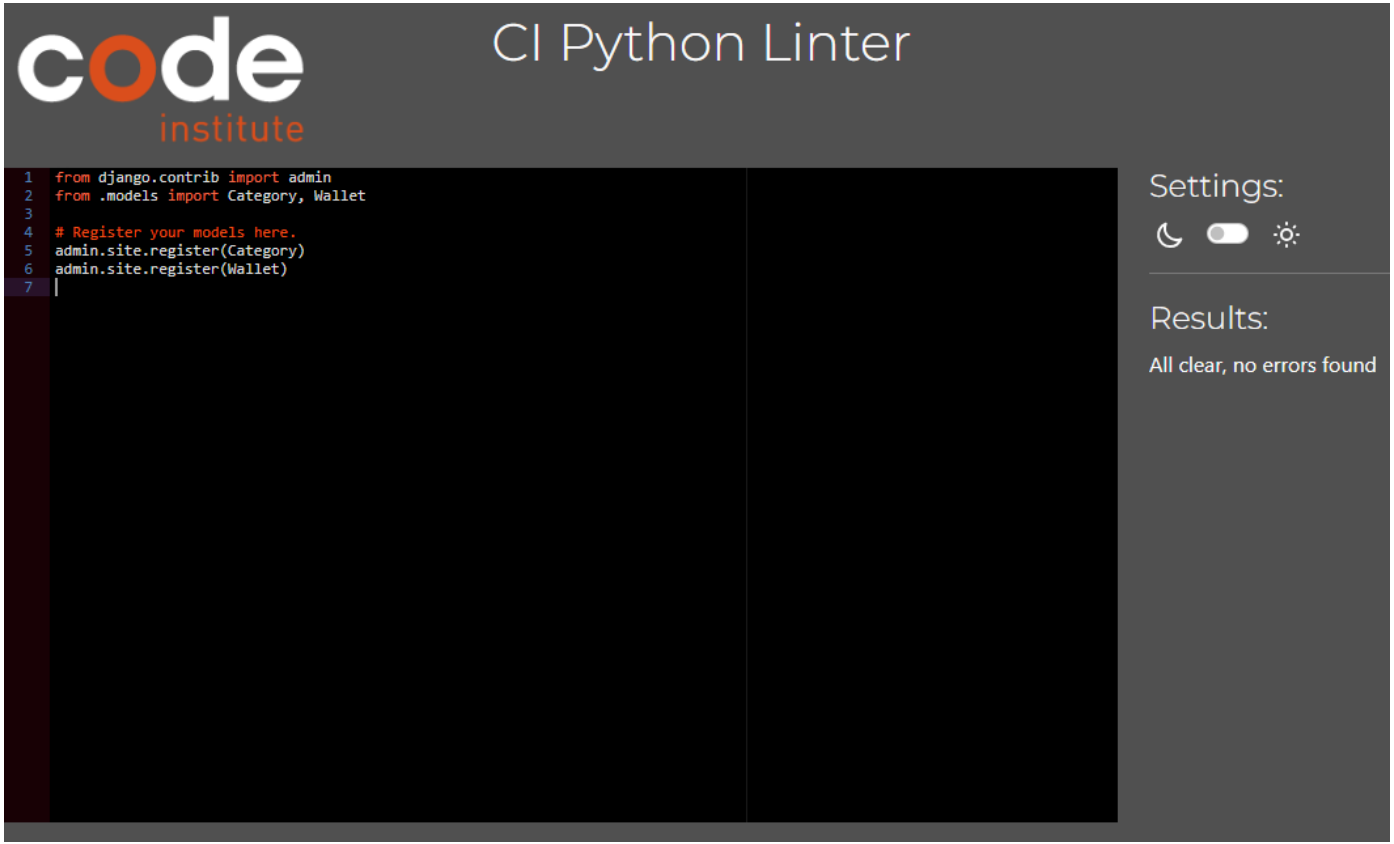
PEP8 VALIDATION

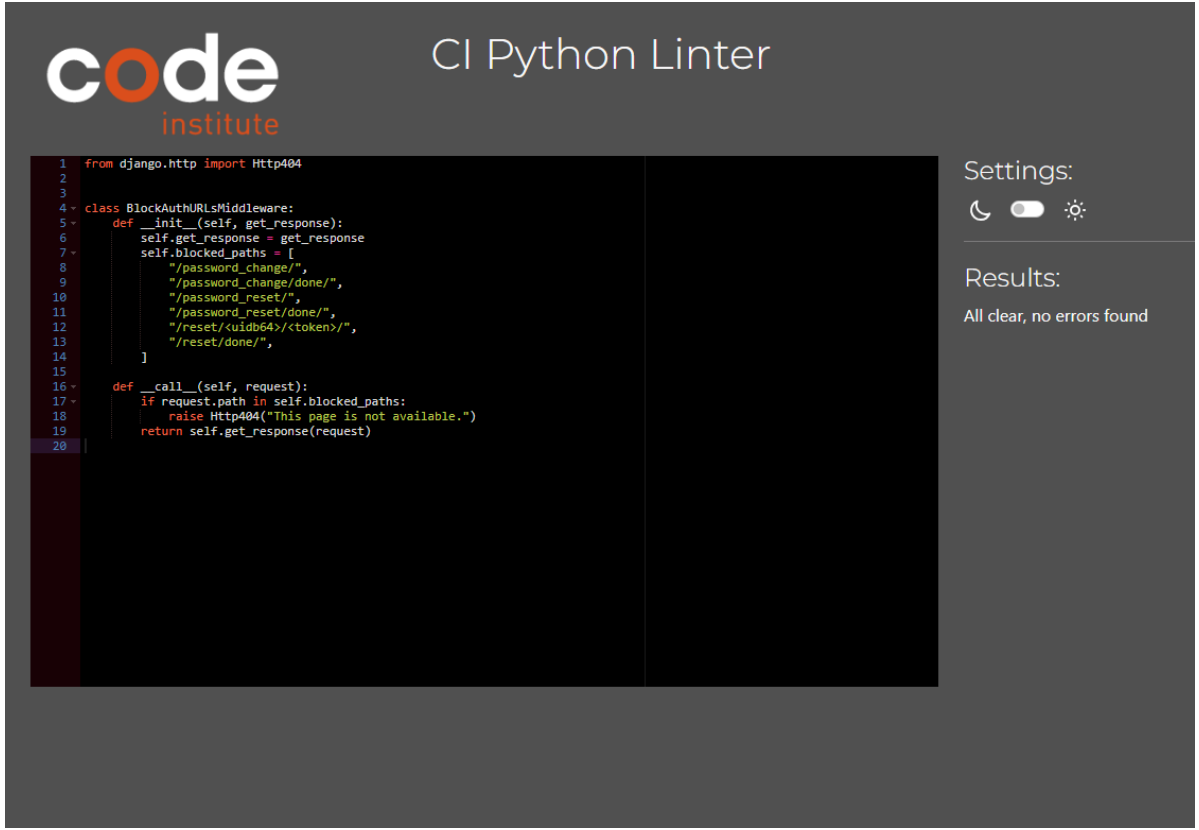
admin.py


code
institute
CI Python Linter
1 from django.contrib import admin
2 from .models import Category, Wallet
3
4 # Register your models here.
5 admin.site.register(Category)
6 admin.site.register(Wallet)
7
Settings:
Results:
All clear, no errors found

Foms.py


```python
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
from django.core.exceptions import ValidationError
from django.utils import formats, timezone

from .models import Transaction, Wallet


class RegisterForm(UserCreationForm):
    """Form for registering a new user with a username and password."""
    class Meta:
        model = User
        fields = ["username", "password1", "password2"]


class TransactionForm(forms.ModelForm):
    """Form for creating a new transaction."""
    class Meta:
        model = Transaction
        # fields = ['wallet', 'category', 'amount', 'date','is_income']
        exclude = ["user"]
        widgets = {
            "date": forms.DateInput(
                format="%d-%m-%Y",
                attrs={
                    "type": "date",
                    "max": formats.date_format(timezone.now(), "Y-m-d"),
                },
            )
        }

    def __init__(self, *args, **kwargs):
        user = kwargs.pop("user", None)
        super().__init__(*args, **kwargs)  # Initialize the form
```

