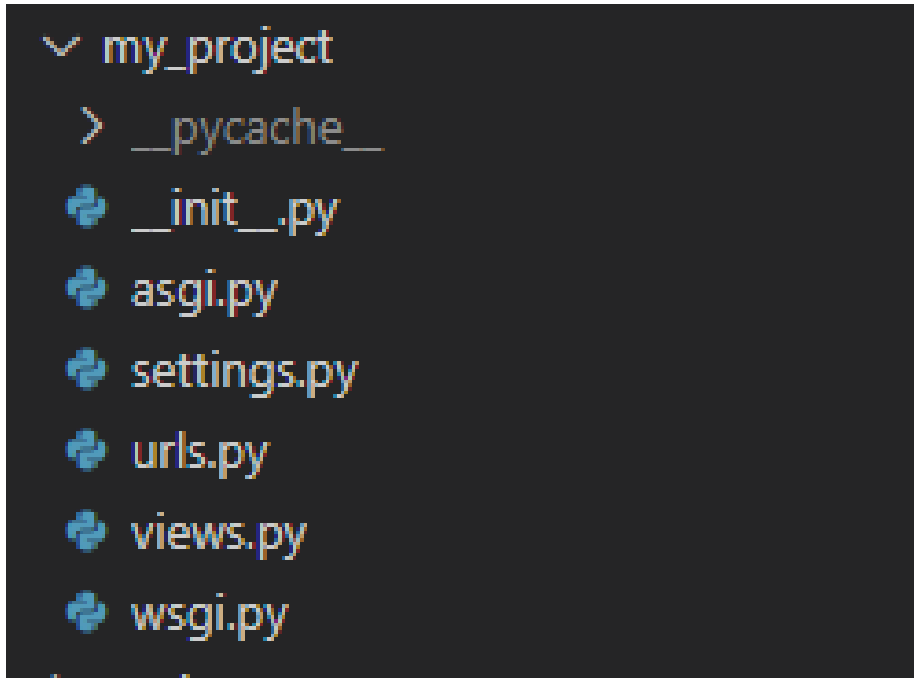
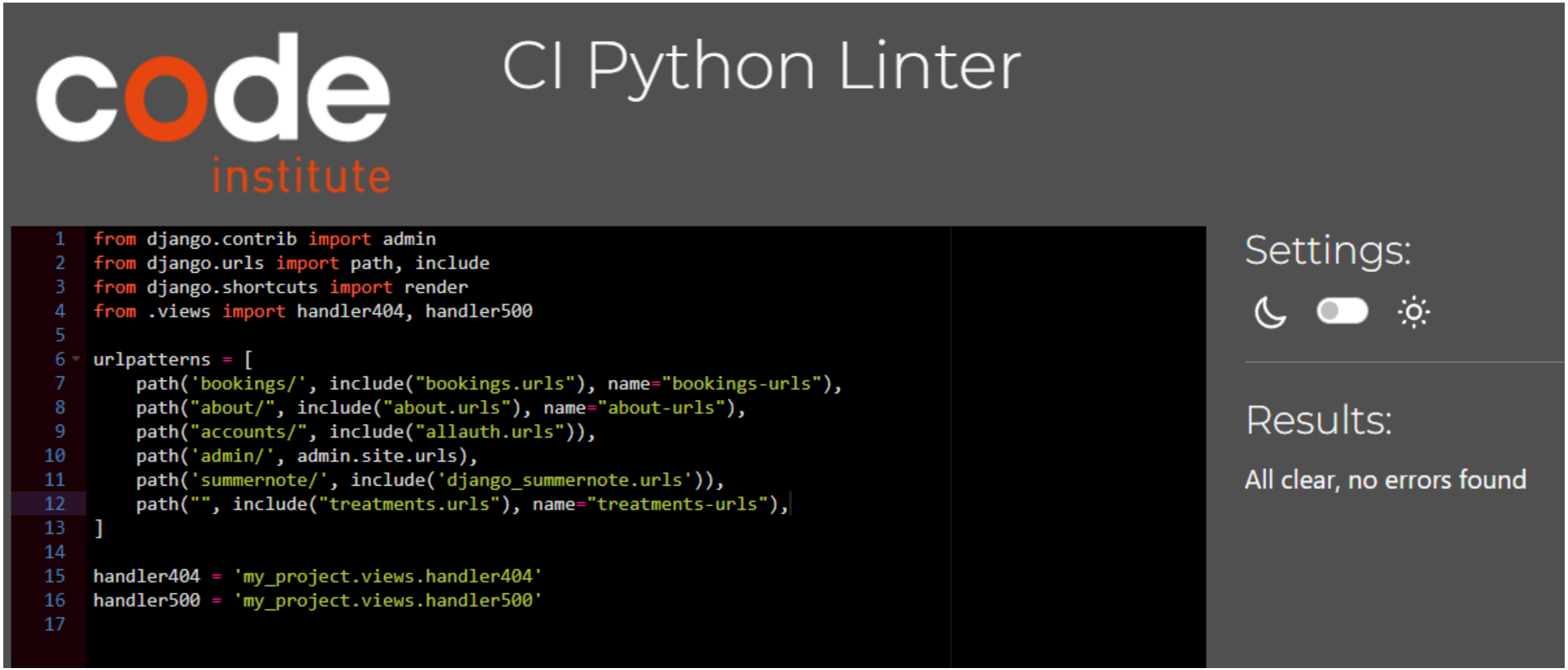
my_project

