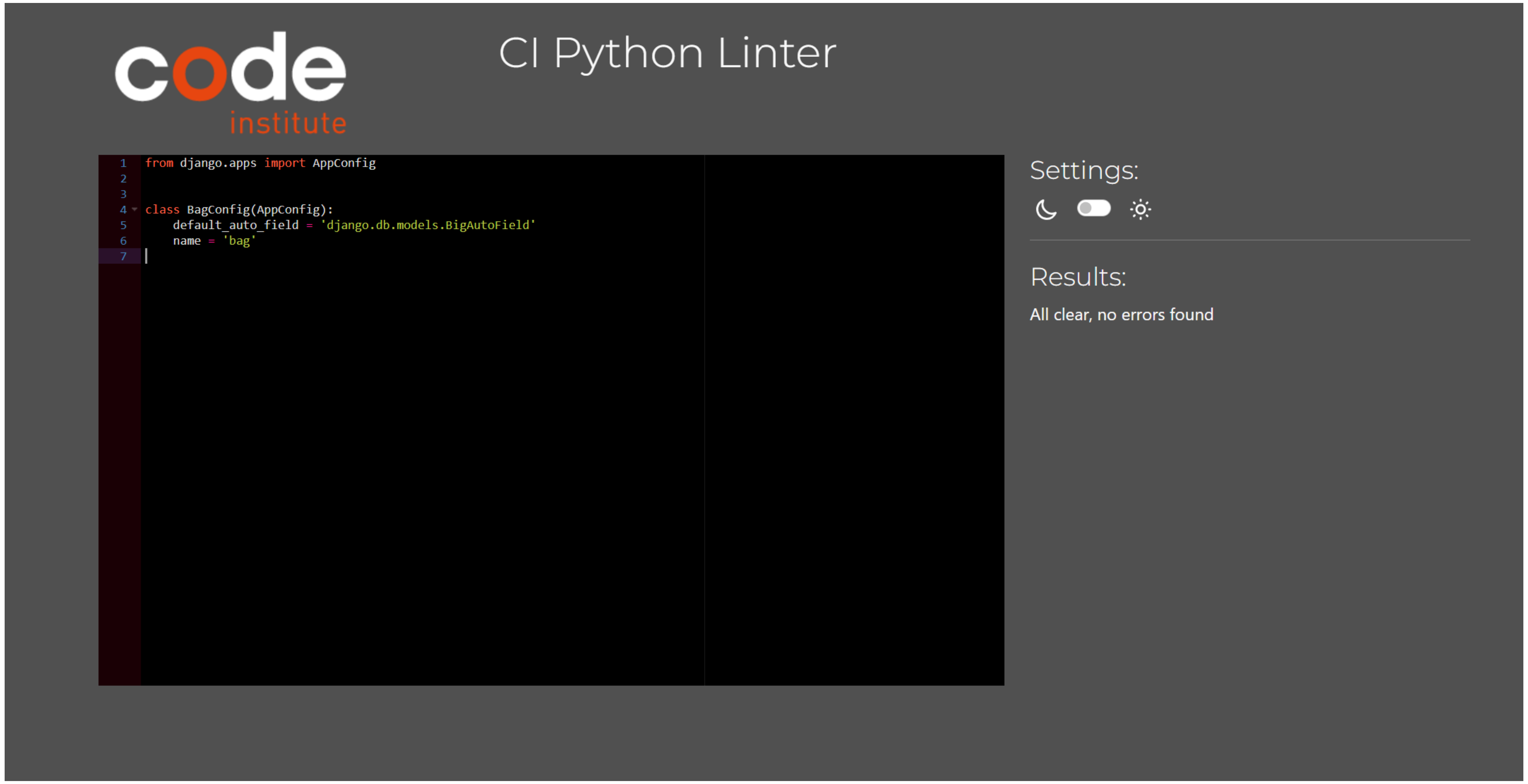
Bag App – apps.py

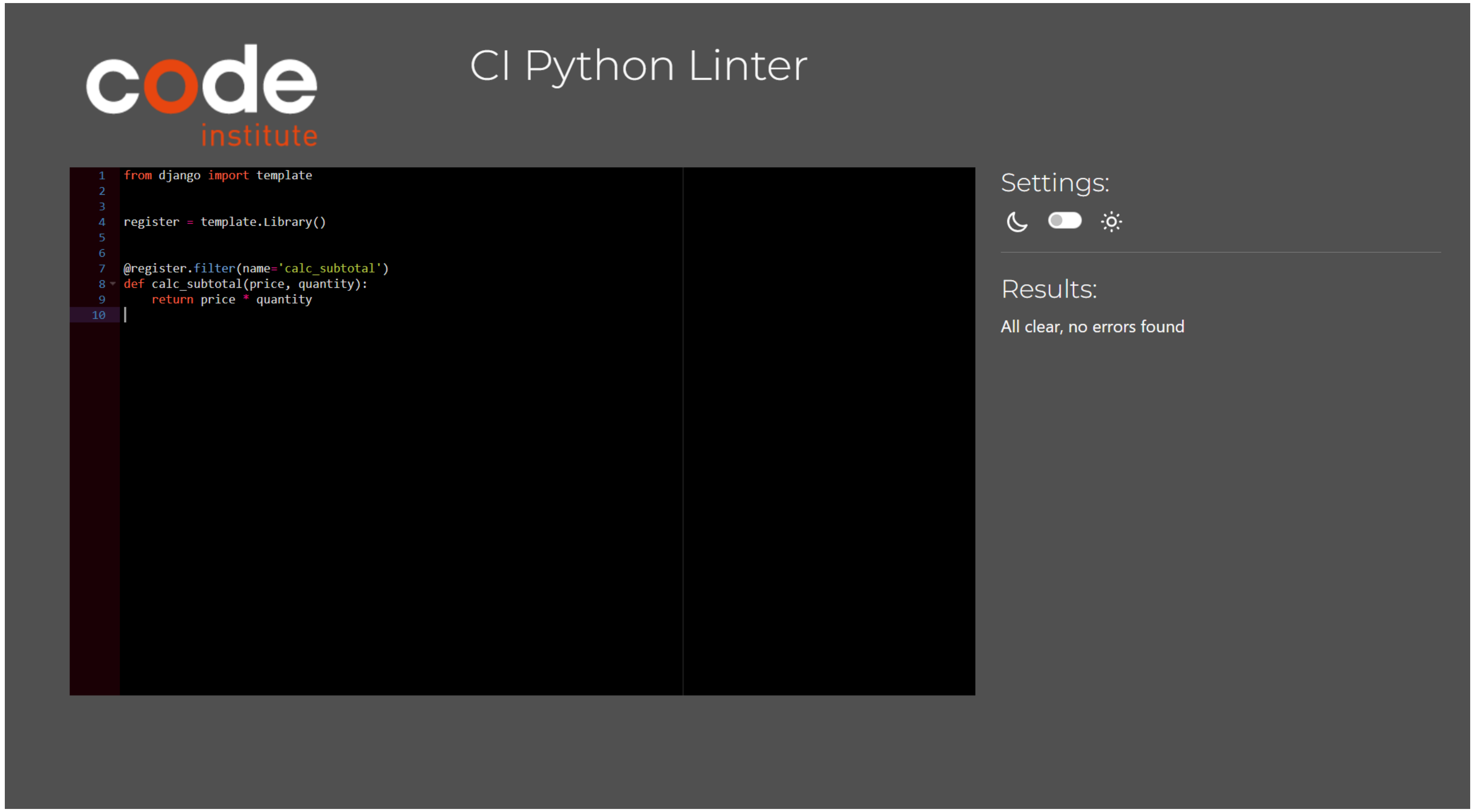


```python
from django.apps import AppConfig


class BagConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'bag'
```

Results:

All clear, no errors found

Bag App – bag_tools.py

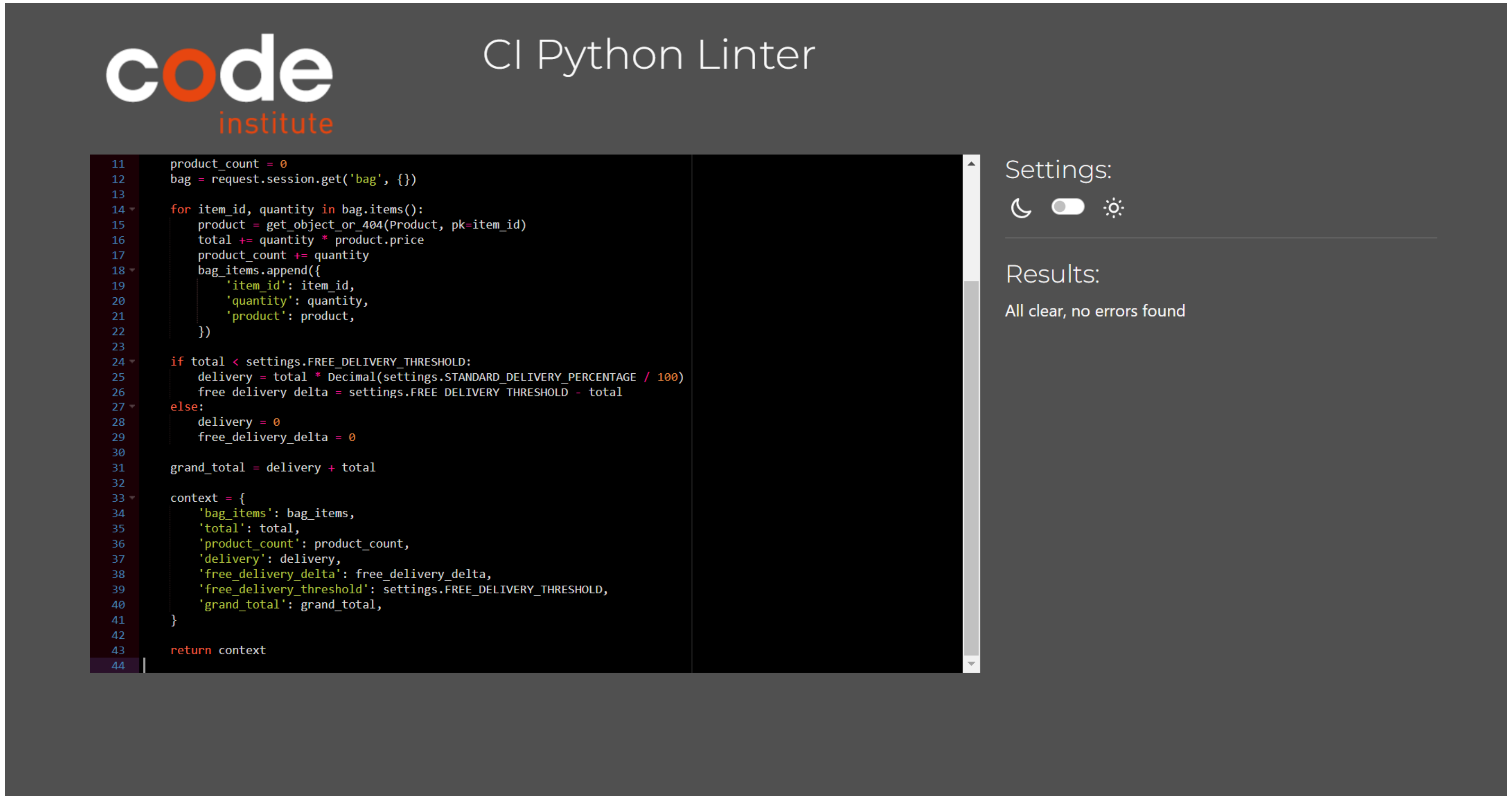


```
from django import template


register = template.Library()


@register.filter(name='calc_subtotal')
def calc_subtotal(price, quantity):
    return price * quantity
```

Results:

All clear, no errors found

Bag App – contexts.py



CI Python Linter

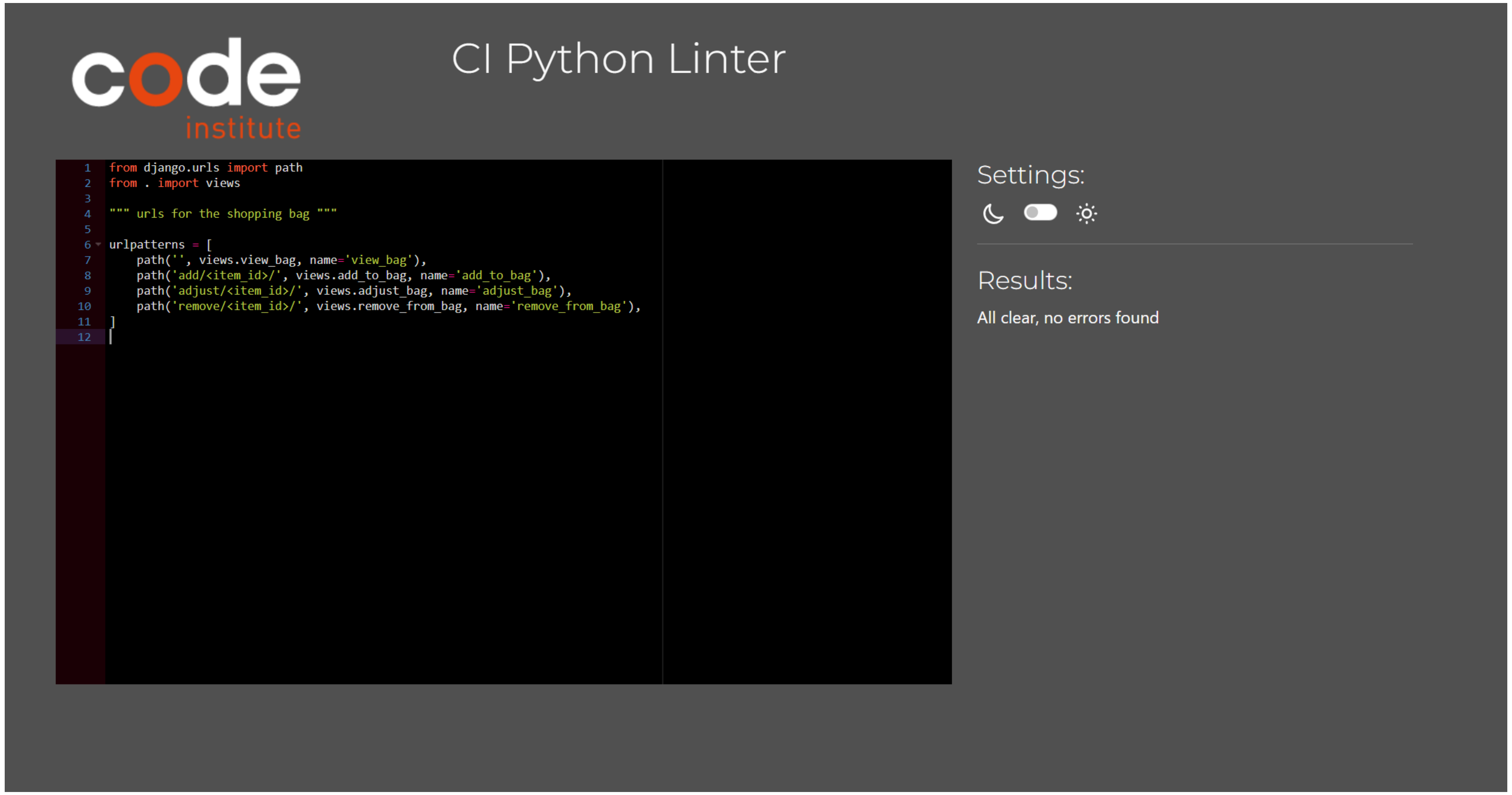
```python
    product_count = 0
    bag = request.session.get('bag', {})

    for item_id, quantity in bag.items():
        product = get_object_or_404(Product, pk=item_id)
        total += quantity * product.price
        product_count += quantity
        bag_items.append({
            'item_id': item_id,
            'quantity': quantity,
            'product': product,
        })

    if total < settings.FREE_DELIVERY_THRESHOLD:
        delivery = total * Decimal(settings.STANDARD_DELIVERY_PERCENTAGE / 100)
        free_delivery_delta = settings.FREE_DELIVERY_THRESHOLD - total
    else:
        delivery = 0
        free_delivery_delta = 0

    grand_total = delivery + total

    context = {
        'bag_items': bag_items,
        'total': total,
        'product_count': product_count,
        'delivery': delivery,
        'free_delivery_delta': free_delivery_delta,
        'free_delivery_threshold': settings.FREE_DELIVERY_THRESHOLD,
        'grand_total': grand_total,
    }

    return context
```

Settings:

Results:

All clear, no errors found

Bag App – urls.py

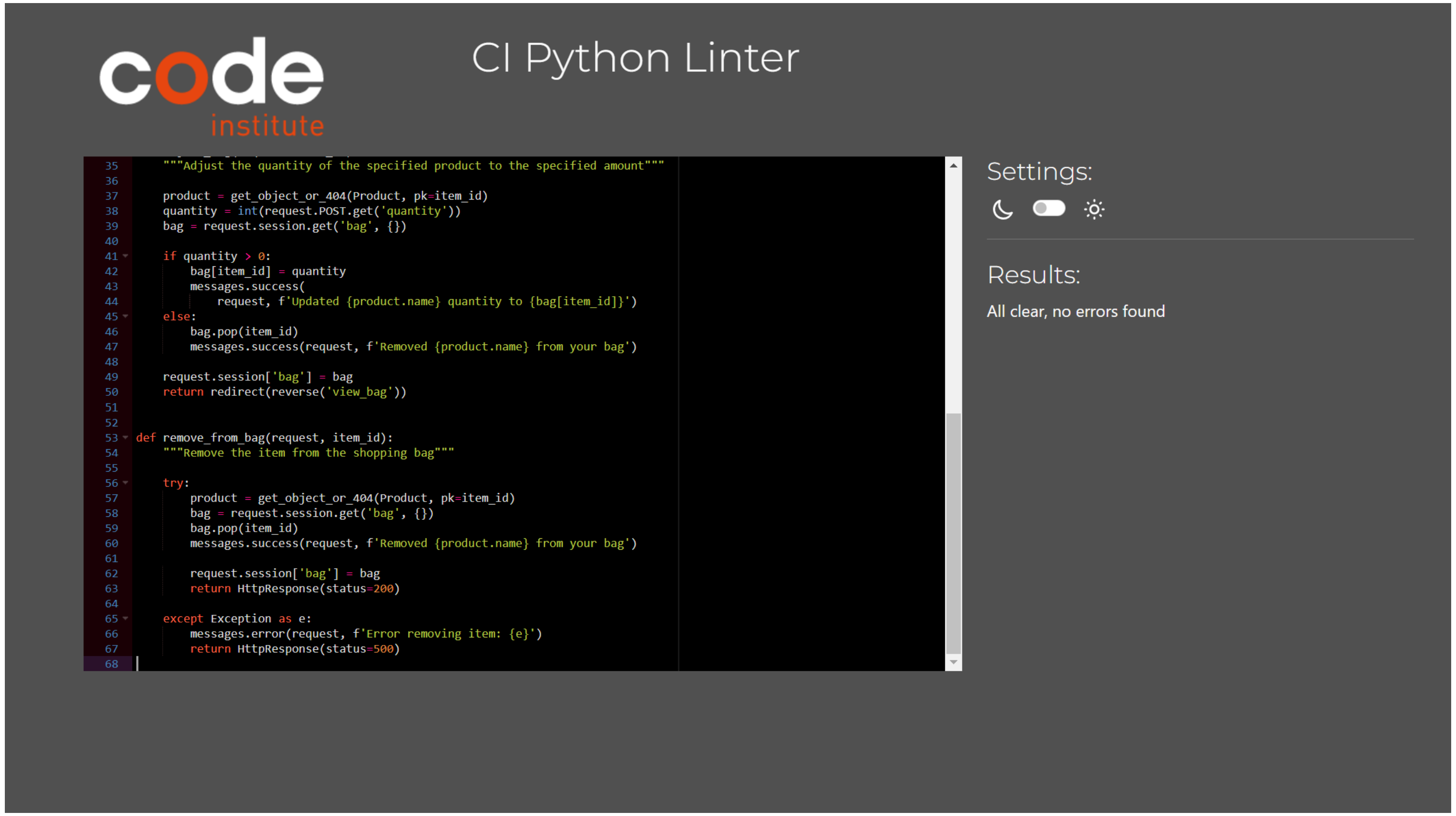


```
from django.urls import path
from . import views

""" urls for the shopping bag """

urlpatterns = [
    path('', views.view_bag, name='view_bag'),
    path('add/<item_id>/', views.add_to_bag, name='add_to_bag'),
    path('adjust/<item_id>/', views.adjust_bag, name='adjust_bag'),
    path('remove/<item_id>/', views.remove_from_bag, name='remove_from_bag'),
]
```

Results:

All clear, no errors found

Bag App – views.py

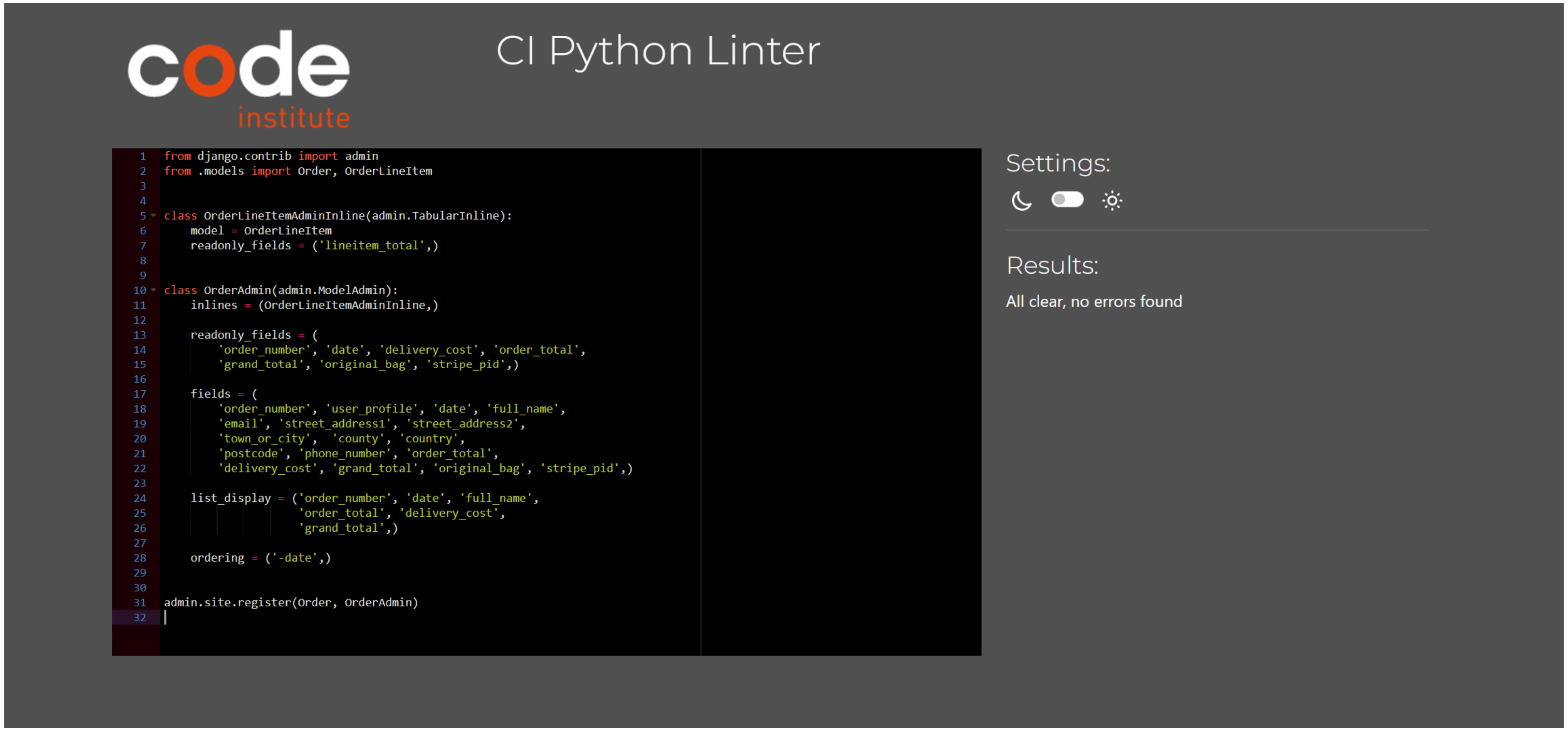


```python
    """Adjust the quantity of the specified product to the specified amount"""

    product = get_object_or_404(Product, pk=item_id)
    quantity = int(request.POST.get('quantity'))
    bag = request.session.get('bag', {})

    if quantity > 0:
        bag[item_id] = quantity
        messages.success(
            request, f'Updated {product.name} quantity to {bag[item_id]}')
    else:
        bag.pop(item_id)
        messages.success(request, f'Removed {product.name} from your bag')

    request.session['bag'] = bag
    return redirect(reverse('view_bag'))


def remove_from_bag(request, item_id):
    """Remove the item from the shopping bag"""

    try:
        product = get_object_or_404(Product, pk=item_id)
        bag = request.session.get('bag', {})
        bag.pop(item_id)
        messages.success(request, f'Removed {product.name} from your bag')

        request.session['bag'] = bag
        return HttpResponse(status=200)

    except Exception as e:
        messages.error(request, f'Error removing item: {e}')
        return HttpResponse(status=500)
```

