

▼ about

> \_\_pycache\_\_

> migrations

> templates

🔗 \_\_init\_\_.py

🔗 apps.py

🔗 urls.py

🔗 views.py

# code Python institute Linter

```
1 from django.apps import AppConfig
2
3
4 class AboutConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'about'
7 |
```

Settings:



Results:

All clear, no errors found

# code Python institute Linter

```
1 from django.shortcuts import render
2
3
4 # Create your views here.
5 def about(request):
6     return render(request, 'about/about.html')
7
```

Settings:



Results:

All clear, no errors found

```
1 import os
2
3 from django.core.asgi import get_asgi_application
4
5 os.environ.setdefault('DJANGO_SETTINGS_MODULE',
6                       'fitsphere.settings')
7
8 application = get_asgi_application()
9
```

Settings:



Results:

All clear, no errors found

# code Python institute Linter

```
1 from django.conf import settings
2
3
4 def social_links(request):
5     return {
6         'social_links': settings.SOCIAL_LINKS,
7     }
8 |
```

Settings:



Results:

All clear, no errors found

▼ pt

> \_\_pycache\_\_

> migrations

> templates

🔗 \_\_init\_\_.py

🔗 apps.py

🔗 urls.py

🔗 views.py

```
1 from django.apps import AppConfig
2
3
4 class PtConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'pt'
7 |
```

Settings:



Results:

All clear, no errors found

# code Python institute Linter

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.pt, name='index'), # Home page route
6 ]
7 |
```

Settings:



Results:

All clear, no errors found



# code Python institute Linter

```
1 from django.shortcuts import render, redirect
2
3
4 # Redirect the user to trainers if signed in
5 def pt(request):
6     if request.user.is_authenticated:
7         return redirect('trainers:trainers')
8     return render(request, 'pt/index.html')
9
```

Settings:



Results:

All clear, no errors found

```
1 from django.urls import path
2 from . import views
3
4 app_name = 'trainers'
5
6 urlpatterns = [
7     path('', views.trainers, name='trainers'),
8     path('trainer/<int:trainer_id>/',
9         views.trainer_profile,
10        name='trainer_profile'),
11     path('feedback/delete/<int:id>/',
12         views.delete_feedback,
13        name='delete_feedback'),
14     path('feedback/edit/<int:feedback_id>/',
15         views.edit_feedback,
16        name='edit_feedback'),
17     path('trainer/<int:trainer_id>/feedback/',
18         views.submit_feedback,
19        name='submit_feedback'),
20 ]
```

Settings:



Results:

All clear, no errors found

```
42 # TrainerFeedback Admin with Summernote
43 class TrainerFeedbackAdmin(SummernoteModelAdmin):
44     # List of fields to display in the admin list view
45     list_display = ('user', 'trainer', 'rating',
46                    'created_at', 'updated_at')
47
48     # Filter options
49     list_filter = ('trainer', 'rating')
50
51     # Search fields
52     search_fields = (
53         'user__username',
54         'trainer__first_name',
55         'trainer__last_name',
56     )
57     # Summernote for the comment field (rich text editor)
58     summernote_fields = ('comment',)
59
60     # Read-only fields for timestamps
61     readonly_fields = ('created_at', 'updated_at')
62
63     # Ordering feedback by created_at
64     ordering = ('-created_at',)
65
66 # Register models
67 admin.site.register(Trainer, TrainerAdmin)
68 admin.site.register(TrainerFeedback,
69                     TrainerFeedbackAdmin)
```

Settings:



Results:

All clear, no errors found

## ▼ trainers

> \_\_pycache\_\_

> migrations

> templates

🌀 \_\_init\_\_.py

🌀 admin.py

🌀 apps.py

🌀 forms.py

🌀 models.py

🌀 urls.py

🌀 views.py

```
1 from django.apps import AppConfig
2
3
4 class TrainersConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'trainers'
7 |
```

Settings:



Results:

All clear, no errors found

```
1 from django import forms
2 from .models import TrainerFeedback
3
4
5 class TrainerFeedbackForm(forms.ModelForm):
6     class Meta:
7         model = TrainerFeedback
8         fields = ['rating', 'comment']
9
10
11 def clean_comment(self):
12     comment = self.cleaned_data.get('comment')
13     # You can add extra validations here if needed
14     if not comment:
15         raise forms.ValidationError('Comment cannot be
16                                     empty.')
17     return comment
18
```

Settings:



Results:

All clear, no errors found

```
72         (4, '4'),
73         (5, '5'),
74     )
75
76     user = models.ForeignKey(User, on_delete=models
77         .CASCADE)
78     trainer = models.ForeignKey(
79         Trainer,
80         on_delete=models.CASCADE,
81         related_name="feedbacks"
82     )
83     rating = models.IntegerField(choices=STARS)
84     comment = models.TextField(null=True, blank=True)
85     created_at = models.DateTimeField(auto_now_add=True)
86
87     updated_at = models.DateTimeField(auto_now=True)
88
89     def __str__(self):
90         return (
91             f"Feedback from {self.user.username} for"
92             f"{self.trainer.first_name}"
93             f"{self.trainer.last_name}"
94         )
95
96     class Meta:
97         ordering = ['-created_at'] # Ordering feedback
98         indexes = [
99             models.Index(fields=['trainer',
100                 'created_at'])
```

Settings:



Results:

All clear, no errors found



```
85         request, "There was an error updating  
            your feedback.")  
86         return redirect('trainers:trainer_profile'  
            , trainer_id=trainer.id)  
87  
88     else:  
89         form = TrainerFeedbackForm(instance=feedback)  
90  
91     return render(request, 'trainer_profile.html', {  
92         'form': form,  
93         'feedback': feedback,  
94         'trainer': trainer,  
95     })  
96  
97  
98     def delete_feedback(request, id):  
99         feedback = get_object_or_404(TrainerFeedback, id  
            =id)  
100         trainer = feedback.trainer  
101  
102         if feedback.user == request.user:  
103             feedback.delete()  
104             messages.success(request, "Feedback deleted  
                successfully!")  
105         else:  
106             messages.error(request, "You can only delete  
                your own feedback.")  
107  
108         return redirect('trainers:trainer_profile',  
            trainer_id=trainer.id)  
109
```

Settings:



Results:

All clear, no errors found