



# 網站使用者的註冊與管理

1

## 學習目標

- 建立網站使用者自動註冊功能
  - [django-registration](#)安裝與設定
  - 建立django-registration所需的模板
  - 整合使用者註冊功能到分享日記網站
- [Pythonanywhere.com](#)免費Python網站開發環境
  - 註冊Pythoanywhere.com帳號
  - 在Pythonanywhere免費網站中建立虛擬環境以及Django網站
  - 建立投票網站基本架構
- 使用[Facebook](#)驗證帳號操作實務
  - 在Pythonanywhere中安裝[django-allauth](#)與設定
  - 到Facebook開發者網頁申請驗證機制
  - 在網站中識別使用者的登入狀態
  - 客製化[django-allauth](#)頁面

## 建立網站使用者的自動化註冊功能

- 會員網站要讓使用者註冊最直覺也是最多網站使用的方法就是使用**電子郵件啟用**的方式。
- 也就是說，使用者在註冊之後要填寫正確的電子郵件位址，接著網站會寄送一封啟用電子郵件到使用者設定的電子郵件信箱，並在信件中提供一個啟用的連結，在使用者按下此連結之後，帳號正式啟用。
- 這些操作的過程只要**使用合適的framework**就可以輕易在自己的Django網站中完成。

## django-registration-redux安裝與設定

- ▶ **django-registration-redux**
  - ▶ 一個整合到Django網站使用者驗證機制中最方便的使用者電子郵件註冊啟用模組
  - ▶ 會使用到電子郵件的寄送功能，所以請務必依照在前面章節中的介紹**完成mailgun介面的相關設定**
  - ▶ 直接複製一份第9堂課的程式做些修改成為**Registration01**
  - ▶ 以pip安裝**django-registration-redux**，如下：
    - ▶ **pip install django-registration-redux**
  - ▶ 由於此framework會運用Django原有的auth架構，所以要**確定在前一堂課操作auth的使用者認證的部份都沒有問題**

## django-registration-redux安裝與設定

- 在settings.py中設定一個常數，用來指定啟用碼的天數，這個常數在檔案的任何一個地方設定均可：
  - `ACCOUNT_ACTIVATION_DAYS = 7`
- 一般習慣上都是設定為7天，它可以是任意的整數。接著，因為它是使用標準自訂網址，所以在urls.py中要加上一行設定，如下所示：
  - `path('accounts/', include('registration.backends.default.urls'))`,
- adding `registration` to the `INSTALLED_APPS`

# django-registration-redux安裝與設定

- 在header.html中，如下所示：

```
<!-- header.html (Registration01) -->
<nav class='navbar navbar-default'>
  <div class='container-fluid'>
    <div class='navbar-header'>
      <div class='navbar-brand' align=center>
        分享日記
      </div>
    </div>
  </div>
  <ul class='nav navbar-nav'>
    <li class='active'><a href='/'>Home</a></li>
    {% if username %}
    <li><a href='/userinfo'>個人資料</a></li>
    <li><a href='/post'>寫日記</a></li>
    <li><a href='/logout'>登出</a></li>
    {% else %}
    <li><a href='/login'>登入</a></li>
    <li><a href='/accounts/register'>註冊</a></li>
    {% endif %}
    <li><a href='/admin'>後台管理</a></li>
  </ul>
</div>
</nav>
```



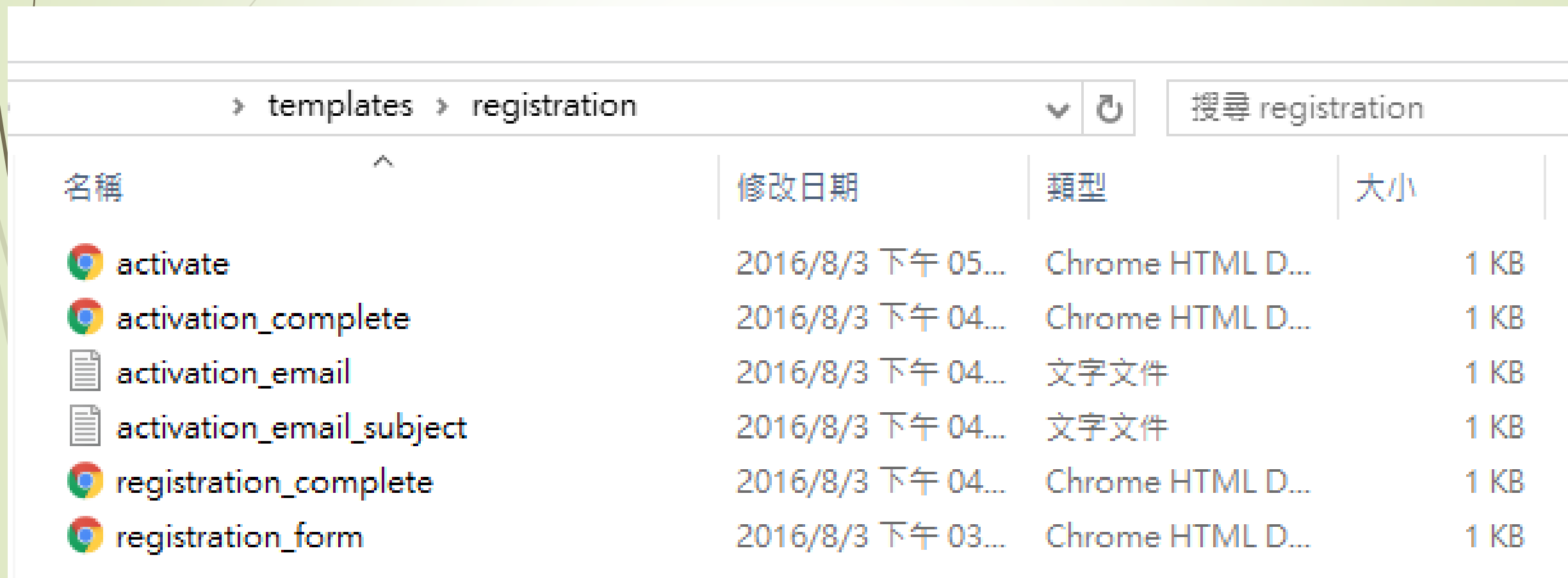
## 建立django-registration所需的模板

- ▶ django-registration所有的**模板**以及**文字檔案**都必須放在我們的**templates**目錄之下的**registration**資料夾之下，所有需要的模板及文字檔案，如下表所示：







模板或檔案名稱	用途說明
registration_form.html	顯示註冊表單的網頁，預設使用form這個變數做為表單各欄位的內容
registration_complete.html	填寫完註冊表單按下送出之後顯示的訊息畫面
activation_complete.html	當帳號順利完成啟用時會顯示的頁面
activate.html	當帳號啟用失敗時會顯示的頁面
activation_email.txt	在寄送啟用信時使用的信件內容
activation_email_subject.txt	在寄送啟用信時使用的信件主旨

# 建立django-registration所需的模板

- 建立自動註冊功能的檔案放置處



File Explorer view showing the directory structure: templates > registration. Search results for 'registration' are displayed below.

名稱	修改日期	類型	大小
 activate	2016/8/3 下午 05...	Chrome HTML D...	1 KB
 activation_complete	2016/8/3 下午 04...	Chrome HTML D...	1 KB
 activation_email	2016/8/3 下午 04...	文字文件	1 KB
 activation_email_subject	2016/8/3 下午 04...	文字文件	1 KB
 registration_complete	2016/8/3 下午 04...	Chrome HTML D...	1 KB
 registration_form	2016/8/3 下午 03...	Chrome HTML D...	1 KB



## 建立django-registration所需的模板

- ▶ registration\_form.html的內容如下：
  - ▶ <!-- registration\_form.html (Registration01) -->
  - ▶ {% extends "base.html" %}
  - ▶ {% block title %}註冊分享日記{% endblock %}
  - ▶ {% block content %}
  - ▶ <div class='container'>
  - ▶ {% for message in messages %}
  - ▶ <div class='alert alert-  
{{message.tags}}'>{{ message }}</div>
  - ▶ {% endfor %}

# 建立django-registration所需的模板

- <div class='row'>
- <div class='col-md-12'>
- <div class='panel panel-default'>
- <div class='panel-heading' align=center>
- <h3>註冊分享日記網站</h3>
- </div>
- </div>
- </div>
- <form action='.' method='POST'>
- {% csrf\_token %}
- <table>
- {{ form.as\_table }}
- </table>
- <input type='submit' value='註冊'><br/>
- </form>
- </div>
- {% endblock %}

## 建立django-registration所需的模板

- 按下submit按鈕（註冊按鈕）之後，django-registraion自動會把註冊的帳號寫入資料庫中（這些動作會自動完成，不需要我們處理），把該帳號之`is_active`設定為`False`，接著即呼叫顯示`registration_complete.html`網頁，用以提醒使用者要回到電子郵件信箱中去收信，然後點擊連結做帳號啟用的動作
- 設計的`registration_complete.html`內容如下：
  - `<!-- registration_complete.html (Registration01) -->`
  - `{% extends "base.html" %}`
  - `{% block title %}分享日記{% endblock %}`
  - `{% block content %}`



## 建立django-registration所需的模板

- 啟用信的內容是什麼呢？由兩個檔案來決定
  - activation\_email\_subject.txt
  - activation\_email.txt
- activation\_email\_subject.txt，主旨使用只有簡單的一句話：「感謝您在分享日記註冊您的帳號，這是啟用信」就好了
- activation\_email.txt程式如下：
  - 感謝您的註冊
  - 使用者：{{ user }}
  - 在網站：{{ site }}註冊
  - 您的連結：  
[http://{{ site }}/accounts/activate/{{ activation\\_key }}](http://{{ site }}/accounts/activate/{{ activation_key }})
  - 將於{{ expiration\_days }}天後到期

# 建立django-registration所需的模板

➤ 剩下的2個檔案分別是在啟用成功與失敗時顯示的網頁，其中成功的網頁activation\_complete設計如下：

```
➤ <!-- activation_complete.html (Registration01) -->
➤ {% extends "base.html" %}
➤ {% block title %}分享日記{% endblock %}
➤ {% block content %}
➤ <div class='container'>
➤   {% for message in messages %}
➤     <div class='alert alert-{{message.tags}}'>{{ message }}</div>
➤   {% endfor %}
➤   <div class='row'>
➤     <div class='col-md-12'>
➤       <div class='panel panel-default'>
➤         <div class='panel-heading' align=center>
➤           <h3>帳號啟用成功，感謝您的註冊！</h3>
➤         </div>
➤       </div>
➤     </div>
➤   </div>
➤ </div>
➤ </div>
➤ {% endblock %}
```



# 建立django-registration所需的模板

- ▶ 啟用失敗的話會呼叫activate.html，內容如下：
  - ▶ `<!-- activate.html (Registration01) -->`
  - ▶ `{% extends "base.html" %}`
  - ▶ `{% block title %}分享日記{% endblock %}`
  - ▶ `{% block content %}`
  - ▶ `<div class='container'>`
  - ▶ `{% for message in messages %}`
  - ▶ `<div class='alert alert-{{message.tags}}'>{{ message }}</div>`
  - ▶ `{% endfor %}`
  - ▶ `<div class='row'>`
  - ▶ `<div class='col-md-12'>`
  - ▶ `<div class='panel panel-default'>`
  - ▶ `<div class='panel-heading' align=center>`
  - ▶ `<h3>啟用失敗，請檢查您的啟用連結，謝謝。</h3>`
  - ▶ `</div>`
  - ▶ `</div>`
  - ▶ `</div>`
  - ▶ `</div>`
  - ▶ `{% endblock %}`



# 整合使用者註冊功能到分享日記網站

- ▶ 使用django-registration的註冊畫面

註冊分享日記

← → ↻ localhost:8000/accounts/register/

分享日記 Home 登入 註冊 後台管理

## 註冊分享日記網站

Username:   
Required. 30 characters or fewer. Letters, digits and @/./+/-/\_ only.