Results:

All clear, no errors found

Checkout App – admin.py



CI Python Linter

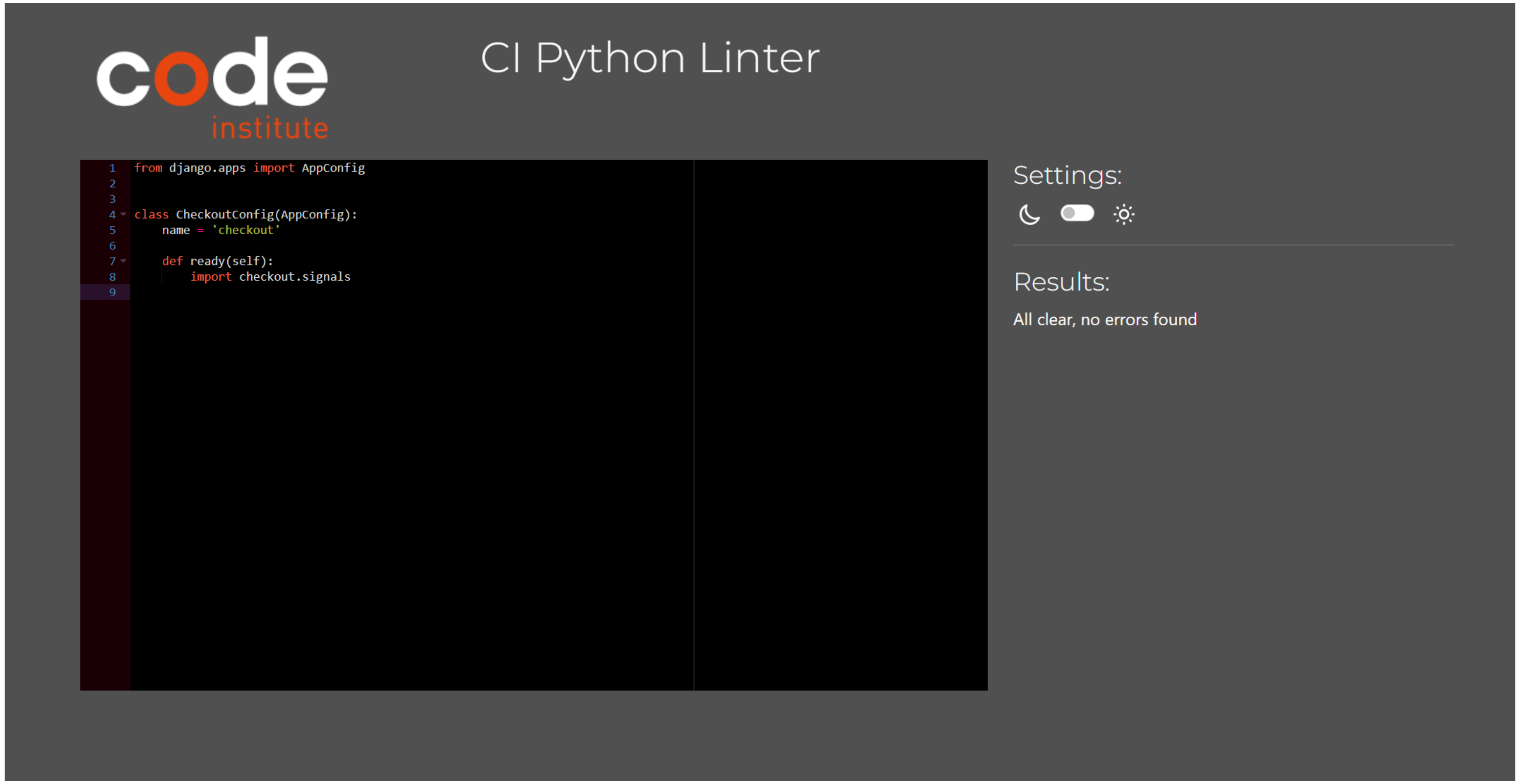
```python
from django.contrib import admin
from .models import Order, OrderLineItem


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


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

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

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

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

    ordering = ('-date',)


admin.site.register(Order, OrderAdmin)

```

Settings:

Results:

All clear, no errors found

Checkout App – apps.py

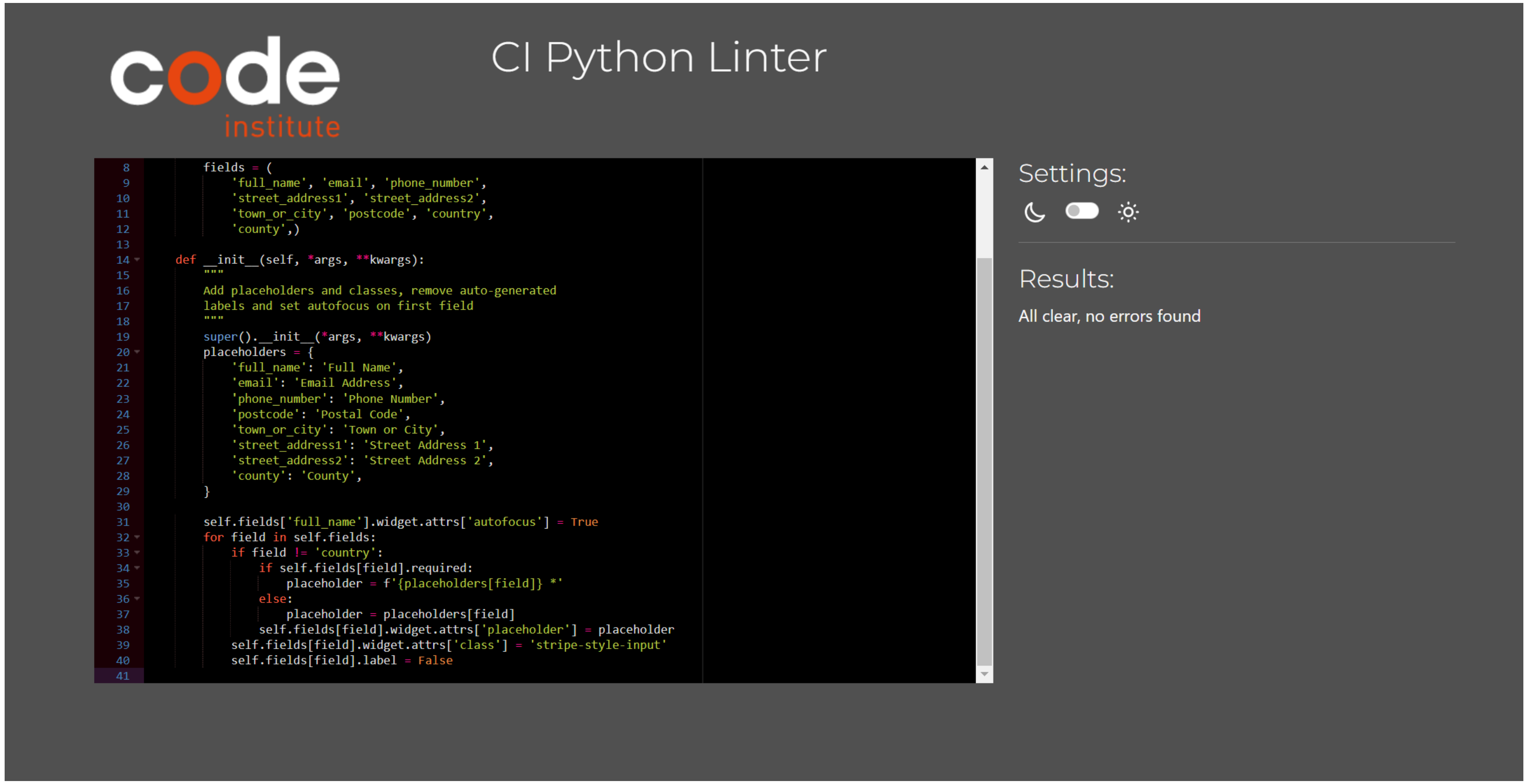


```
from django.apps import AppConfig


class CheckoutConfig(AppConfig):
    name = 'checkout'

    def ready(self):
        import checkout.signals
```

Results:

All clear, no errors found

Checkout App – forms.py



CI Python Linter

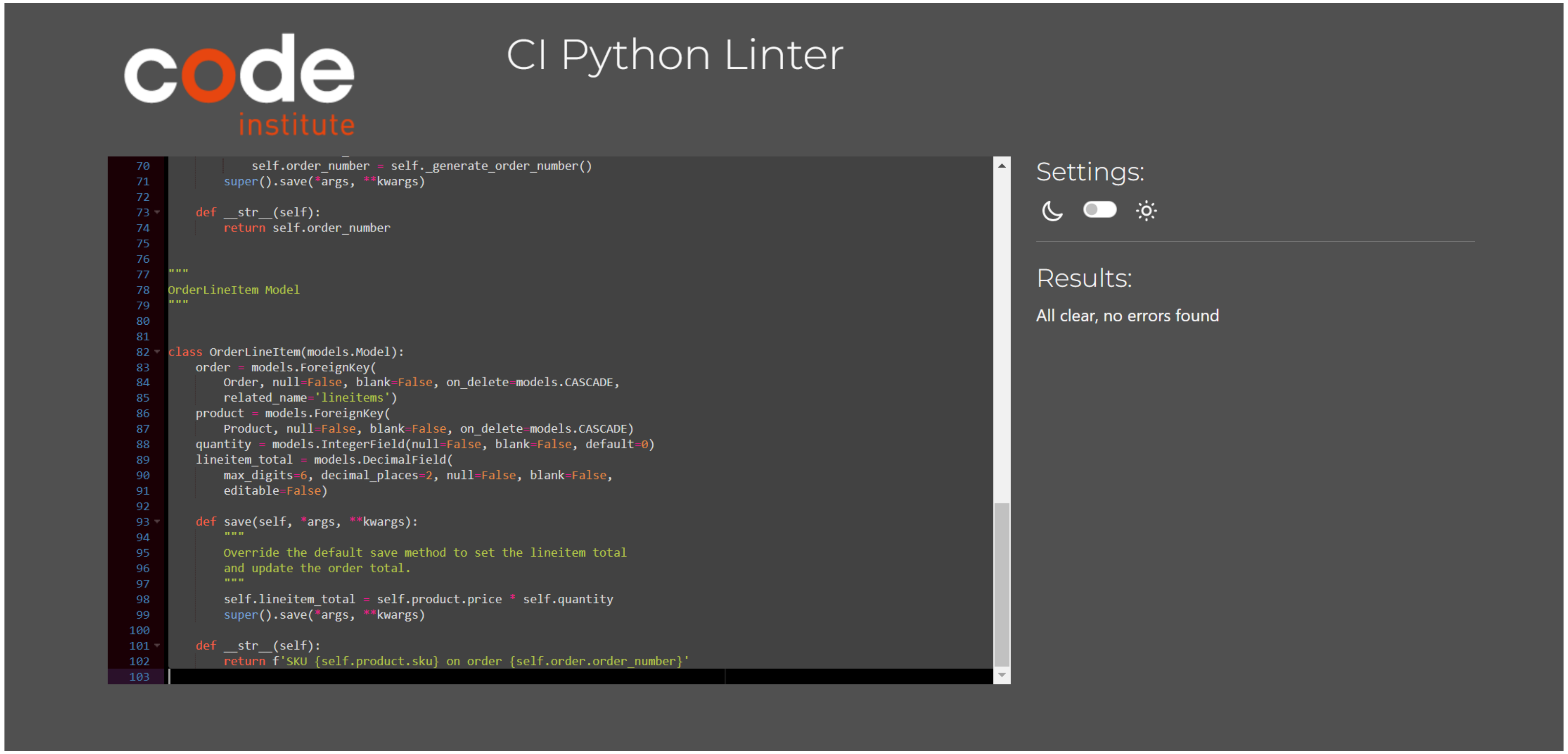
```
        fields = (
            'full_name', 'email', 'phone_number',
            'street_address1', 'street_address2',
            'town_or_city', 'postcode', 'country',
            'county',)

    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',
        }

        self.fields['full_name'].widget.attrs['autofocus'] = True
        for field in self.fields:
            if field != 'country':
                if self.fields[field].required:
                    placeholder = f'{placeholders[field]} *'
                else:
                    placeholder = placeholders[field]
                self.fields[field].widget.attrs['placeholder'] = placeholder
            self.fields[field].widget.attrs['class'] = 'stripe-style-input'
            self.fields[field].label = False
```

Settings:

Results:

All clear, no errors found

Checkout App – models.py

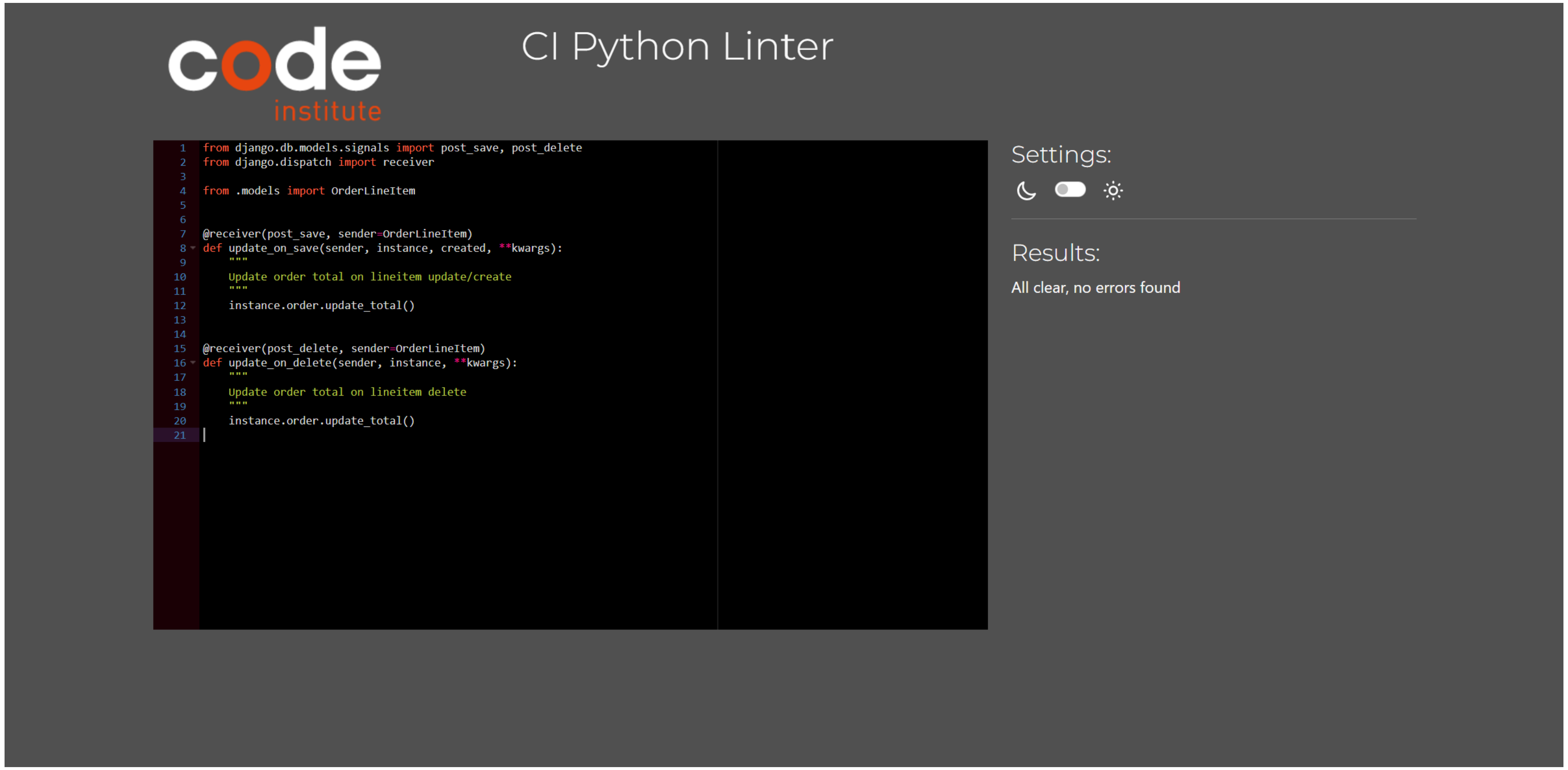


```
            self.order_number = self._generate_order_number()
        super().save(*args, **kwargs)

    def __str__(self):
        return self.order_number


"""
OrderLineItem Model
"""


class OrderLineItem(models.Model):
    order = models.ForeignKey(
        Order, null=False, blank=False, on_delete=models.CASCADE,
        related_name='lineitems')
    product = models.ForeignKey(
        Product, null=False, blank=False, on_delete=models.CASCADE)
    quantity = models.IntegerField(null=False, blank=False, default=0)
    lineitem_total = models.DecimalField(
        max_digits=6, decimal_places=2, null=False, blank=False,
        editable=False)

    def save(self, *args, **kwargs):
        """
        Override the default save method to set the lineitem total
        and update the order total.
        """
        self.lineitem_total = self.product.price * self.quantity
        super().save(*args, **kwargs)

    def __str__(self):
        return f'SKU {self.product.sku} on order {self.order.order_number}'
```

Results:

All clear, no errors found

Checkout App – signals.py



CI Python Linter

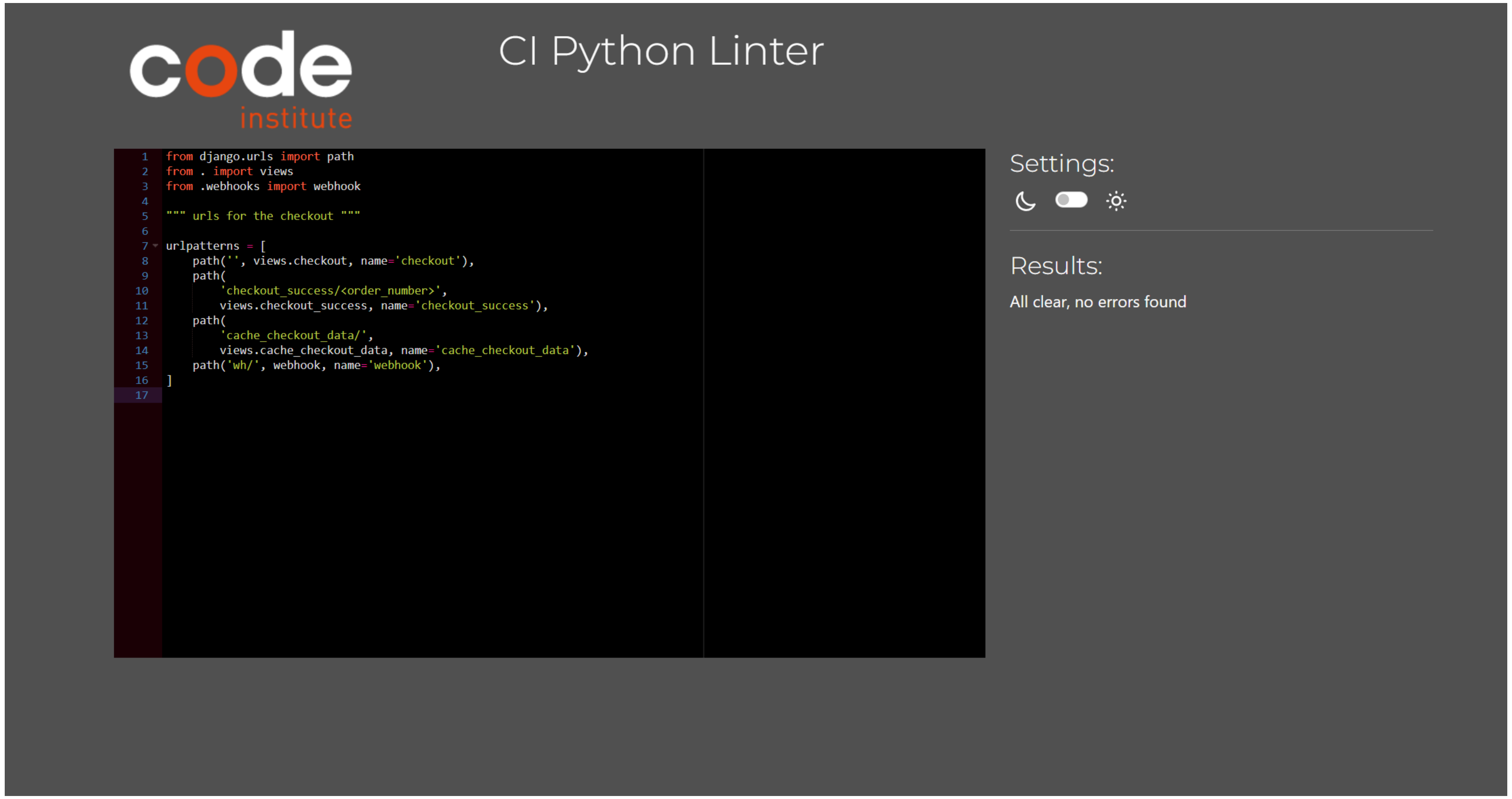
```python
from django.db.models.signals import post_save, post_delete
from django.dispatch import receiver

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()
```

Settings:

Results:

All clear, no errors found

Checkout App – urls.py



CI Python Linter

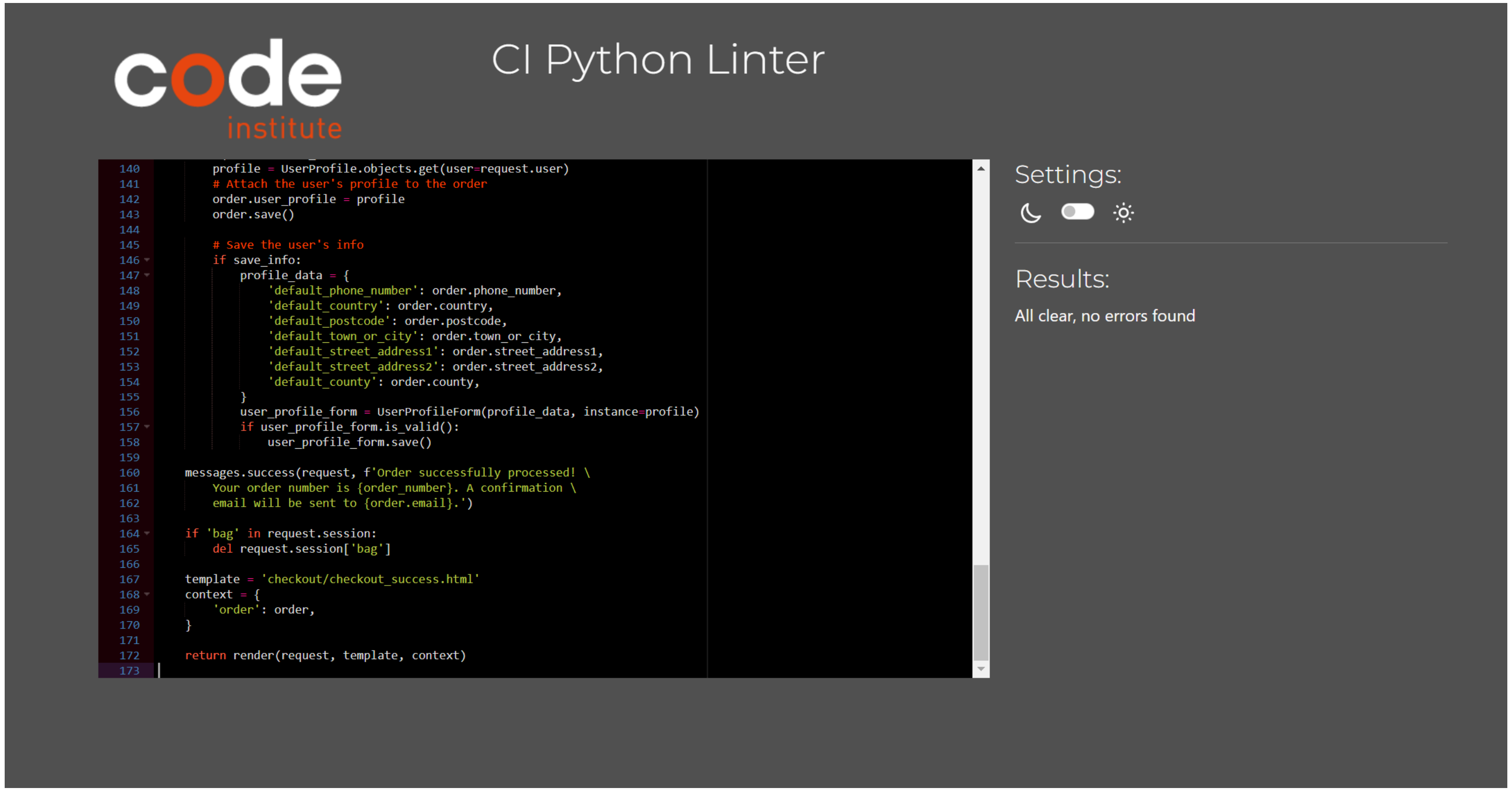
```
from django.urls import path
from . import views
from .webhooks import webhook

""" urls for the checkout """

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

Checkout App – views.py



CI Python Linter

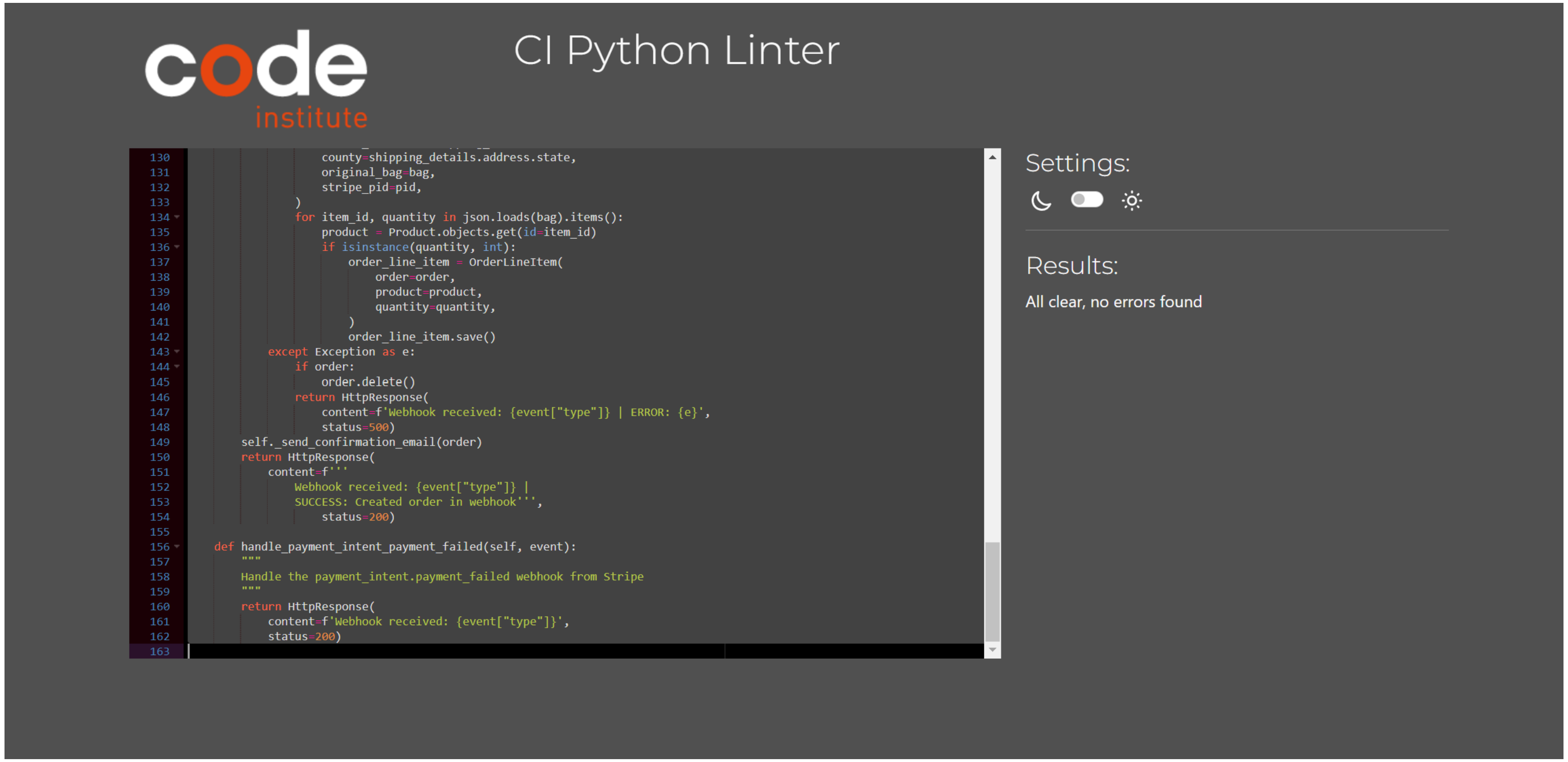
```
        profile = UserProfile.objects.get(user=request.user)
        # Attach the user's profile to the order
        order.user_profile = profile
        order.save()

        # Save the user's info
        if save_info:
            profile_data = {
                'default_phone_number': order.phone_number,
                'default_country': order.country,
                'default_postcode': order.postcode,
                'default_town_or_city': order.town_or_city,
                'default_street_address1': order.street_address1,
                'default_street_address2': order.street_address2,
                'default_county': order.county,
            }
            user_profile_form = UserProfileForm(profile_data, instance=profile)
            if user_profile_form.is_valid():
                user_profile_form.save()

    messages.success(request, f'Order successfully processed! \
        Your order number is {order_number}. A confirmation \
        email will be sent to {order.email}.')

    if 'bag' in request.session:
        del request.session['bag']

    template = 'checkout/checkout_success.html'
    context = {
        'order': order,
    }

    return render(request, template, context)
```

Settings:

Results:

All clear, no errors found

Checkout App – webhook_handler.py



CI Python Linter

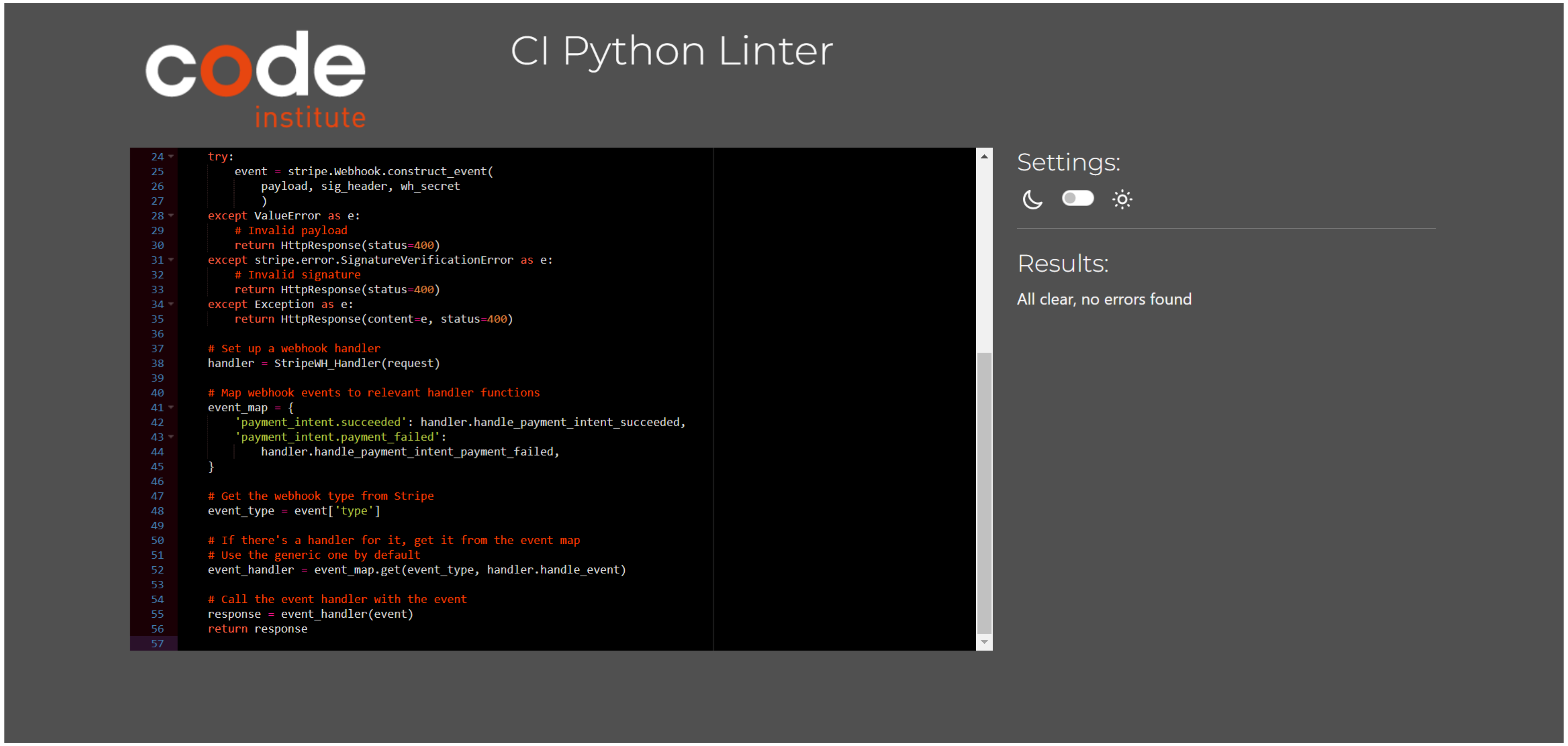
```
                    county=shipping_details.address.state,
                    original_bag=bag,
                    stripe_pid=pid,
                )
                for item_id, quantity in json.loads(bag).items():
                    product = Product.objects.get(id=item_id)
                    if isinstance(quantity, int):
                        order_line_item = OrderLineItem(
                            order=order,
                            product=product,
                            quantity=quantity,
                        )
                        order_line_item.save()
            except Exception as e:
                if order:
                    order.delete()
                return HttpResponse(
                    content=f'Webhook received: {event["type"]} | ERROR: {e}',
                    status=500)
        self._send_confirmation_email(order)
        return HttpResponse(
            content=f'''
                Webhook received: {event["type"]} |
                SUCCESS: Created order in webhook''',
                    status=200)

    def handle_payment_intent_payment_failed(self, event):
        """
        Handle the payment_intent.payment_failed webhook from Stripe
        """
        return HttpResponse(
            content=f'Webhook received: {event["type"]}',
            status=200)
```

Settings:

Results:

All clear, no errors found

Checkout App – webhooks.py



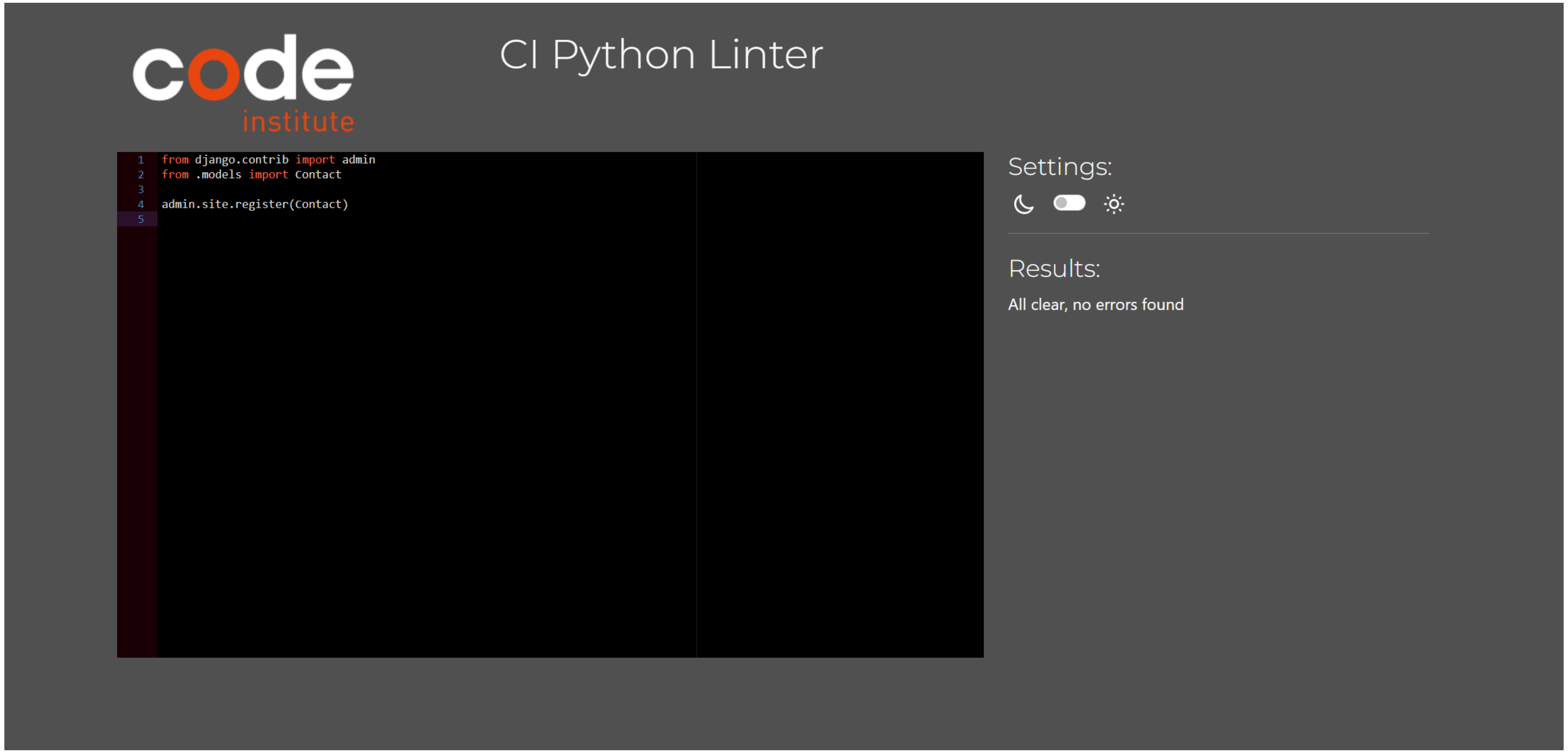
```
    try:
        event = stripe.Webhook.construct_event(
            payload, sig_header, wh_secret
            )
    except ValueError as e:
        # Invalid payload
        return HttpResponse(status=400)
    except stripe.error.SignatureVerificationError as e:
        # Invalid signature
        return HttpResponse(status=400)
    except Exception as e:
        return HttpResponse(content=e, status=400)

    # Set up a webhook handler
    handler = StripeWH_Handler(request)

    # Map webhook events to relevant handler functions
    event_map = {
        'payment_intent.succeeded': handler.handle_payment_intent_succeeded,
        'payment_intent.payment_failed':
            handler.handle_payment_intent_payment_failed,
    }

    # Get the webhook type from Stripe
    event_type = event['type']

    # If there's a handler for it, get it from the event map
    # Use the generic one by default
    event_handler = event_map.get(event_type, handler.handle_event)

    # Call the event handler with the event
    response = event_handler(event)
    return response
```

Settings:

Results:

All clear, no errors found

Contact App – admin.py


CI Python Linter

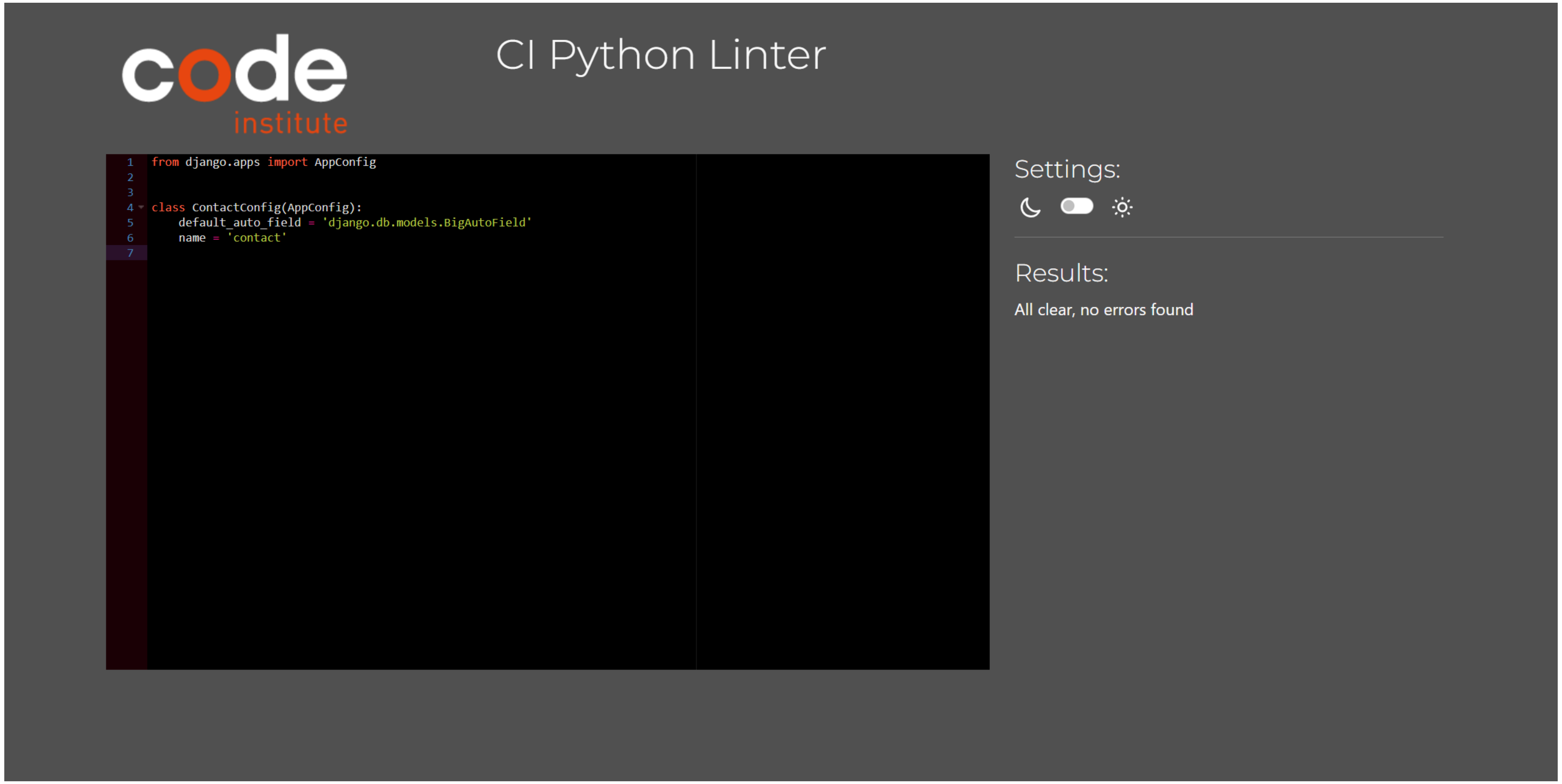
```
from django.contrib import admin
from .models import Contact

admin.site.register(Contact)

```

Settings:

Results:

All clear, no errors found

Contact App – apps.py

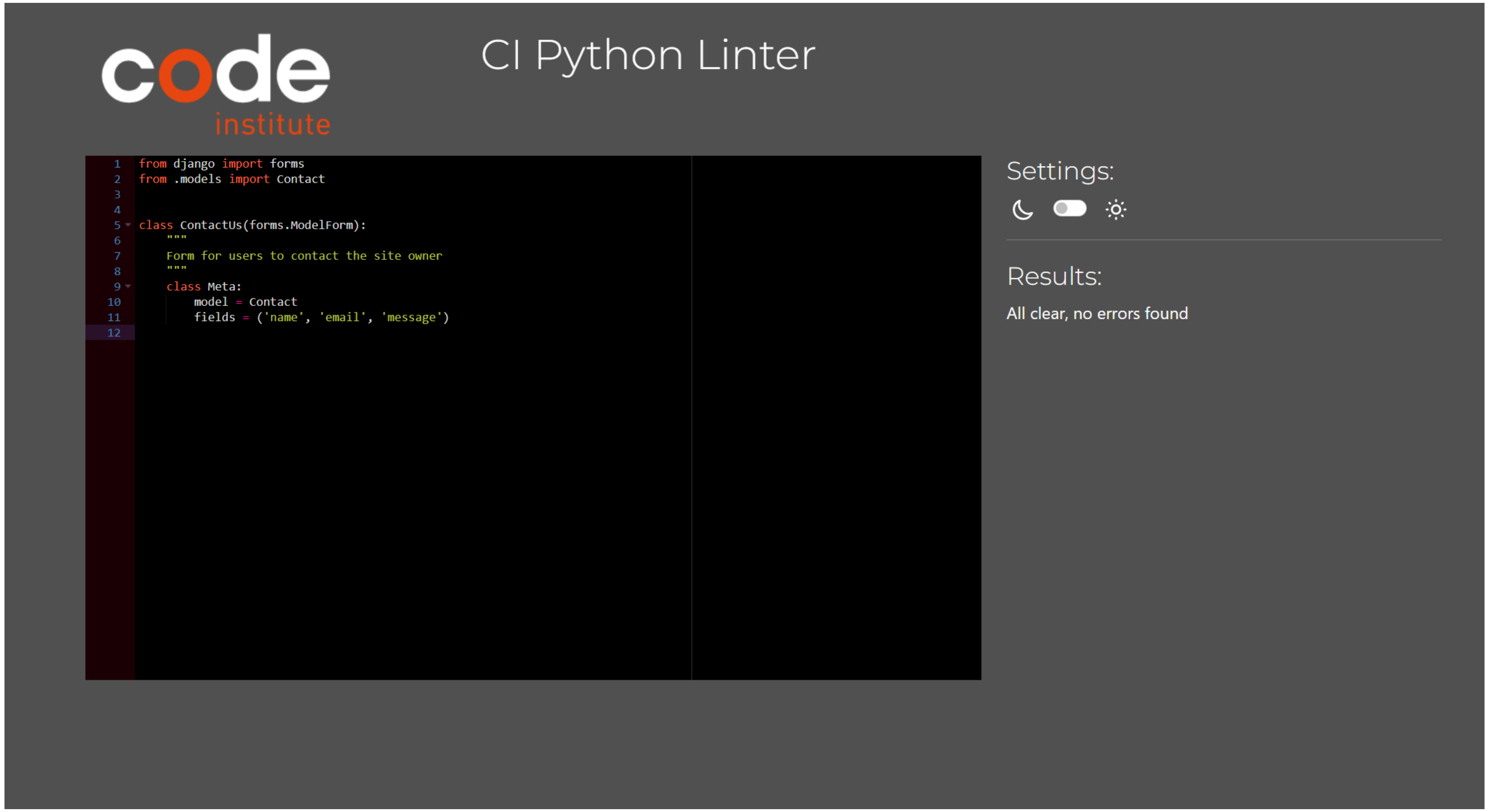