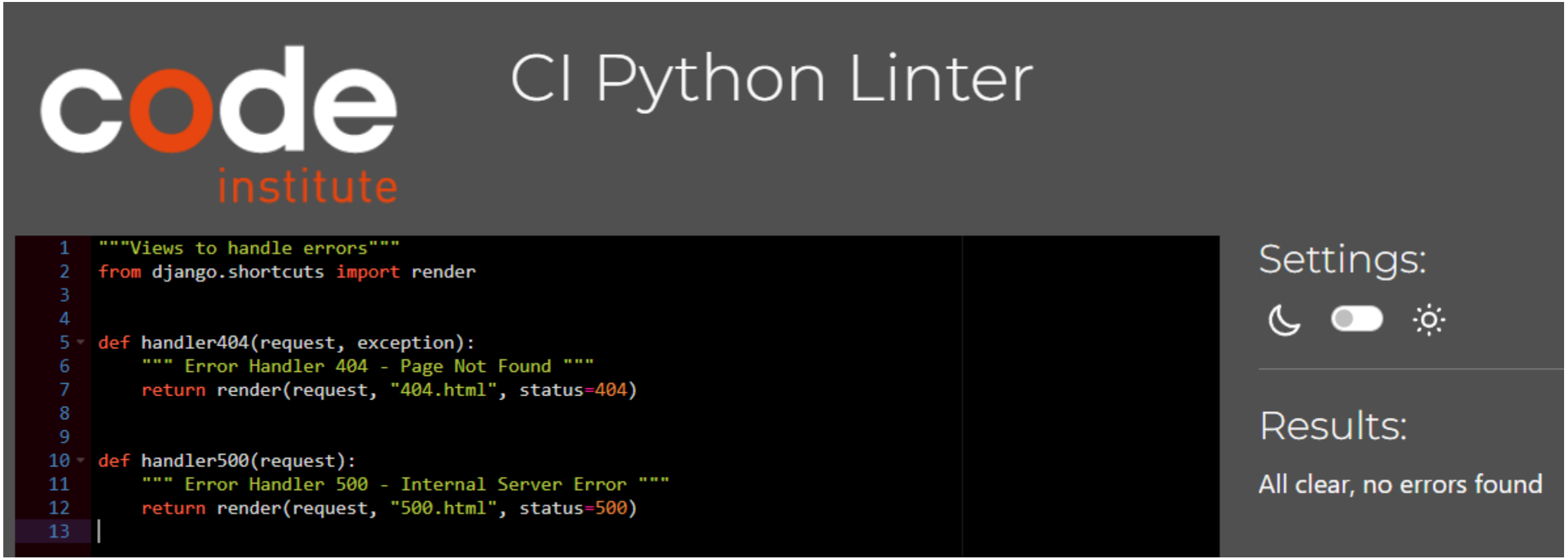

my_project
__pycache__
__init__.py
asgi.py
settings.py
urls.py
views.py
wsgi.py

urls.py

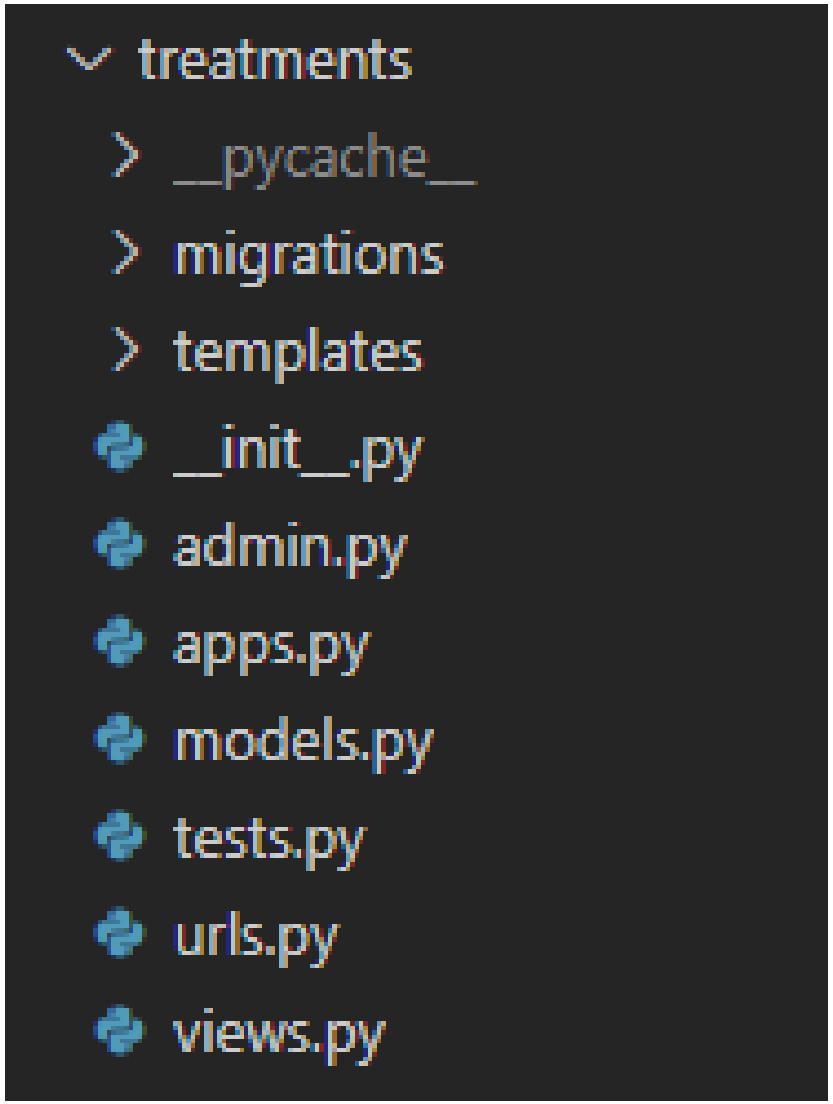



```python
from django.contrib import admin
from django.urls import path, include
from django.shortcuts import render
from .views import handler404, handler500

urlpatterns = [
    path('bookings/', include("bookings.urls"), name="bookings-urls"),
    path("about/", include("about.urls"), name="about-urls"),
    path("accounts/", include("allauth.urls")),
    path('admin/', admin.site.urls),
    path('summernote/', include('django_summernote.urls')),
    path("", include("treatments.urls"), name="treatments-urls"),
]

handler404 = 'my_project.views.handler404'
handler500 = 'my_project.views.handler500'
```