Middleware.py



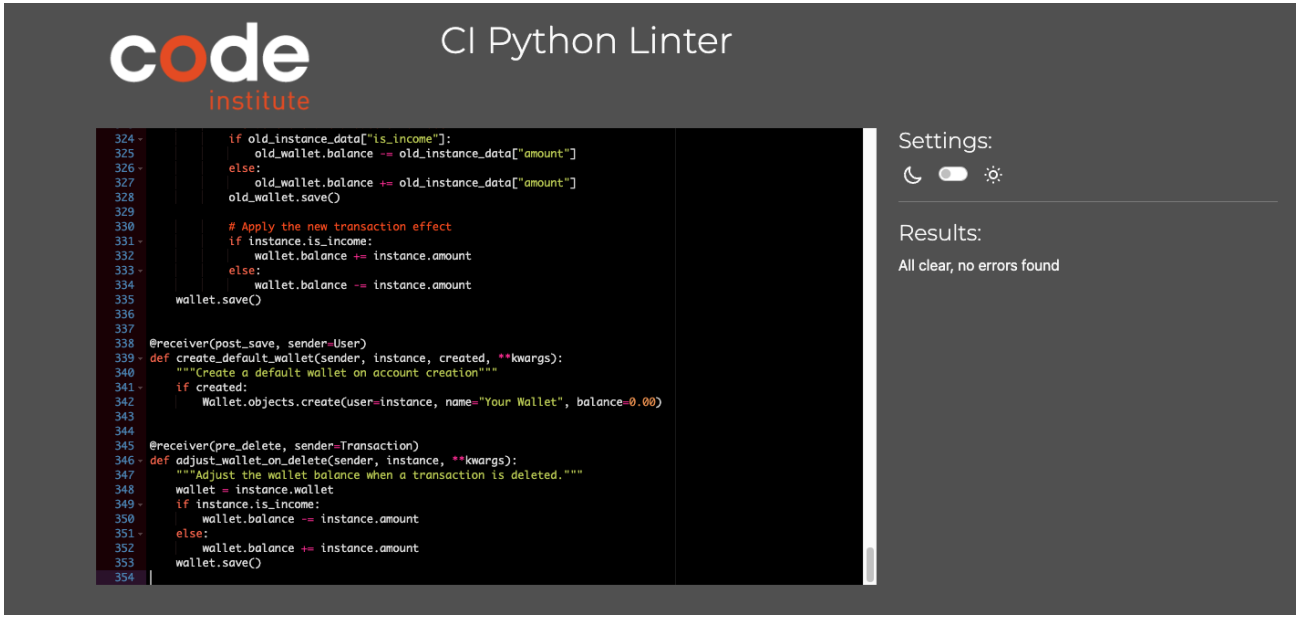
Models.py


```
    name = models.CharField(
        max_length=100,
        unique=True,
        help_text="The unique name of the category.",
    )  # Unique category name

    def __str__(self):
        return self.name  # For readable display

    def delete(self, *args, **kwargs):
        raise Exception("Deleting categories is not allowed.")


class Transaction(models.Model):
    wallet = models.ForeignKey(
        Wallet, on_delete=models.CASCADE
    )  # One-to-many link to Wallet
    category = models.ForeignKey(
        Category, on_delete=models.SET_NULL, null=True
    )  # Many-to-one link to Category
    amount = models.DecimalField(
        max_digits=10,
        decimal_places=2,
        validators=[MinValueValidator(Decimal("0.00"))],
    )  # Transaction amount
    date = models.DateField()  # Transaction date
    is_income = models.BooleanField(default=False)
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        category_name = (
            self.category.name if self.category else "Uncategorized"
        )
        return f"{self.amount} - {category_name}"
```

URLS.py


code institute
CI Python Linter
path('personal/wallets',views.wallets,name="wallets"),
path(
"personal/transactions/<int:pk>/update",
views.UpdateTransaction.as_view(),
name="update_transaction",
),
path(
"personal/transactions/<int:pk>/delete",
views.DeleteTransaction.as_view(),
name="delete_transaction",
),
Wallet
path(
"personal/wallet/create",
views.WalletCreateView.as_view(),
name="create_wallet",
),
path("personal/wallets", views.ListWallet.as_view(), name="list_wallet"),
path(
"personal/wallets/<int:pk>/update",
views.UpdateWallet.as_view(),
name="update_wallet",
),
path(
"personal/wallets/<int:pk>/delete",
views.DeleteWallet.as_view(),
name="delete_wallet",
),
]
# Serve static files during development
if settings.DEBUG:
urlpatterns += static(
settings.STATIC_URL, document_root=settings.STATIC_ROOT
)
Settings:
Results:
All clear, no errors found

Views.py




```
            if old_instance_data["is_income"]:
                old_wallet.balance -= old_instance_data["amount"]
            else:
                old_wallet.balance += old_instance_data["amount"]
            old_wallet.save()

            # Apply the new transaction effect
            if instance.is_income:
                wallet.balance += instance.amount
            else:
                wallet.balance -= instance.amount
    wallet.save()


@receiver(post_save, sender=User)
def create_default_wallet(sender, instance, created, **kwargs):
    """Create a default wallet on account creation"""
    if created:
        Wallet.objects.create(user=instance, name="Your Wallet", balance=0.00)


@receiver(pre_delete, sender=Transaction)
def adjust_wallet_on_delete(sender, instance, **kwargs):
    """Adjust the wallet balance when a transaction is deleted."""
    wallet = instance.wallet
    if instance.is_income:
        wallet.balance -= instance.amount
    else:
        wallet.balance += instance.amount
    wallet.save()
```