```python
from django.apps import AppConfig


class ContactConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'contact'

```

Results:

All clear, no errors found

Contact App – forms.py



CI Python Linter

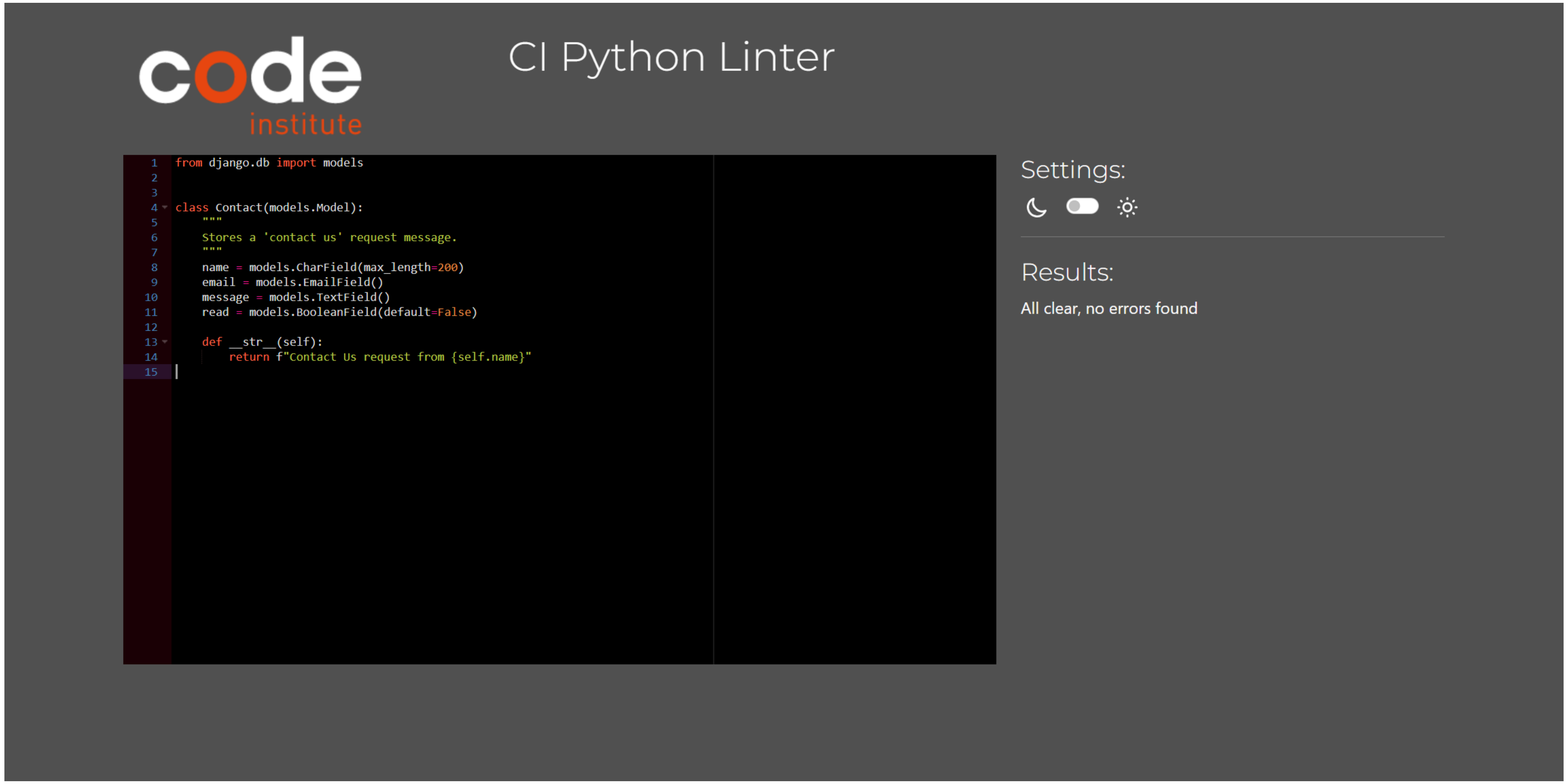
```python
from django import forms
from .models import Contact


class ContactUs(forms.ModelForm):
    """
    Form for users to contact the site owner
    """
    class Meta:
        model = Contact
        fields = ('name', 'email', 'message')
```

Settings:

Results:

All clear, no errors found

Contact App – models.py


CI Python Linter

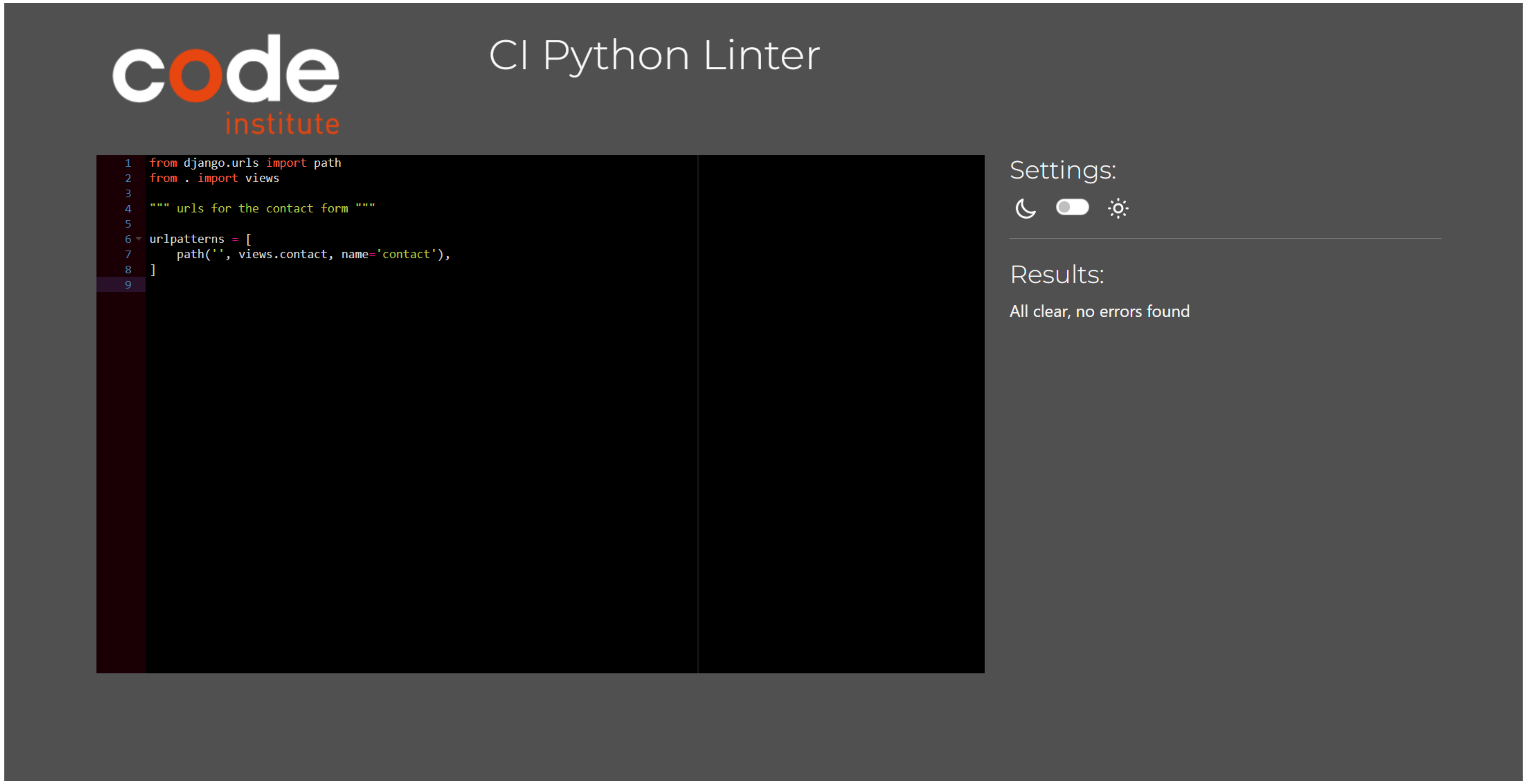
```python
from django.db import models


class Contact(models.Model):
    """
    Stores a 'contact us' request message.
    """
    name = models.CharField(max_length=200)
    email = models.EmailField()
    message = models.TextField()
    read = models.BooleanField(default=False)

    def __str__(self):
        return f"Contact Us request from {self.name}"

```

Settings:

Results:

All clear, no errors found

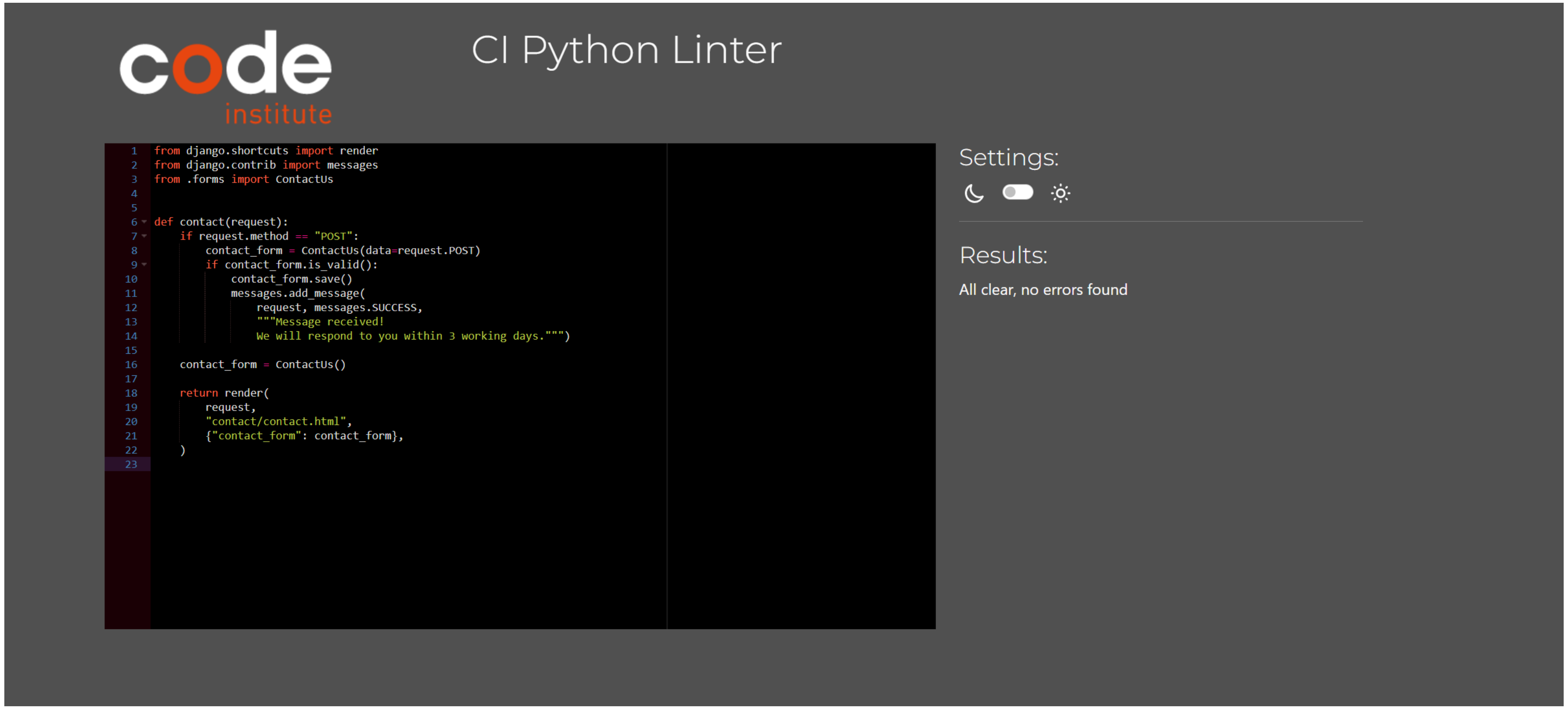
Contact App – urls.py



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

""" urls for the contact form """

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

All clear, no errors found

Contact App – views.py



CI Python Linter

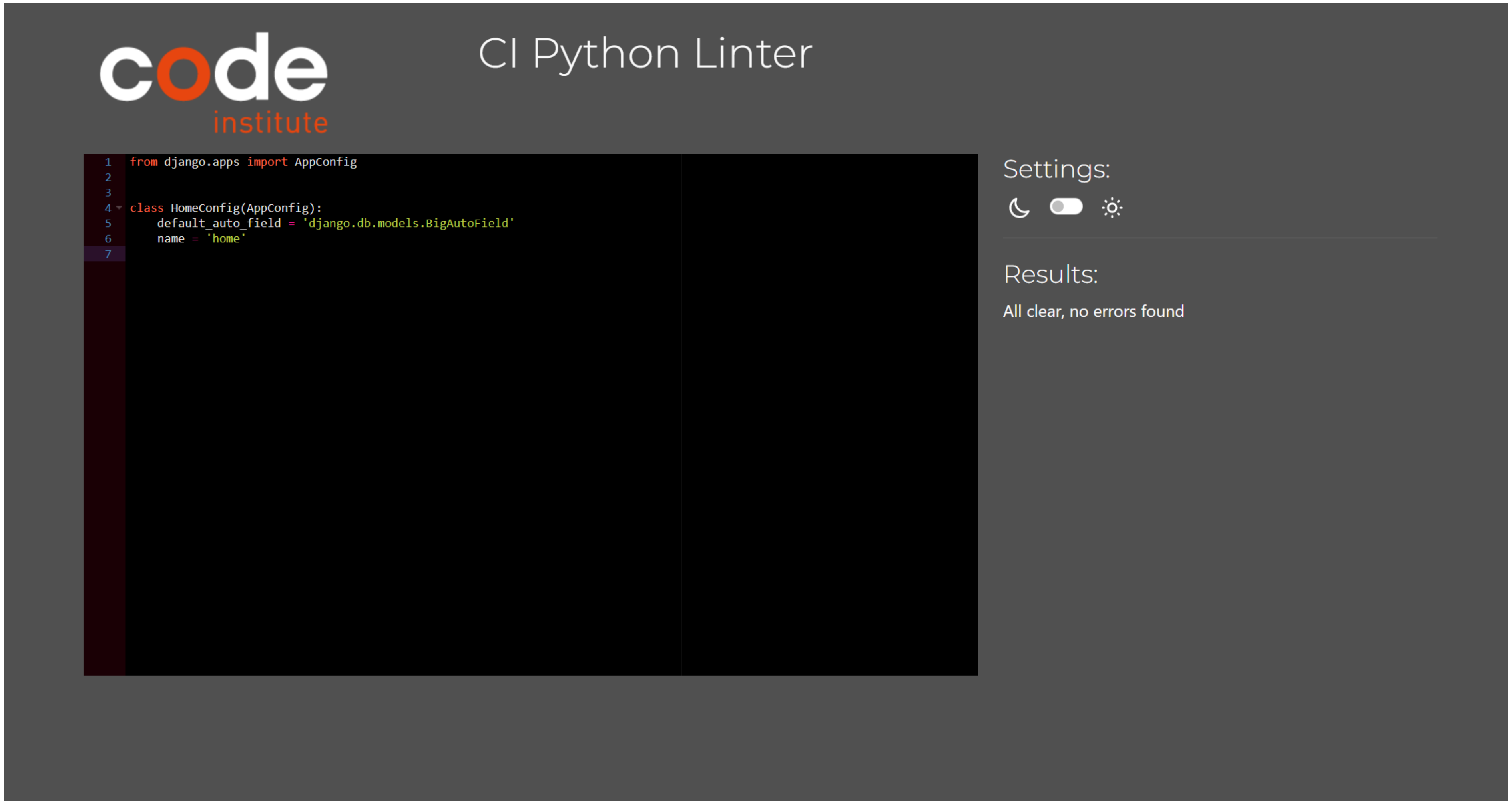
```python
from django.shortcuts import render
from django.contrib import messages
from .forms import ContactUs


def contact(request):
    if request.method == "POST":
        contact_form = ContactUs(data=request.POST)
        if contact_form.is_valid():
            contact_form.save()
            messages.add_message(
                request, messages.SUCCESS,
                """Message received!
                We will respond to you within 3 working days.""")

    contact_form = ContactUs()

    return render(
        request,
        "contact/contact.html",
        {"contact_form": contact_form},
    )
```

Settings:

Results:

All clear, no errors found

Home App – apps.py

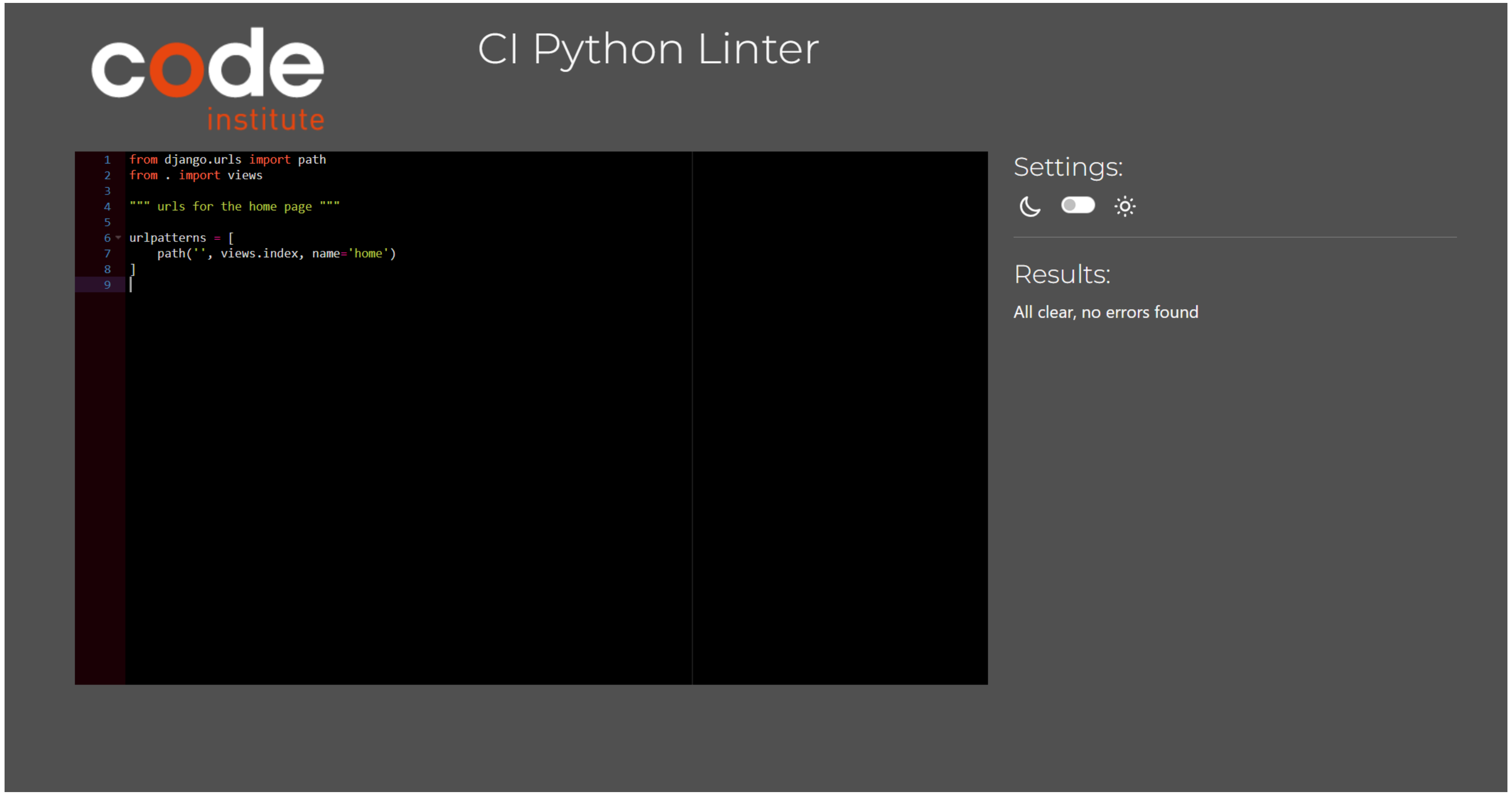


