Exercise 30: Creating an Email Utility Function

10 to 15 minutes

As you may create many apps that make use of email, it makes sense to create a `send_email()` utility function in the common app.

Create a new `email.py` file within `common/utils` with the following content:[51]

Exercise Code 30.1: common/utils/email.py

```python
import sendgrid
from sendgrid.helpers.mail import Mail

from django.conf import settings


def send_email(to, subject, content, sender='admin@example.com'):
    sg = sendgrid.SendGridAPIClient(settings.SENDGRID_API_KEY)
    mail = Mail(
        from_email=sender,
        to_emails=to,
        subject=subject,
        html_content=content
    )
    return sg.send(mail)
```

Code Explanation

This encapsulates everything you did at the Django shell into a function. The `to`, `subject`, and `content` arguments are all required, and `sender` defaults to `'admin@example.com'`.

❖ E30.1. Try It Out

1. With `djangojokes.com` open in the terminal, run the following to open the shell:

```
(.venv) …/projects/djangojokes.com> python manage.py shell
```

51. **Don't want to type?** Copy from `starter-code/sendgrid/email.py`.

2. Create your variables (don't forget to replace `'you@example.com'`) and send the email:

```
>>> from common.utils import email
>>> to = 'you@example.com'
>>> subject = 'SendGrid Test 2'
>>> content = '<h1>It worked!</h1><p>So cool!</p>'
>>> email.send_email(to, subject, content)
```

3. Check your email. If you received an email with the subject "SendGrid Test 2," you are all set.

Git Commit

1. Open the `.gitignore` file.
2. Confirm that it contains `local_settings.py`. You don't want that file in source control as it contains sensitive data.
3. Commit your code to Git.

Conclusion

In this lesson, you have set up Django to use SendGrid to send email. You are now ready to create forms that autogenerate emails.