E-mail:

Password:

Password confirmation:   
Enter the same password as above, for verification.

## 整合使用者註冊功能到分享日記網站

- 提醒使用者去檢查啟用電子郵件的畫面



# 整合使用者註冊功能到分享日記網站

## ► 啟用電子郵件的內容

感謝您在分享日記註冊您的帳號，這是啟用信

收件匣 x



**webmaster@localhost** 透過 drho.tw

寄給我 ▾

感謝您的註冊

使用者：Richard

在網站：localhost:8000註冊

您的連結：<http://localhost:8000/accounts/activate/IIJpY2hhcmQi:1bUswD:z6gUajilEZ3nkQG-29L-kjTalws>

將於7天後到期

# 整合使用者註冊功能到分享日記網站

- 網站帳號成功啟用的訊息畫面



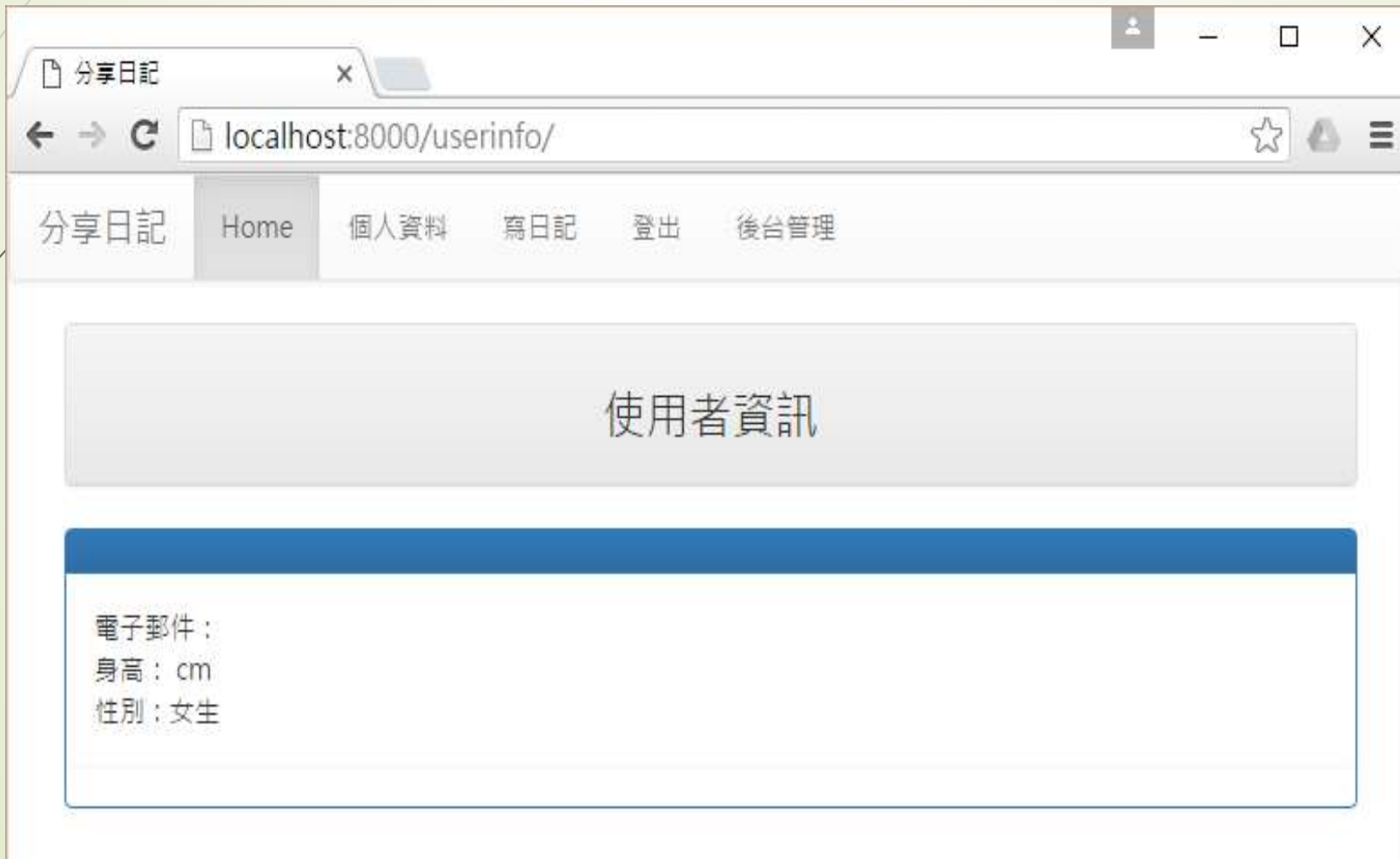
# 整合使用者註冊功能到分享日記網站

## ➤ 新帳號的登入畫面



# 整合使用者註冊功能到分享日記網站

- 新建立的使用者無法顯示出正確的個人資料



## 整合使用者註冊功能到分享日記網站

- 到此的範例網站等可以開放給使用者自由註冊，並在註冊之後即可馬上啟用，成為更實用的會員制網站了。
- 不過，在顯示個人資料的時候卻沒有出現正確的訊息，連電子郵件都不行
- 因為此網站個人資料是存放在Profile的資料表中
- 新註冊的使用者並沒有建立第一個使用者的Profile項目
- 新註冊的使用者並沒有相對應的Profile資料記錄



## 整合使用者註冊功能到分享日記網站

➤ 在 `forms.py` 中建立一個 `ModelForm` 名為 `ProfileForm`，如下所示：

➤ `class ProfileForm(forms.ModelForm):`

➤ `class Meta:`

➤ `model = models.Profile`

➤ `fields = ['height', 'male', 'website']`

➤ `def __init__(self, *args, **kwargs):`

➤ `super(ProfileForm, self).__init__(*args, **kwargs)`

➤ `self.fields['height'].label = '身高(cm)'`

➤ `self.fields['male'].label = '是男生嗎'`

➤ `self.fields['website'].label = '個人網站'`

# 整合使用者註冊功能到分享日記網站

- 為了能夠顯示使用者訊息（命名為profile）以及修改用的表單（命名為profile\_form），新版的userinfo.html要修改為如下所示的內容：

- <!-- userinfo.html (ch10www project) -->
- {% extends "base.html" %}
- {% block title %}分享日記{% endblock %}
- {% block content %}
- <div class='container'>
- {% for message in messages %}
- <div class='alert alert-{{message.tags}}'>{{ message }}</div>
- {% endfor %}
- <div class='row'>
- <div class='col-md-12'>
- <div class='panel panel-default'>
- <div class='panel-heading' align=center>
- <h3>使用者資訊</h3>
- </div>
- </div>
- </div>
- </div>

# 整合使用者註冊功能到分享日記網站

```

- <div class='row'>
-   <div class='col-md-12'>
-     <div class='panel panel-primary'>
-       <div class='panel-heading' align=center>
-         {{ profile.user.username | upper }}
-       </div>
-       <div class='panel panel-body'>
-         電子郵件 : {{ profile.user.email }}<br/>
-         身高 : {{ profile.height }} cm<br/>
-         性別 : {{ profile.male | yesno:"男生,女生"}}<br/>
-         個人網站 : <a href='{{ profile.website }}'>{{ profile.website }}</a>
-       </div>
-     </div>
-   </div>
- </div>
- <form name='myname' action='.' method='POST'>
-   {% csrf_token %}
-   <table>
-     {{ profile_form.as_table }}
-   </table>
-   <input type='submit' value='修改個人資料'>
- </form>
- </div>
- {% endblock %}

```

# 整合使用者註冊功能到分享日記網站

## ➡ 新版的使用者資訊

分享日記 x

localhost:8000/userinfo/

分享日記 Home 個人資料 寫日記 登出 後台管理

使用者資訊

RICHARD

電子郵件 : p179457@gmail.com  
身高 : 174 cm  
性別 : 男生  
個人網站 : <http://hophd.com>

身高(cm):

是男生嗎:

個人網站:

修改個人資料

## 整合使用者註冊功能到分享日記網站

- 上圖在中間顯示使用者資訊之下，再加上一個表單，此表單即為 ProfileForm 的執行實例
- 表單中只要填入想要修改的資料，表單即會交由 views.userinfo 處理
- 以下是 views.userinfo 的程式碼片段：
  - @login\_required(login\_url='/login/')
  - def userinfo(request):
  - if request.user.is\_authenticated:
  - username = request.user.username
  - user = User.objects.get(username=username)
  - try:
  - profile = models.Profile.objects.get(user=user)
  - except:
  - profile = models.Profile(user=user)

## 整合使用者註冊功能到分享日記網站

- ▶ `if request.method == 'POST':`
- ▶ `profile_form = forms.ProfileForm(request.POST, instance=profile)`
- ▶ `if profile_form.is_valid():`
- ▶ `messages.add_message(request, messages.INFO, "個人資料已儲存")`
- ▶ `profile_form.save()`
- ▶ `return HttpResponseRedirect('/userinfo')`
- ▶ `else:`
- ▶ `messages.add_message(request, messages.INFO, '要修改個人資料，每一個欄位都要填...')`
- ▶ `else:`
- ▶ `profile_form = forms.ProfileForm()`
- ▶ `return render(request, 'userinfo.html', locals())`



# Pythonanywhere.com免費Python網站開發環境

29

- 接下來要為我們的網站加入第三方網站的驗證機制，也就是使用Facebook或是Google+等知名的網站幫我們驗證使用者的身份，如此使用者就不需要為了加入我們的網站而另外註冊帳號和設定密碼，這是目前會員網站最流行也是最方便的方式。為了方便示範起見，在本節先從如何免費建立有自有網址的Django網站開始。

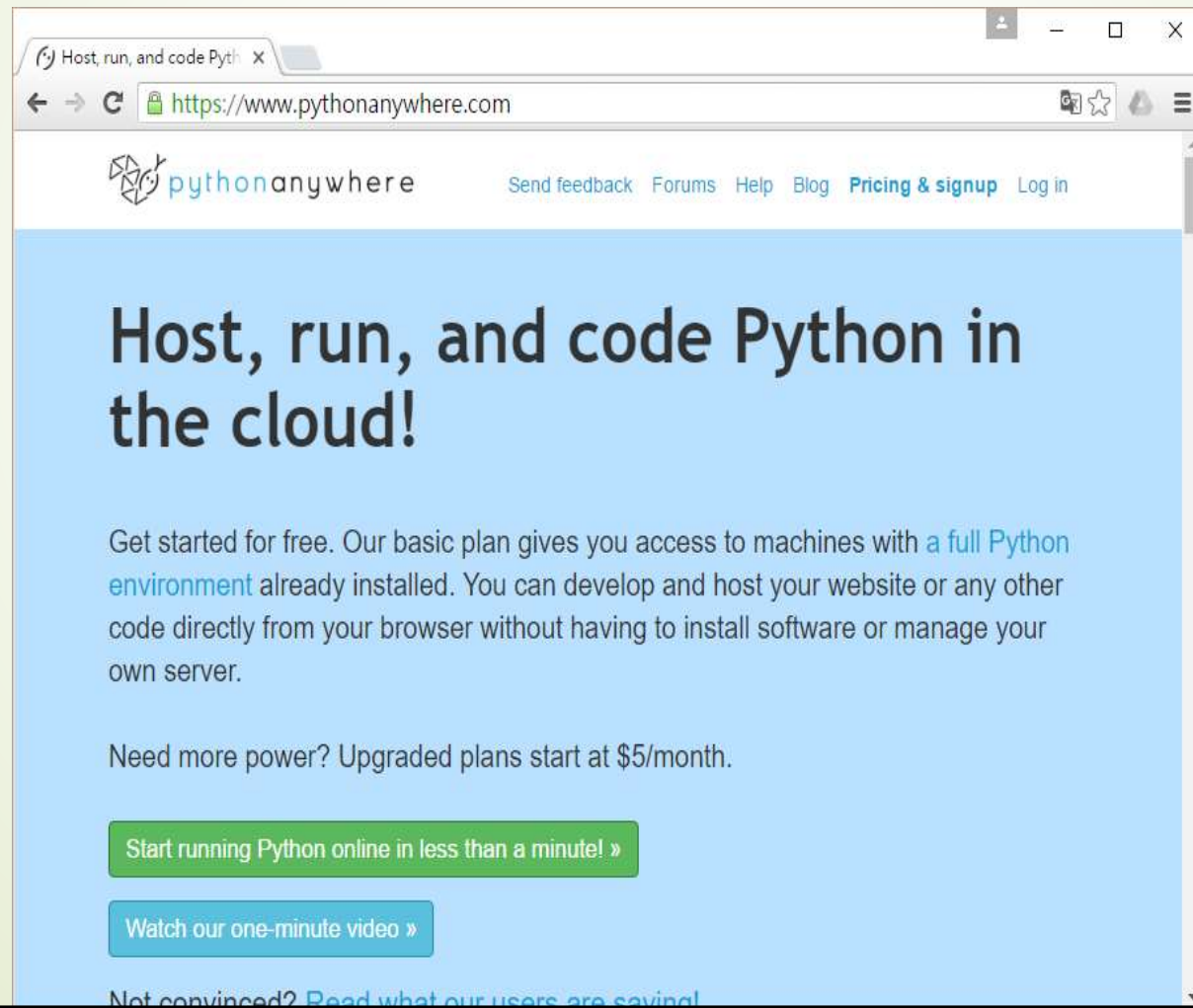


## 註冊Pythonanywhere.com帳號

- 讓學習者或開發人員可以在線上直接編輯以及執行Python程式的網站
- 提供Bash（作業系統終端機）環境，可以直接在終端機中操作Python以及Django程式
- 提供MySQL伺服器以及網站寄存服務，從開發到上線部署一站搞定