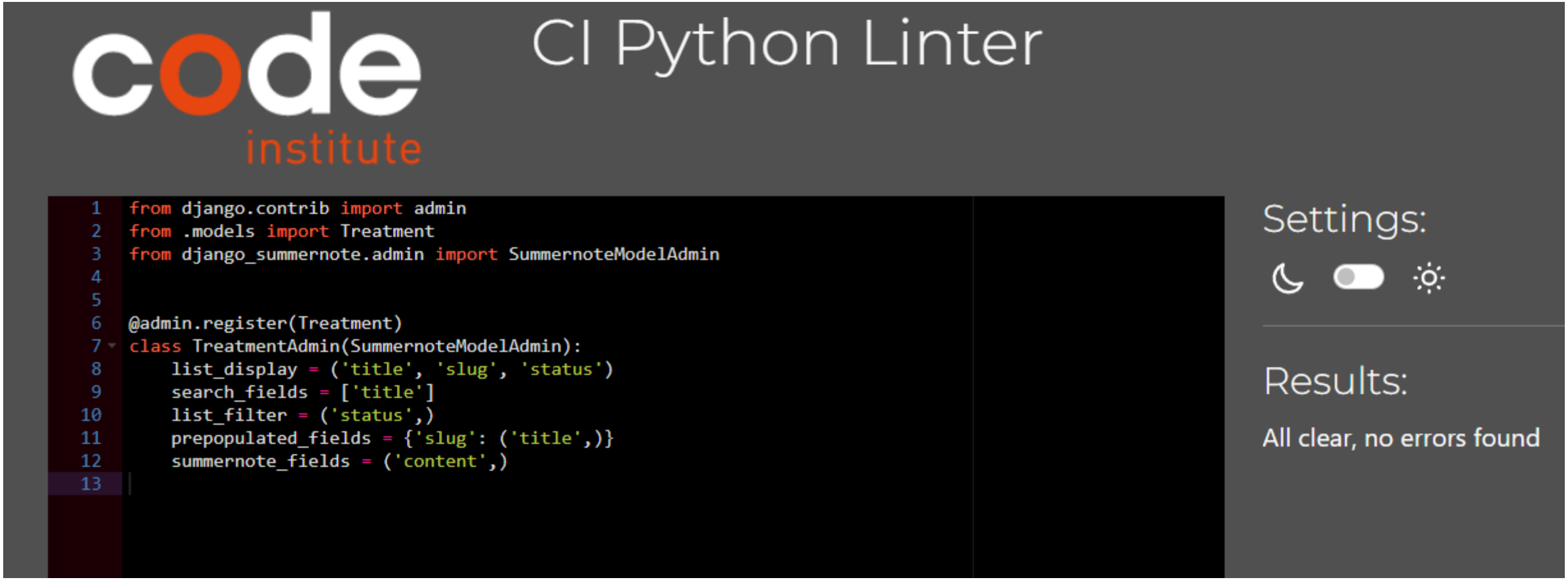
views.py


code
institute
CI Python Linter
"""Views to handle errors"""
from django.shortcuts import render
def handler404(request, exception):
""" Error Handler 404 - Page Not Found """
return render(request, "404.html", status=404)
def handler500(request):
""" Error Handler 500 - Internal Server Error """
return render(request, "500.html", status=500)
Settings:
Results:
All clear, no errors found

