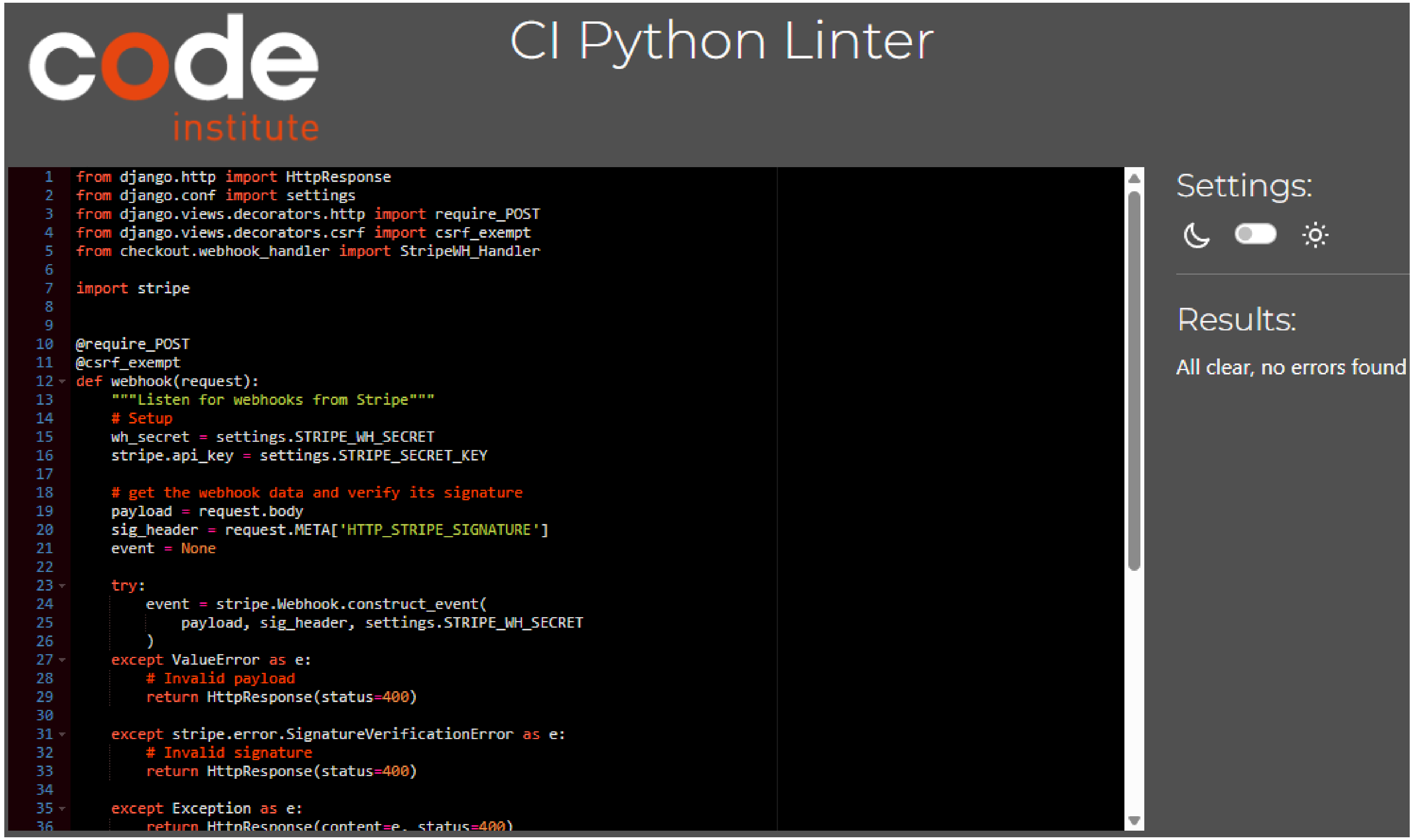
```
from django.http import HttpResponse
from django.core.mail import send_mail
from django.template.loader import render_to_string
from django.conf import settings

from .models import Order, OrderLineItem
from products.models import Product
from profiles.models import UserProfile

import json
import stripe
import time


class StripeWH_Handler:
    """Handle Stripe webhooks"""

    def __init__(self, request):
        self.request = request

    def _send_confirmation_email(self, order):
        """Send the user a confirmation email"""
        cust_email = order.email
        subject = render_to_string(
            'checkout/confirmation_emails/confirmation_email_subject.txt',
            {'order': order})
        body = render_to_string(
            'checkout/confirmation_emails/confirmation_email_body.txt',
            {'order': order, 'contact_email': settings.DEFAULT_FROM_EMAIL})

        send_mail(
            subject,
            body,
            settings.DEFAULT_FROM_EMAIL,
            [cust_email]
        )
```

Results:

All clear, no errors found

Webhook_handler.py

PEP8 of checkout app



```python
from django.http import HttpResponse
from django.conf import settings
from django.views.decorators.http import require_POST
from django.views.decorators.csrf import csrf_exempt
from checkout.webhook_handler import StripeWH_Handler

import stripe


@require_POST
@csrf_exempt
def webhook(request):
    """Listen for webhooks from Stripe"""
    # Setup
    wh_secret = settings.STRIPE_WH_SECRET
    stripe.api_key = settings.STRIPE_SECRET_KEY

    # get the webhook data and verify its signature
    payload = request.body
    sig_header = request.META['HTTP_STRIPE_SIGNATURE']
    event = None

    try:
        event = stripe.Webhook.construct_event(
            payload, sig_header, settings.STRIPE_WH_SECRET
        )
    except ValueError as e:
        # Invalid payload
        return HttpResponse(status=400)

    except stripe.error.SignatureVerificationError as e:
        # Invalid signature
        return HttpResponse(status=400)

    except Exception as e:
        return HttpResponse(content=e, status=400)
```

Settings:

Results:

All clear, no errors found

webhooks.py