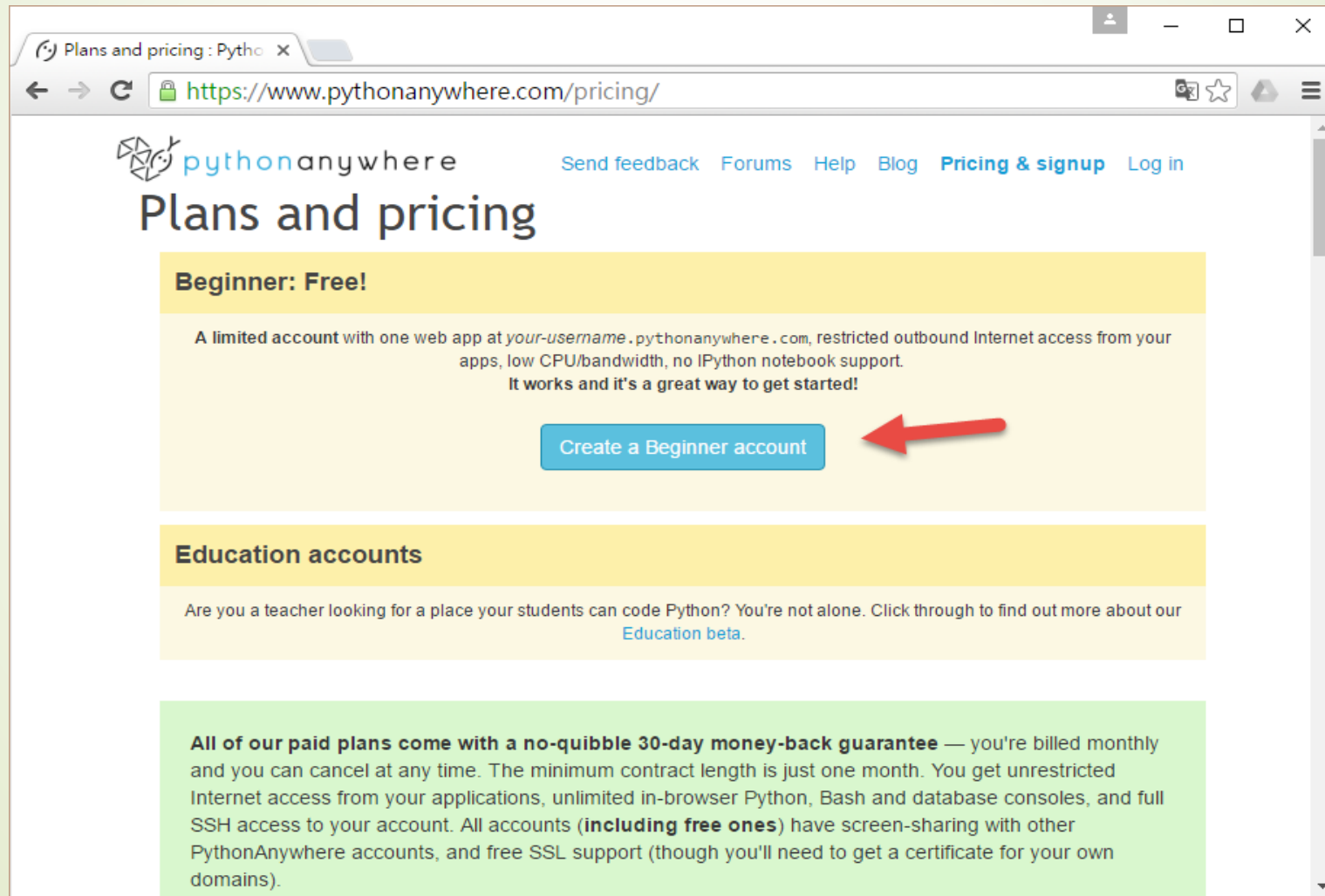
# 註冊Pythonanywhere.com帳號

## ➡ Pythonanywhere.com首頁



# 註冊Pythonanywhere.com帳號

- ▶ 點擊右上角的Pricing & signup，在Pythonanywhere.com中建立免費帳號



Plans and pricing: Python x

← → ↻ <https://www.pythonanywhere.com/pricing/> ☆ 🗑️ ☰

pythonanywhere [Send feedback](#) [Forums](#) [Help](#) [Blog](#) [Pricing & signup](#) [Log in](#)

## Plans and pricing

**Beginner: Free!**

A limited account with one web app at *your-username.pythonanywhere.com*, restricted outbound Internet access from your apps, low CPU/bandwidth, no IPython notebook support.  
It works and it's a great way to get started!

[Create a Beginner account](#)

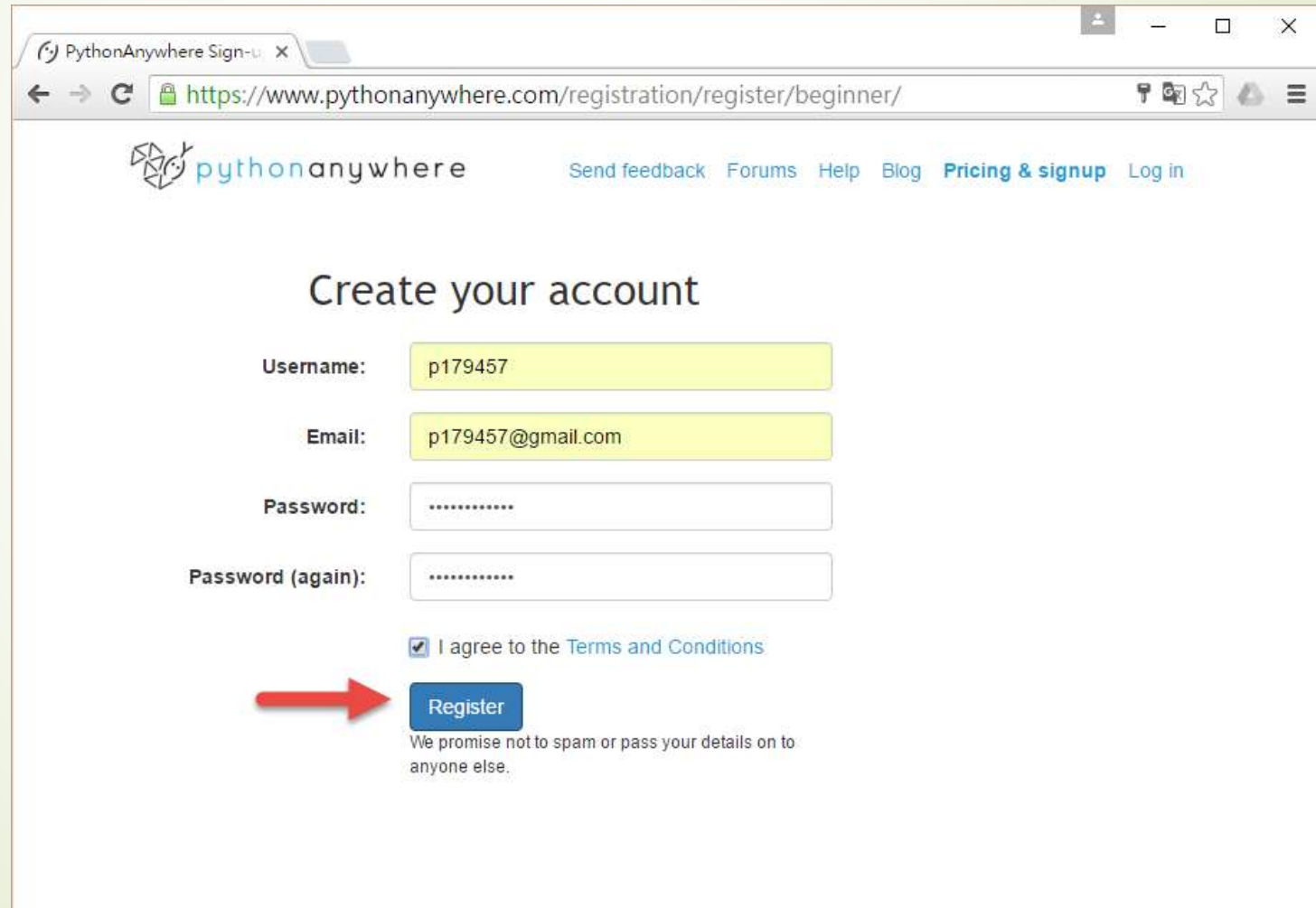
**Education accounts**

Are you a teacher looking for a place your students can code Python? You're not alone. Click through to find out more about our [Education beta](#).

All of our paid plans come with a no-quibble 30-day money-back guarantee — you're billed monthly and you can cancel at any time. The minimum contract length is just one month. You get unrestricted Internet access from your applications, unlimited in-browser Python, Bash and database consoles, and full SSH access to your account. All accounts (**including free ones**) have screen-sharing with other PythonAnywhere accounts, and free SSL support (though you'll need to get a certificate for your own domains).