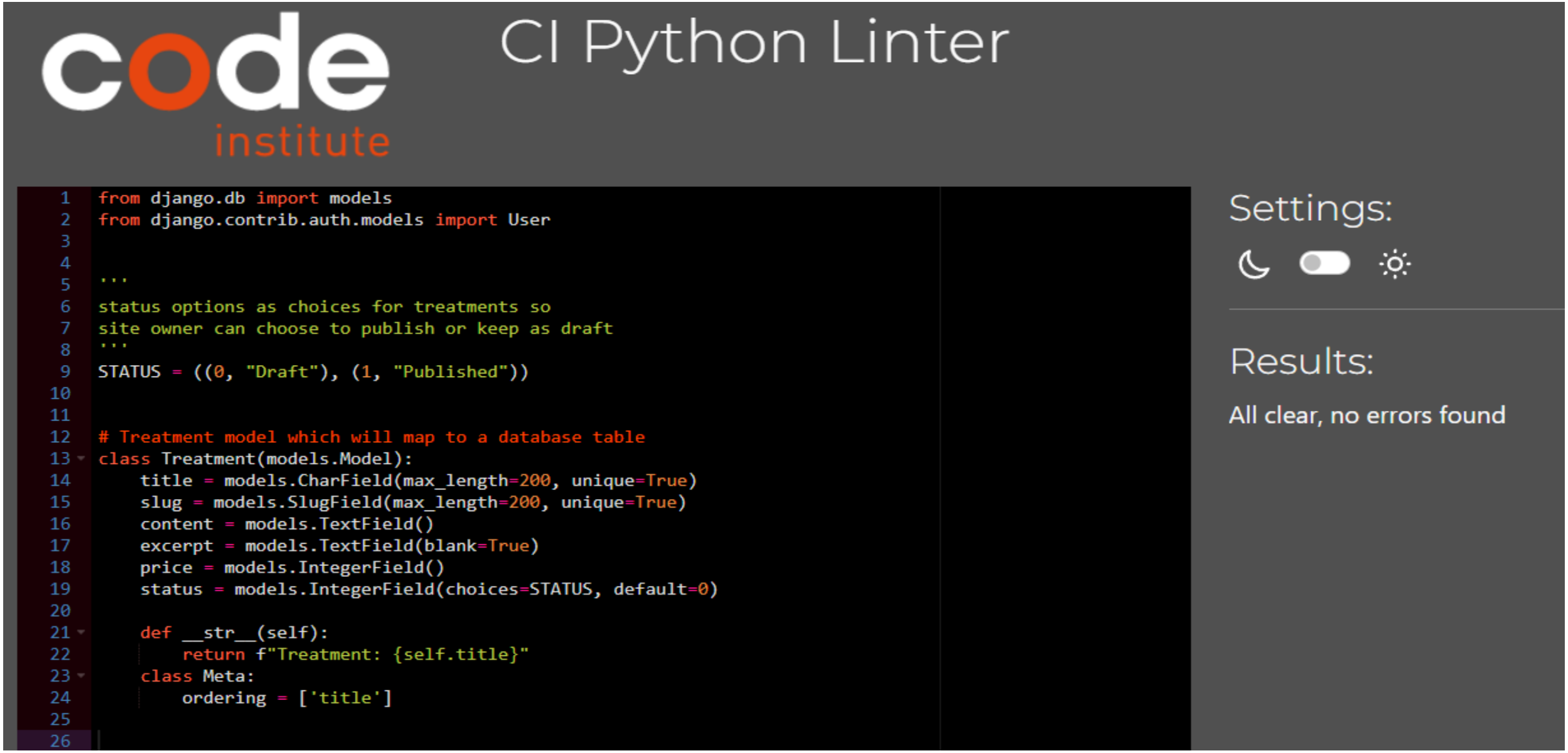
App: treatments



Admin.py

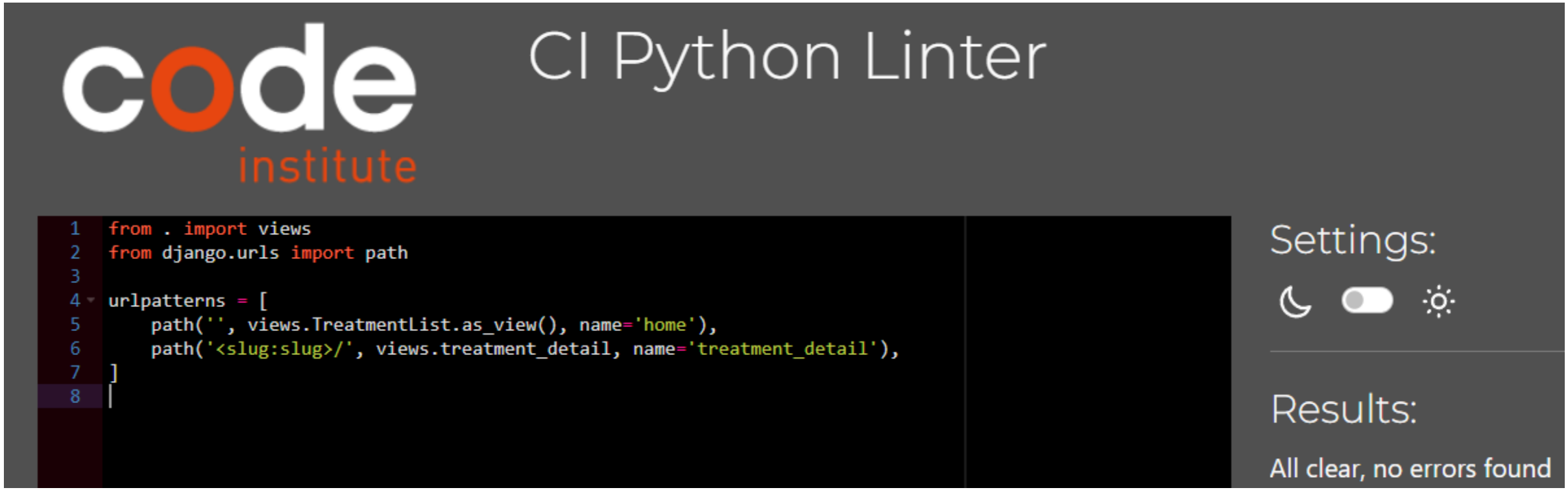

code
institute
CI Python Linter
from django.contrib import admin
from .models import Treatment
from django_summernote.admin import SummernoteModelAdmin
@admin.register(Treatment)
class TreatmentAdmin(SummernoteModelAdmin):
list_display = ('title', 'slug', 'status')
search_fields = ['title']
list_filter = ('status',)
prepopulated_fields = {'slug': ('title',)}
summernote_fields = ('content',)
Settings:
Results:
All clear, no errors found

models.py




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


'''
status options as choices for treatments so
site owner can choose to publish or keep as draft
'''
STATUS = ((0, "Draft"), (1, "Published"))


# Treatment model which will map to a database table
class Treatment(models.Model):
    title = models.CharField(max_length=200, unique=True)
    slug = models.SlugField(max_length=200, unique=True)
    content = models.TextField()
    excerpt = models.TextField(blank=True)
    price = models.IntegerField()
    status = models.IntegerField(choices=STATUS, default=0)

    def __str__(self):
        return f"Treatment: {self.title}"
    class Meta:
        ordering = ['title']
```

urls.py


code institute
CI Python Linter
from . import views
from django.urls import path
urlpatterns = [
path('', views.TreatmentList.as_view(), name='home'),
path('<slug:slug>/', views.treatment_detail, name='treatment_detail'),
]
Settings:
Results:
All clear, no errors found

views.py

CI Python Linter

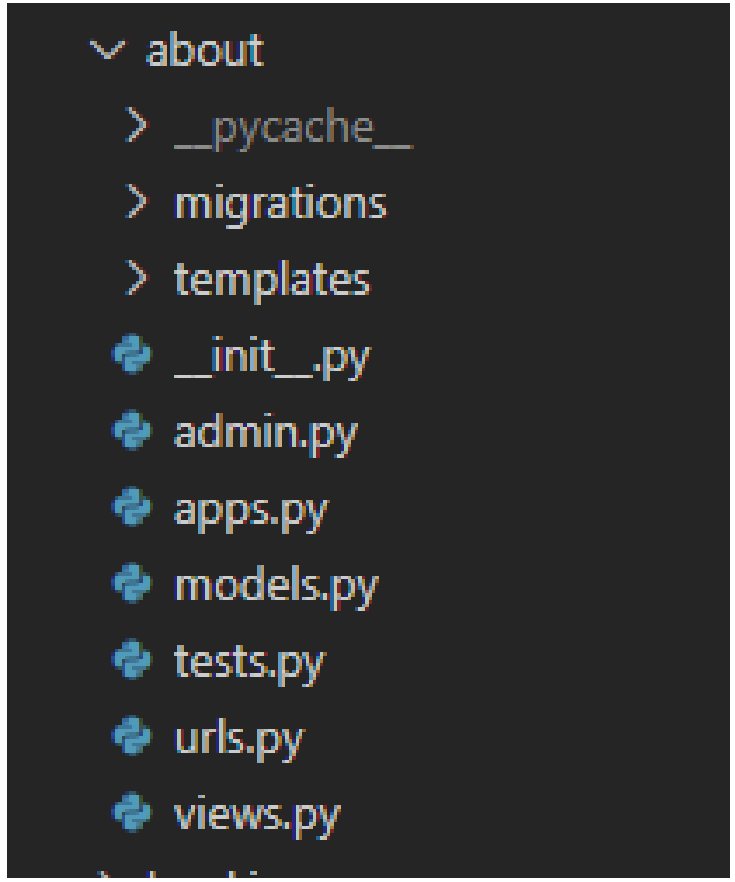
```python
from django.shortcuts import render, get_object_or_404
from django.views import generic
from .models import Treatment


