創建虛擬環境

開啟終端機後，輸入以下指令：`python -m venv 虛擬環境名稱`

切換虛擬環境

開啟終端機後，輸入以下指令：`虛擬環境名稱\Scripts\activate`

```
###自己的:django_venv\Scripts\activate
```

在終端機內輸入以下指令：

```
heroku login
```

安裝 Heroku 用的 Python 套件

確認終端機有在虛擬環境內。若沒有，則先參照前面的教學，進入虛擬環境。接著輸入：

```
pip install dj-database-url gunicorn dj-static
```

`requirements.txt`

利用以下的指令將此虛擬環境裡的 Python 套件全部條列出來，包括套件名稱與版本資訊，並儲存於 requirements.txt：

```
pip freeze > requirements.txt
```

由於 Heroku 使用 PostgreSQL 資料庫，我們還需要手動在 `requirements.txt` 最後面加上 **psycopg2==2.9.1**

在網站根目錄(即 manage.py 的所在目錄)建立 Procfile 檔案
注意，沒有附檔名喔！而且 P 必須為大寫
內容輸入

```
web: gunicorn --pythonpath mysite mysite.wsgi --log-file -
```

這個指令告訴heroku用gunicorn執行mysite.wsgi(即下層mysite的wsgi)
同時也是heroku執行app的進入點

修改 wsgi.py

heroku 處理靜態檔案的方式與本機不相同
而且為了讓 heroku 能夠透過 wsgi 與我們的網站溝通
將原本的 wsgi.py

```
import os
from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'mysite.setting
s')
application = get_wsgi_application()
```

修改為

```
import os
from django.core.wsgi import get_wsgi_application
from dj_static import Cling # <- 加入

os.environ.setdefault("DJANGO_SETTINGS_MODULE",
"mysite.settings")
application = Cling(get_wsgi_application()) # <- 修改
```

建立 production_settings.py

通常正式上線（production）時的環境會和開發時所做的 settings.py 設定有所不同
而且我們希望網站上線時使用一個全新的、空白的資料庫
以便與原本測試用的資料庫來做區分
所以我們另外建立一個 production_settings.py，放在原本的 settings.py 旁邊

```
import dj_database_url
from .settings import * # 含入原本的settings.py 所有設定
# heroku 使用的資料庫為 PostgreSQL，所以要修改資料庫設定
DATABASES = {
    'default': dj_database_url.config(),
}
STATIC_ROOT = 'staticfiles' # 設定網站正式上線時靜態檔案目錄位置
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https') #
 設定 HTTP 連線方式
ALLOWED_HOSTS = ['*'] # 讓所有的網域都能瀏覽本網站
DEBUG = False # 關閉除錯模式
```

建立 .gitignore

為了要節省伺服器空間，我們不會把開發時使用的檔案，例如虛擬環境、本機資料庫及測試用檔案，一起部署上去
因此，在網站根目錄建立一個.gitignore 檔案，git 在 commit 時會自動略過它們(後面會講)
注意這個檔案就是以點.開頭，然後 gitignore 後面沒有其他的副檔名

```
.vscode
*.pyc
__pycache__
db.sqlite3
Include
Lib
Scripts
tcl
```

Include、Lib、Scripts、tcl 這些即為當初建立虛擬環境時所產生的檔案目錄

新增新的 Heroku App

接下來，我們需要新增一個可以上傳 repository 的地方：（如果你之前已經新增過，請直接跳到下一個步驟）

```
heroku create 自己 app 名稱
```

在本機初始化 git 並新建一個 git 儲存庫(repository)來存放和記錄網站檔案

```
git init
```

再將此 git 儲存庫與剛才你建立的 heroku 應用程式建立連結

```
heroku git:remote -a <你的 heroku App name>
```

-a 表示指定哪個應用程式

設定 heroku 使用 production_settings.py 做為網站設定檔

```
heroku config:set DJANGO_SETTINGS_MODULE=mysite.production_settings
```

將網站所有檔案加入 git 的追蹤

```
git add .
```

將所有追蹤的檔案加入 git 儲存庫，並將此次 commit 的動作命名為 first commit

```
git commit -m "first commit"
```

commit 好後，將檔案推送至 heroku

```
git push heroku master
```

部署完成後，因為我們做的設定是使用全新的空白資料庫
所以要同步資料庫並建立新的超級使用者

```
heroku run python manage.py migrate
heroku run python manage.py createsuperuser
```

最後讓 heroku 運行網站

```
heroku ps:scale web=1
```

如果忘記你的網站網址的話
也可以直接執行

```
heroku open
```

以上內容參考
https://ithelp.ithome.com.tw/articles/10212659?sc=rss.qu

Django dumpdata and loaddata
https://coderwall.com/p/mvsoyg/django-dumpdata-and-loaddata

dumpdata (--format)

- By default, dumpdata will format its output in JSON
- You can specify the format using --format option
- Command supports for following formats(serialization formats)

1. json
2. xml
3. yaml

```
./manage.py dumpdata auth.user --indent 2 --format xml > user.xml
```

How to transfer your local sqlite3 data to heroku?

https://stackoverflow.com/questions/61336258/how-to-transfer-your-local-sqlite3-data-to-heroku

First of all, dump your database into a json file, like shown below:

```
python manage.py dumpdata --exclude contenttypes > db.json
```

Once this is done, you need to commit and push this changes to local as well as heroku's git repositories. This step is needed, because we need to load the data from this "db.json" file and for that it should be present in heroku's repo. so, type these commands:

```
git add .
git commit -m "added db"
git push
git push heroku main
```

After pushing to heroku main, you may want to run `heroku run ls` which will list all the files in the repo and make sure, your db.json is present.

Finally run this to load the data into heroku:

```
heroku run python3 manage.py loaddata db.json
```

###git push heroku master