# 註冊Pythonanywhere.com帳號

- 按下「Create a Beginner account」按鈕輸入註冊用的資料



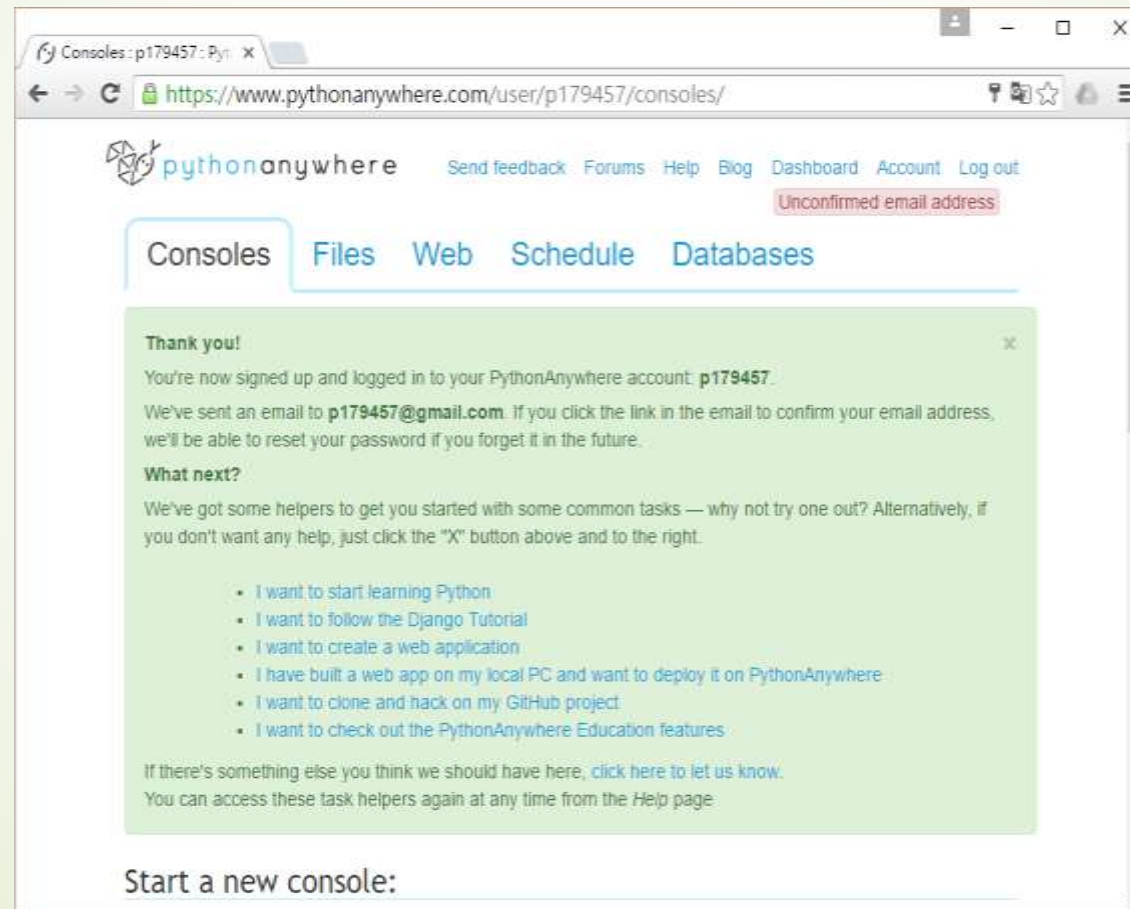
The screenshot shows a web browser window with the URL <https://www.pythonanywhere.com/registration/register/beginner/>. The page title is "Create your account". The form contains the following fields:

- Username:** p179457
- Email:** p179457@gmail.com
- Password:** .....
- Password (again):** .....

Below the password fields, there is a checkbox labeled "I agree to the [Terms and Conditions](#)" which is checked. A red arrow points to the "Register" button. Below the button, there is a small disclaimer: "We promise not to spam or pass your details on to anyone else."

# 註冊Pythonanywhere.com帳號

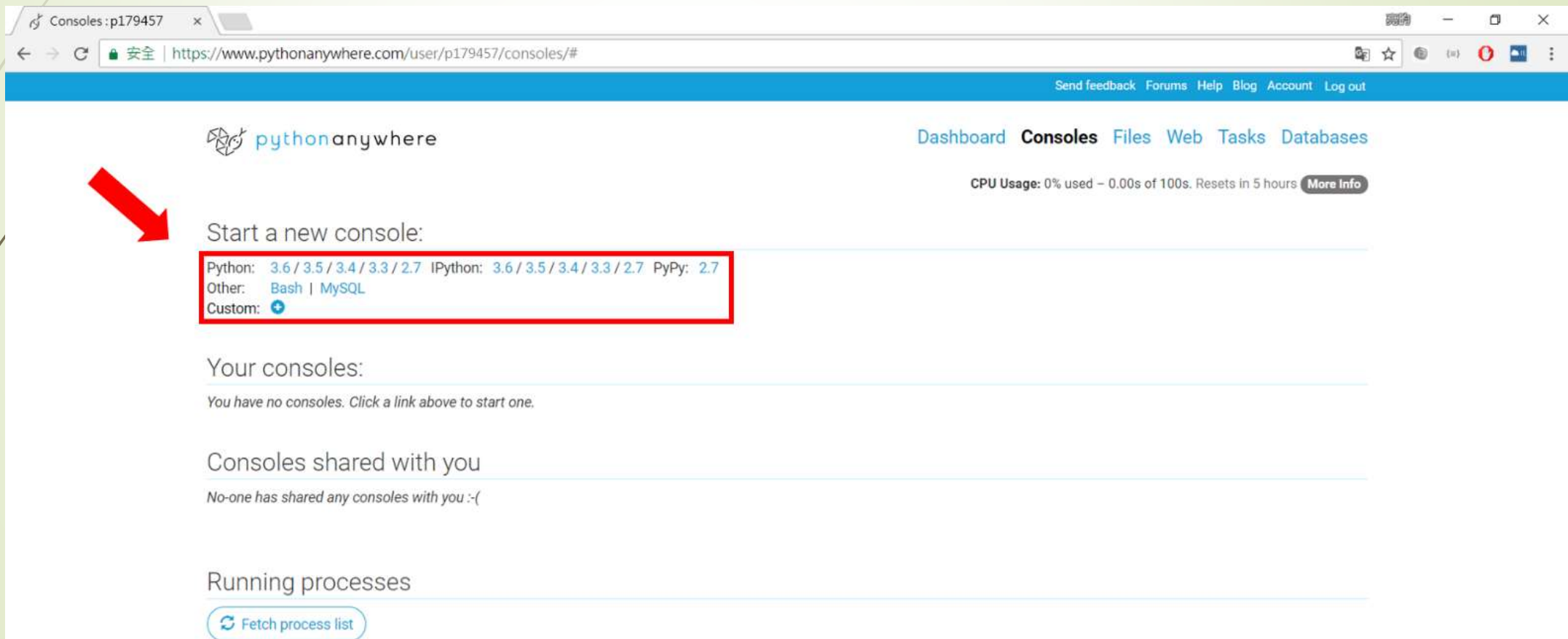
- 在按下註冊按鈕之後，即進入下圖





# 註冊Pythonanywhere.com帳號

- Pythonanywhere.com主畫面，方框框起來的地方就是可以使用的Shell種類，包括各種版本的Python Shell



The screenshot shows the Pythonanywhere.com user dashboard for user p179457. The page title is "Consoles : p179457" and the URL is "https://www.pythonanywhere.com/user/p179457/consoles/#". The dashboard includes navigation links for "Send feedback", "Forums", "Help", "Blog", "Account", and "Log out". The main content area features the Pythonanywhere logo and a navigation menu with "Dashboard", "Consoles", "Files", "Web", "Tasks", and "Databases". A "CPU Usage" indicator shows "0% used - 0.00s of 100s. Resets in 5 hours" with a "More info" button. The "Start a new console:" section is highlighted with a red box and a red arrow pointing to it. This section lists available shell options: "Python: 3.6 / 3.5 / 3.4 / 3.3 / 2.7", "IPython: 3.6 / 3.5 / 3.4 / 3.3 / 2.7", "PyPy: 2.7", "Other: Bash | MySQL", and "Custom: +". Below this, the "Your consoles:" section states "You have no consoles. Click a link above to start one." The "Consoles shared with you" section states "No-one has shared any consoles with you :-(". The "Running processes" section includes a "Fetch process list" button.

# 註冊Pythonanywhere.com帳號

## Files頁籤的內容

pythonanywhere

Dashboard Consoles **Files** Web Tasks Databases

/home/ p179457 Open Bash console here **14% full** – 72.6 MB of your 512.0 MB quota

Directories

Enter new directory name New directory

- .cache/
- .local/
- .virtualenvs/
- VENV/
- mvote/

Files

Enter new file name, eg hello.py New file

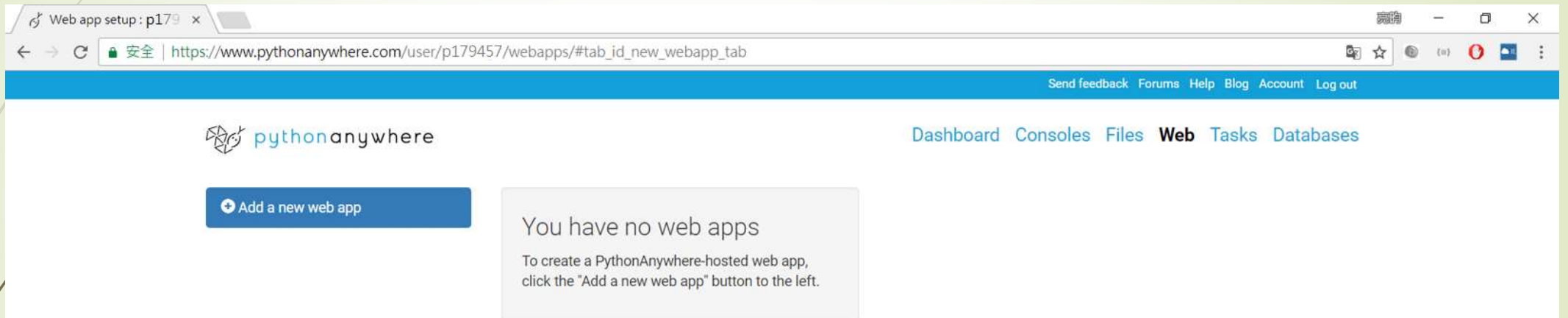
.bash_history	2017-11-18 22:09	992 bytes
.bashrc	2017-09-18 02:26	559 bytes
.gitconfig	2017-10-20 01:53	318 bytes
.profile	2017-09-18 02:26	79 bytes
.pythonstartup.py	2017-09-18 02:26	77 bytes
.vimrc	2017-09-18 02:26	4.6 KB
README.txt	2017-09-18 02:26	235 bytes
mysite.zip	2017-09-18 02:27	18.9 KB
scap.py	2017-11-18 12:45	309 bytes
test.py	2017-11-17 03:11	222 bytes

Upload a file



# 註冊Pythonanywhere.com帳號

## Web頁籤的內容



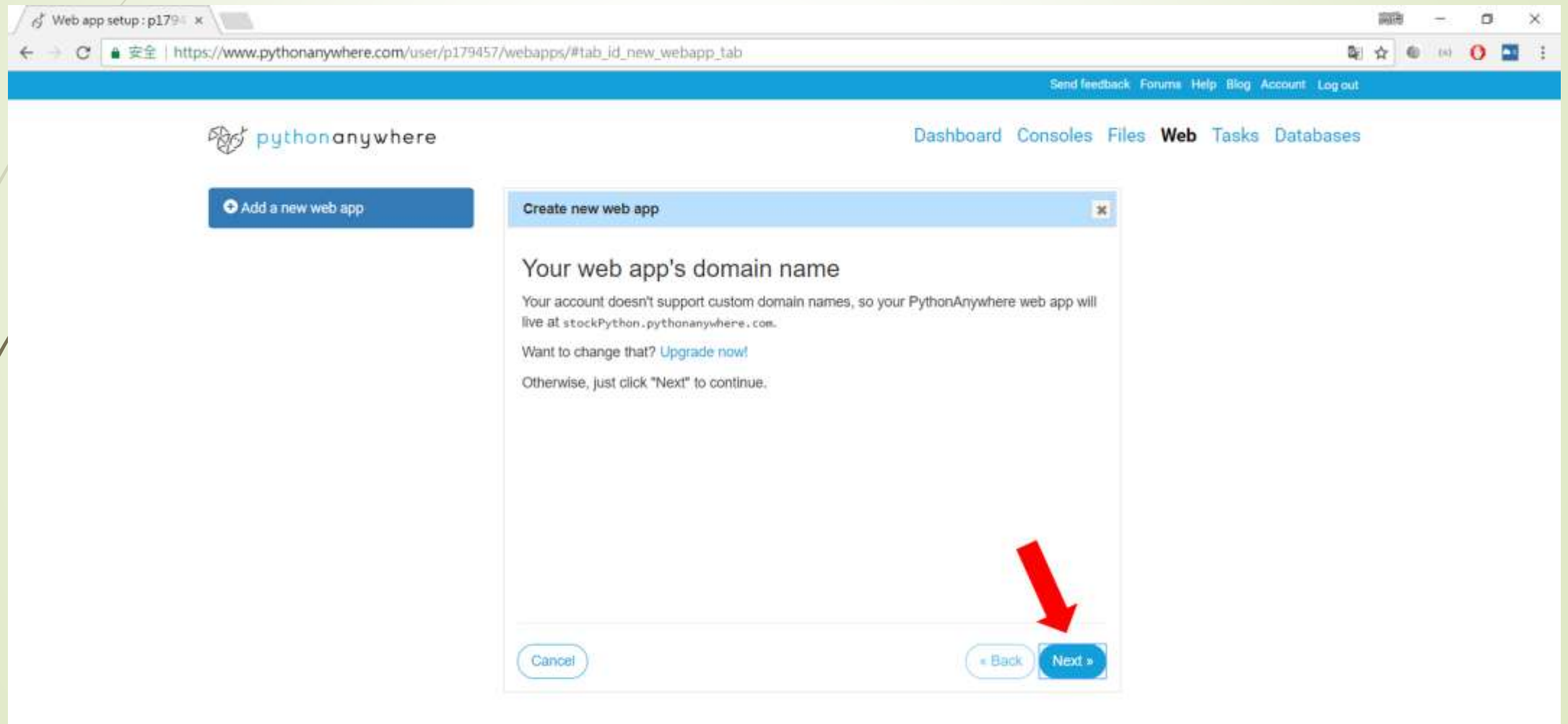
The screenshot shows a web browser window with the URL [https://www.pythonanywhere.com/user/p179457/webapps/#tab\\_id\\_new\\_webapp\\_tab](https://www.pythonanywhere.com/user/p179457/webapps/#tab_id_new_webapp_tab). The page features the PythonAnywhere logo and a navigation menu with links for Dashboard, Consoles, Files, Web, Tasks, and Databases. A prominent blue button labeled "Add a new web app" is visible. A message box states: "You have no web apps. To create a PythonAnywhere-hosted web app, click the 'Add a new web app' button to the left."