# view for listing treatments
class TreatmentList(generic.ListView):
    queryset = Treatment.objects.filter(status=1)
    template_name = "treatments/index.html"
    # paginate by 3 treatments per page
    paginate_by = 3


def treatment_detail(request, slug):

    queryset = Treatment.objects.filter(status=1)
    treatment = get_object_or_404(queryset, slug=slug)

    return render(
        request,
        "treatments/treatment_detail.html",
        {"treatment": treatment},
    )

```

Settings:

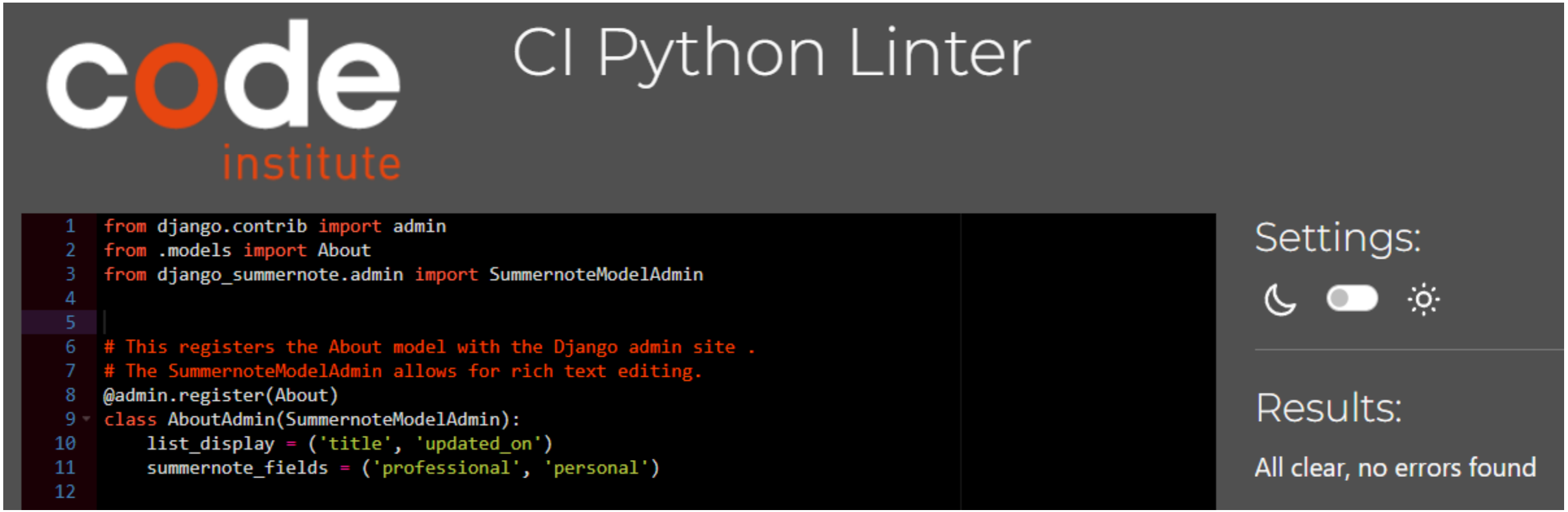
Results:

All clear, no errors found

App: about

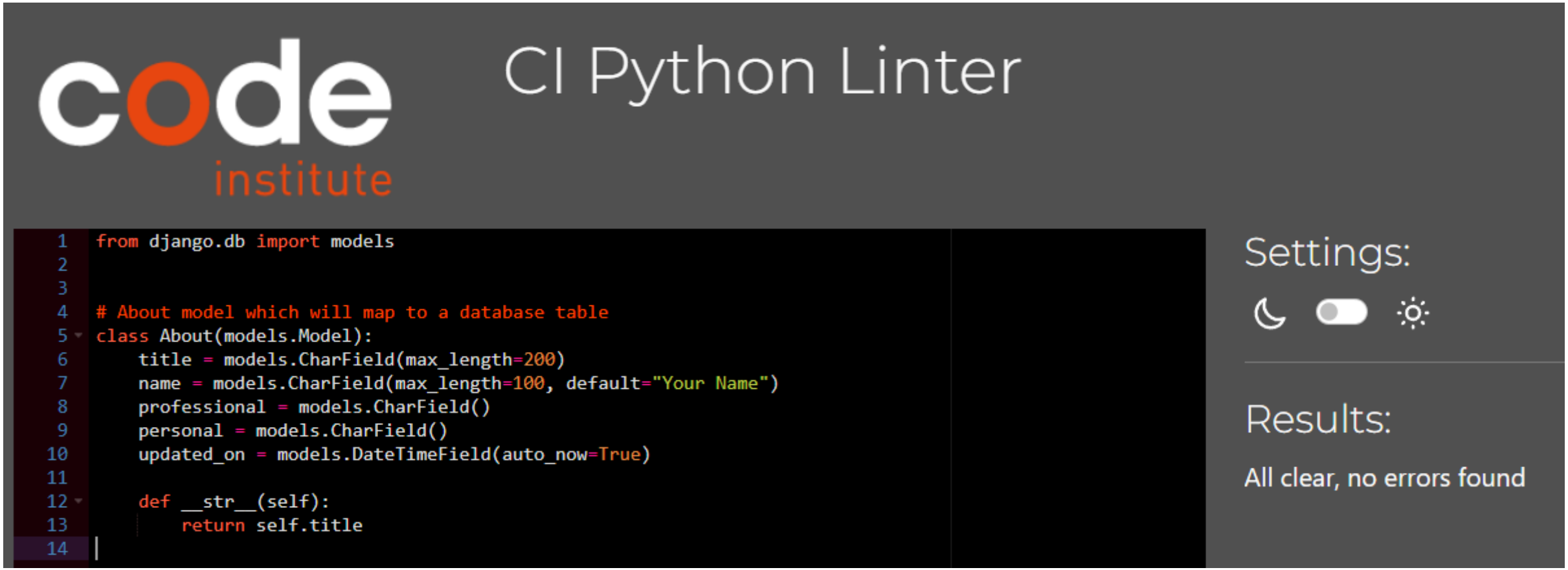

about
__pycache__
migrations
templates
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py

Admin.py



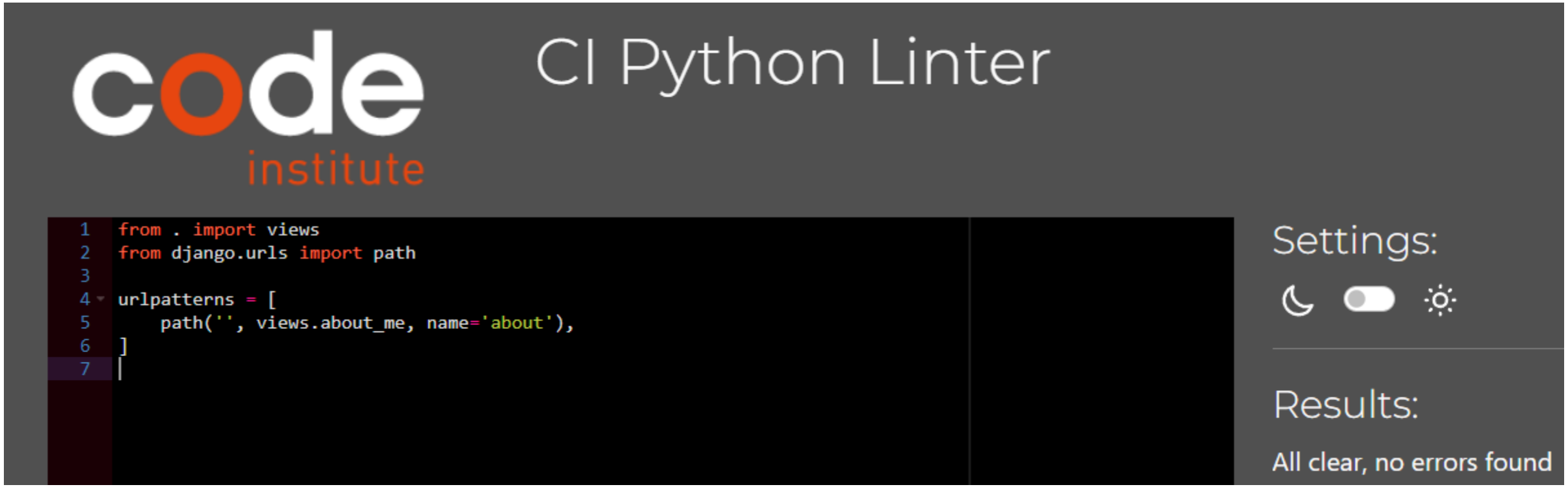