```
from django.apps import AppConfig


class HomeConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'home'
```

Results:

All clear, no errors found

Home App – urls.py



CI Python Linter

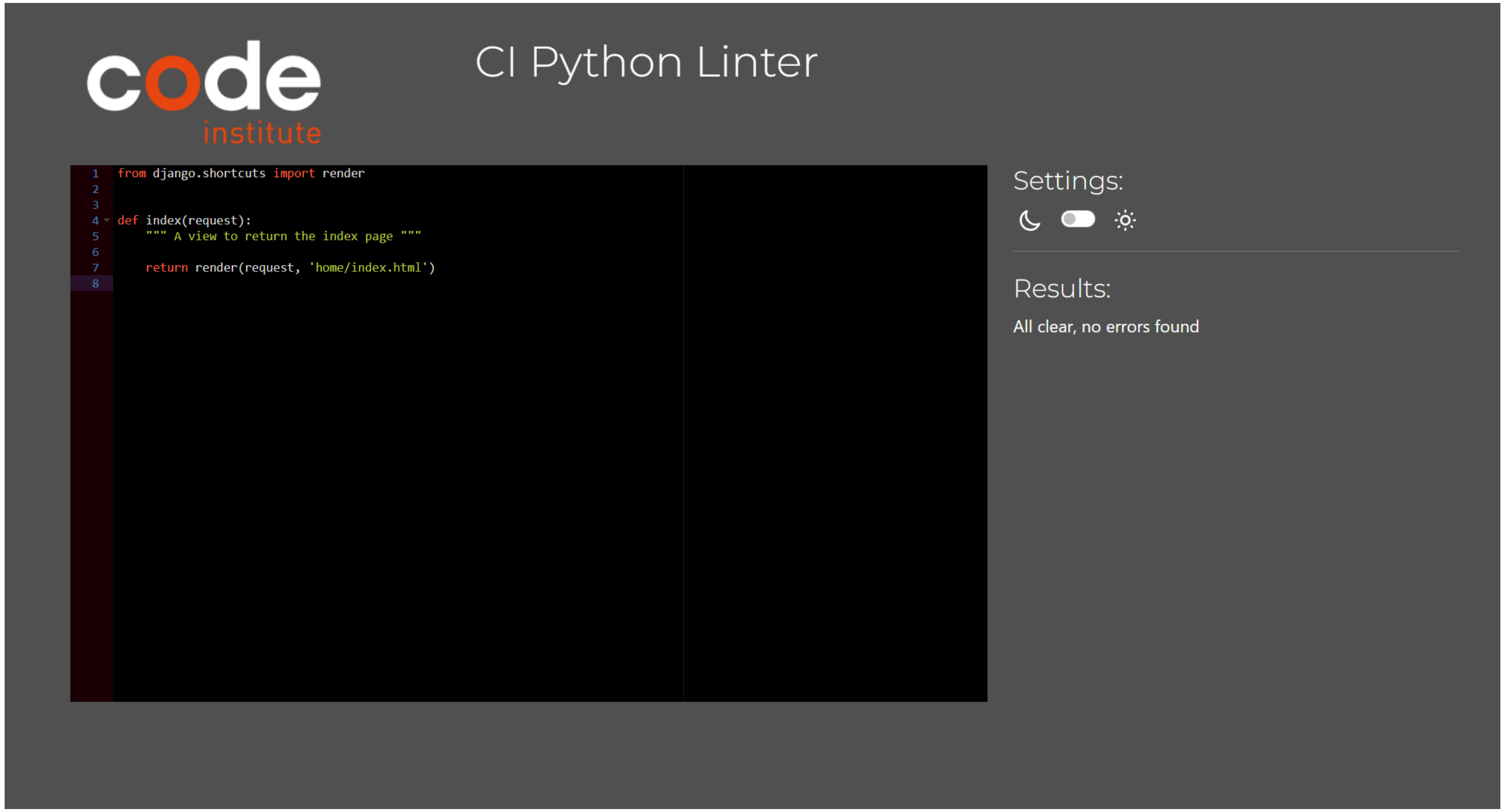
```
from django.urls import path
from . import views

""" urls for the home page """

urlpatterns = [
    path('', views.index, name='home')
]
```

Settings:

Results:

All clear, no errors found

Home App – views.py

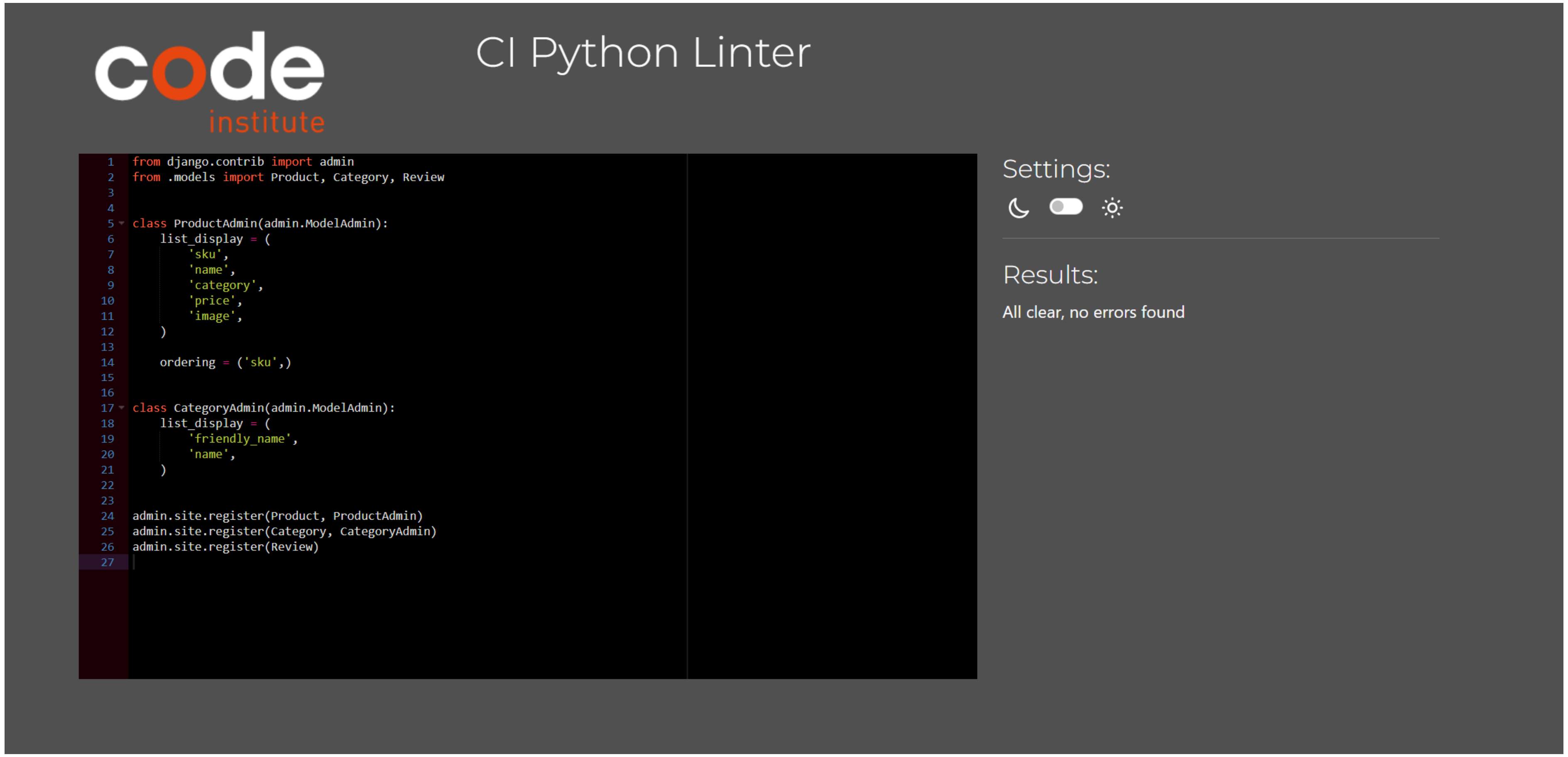


```
from django.shortcuts import render


def index(request):
    """ A view to return the index page """

    return render(request, 'home/index.html')
```

Results:

All clear, no errors found

Products App – admin.py



CI Python Linter

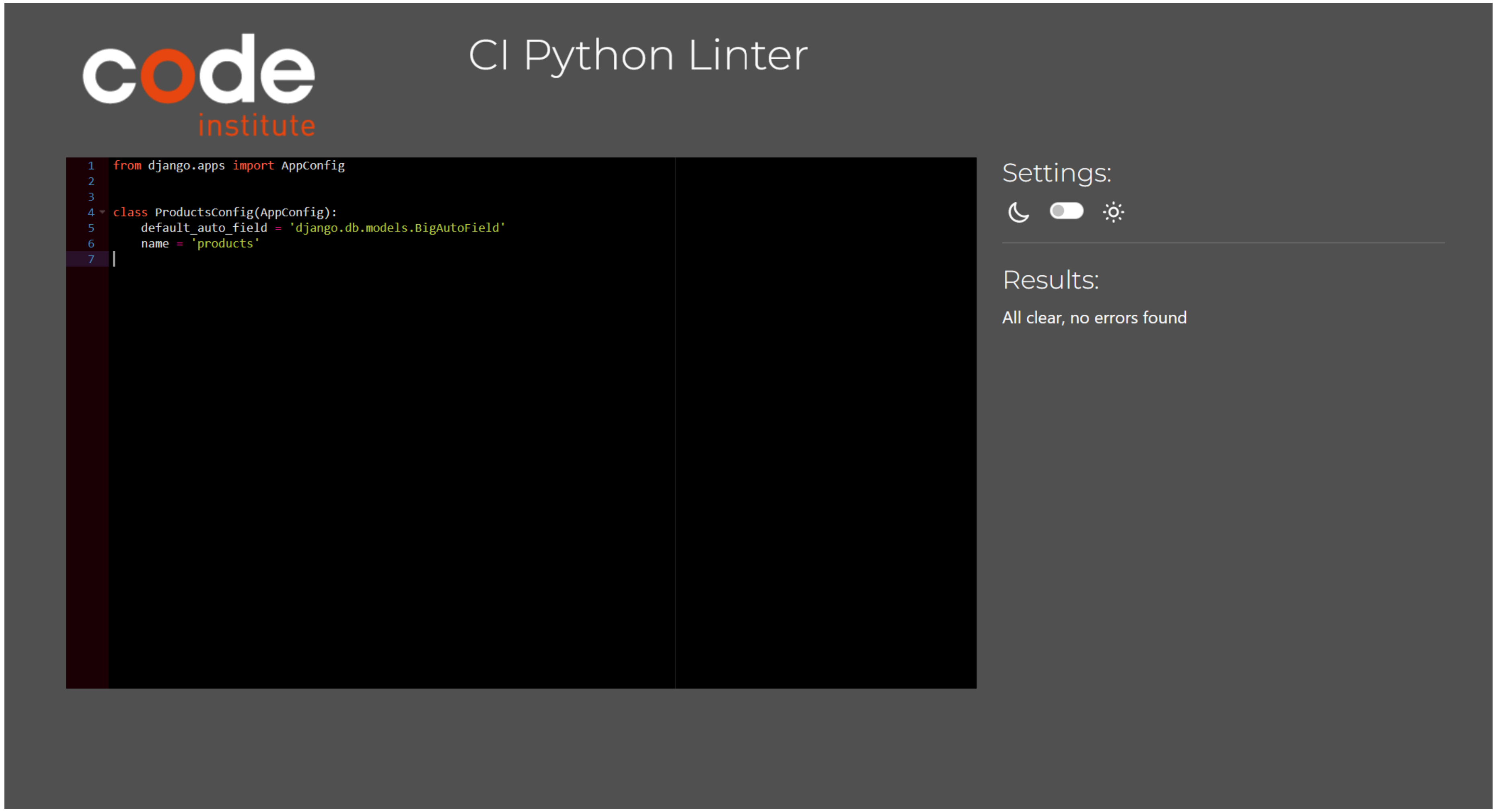
```python
from django.contrib import admin
from .models import Product, Category, Review


class ProductAdmin(admin.ModelAdmin):
    list_display = (
        'sku',
        'name',
        'category',
        'price',
        'image',
    )

    ordering = ('sku',)


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


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

Settings:

Results:

All clear, no errors found

Products App – apps.py

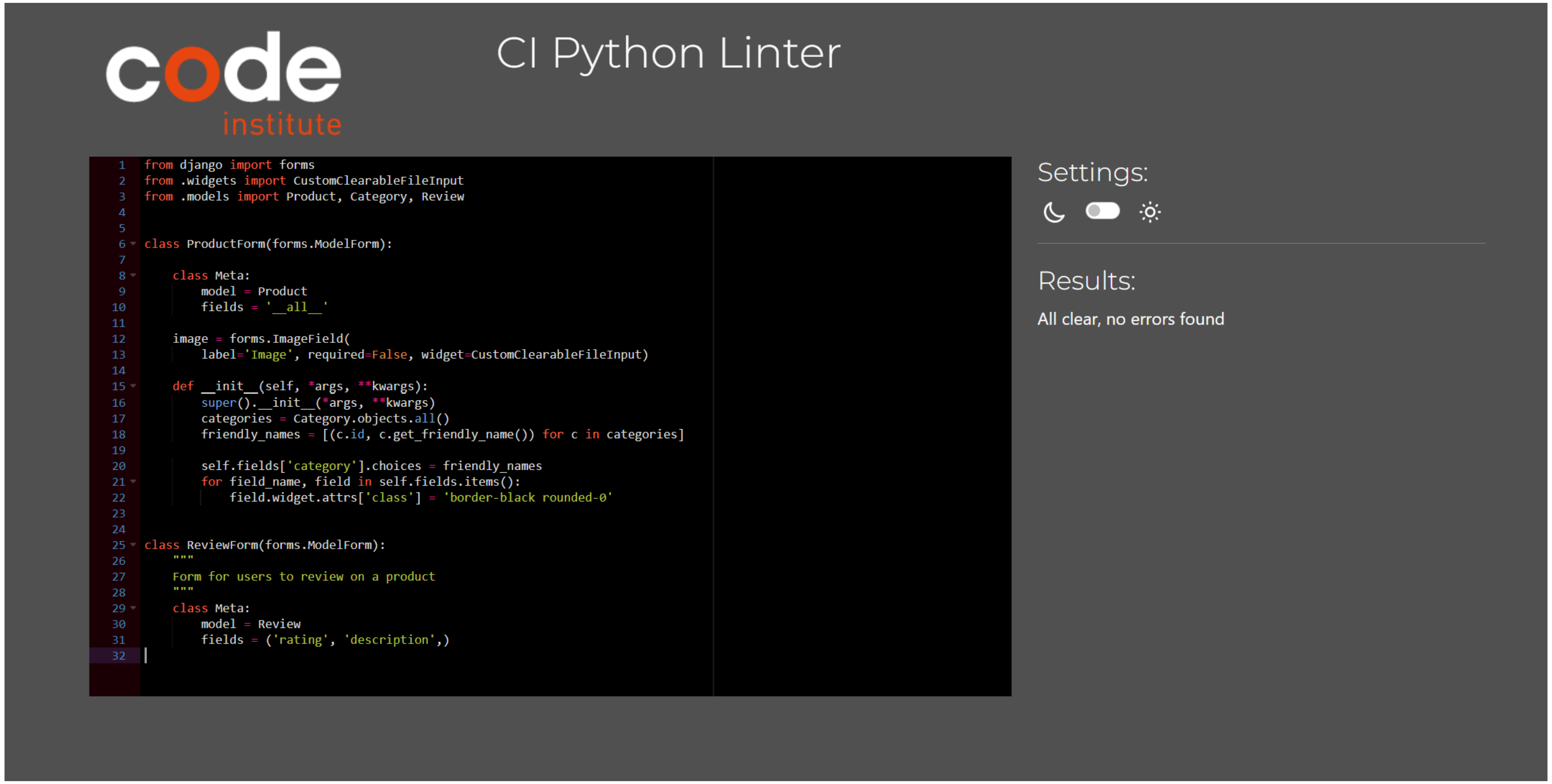


```
from django.apps import AppConfig


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

Results:

All clear, no errors found

Products App – forms.py

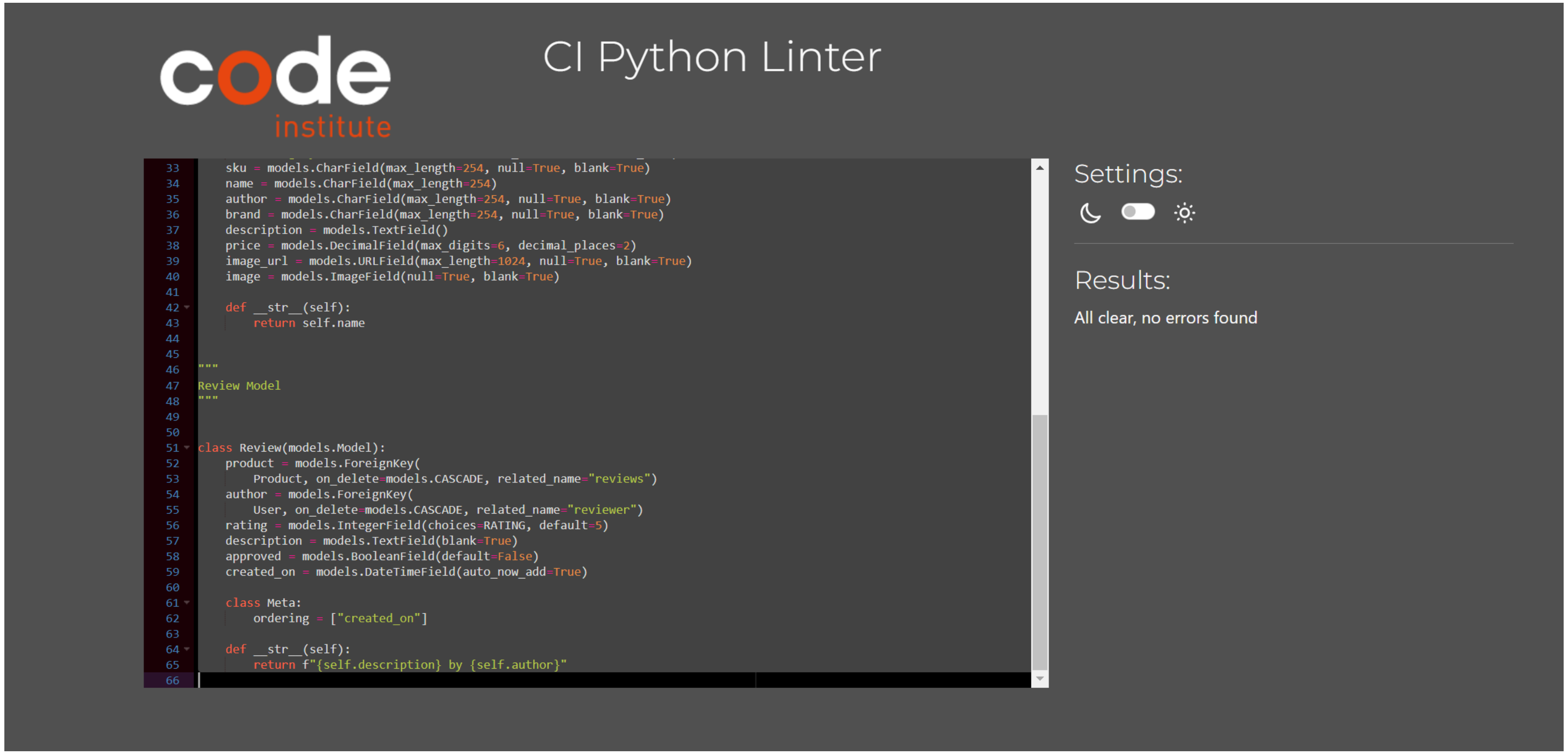


```python
from django import forms
from .widgets import CustomClearableFileInput
from .models import Product, Category, Review


class ProductForm(forms.ModelForm):

    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'


class ReviewForm(forms.ModelForm):
    """
    Form for users to review on a product
    """
    class Meta:
        model = Review
        fields = ('rating', 'description',)

```

Results:

All clear, no errors found

Products App – models.py


code institute

CI Python Linter

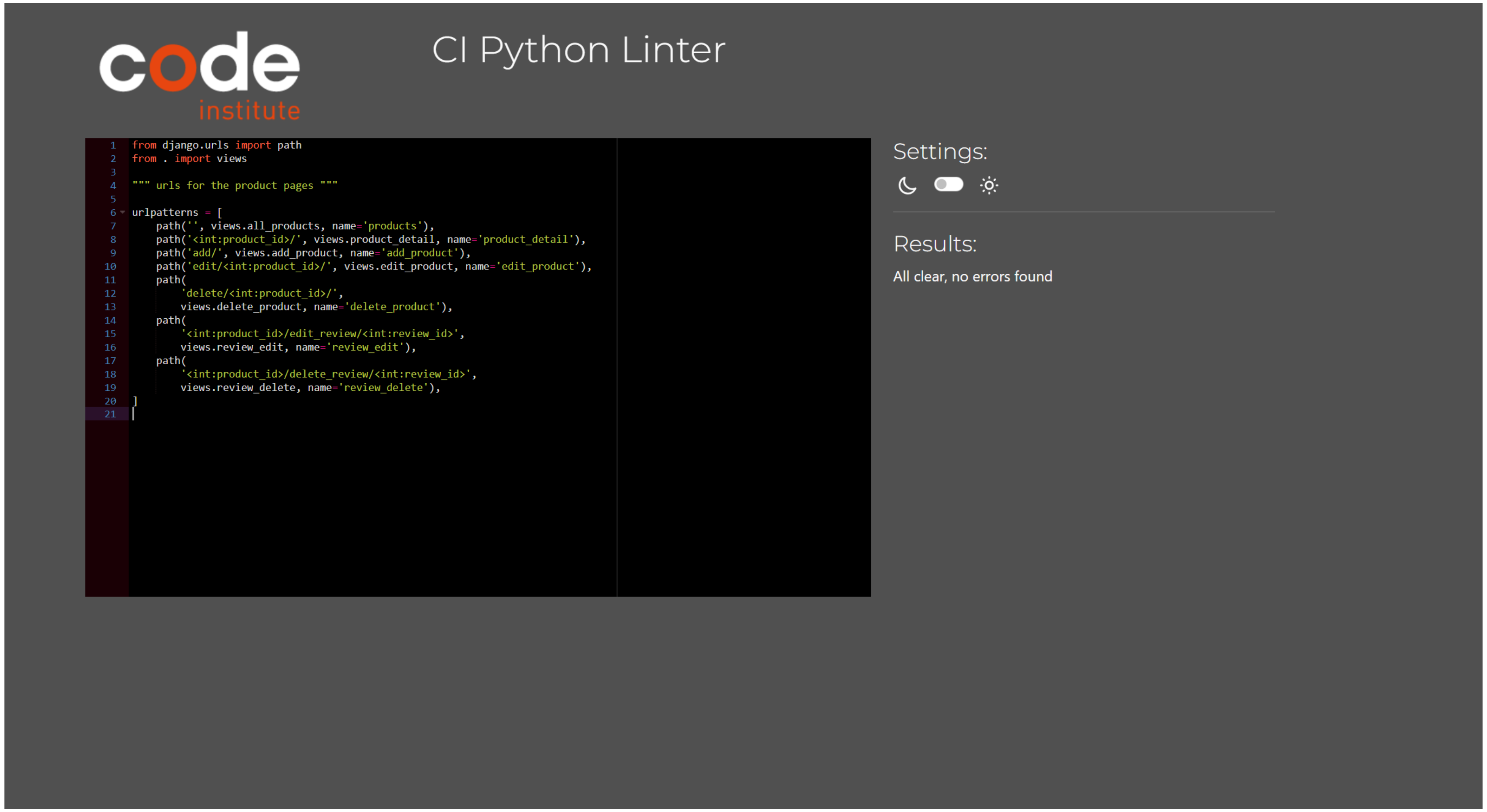
```
    sku = models.CharField(max_length=254, null=True, blank=True)
    name = models.CharField(max_length=254)
    author = models.CharField(max_length=254, null=True, blank=True)
    brand = models.CharField(max_length=254, null=True, blank=True)
    description = models.TextField()
    price = models.DecimalField(max_digits=6, decimal_places=2)
    image_url = models.URLField(max_length=1024, null=True, blank=True)
    image = models.ImageField(null=True, blank=True)