## 註冊Pythonanywhere.com帳號

- 點擊「[Add a new web app](#)」按鈕，即可開始建立免費網站步驟（第一個免費）
- 第一個步驟可以設定domain名稱，不過免費的帳號只能使用自己的帳號ID當作是網址，因此直接點擊「Next」進入下一個步驟

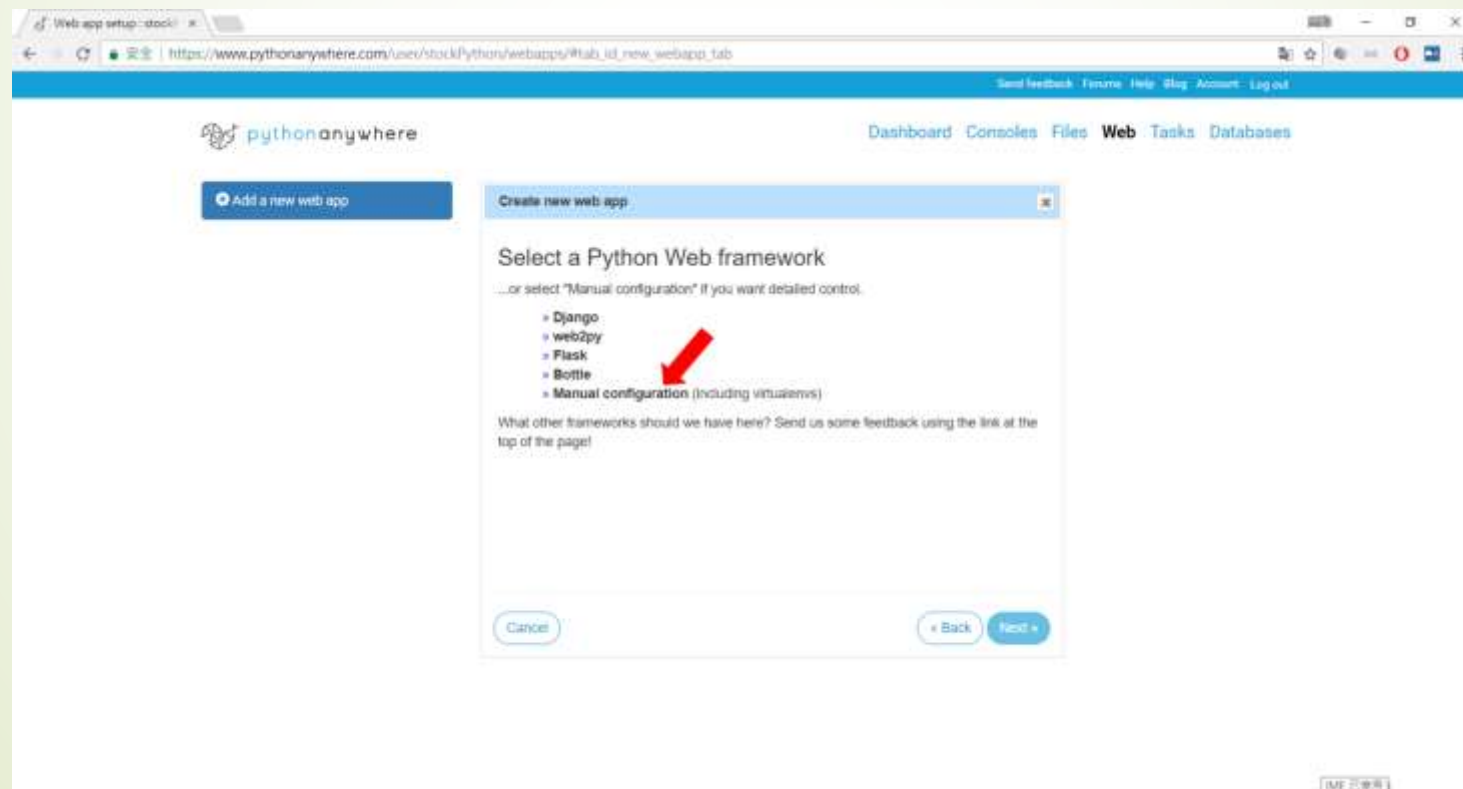
# 註冊Pythonanywhere.com帳號

➤ 在Pythonanywhere.com建立網站的第一個步驟



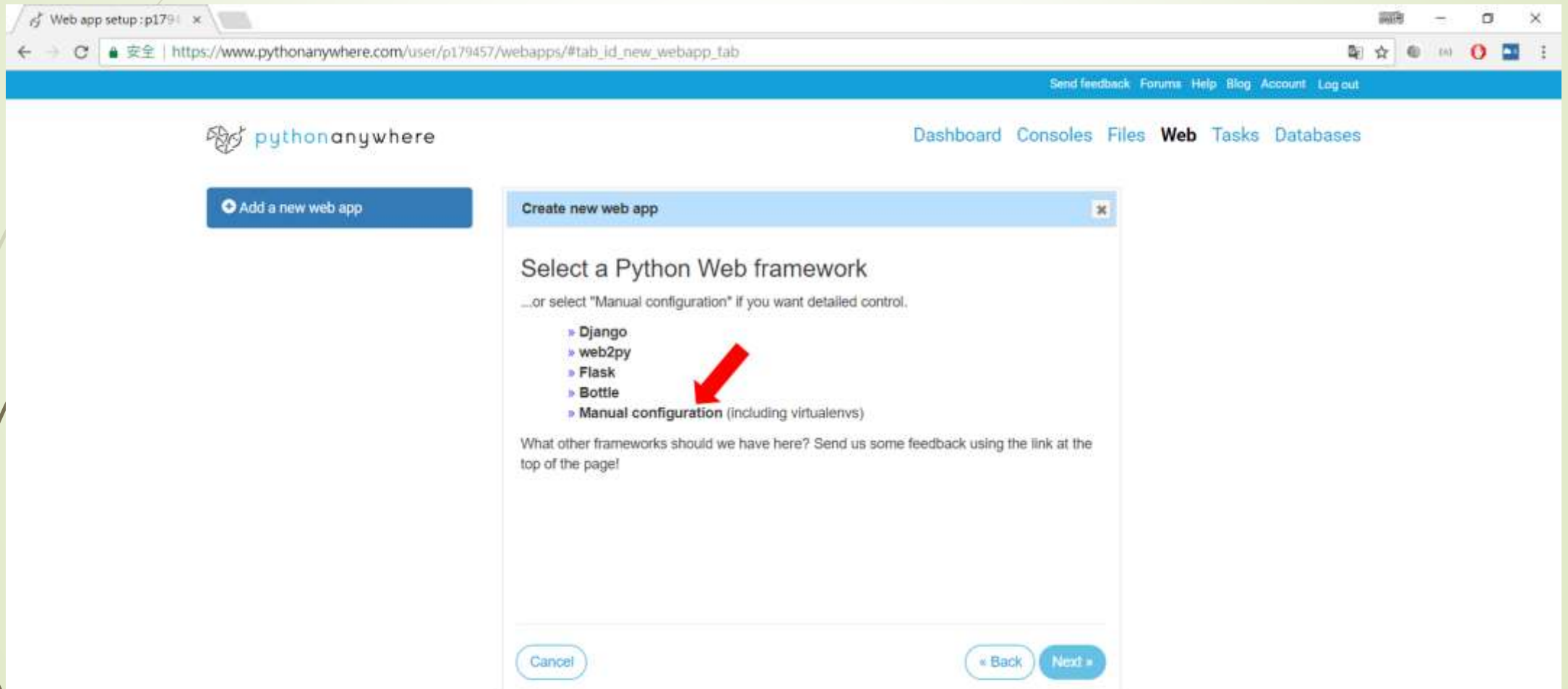
# 註冊Pythonanywhere.com帳號

- 第二個步驟中可以指定要使用的網站Framework
- 我們使用「Manual configuration」，用手動的方式自行加入 Django Framework