```python
from django.contrib import admin
from .models import About
from django_summernote.admin import SummernoteModelAdmin


# This registers the About model with the Django admin site .
# The SummernoteModelAdmin allows for rich text editing.
@admin.register(About)
class AboutAdmin(SummernoteModelAdmin):
    list_display = ('title', 'updated_on')
    summernote_fields = ('professional', 'personal')

```

models.py

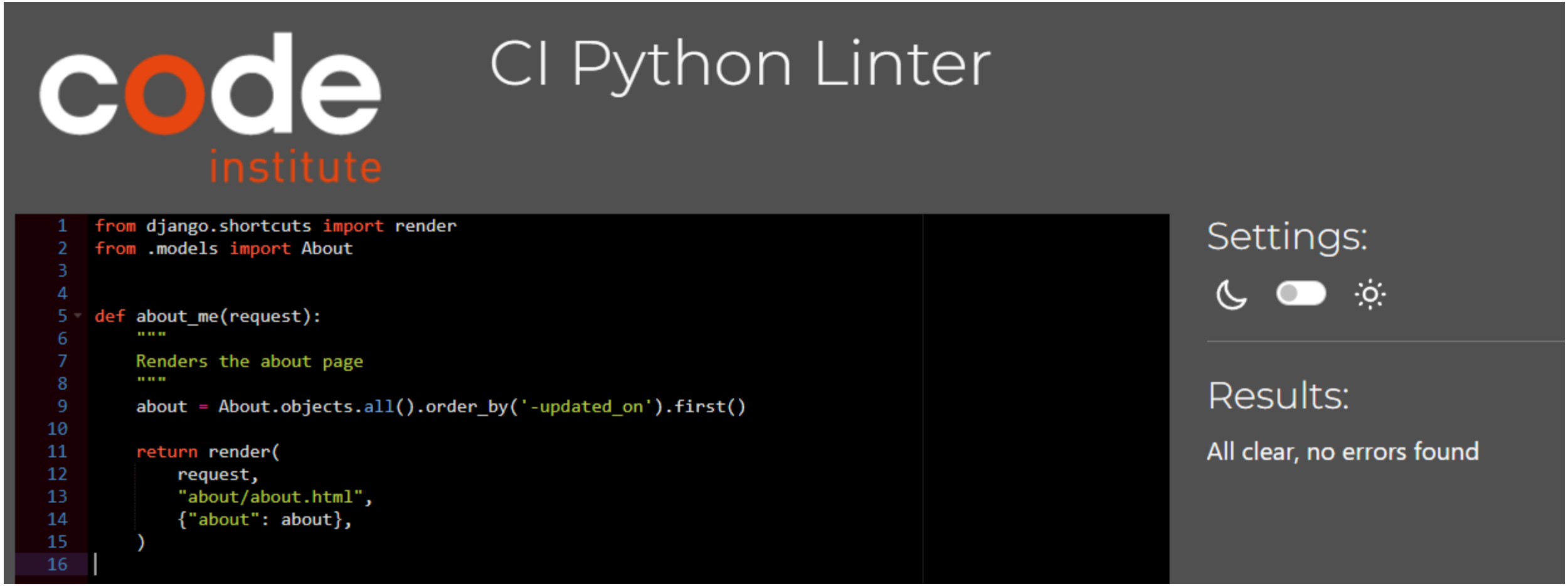



```python
from django.db import models


# About model which will map to a database table
class About(models.Model):
    title = models.CharField(max_length=200)
    name = models.CharField(max_length=100, default="Your Name")
    professional = models.CharField()
    personal = models.CharField()
    updated_on = models.DateTimeField(auto_now=True)