    def __str__(self):
        return self.name


"""
Review Model
"""


class Review(models.Model):
    product = models.ForeignKey(
        Product, on_delete=models.CASCADE, related_name="reviews")
    author = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name="reviewer")
    rating = models.IntegerField(choices=RATING, default=5)
    description = models.TextField(blank=True)
    approved = models.BooleanField(default=False)
    created_on = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ["created_on"]

    def __str__(self):
        return f"{self.description} by {self.author}"
```

Settings:

Results:

All clear, no errors found

Products App – urls.py



CI Python Linter

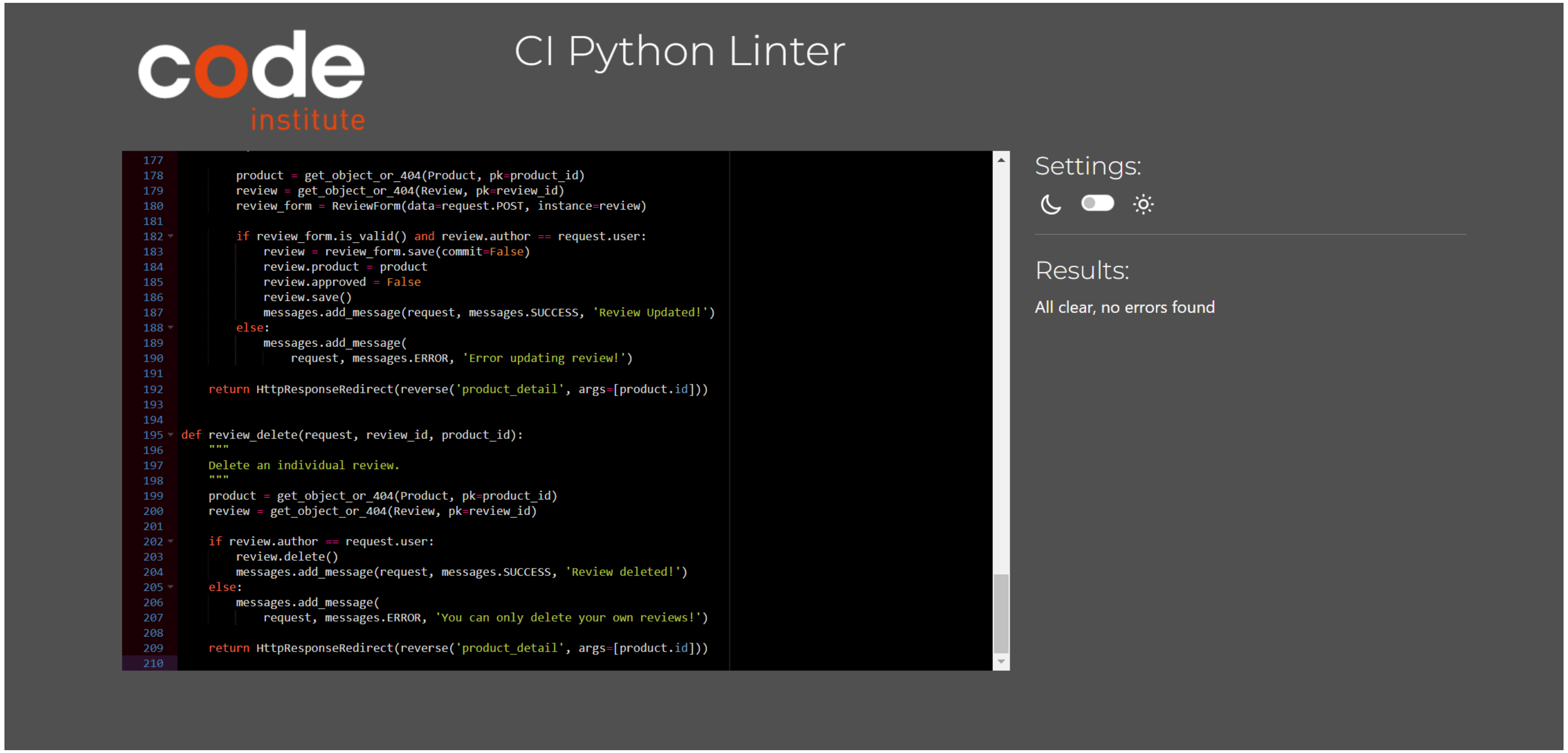
```
from django.urls import path
from . import views

""" urls for the product pages """

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(
        '<int:product_id>/edit_review/<int:review_id>',
        views.review_edit, name='review_edit'),
    path(
        '<int:product_id>/delete_review/<int:review_id>',
        views.review_delete, name='review_delete'),
]
```

Settings:

Results:

All clear, no errors found

Products App – views.py

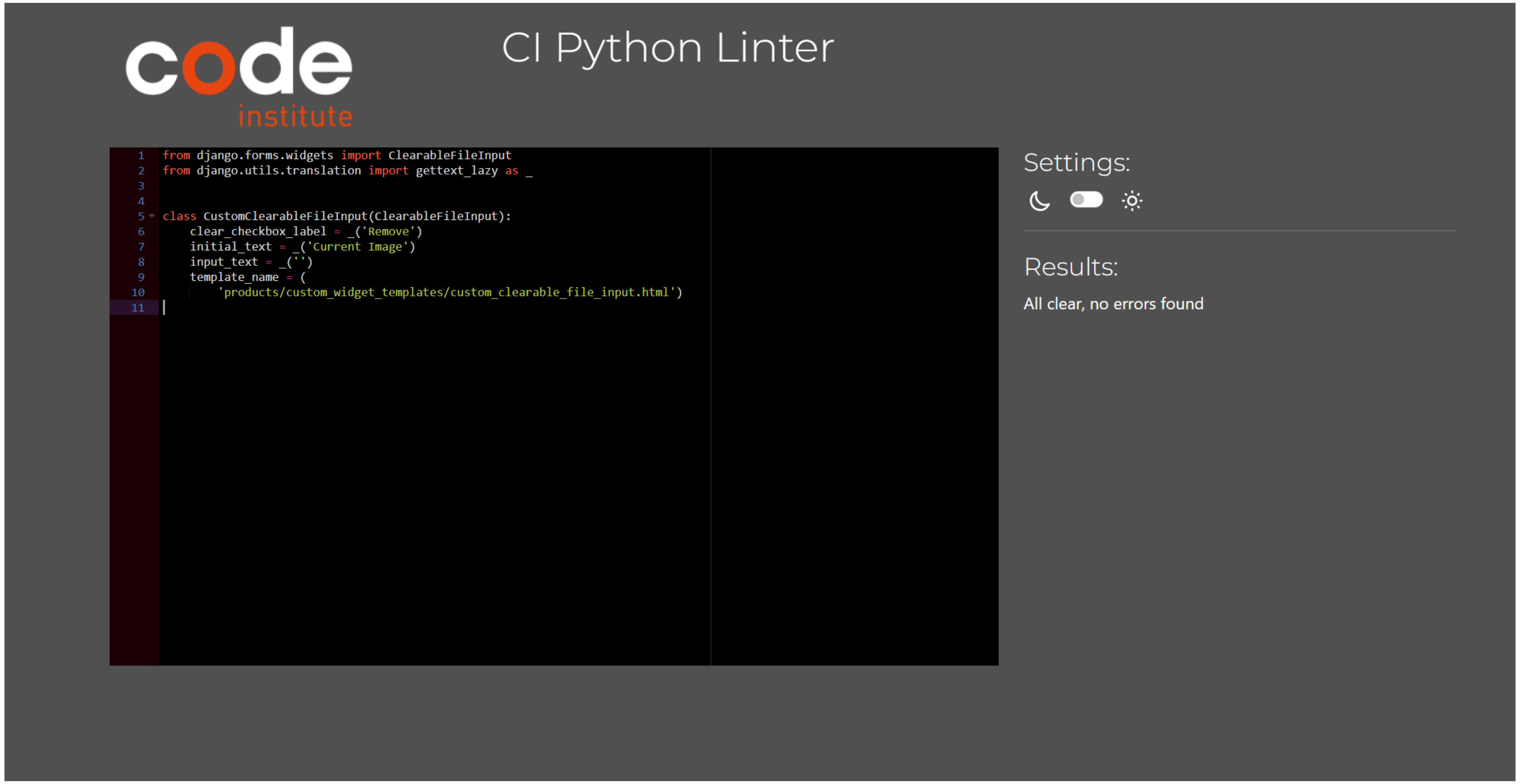


```
        product = get_object_or_404(Product, pk=product_id)
        review = get_object_or_404(Review, pk=review_id)
        review_form = ReviewForm(data=request.POST, instance=review)

        if review_form.is_valid() and review.author == request.user:
            review = review_form.save(commit=False)
            review.product = product
            review.approved = False
            review.save()
            messages.add_message(request, messages.SUCCESS, 'Review Updated!')
        else:
            messages.add_message(
                request, messages.ERROR, 'Error updating review!')

    return HttpResponseRedirect(reverse('product_detail', args=[product.id]))


def review_delete(request, review_id, product_id):
    """
    Delete an individual review.
    """
    product = get_object_or_404(Product, pk=product_id)
    review = get_object_or_404(Review, pk=review_id)

    if review.author == request.user:
        review.delete()
        messages.add_message(request, messages.SUCCESS, 'Review deleted!')
    else:
        messages.add_message(
            request, messages.ERROR, 'You can only delete your own reviews!')

    return HttpResponseRedirect(reverse('product_detail', args=[product.id]))
```

Results:

All clear, no errors found

Products App – widgets.py


CI Python Linter

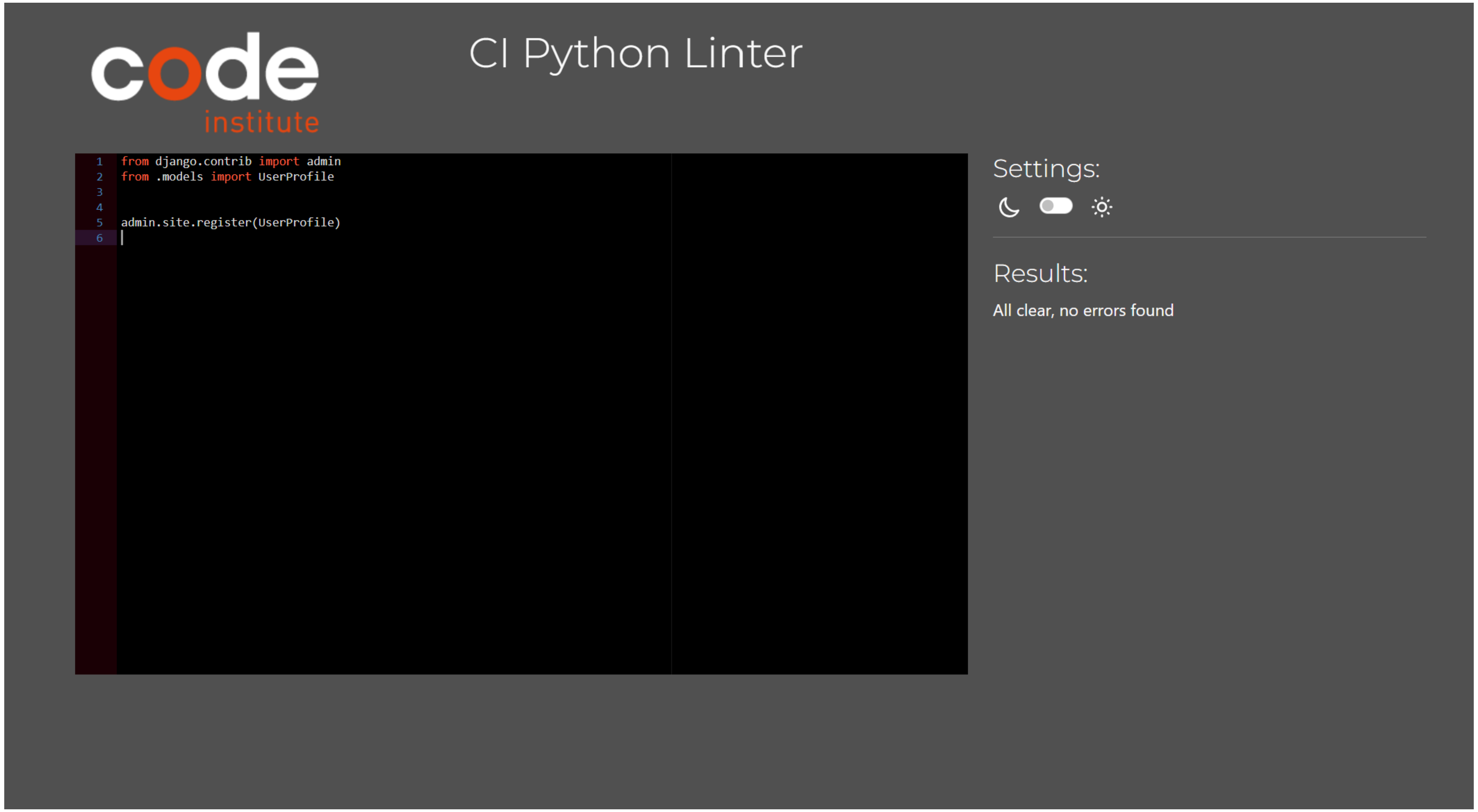
```
from django.forms.widgets import ClearableFileInput
from django.utils.translation import gettext_lazy as _


class CustomClearableFileInput(ClearableFileInput):
    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

Profiles App – admin.py



CI Python Linter

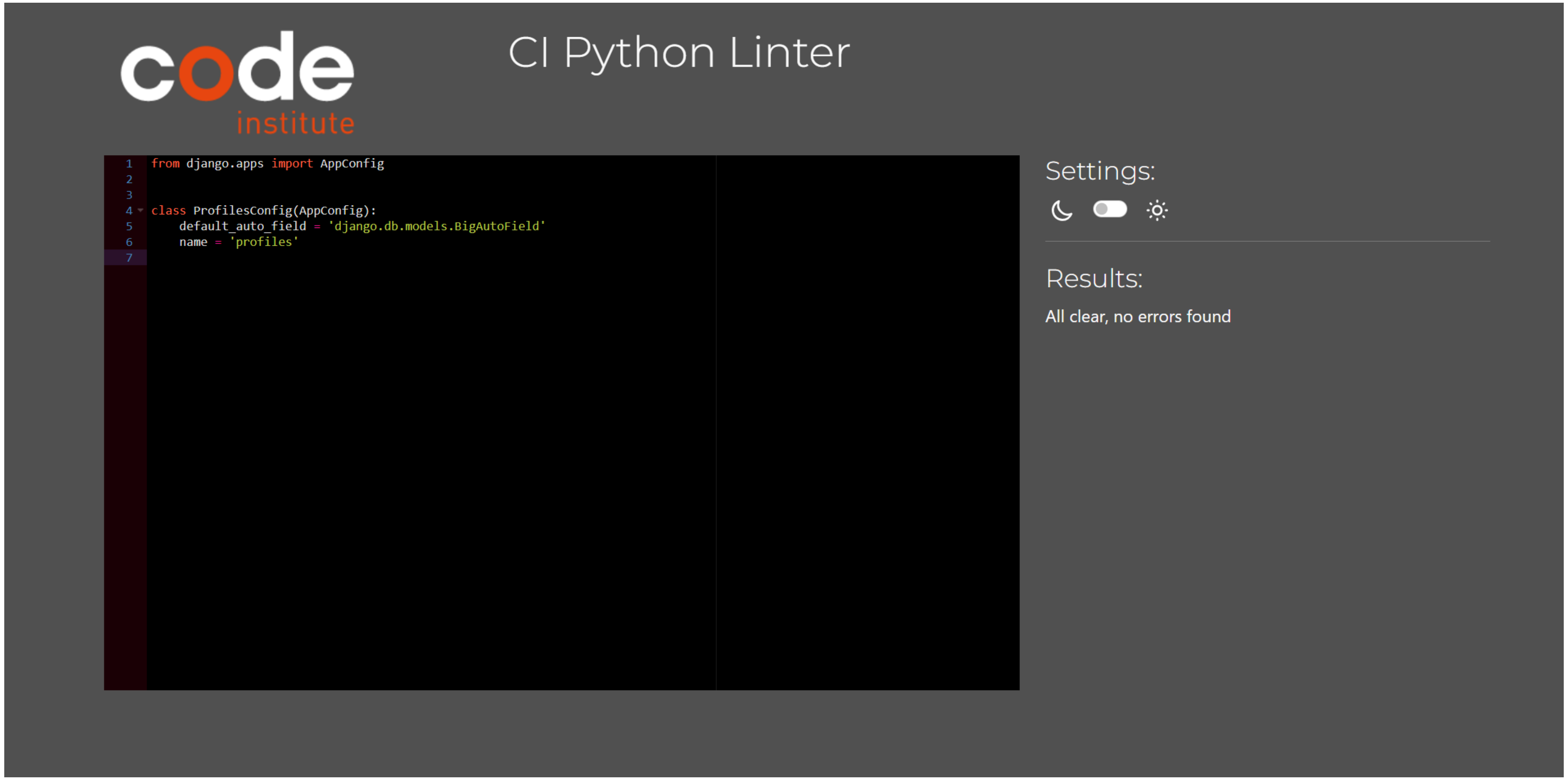
```
from django.contrib import admin
from .models import UserProfile


admin.site.register(UserProfile)
```

Settings:

Results:

All clear, no errors found

Profiles App – apps.py



CI Python Linter

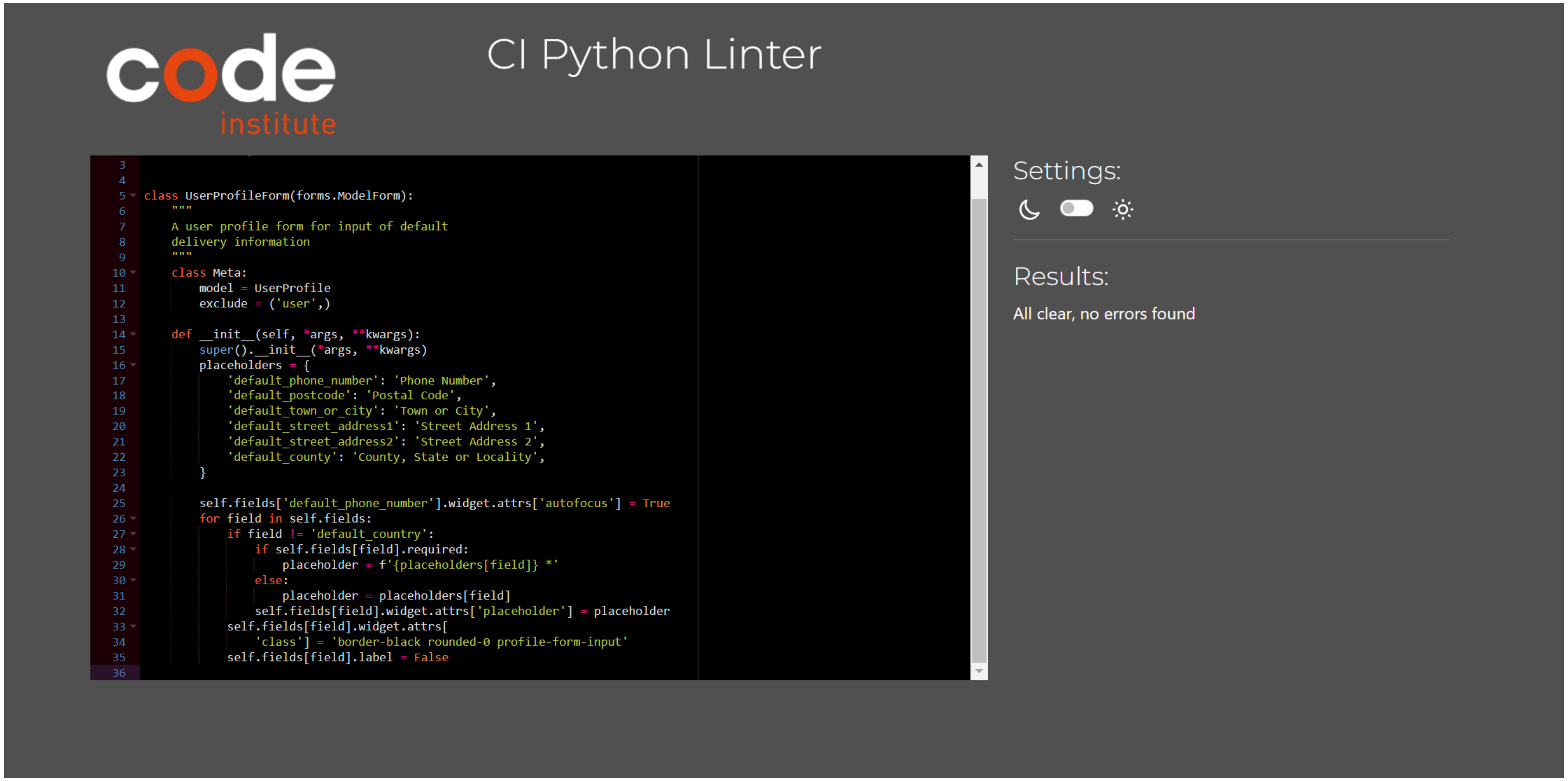
```
from django.apps import AppConfig


class ProfilesConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'profiles'
```

Settings:

Results:

All clear, no errors found

Profiles App – forms.py



CI Python Linter

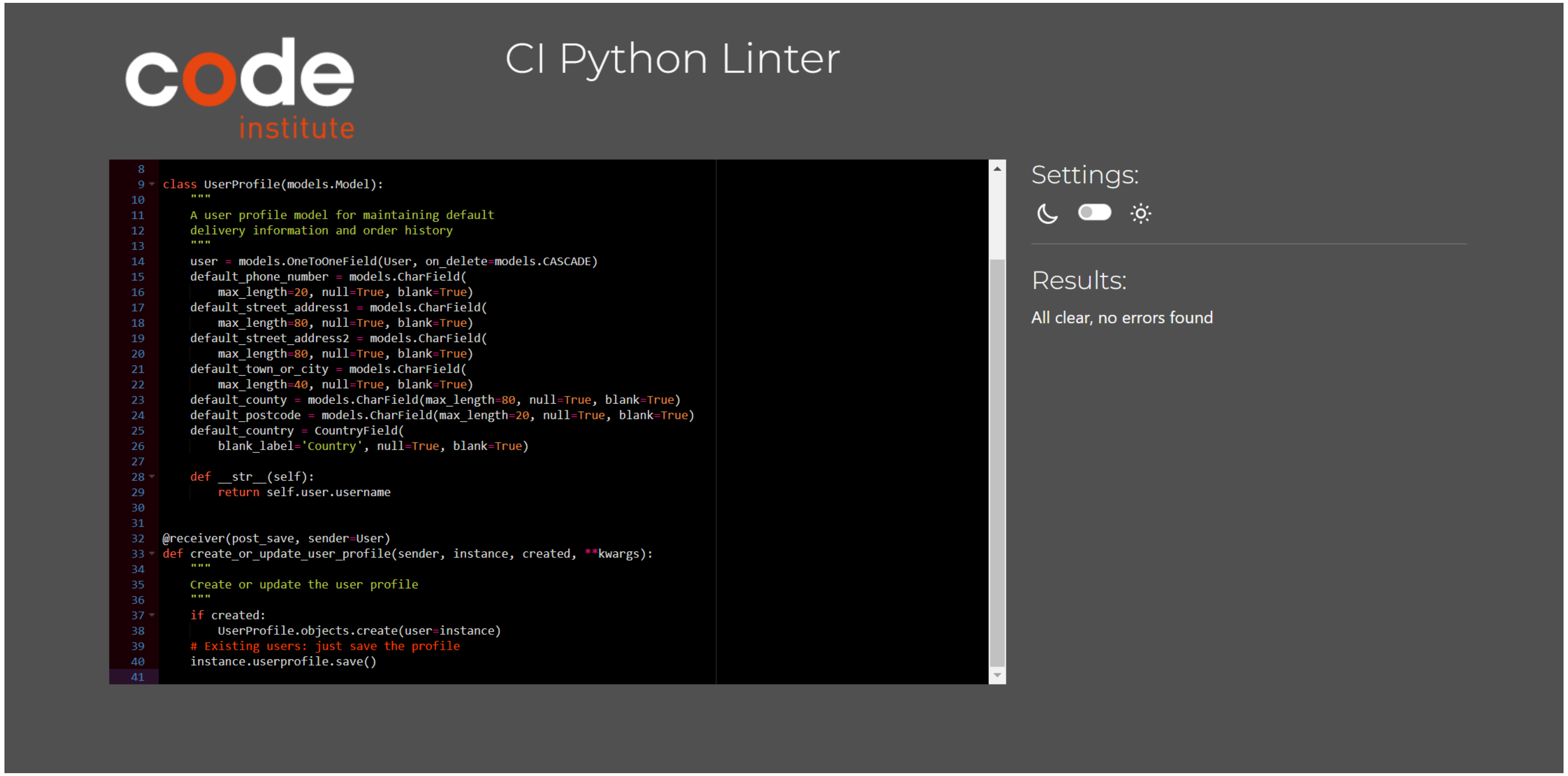
```python


class UserProfileForm(forms.ModelForm):
    """
    A user profile form for input of default
    delivery information
    """
    class Meta:
        model = UserProfile
        exclude = ('user',)

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        placeholders = {
            'default_phone_number': 'Phone Number',
            'default_postcode': 'Postal Code',
            'default_town_or_city': 'Town or City',
            'default_street_address1': 'Street Address 1',
            'default_street_address2': 'Street Address 2',
            'default_county': 'County, State or Locality',
        }

        self.fields['default_phone_number'].widget.attrs['autofocus'] = True
        for field in self.fields:
            if field != 'default_country':
                if self.fields[field].required:
                    placeholder = f'{placeholders[field]} *'
                else:
                    placeholder = placeholders[field]
                self.fields[field].widget.attrs['placeholder'] = placeholder
            self.fields[field].widget.attrs[
                'class'] = 'border-black rounded-0 profile-form-input'
            self.fields[field].label = False

```

Settings:

Results:

All clear, no errors found

Profiles App – models.py



CI Python Linter

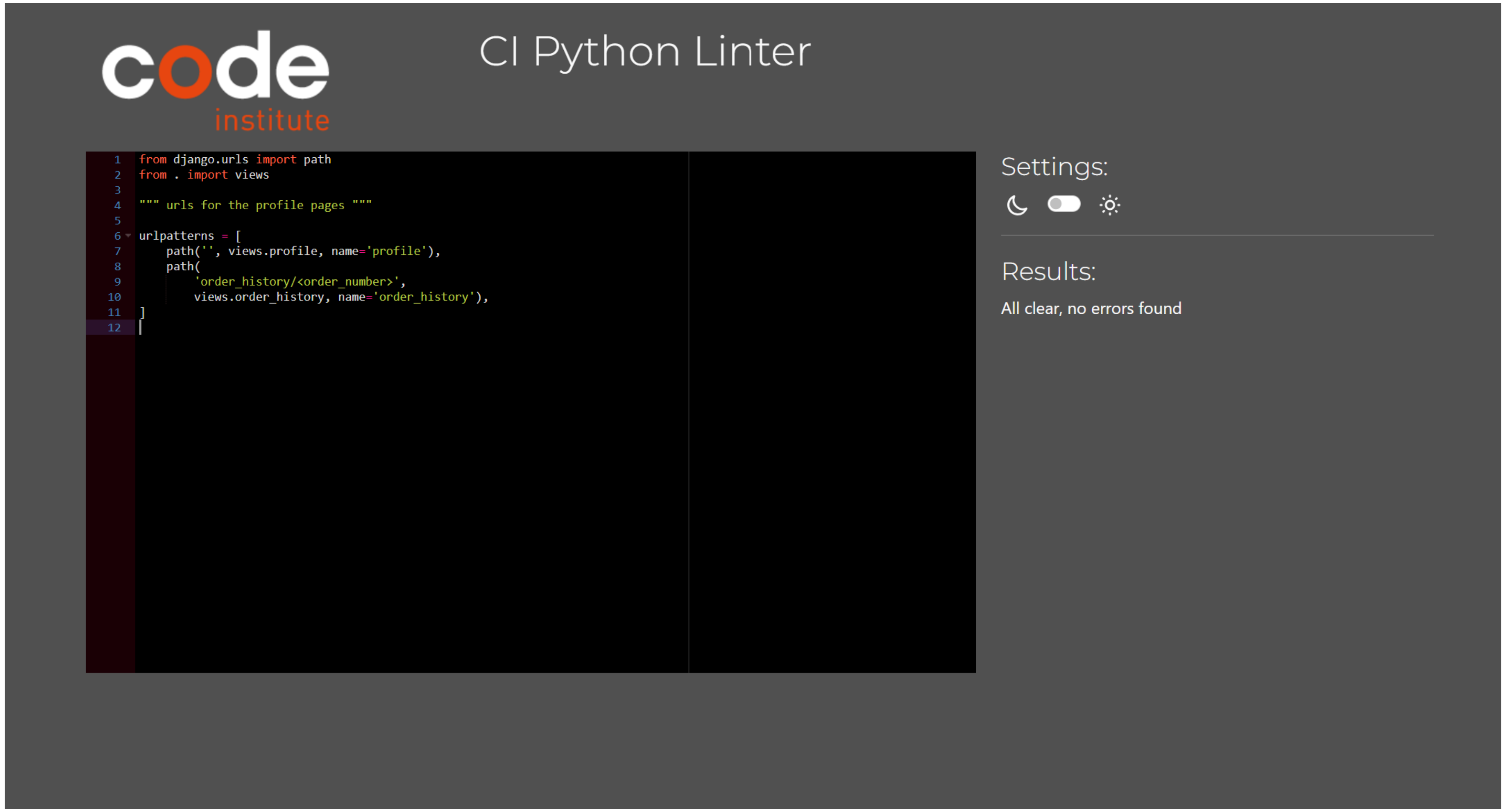
```python

class UserProfile(models.Model):
    """
    A user profile model for maintaining default
    delivery information and order history
    """
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    default_phone_number = models.CharField(
        max_length=20, null=True, blank=True)
    default_street_address1 = models.CharField(
        max_length=80, null=True, blank=True)
    default_street_address2 = models.CharField(
        max_length=80, null=True, blank=True)
    default_town_or_city = models.CharField(
        max_length=40, null=True, blank=True)
    default_county = models.CharField(max_length=80, null=True, blank=True)
    default_postcode = models.CharField(max_length=20, null=True, blank=True)
    default_country = CountryField(
        blank_label='Country', null=True, blank=True)

    def __str__(self):
        return self.user.username


@receiver(post_save, sender=User)
def create_or_update_user_profile(sender, instance, created, **kwargs):
    """
    Create or update the user profile
    """
    if created:
        UserProfile.objects.create(user=instance)
    # Existing users: just save the profile
    instance.userprofile.save()

```

Settings:

Results:

All clear, no errors found

Profiles App – urls.py



CI Python Linter

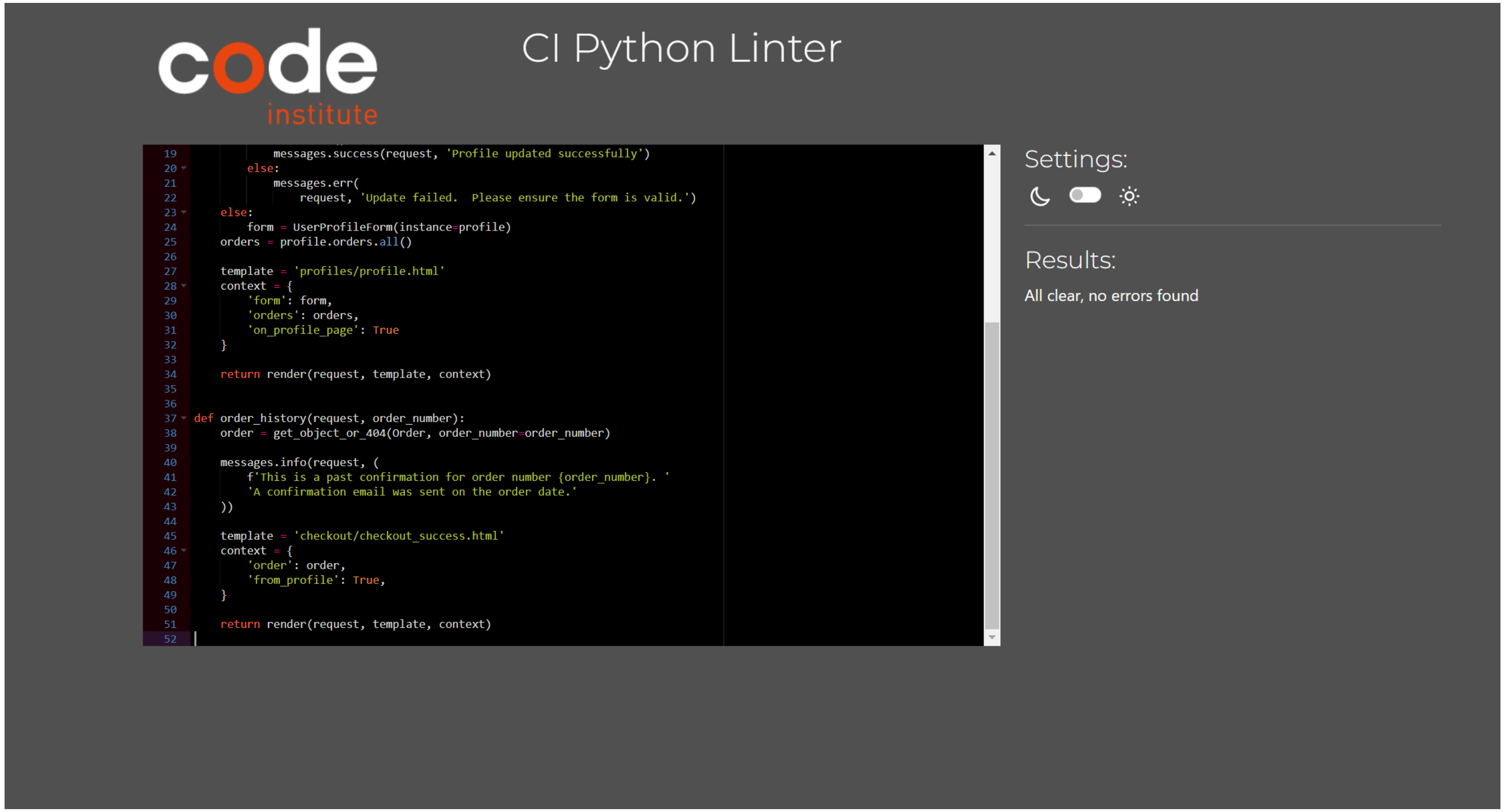
```
from django.urls import path
from . import views

""" urls for the profile pages """

urlpatterns = [
    path('', views.profile, name='profile'),
    path(
        'order_history/<order_number>',
        views.order_history, name='order_history'),
]
```

Settings:

Results:

All clear, no errors found

Profiles App – views.py



CI Python Linter

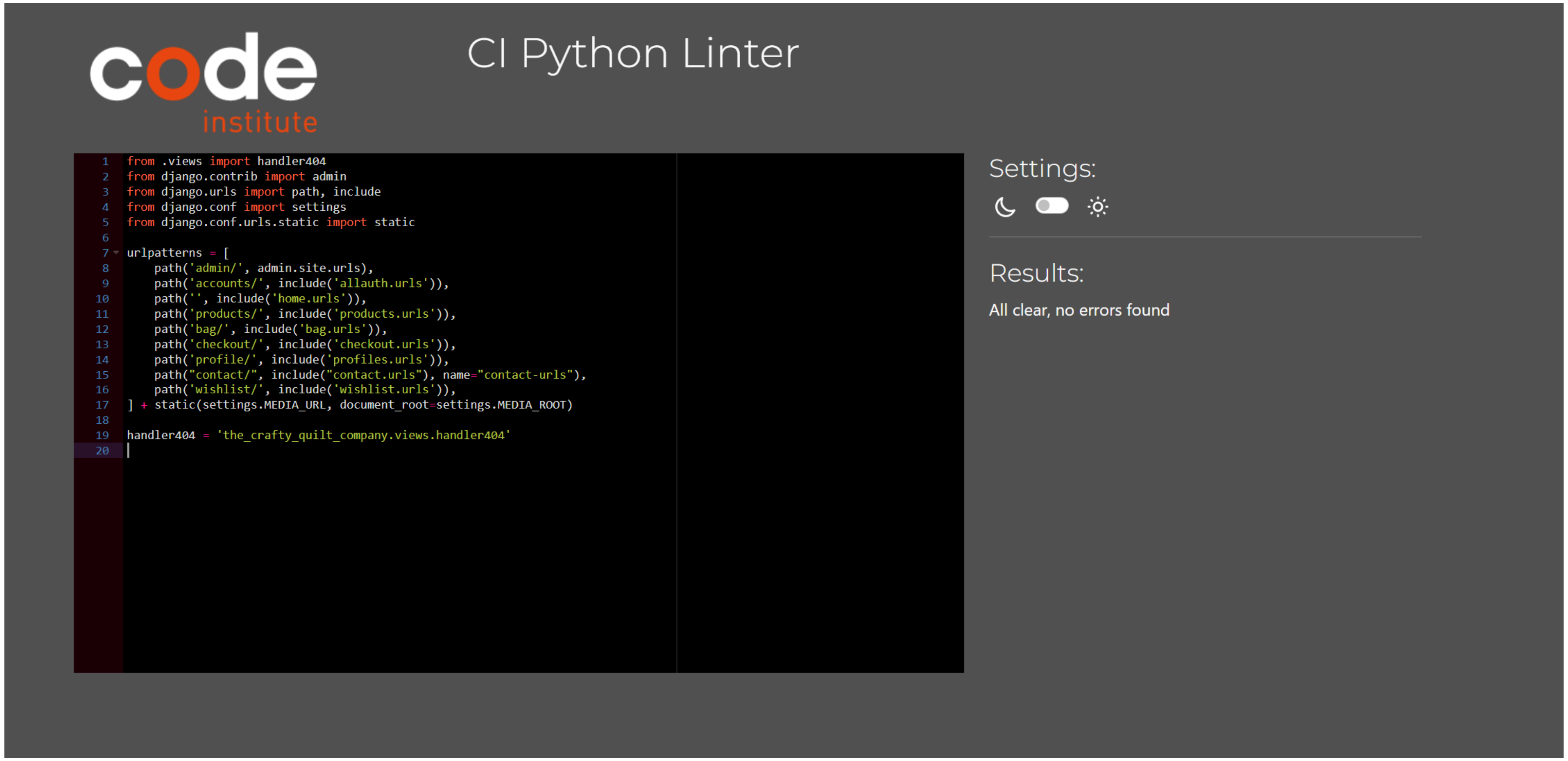
```
                messages.success(request, 'Profile updated successfully')
            else:
                messages.err(
                    request, 'Update failed.  Please ensure the form is valid.')
        else:
            form = UserProfileForm(instance=profile)
        orders = profile.orders.all()

        template = 'profiles/profile.html'
        context = {
            'form': form,
            'orders': orders,
            'on_profile_page': True
        }

        return render(request, template, context)


    def order_history(request, order_number):
        order = get_object_or_404(Order, order_number=order_number)

        messages.info(request, (
            f'This is a past confirmation for order number {order_number}. '
            'A confirmation email was sent on the order date.'
        ))

        template = 'checkout/checkout_success.html'
        context = {
            'order': order,
            'from_profile': True,
        }

        return render(request, template, context)
```

Settings:

Results:

All clear, no errors found

The Crafty Quilt Company App – urls.py



CI Python Linter

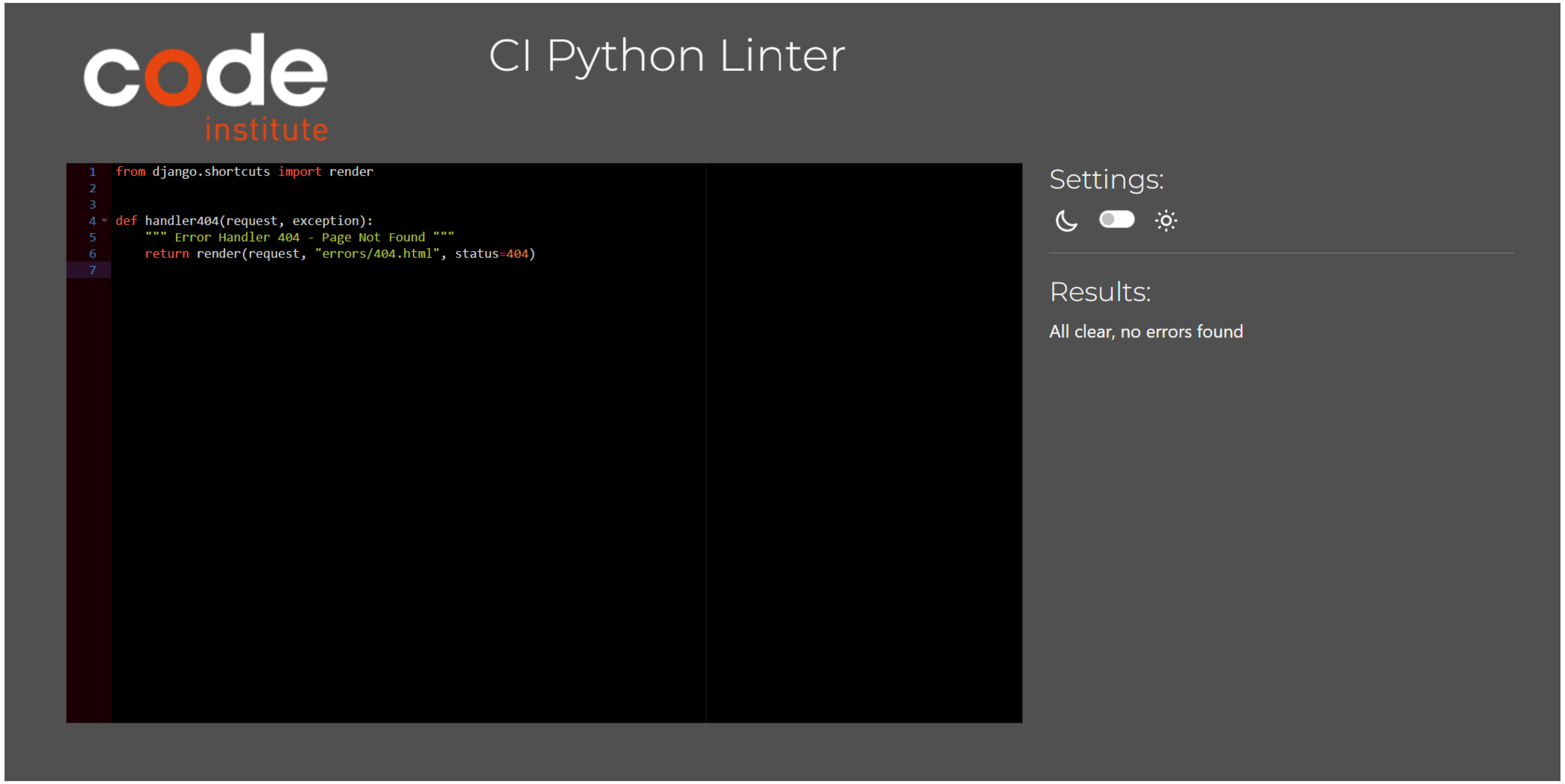
```python
from .views import handler404
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('allauth.urls')),
    path('', include('home.urls')),
    path('products/', include('products.urls')),
    path('bag/', include('bag.urls')),
    path('checkout/', include('checkout.urls')),
    path('profile/', include('profiles.urls')),
    path("contact/", include("contact.urls"), name="contact-urls"),
    path('wishlist/', include('wishlist.urls')),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

handler404 = 'the_crafty_quilt_company.views.handler404'
```

Settings:

Results:

All clear, no errors found

The Crafty Quilt Company App – views.py


CI Python Linter

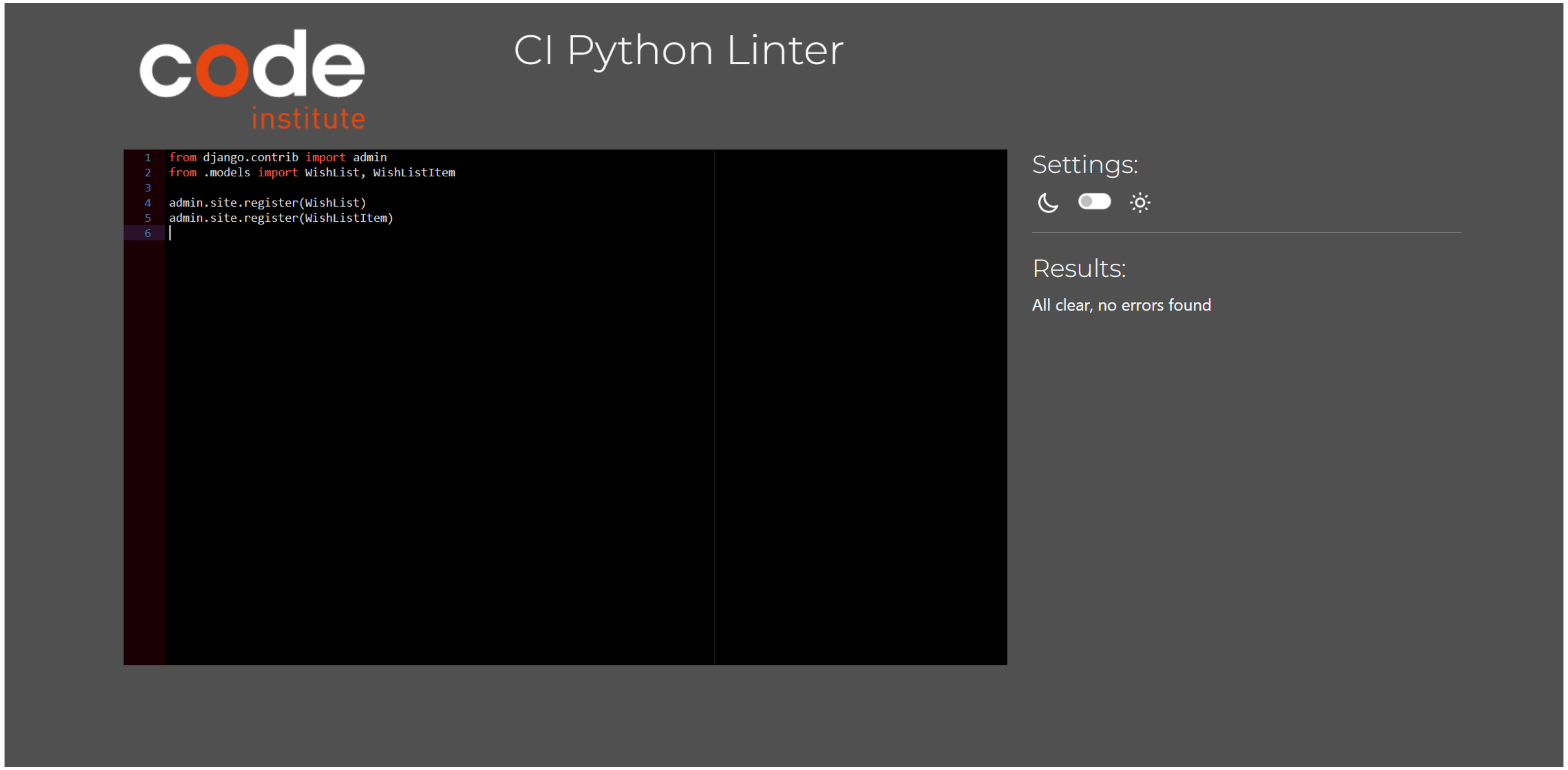
```python
from django.shortcuts import render


def handler404(request, exception):
    """ Error Handler 404 - Page Not Found """
    return render(request, "errors/404.html", status=404)