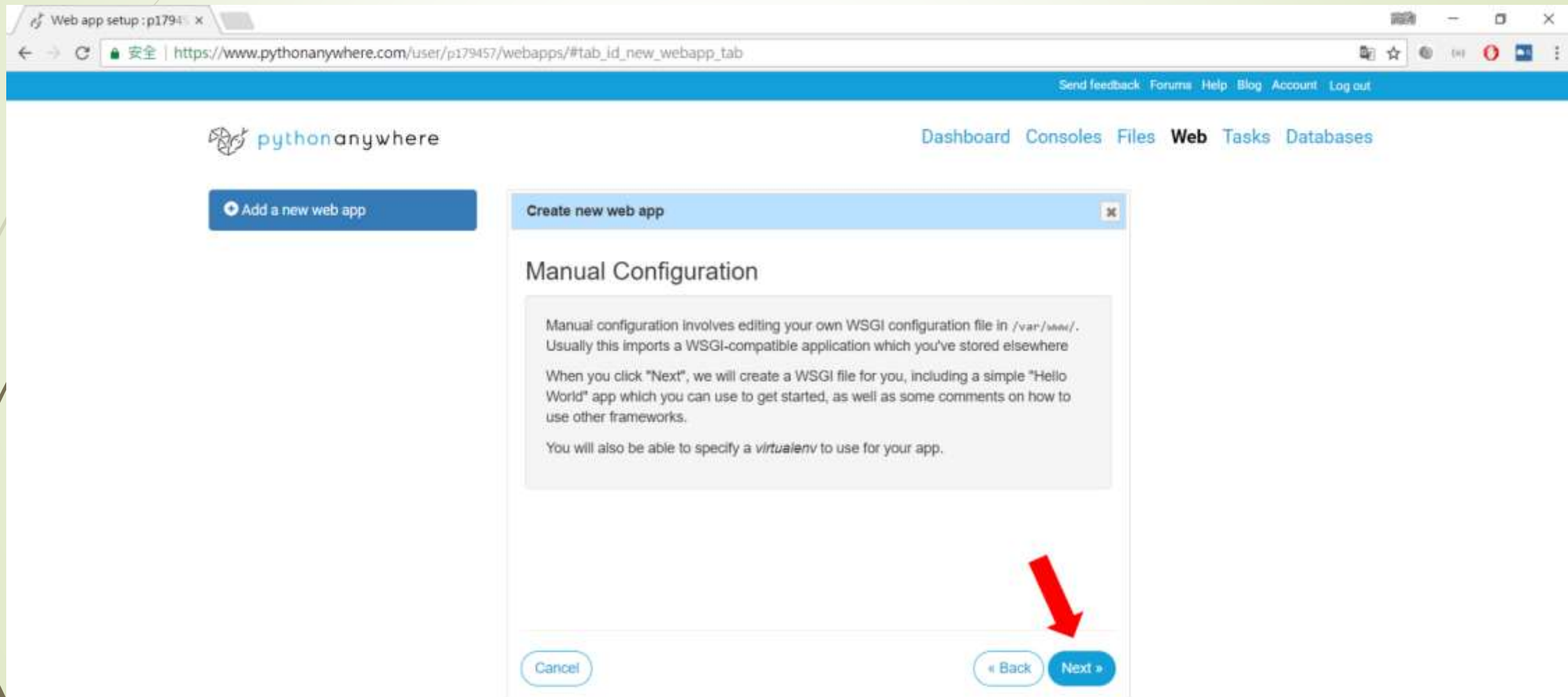
# 註冊Pythonanywhere.com帳號

- 設定網站要使用的Python版本，選用Python 3.5



# 註冊Pythonanywhere.com帳號

➡ 使用Manual Configuration之最後說明頁



The screenshot shows a web browser window with the URL `https://www.pythonanywhere.com/user/p179457/webapps/#tab_id_new_webapp_tab`. The page title is "Web app setup: p179457". The browser's address bar shows "安全" (Secure) and the URL. The page content includes the PythonAnywhere logo, a navigation menu with "Dashboard", "Consoles", "Files", "Web", "Tasks", and "Databases", and a "Send feedback" link. A blue button labeled "Add a new web app" is visible. A modal window titled "Create new web app" is open, displaying the "Manual Configuration" section. The text in the modal explains that manual configuration involves editing a WSGI configuration file in `/var/www/` and that clicking "Next" will create a WSGI file for a "Hello World" app. A red arrow points to the "Next" button at the bottom right of the modal.

Web app setup: p179457 x

安全 | [https://www.pythonanywhere.com/user/p179457/webapps/#tab\\_id\\_new\\_webapp\\_tab](https://www.pythonanywhere.com/user/p179457/webapps/#tab_id_new_webapp_tab)

Send feedback Forums Help Blog Account Log out

pythonanywhere

Dashboard Consoles Files **Web** Tasks Databases

Add a new web app

Create new web app

### Manual Configuration

Manual configuration involves editing your own WSGI configuration file in `/var/www/`. Usually this imports a WSGI-compatible application which you've stored elsewhere

When you click "Next", we will create a WSGI file for you, including a simple "Hello World" app which you can use to get started, as well as some comments on how to use other frameworks.

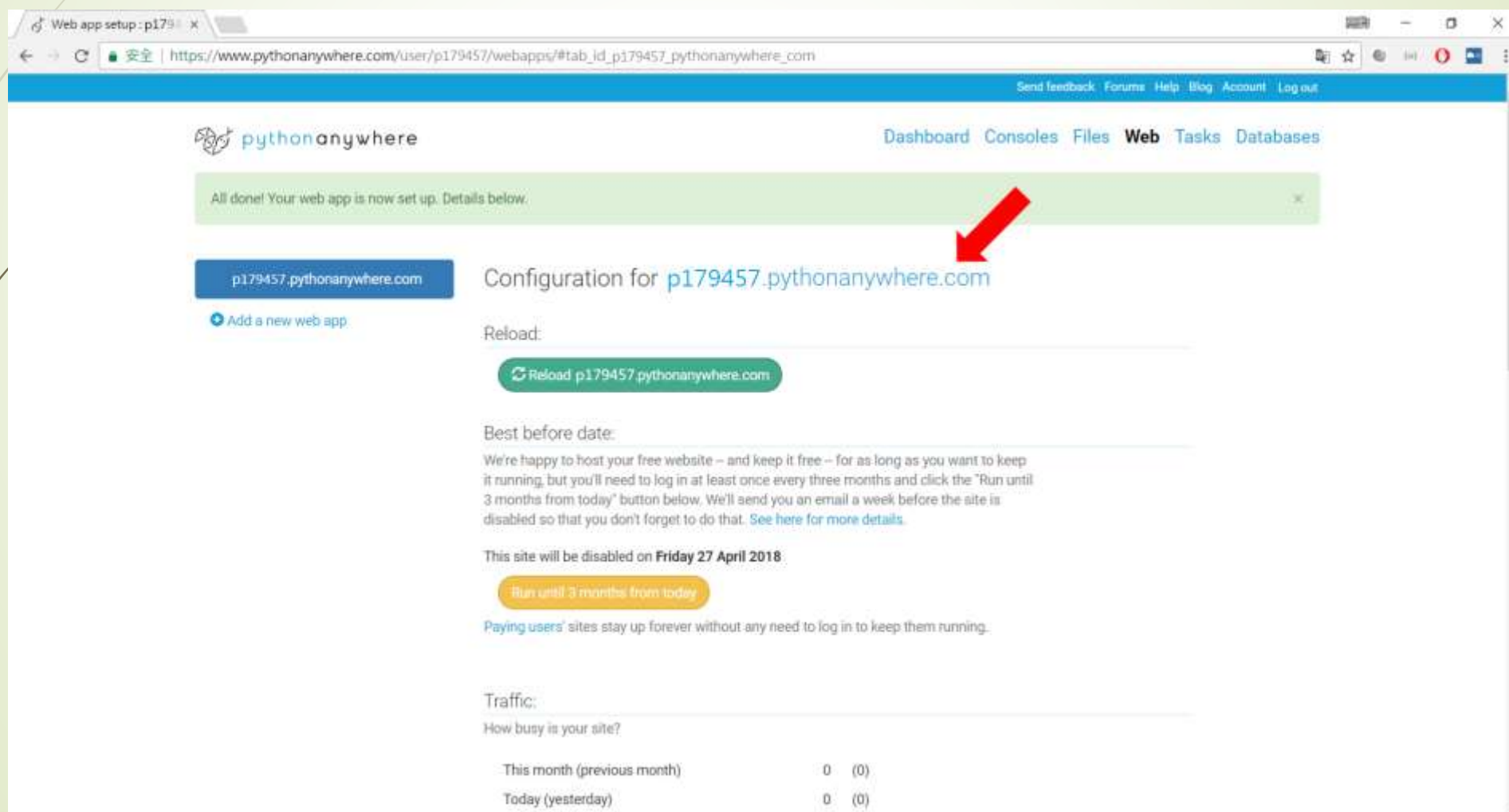
You will also be able to specify a *virtualenv* to use for your app.

Cancel « Back Next »



# 註冊Pythonanywhere.com帳號

## ➡ 網站建立完成後的畫面

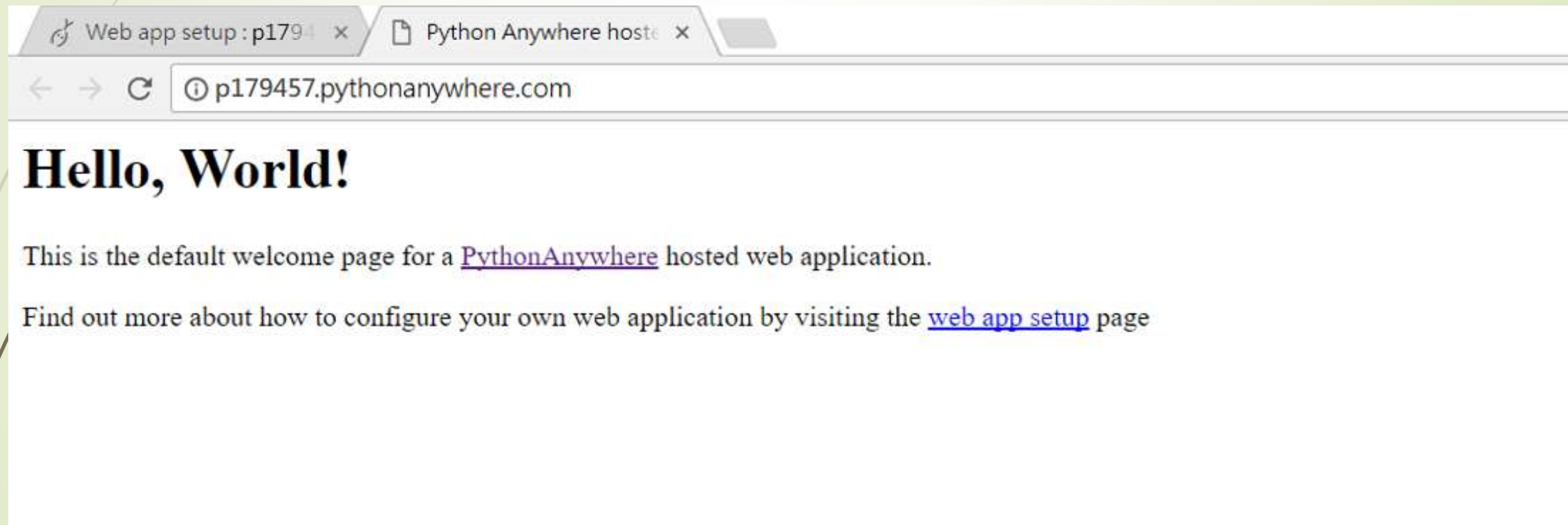


The screenshot shows the PythonAnywhere dashboard for a user. The browser address bar indicates the URL: [https://www.pythonanywhere.com/user/p179457/webapps/#tab\\_id\\_p179457\\_pythonanywhere\\_com](https://www.pythonanywhere.com/user/p179457/webapps/#tab_id_p179457_pythonanywhere_com). The dashboard features a blue navigation bar with links for "Send feedback", "Forums", "Help", "Blog", "Account", and "Log out". Below the navigation bar, the PythonAnywhere logo is on the left, and a menu with "Dashboard", "Consoles", "Files", "Web", "Tasks", and "Databases" is on the right. A green notification box at the top states: "All done! Your web app is now set up. Details below." Below this, a blue button displays the domain "p179457.pythonanywhere.com" and a link to "Add a new web app". The main content area is titled "Configuration for p179457.pythonanywhere.com" and includes a "Reload" section with a green "Reload p179457.pythonanywhere.com" button. The "Best before date" section explains the free hosting policy and includes a yellow "Run until 3 months from today" button. A warning states "This site will be disabled on Friday 27 April 2018". The "Traffic" section shows a table of site activity:

How busy is your site?	
This month (previous month)	0 (0)
Today (yesterday)	0 (0)