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

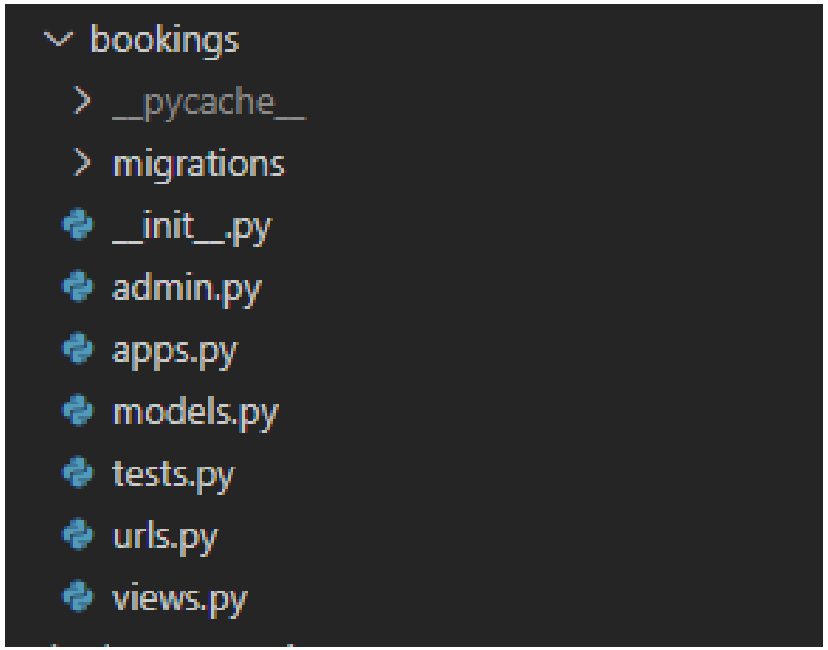
urls.py


code
institute
CI Python Linter
from . import views
from django.urls import path
urlpatterns = [
path('', views.about_me, name='about'),
]
Settings:
Results:
All clear, no errors found

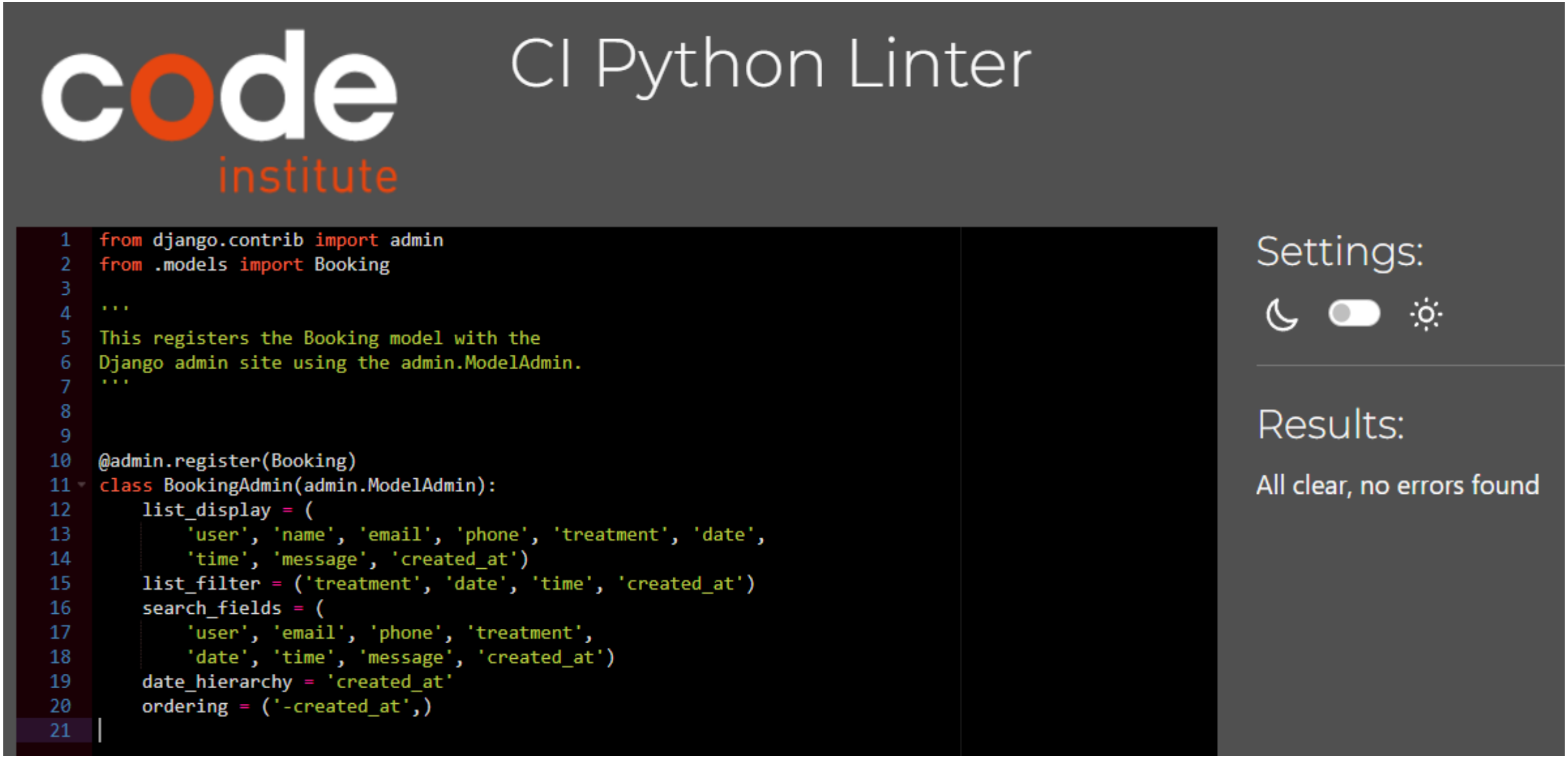
views.py


code
institute
CI Python Linter
from django.shortcuts import render
from .models import About
def about_me(request):
"""
Renders the about page
"""
about = About.objects.all().order_by('-updated_on').first()
return render(
request,
"about/about.html",
{"about": about},
)
Settings:
Results:
All clear, no errors found

App: bookings



Admin.py




```python
from django.contrib import admin
from .models import Booking

'''
This registers the Booking model with the
Django admin site using the admin.ModelAdmin.
'''


@admin.register(Booking)
class BookingAdmin(admin.ModelAdmin):
    list_display = (
        'user', 'name', 'email', 'phone', 'treatment', 'date',
        'time', 'message', 'created_at')
    list_filter = ('treatment', 'date', 'time', 'created_at')
    search_fields = (
        'user', 'email', 'phone', 'treatment',
        'date', 'time', 'message', 'created_at')
    date_hierarchy = 'created_at'
    ordering = ('-created_at',)
```