```

Settings:

Results:

All clear, no errors found

Wishlist App – admin.py



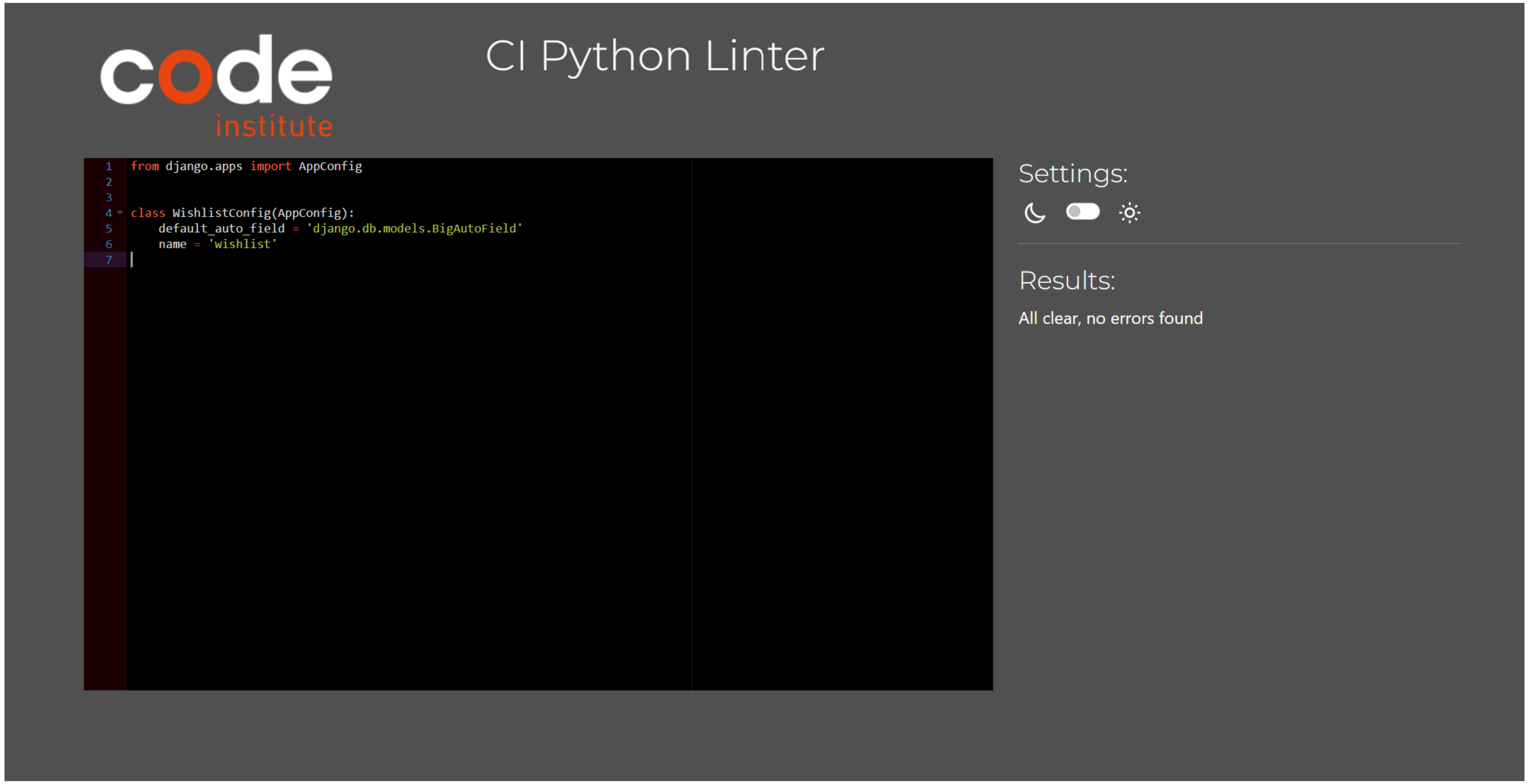
```
from django.contrib import admin
from .models import WishList, WishListItem

admin.site.register(WishList)
admin.site.register(WishListItem)
```

Settings:

Results:

All clear, no errors found

Wishlist App – apps.py



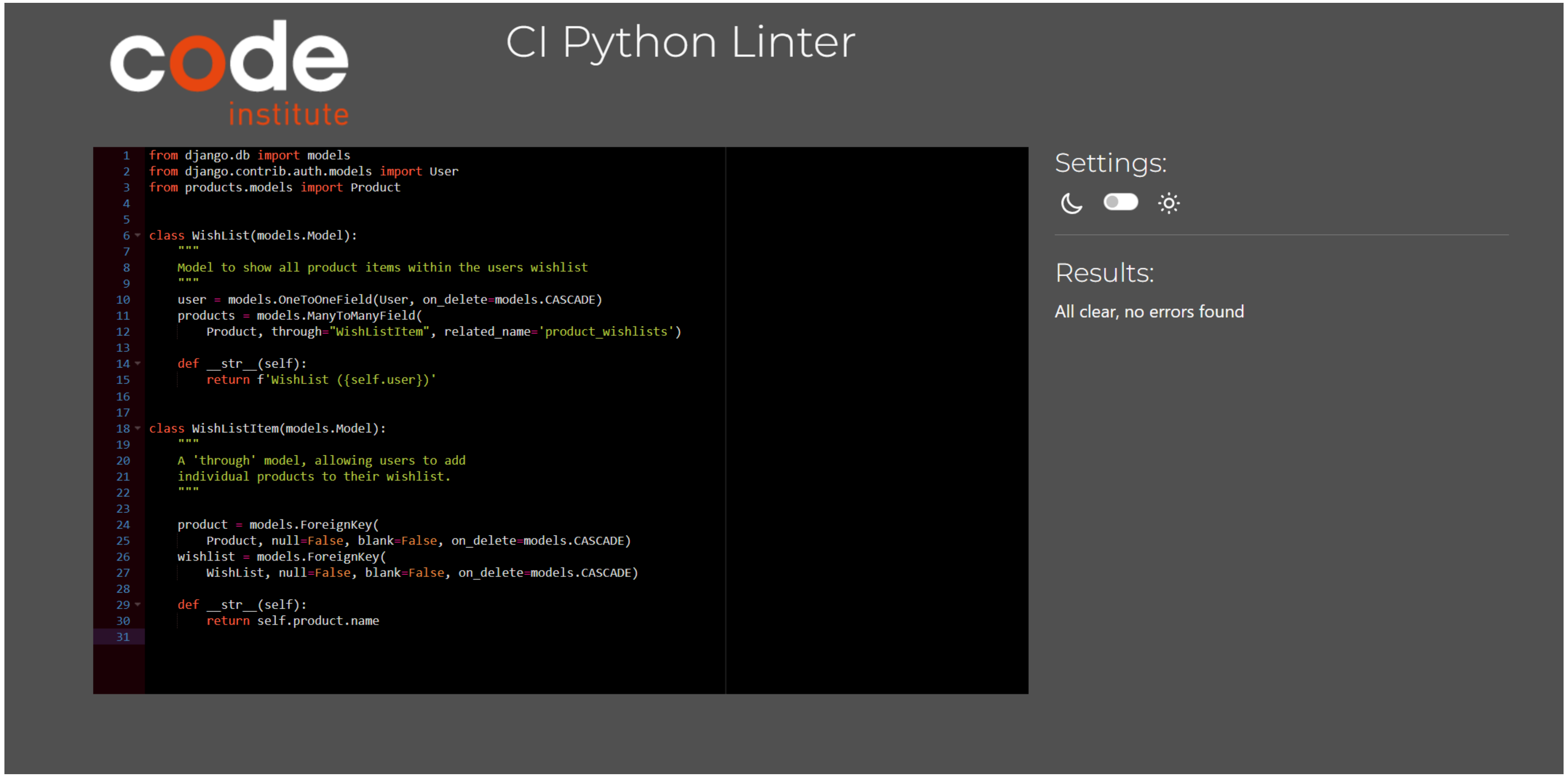
CI Python Linter

```python
from django.apps import AppConfig


class WishlistConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'wishlist'

```

Results:

All clear, no errors found

Wishlist App – models.py



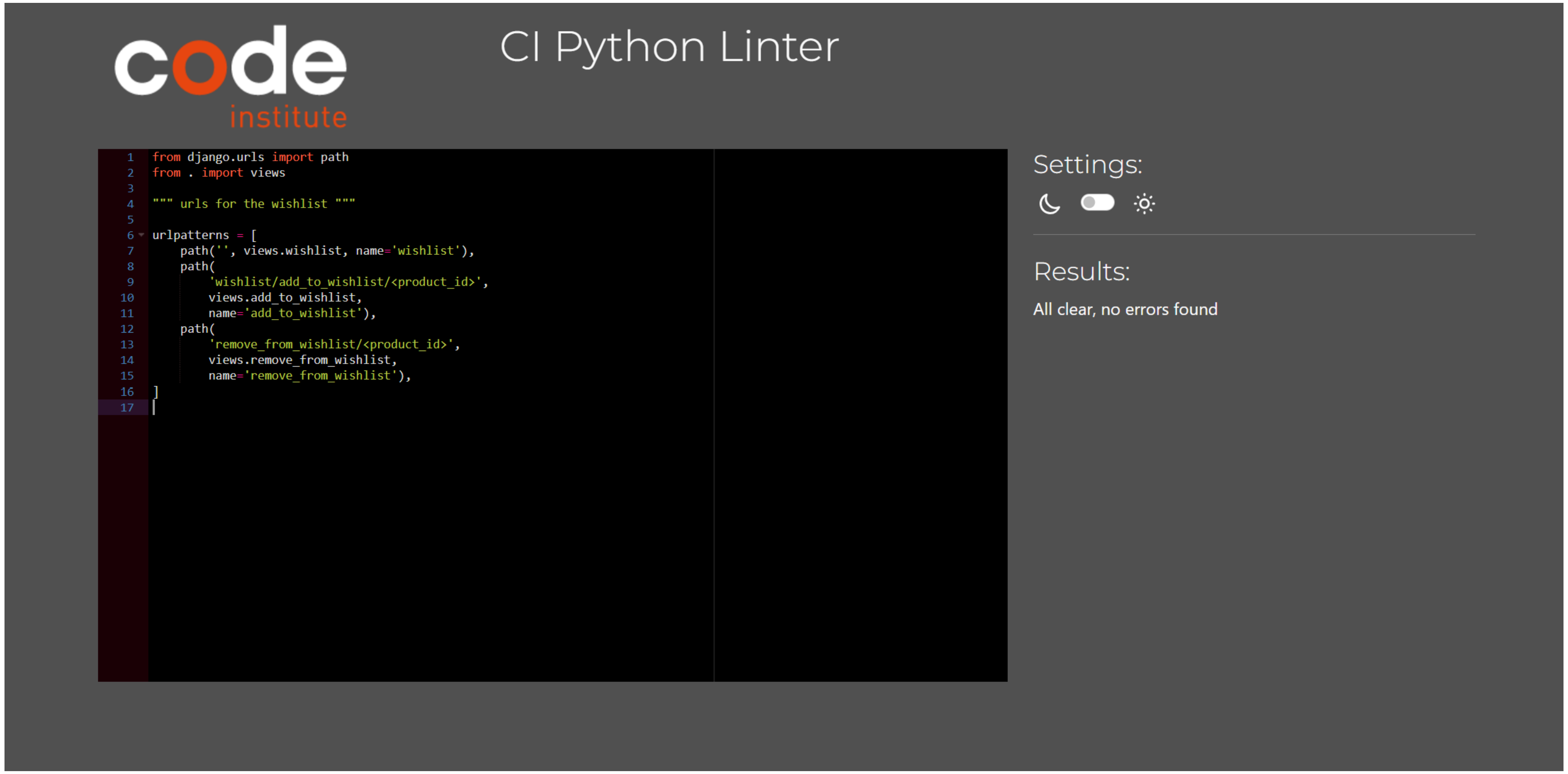
```python
from django.db import models
from django.contrib.auth.models import User
from products.models import Product


class WishList(models.Model):
    """
    Model to show all product items within the users wishlist
    """
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    products = models.ManyToManyField(
        Product, through="WishListItem", related_name='product_wishlists')

    def __str__(self):
        return f'WishList ({self.user})'


class WishListItem(models.Model):
    """
    A 'through' model, allowing users to add
    individual products to their wishlist.
    """

    product = models.ForeignKey(
        Product, null=False, blank=False, on_delete=models.CASCADE)
    wishlist = models.ForeignKey(
        WishList, null=False, blank=False, on_delete=models.CASCADE)

    def __str__(self):
        return self.product.name
```

Settings:

Results:

All clear, no errors found

Wishlist App – urls.py

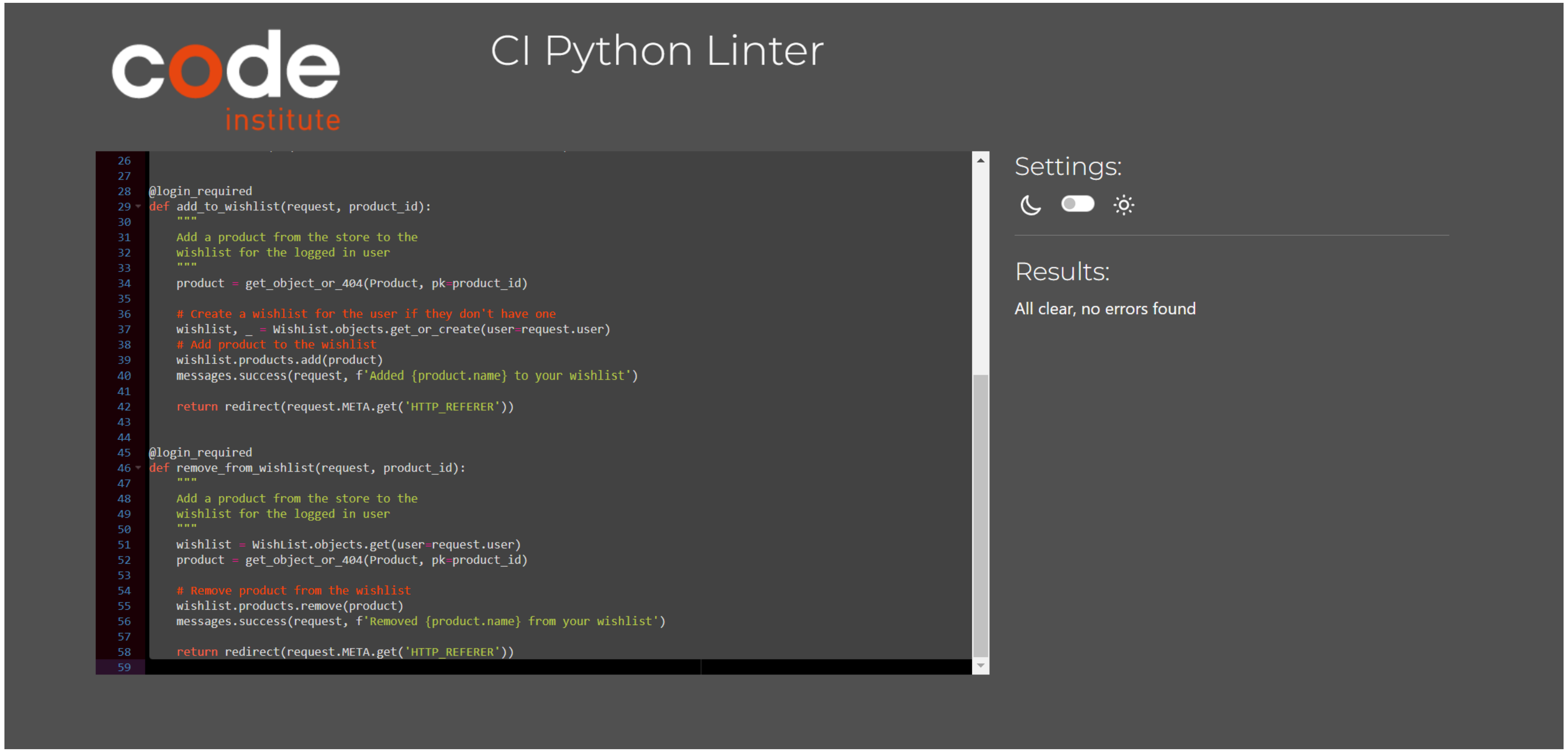


```
from django.urls import path
from . import views

""" urls for the wishlist """

urlpatterns = [
    path('', views.wishlist, name='wishlist'),
    path(
        'wishlist/add_to_wishlist/<product_id>',
        views.add_to_wishlist,
        name='add_to_wishlist'),
    path(
        'remove_from_wishlist/<product_id>',
        views.remove_from_wishlist,
        name='remove_from_wishlist'),
]
```

Results:

All clear, no errors found

Wishlist App – views.py



CI Python Linter

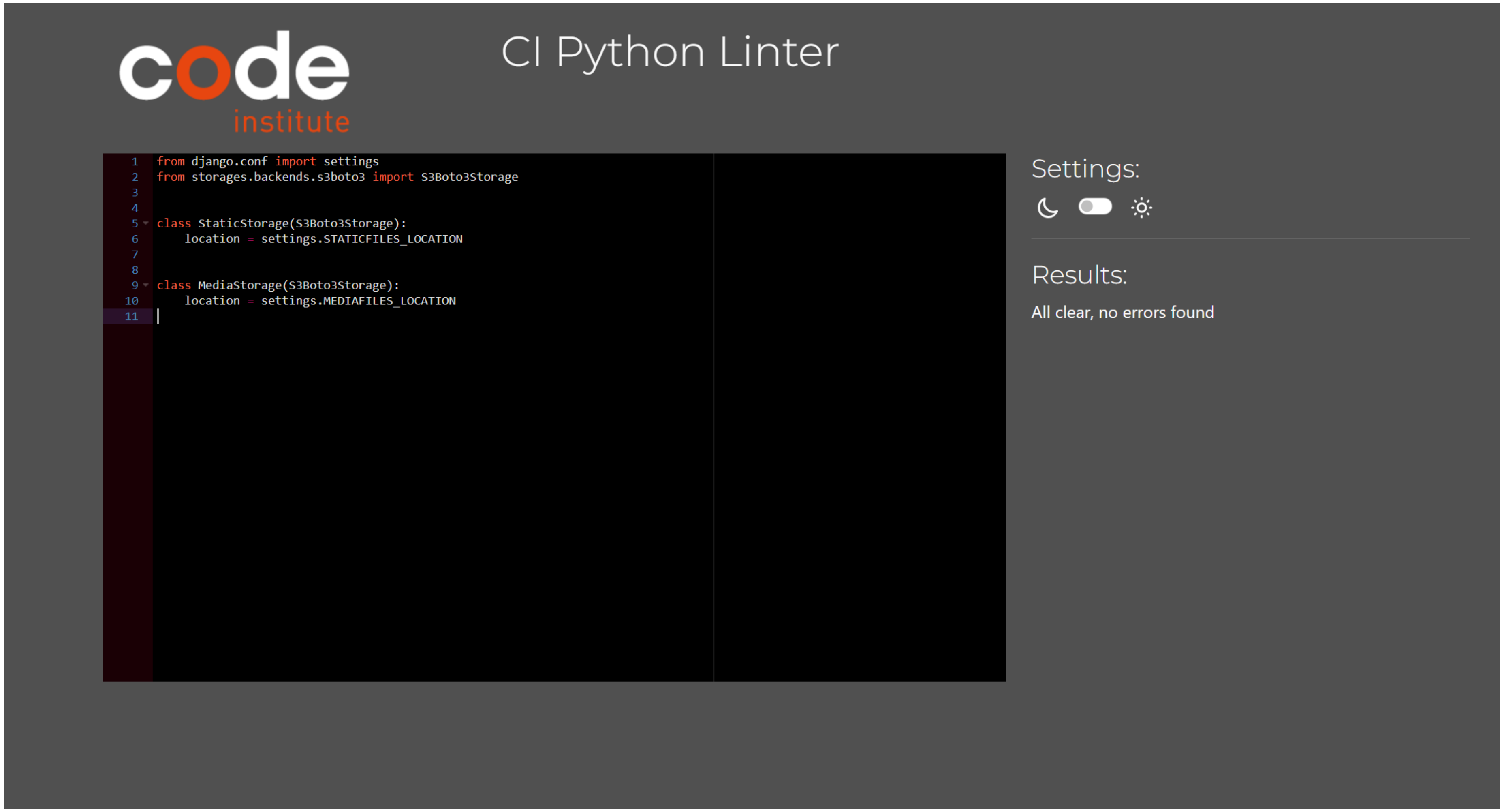
```python


@login_required
def add_to_wishlist(request, product_id):
    """
    Add a product from the store to the
    wishlist for the logged in user
    """
    product = get_object_or_404(Product, pk=product_id)

    # Create a wishlist for the user if they don't have one
    wishlist, _ = WishList.objects.get_or_create(user=request.user)
    # Add product to the wishlist
    wishlist.products.add(product)
    messages.success(request, f'Added {product.name} to your wishlist')

    return redirect(request.META.get('HTTP_REFERER'))


@login_required
def remove_from_wishlist(request, product_id):
    """
    Add a product from the store to the
    wishlist for the logged in user
    """
    wishlist = WishList.objects.get(user=request.user)
    product = get_object_or_404(Product, pk=product_id)

    # Remove product from the wishlist
    wishlist.products.remove(product)
    messages.success(request, f'Removed {product.name} from your wishlist')

    return redirect(request.META.get('HTTP_REFERER'))

```

Settings:

Results:

All clear, no errors found

custom_storages.py



```python
from django.conf import settings
from storages.backends.s3boto3 import S3Boto3Storage


class StaticStorage(S3Boto3Storage):
    location = settings.STATICFILES_LOCATION


class MediaStorage(S3Boto3Storage):
    location = settings.MEDIAFILES_LOCATION
```

Results:

All clear, no errors found