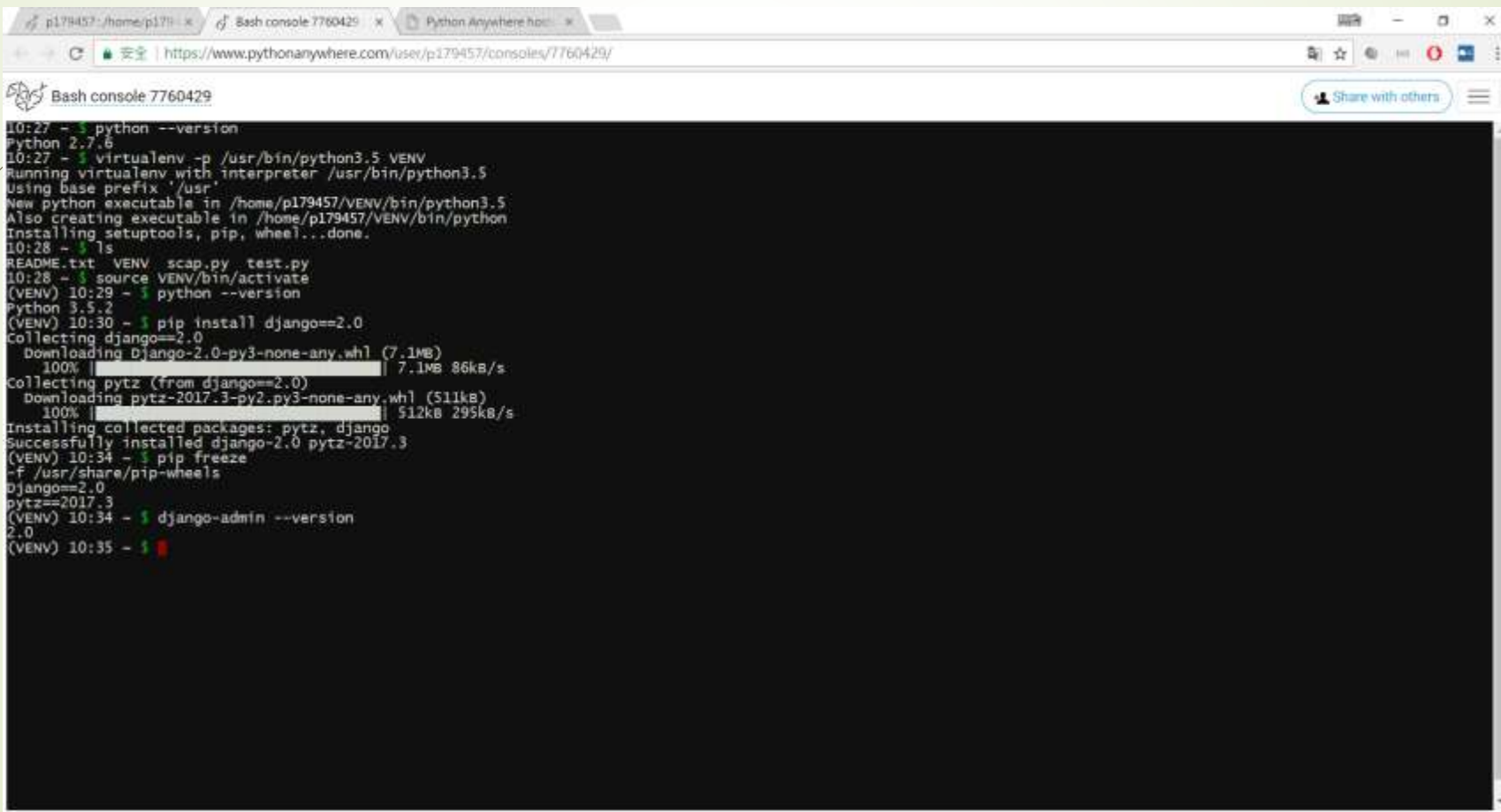
# 註冊Pythonanywhere.com帳號

- 選用Manual Configuration建立的預設網站首頁畫面 )



# 在Pythonanywhere免費網站中建立虛擬環境 以及Django網站

- ➔ 進入Bash終端機環境，透過指令操作安裝虛擬環境virtualenv以及Django



```
10:27 ~ $ python --version
Python 2.7.6
10:27 ~ $ virtualenv -p /usr/bin/python3.5 VENV
Running virtualenv with interpreter /usr/bin/python3.5
Using base prefix '/usr'
New python executable in /home/p179457/VENV/bin/python3.5
Also creating executable in /home/p179457/VENV/bin/python
Installing setuptools, pip, wheel...done.
10:28 ~ $ ls
README.txt VENV scap.py test.py
10:28 ~ $ source VENV/bin/activate
(VENV) 10:29 ~ $ python --version
Python 3.5.2
(VENV) 10:30 ~ $ pip install django==2.0
Collecting django==2.0
  Downloading Django-2.0-py3-none-any.whl (7.1MB)
    100% |#####| 7.1MB 86kB/s
Collecting pytz (from django==2.0)
  Downloading pytz-2017.3-py2.py3-none-any.whl (511kB)
    100% |#####| 512kB 295kB/s
Installing collected packages: pytz, django
Successfully installed django-2.0 pytz-2017.3
(VENV) 10:34 ~ $ pip freeze
-f /usr/share/pip-wheels
django==2.0
pytz==2017.3
(VENV) 10:34 ~ $ django-admin --version
2.0
(VENV) 10:35 ~ $
```

## 在Pythonanywhere免費網站中建立虛擬環境以及Django網站

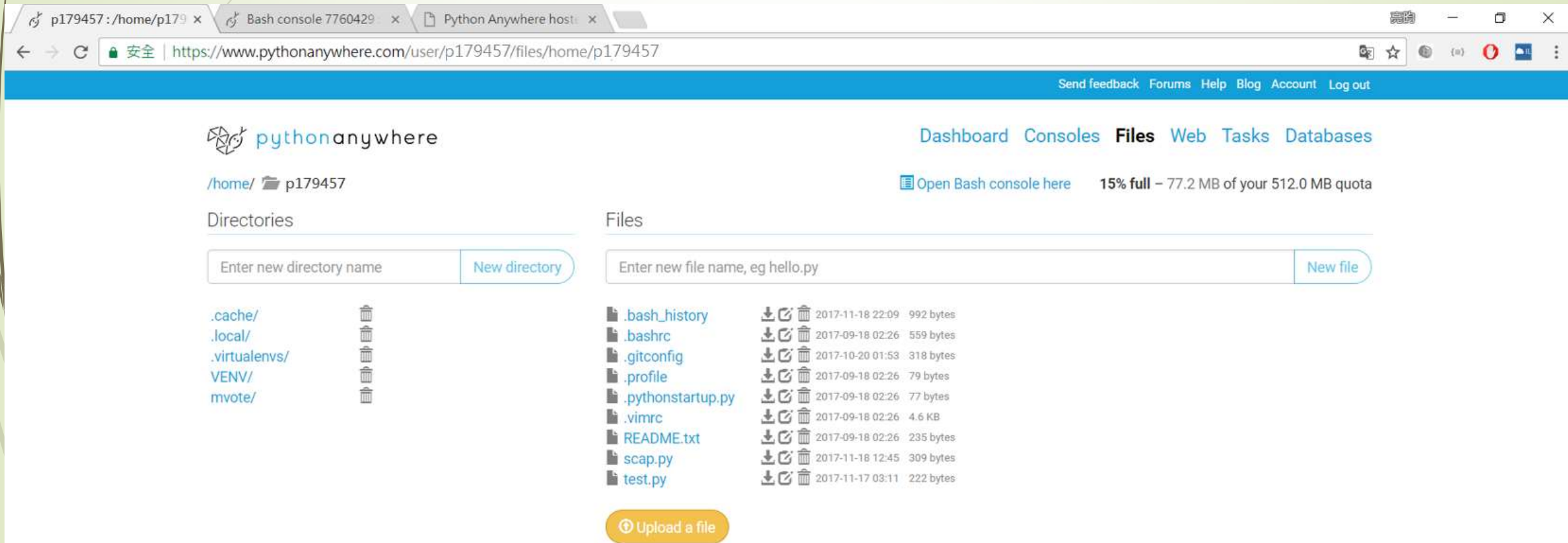
- 以下的指令即可啟用（ 假設我們要建立的網站叫做mvote ）：
  - (VENV) 10:35 ~ \$ django-admin startproject mvote
  - (VENV) 10:39 ~ \$ cd mvote
  - (VENV) 10:39 ~ /mvote \$ python manage.py startapp mysite
  - (VENV) 10:41 ~ /mvote \$ cd ..

- (VENV) 10:41 ~ \$ tree mvote
- mvote
- |— manage.py
- |— mvote
- | |— \_\_init\_\_.py
- | |— \_\_pycache\_\_
- | | |— \_\_init\_\_.cpython-35.pyc
- | | |— settings.cpython-35.pyc
- | |— settings.py
- | |— urls.py
- | |— wsgi.py
- |— mysite
- | |— \_\_init\_\_.py
- | |— admin.py
- | |— apps.py
- | |— migrations
- | | |— \_\_init\_\_.py
- | |— models.py
- | |— tests.py
- | |— views.py
- 4 directories, 14 files



# 在Pythonanywhere免費網站中建立虛擬環境以及Django網站

➔ 建立了Django的Project之後的檔案列表



The screenshot shows the PythonAnywhere dashboard for user p179457. The page displays the file listing for the directory /home/p179457. The dashboard includes navigation links for Dashboard, Consoles, Files, Web, Tasks, and Databases. A quota indicator shows 15% full (77.2 MB of 512.0 MB). The file listing includes a table of files and directories with their creation dates and sizes.

pythonanywhere

Dashboard Consoles **Files** Web Tasks Databases

/home/ p179457

Open Bash console here 15% full – 77.2 MB of your 512.0 MB quota

Directories

Enter new directory name

- .cache/
- .local/
- .virtualenvs/
- VENV/
- mvote/

Files

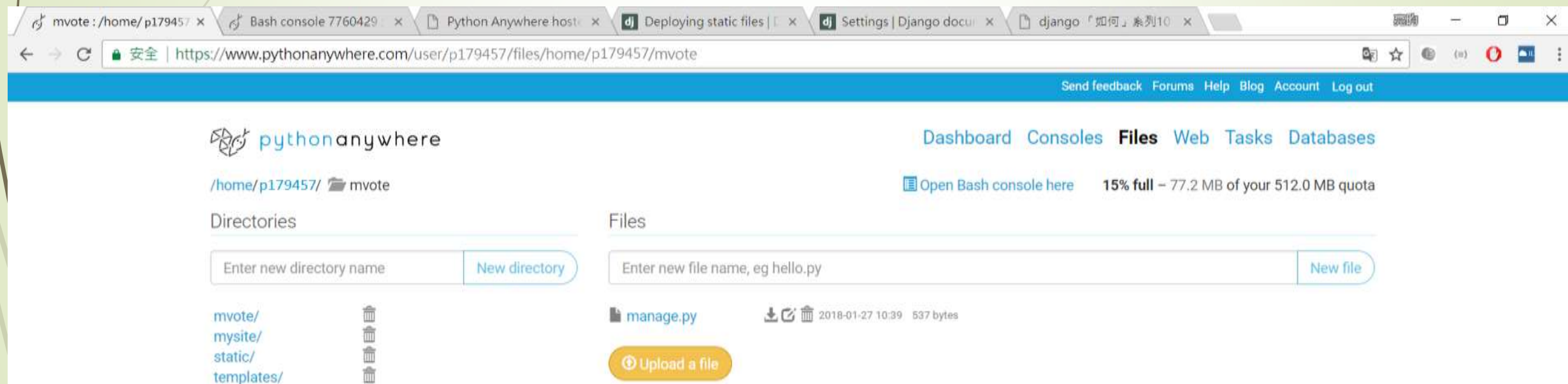
Enter new file name, eg hello.py

.bash_history	2017-11-18 22:09	992 bytes
.bashrc	2017-09-18 02:26	559 bytes
.gitconfig	2017-10-20 01:53	318 bytes
.profile	2017-09-18 02:26	79 bytes
.pythonstartup.py	2017-09-18 02:26	77 bytes
.vimrc	2017-09-18 02:26	4.6 KB
README.txt	2017-09-18 02:26	235 bytes
scap.py	2017-11-18 12:45	309 bytes
test.py	2017-11-17 03:11	222 bytes



# 在Pythonanywhere免費網站中建立虛擬環境以及Django網站

➔ 建立templates等資料夾所在的位置



The screenshot shows the PythonAnywhere web interface. The browser address bar displays the URL <https://www.pythonanywhere.com/user/p179457/files/home/p179457/mvote>. The page header includes navigation links: [Send feedback](#), [Forums](#), [Help](#), [Blog](#), [Account](#), and [Log out](#). The main content area is divided into two sections: **Directories** and **Files**. The **Directories** section has a text input field for "Enter new directory name" and a "New directory" button. Below it, a list of existing directories is shown: `mvote/`, `mysite/`, `static/`, and `templates/`, each with a trash icon. The **Files** section has a text input field for "Enter new file name, eg hello.py" and a "New file" button. Below it, a file named `manage.py` is listed with a download icon, a trash icon, the date `2018-01-27 10:39`, and the size `537 bytes`. A yellow "Upload a file" button is also present. The top right of the page shows navigation tabs: [Dashboard](#), [Consoles](#), [Files](#) (active), [Web](#), [Tasks](#), and [Databases](#). A status message indicates "15% full - 77.2 MB of your 512.0 MB quota".