models.py

code institute

CI Python Linter

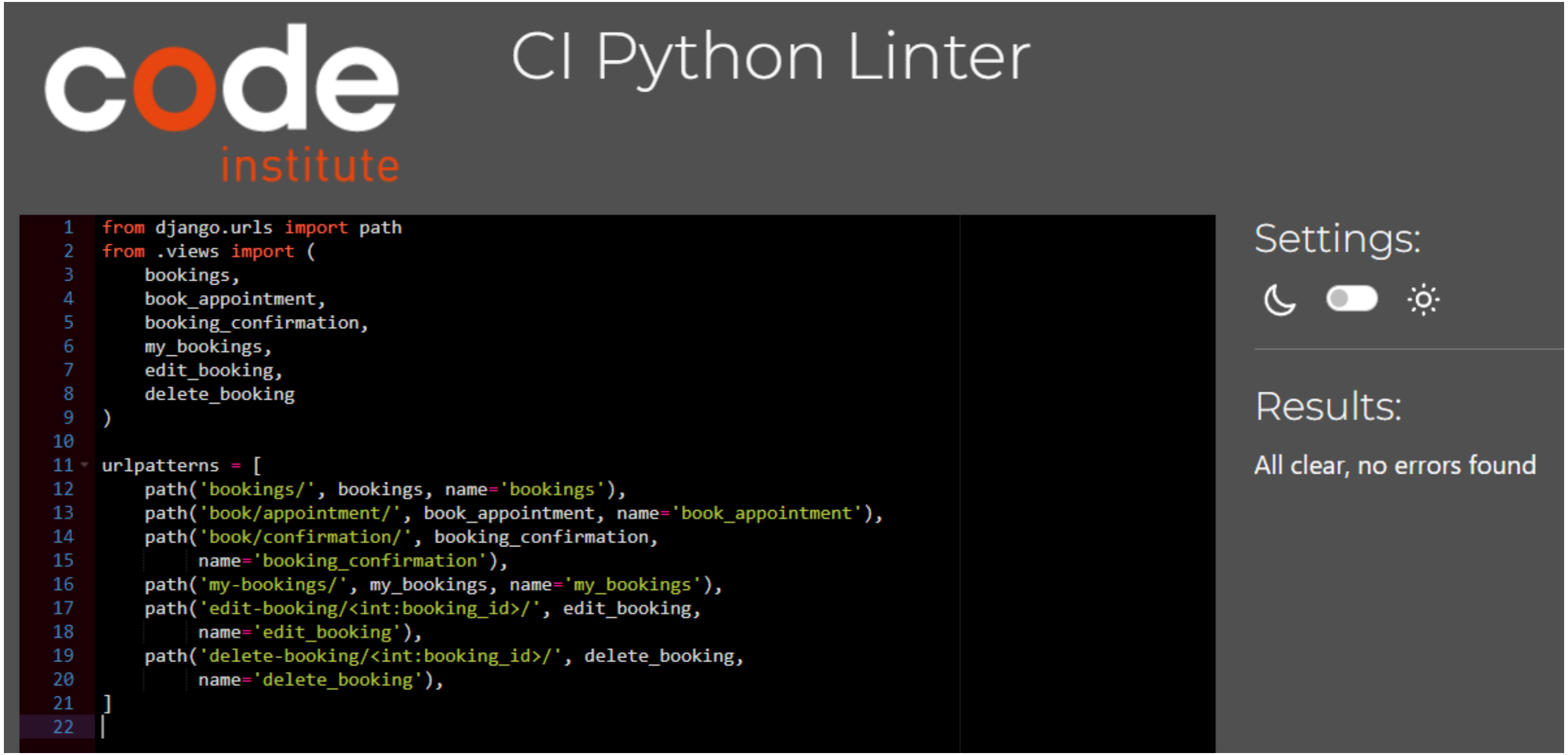
```python
from django.db import models
from django.contrib.auth.models import User
from django.core.validators import MaxLengthValidator, MinLengthValidator

# Time slots as choices for bokking times
TIME_SLOTS = (
    (0, '9:00-9:45'),
    (1, '10:00-10:45'),
    (2, '11:00-11:45'),
    (3, '14:00-14:45'),
    (4, '15:00-15:45'),
    (5, '16:00-16:45'),
    (6, '17:00-17:45'),
    (7, '18:00-18:45'),
)

# treatment options as choices for booking treatments
TREATMENTS = (
    (0, 'Botox Therapy(chronic pain relief)'),
    (1, 'Botox Treatment (wrinkle reduction/prevention)'),
    (2, 'Chemical Peel (skin rejuvenation)'),
    (3, 'Dermal Fillers (facce sculpting/wrinkle reduction)'),
    (4, 'Hyperhidrosis (excessive sweating treatment)'),
    (5, 'Lip filler (lip augmentation)'),
    (6, 'Skin: Hydrafacial'),
    (7, 'Skin: Profhilo Treatment'),
    (8, 'Wrinkle softening injections'),
)


# Booking model which will map to a database table
class Booking(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=100, default="Your Name")
    email = models.EmailField()
```

Settings:

Results:

All clear, no errors found

urls.py




```python
from django.urls import path
from .views import (
    bookings,
    book_appointment,
    booking_confirmation,
    my_bookings,
    edit_booking,
    delete_booking
)

urlpatterns = [
    path('bookings/', bookings, name='bookings'),
    path('book/appointment/', book_appointment, name='book_appointment'),
    path('book/confirmation/', booking_confirmation,
         name='booking_confirmation'),
    path('my-bookings/', my_bookings, name='my_bookings'),
    path('edit-booking/<int:booking_id>/', edit_booking,
         name='edit_booking'),
    path('delete-booking/<int:booking_id>/', delete_booking,
         name='delete_booking'),
]
```

views.py

code institute

CI Python Linter

```python
from django.shortcuts import (
    render, redirect, HttpResponseRedirect, get_object_or_404
)
from django.urls import reverse
from .models import Booking, TREATMENTS, TIME_SLOTS
from django.contrib.auth.decorators import login_required
from django import forms
import datetime


@login_required
def bookings(request):
    return render(
        request, 'bookings.html',
        {'treatments': TREATMENTS, 'time_slots': TIME_SLOTS}
    )


@login_required
def book_appointment(request):
    if request.method == 'POST':
        # Check if all required fields are present
        required_fields = [
            'treatment', 'time_slot', 'date', 'message', 'name', 'phone'
        ]
        missing_fields = [
            field for field in required_fields if field not in request.POST
        ]
        if missing_fields:
            # Construct an error message indicating which fields are missing
            error_message = "Please fill out all required fields."
            for field in missing_fields:
                error_message += f" {field.capitalize()} is required."

            # Redirect back to the booking page with the error message
```

Settings:

Results:

All clear, no errors found