# 在Pythonanywhere免費網站中建立虛擬環境以及Django網站

The screenshot shows the Pythonanywhere dashboard for a user's account. The browser address bar indicates the URL: `https://www.pythonanywhere.com/user/stockPython/files/home/stockPython/mvote/mysite`. The dashboard includes a navigation bar with links for `Send feedback`, `Forums`, `Help`, `Blog`, `Account`, and `Log out`. The main content area is divided into two sections: **Directories** and **Files**.

**Directories:** A form to create a new directory is visible, with the input field containing "Enter new directory name" and a "New directory" button. Below the form, a list of existing directories is shown:

- migrations/
- static/
- templates/

**Files:** A form to create a new file is visible, with the input field containing "Enter new file name, eg hello.py" and a "New file" button. Below the form, a list of existing files is shown:

File Name	Download	Share	Delete	Date	Size
<code>__init__.py</code>	↓	↗	🗑️	2018-01-27 10:41	0 bytes
<code>admin.py</code>	↓	↗	🗑️	2018-01-27 10:41	63 bytes
<code>apps.py</code>	↓	↗	🗑️	2018-01-27 10:41	87 bytes
<code>models.py</code>	↓	↗	🗑️	2018-01-27 10:41	57 bytes
<code>tests.py</code>	↓	↗	🗑️	2018-01-27 10:41	60 bytes
<code>views.py</code>	↓	↗	🗑️	2018-01-27 10:41	63 bytes

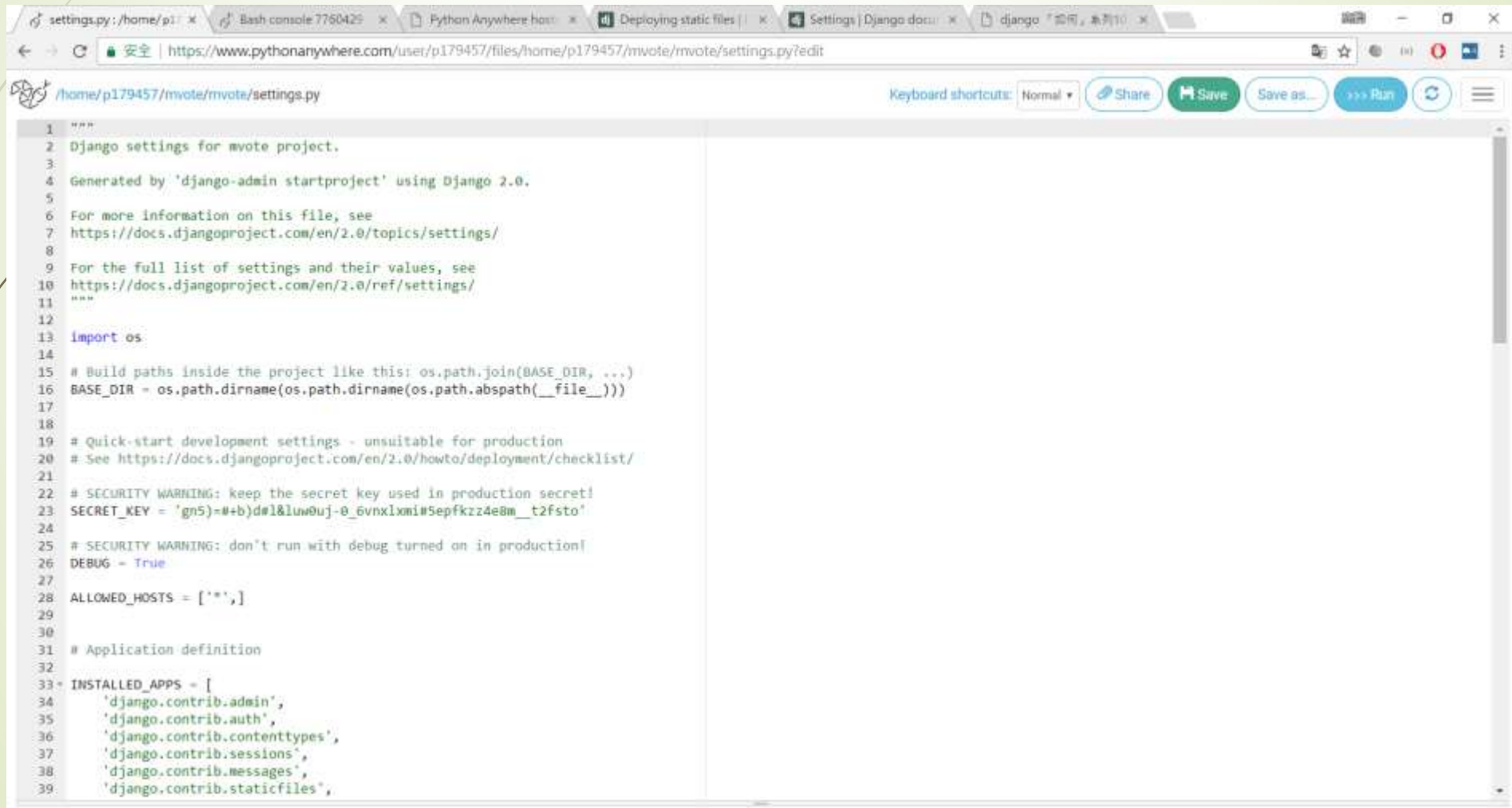
At the bottom of the Files section, there is an "Upload a file" button.

- ▶ 要改的內容如同前面幾堂課中的介紹包括
  - ▶ templates以及static files的設定
  - ▶ 在INSTALLED\_APP中加入'mysite'
  - ▶ 及ALLOWED\_HOSTS的串列中要加上'\*'
  - ▶ 把DEBUG設為False
- ▶ 此外還包括對於靜態檔案的設定內容：
  - ▶ (... 之前的內容省略 ...)
  - ▶ # Internationalization
  - ▶ # <https://docs.djangoproject.com/en/1.8/topics/i18n/>
  - ▶
  - ▶ LANGUAGE\_CODE = 'zh-Hant'
  - ▶
  - ▶ TIME\_ZONE = 'Asia/Taipei'
  - ▶
  - ▶ USE\_I18N = True

- `USE_L10N = True`
- 
- `USE_TZ = True`
- `# Static files (CSS, JavaScript, Images)`
- `# https://docs.djangoproject.com/en/1.8/howto/static-files/`
- 
- `STATIC_URL = '/static/'`
- `STATICFILES_DIRS = [`
- `os.path.join(BASE_DIR, 'static'),`
- `]`
- `STATIC_ROOT = '/home/stockPython/mvote/static/'`

# 在Pythonanywhere免費網站中Django網站

- 在pythonanywhere的Files介面中開啟settings.py



```
1 """
2 Django settings for mvote project.
3
4 Generated by 'django-admin startproject' using Django 2.0.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/2.0/topics/settings/
8
9 For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/2.0/ref/settings/
11 """
12
13 import os
14
15 # Build paths inside the project like this: os.path.join(BASE_DIR, ...)
16 BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'gn5)=#b)d#l&lun0uj-0_6vnxlxmi#5epfkzz4e8m_t2fsto'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = ['*'],
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
```

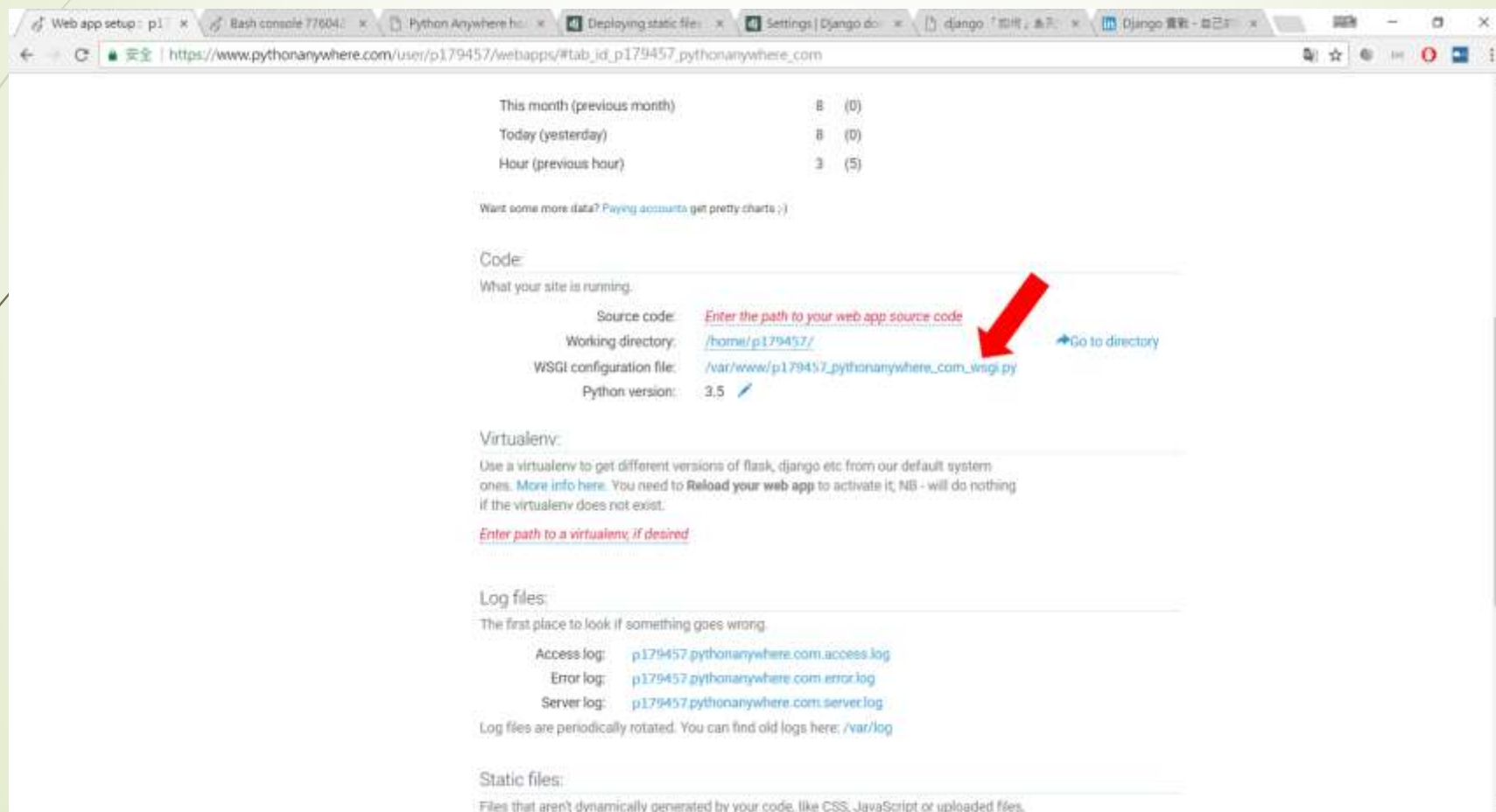


- 此例中帳號為stockPython，而使用django-admin產生的專案名為mvote，並利用python manage.py startapp產生mysite這個APP，請留意資料夾設定的內容，網站才能夠順利被運行。如箭頭所指的檔案還必須要點擊之後加以編輯，把原有的內容全部刪除，然後貼上如下所示的內容才行：

- # ++++++ DJANGO ++++++
- import os
- import sys
- 
- path = '/home/stockPython/mvote'
- if path not in sys.path:
- sys.path.append(path)
- 
- os.environ['DJANGO\_SETTINGS\_MODULE'] = 'mvote.settings'
- from django.core.wsgi import get\_wsgi\_application
- application = get\_wsgi\_application()



## ➡ 設定Web的相關參數之一



The screenshot shows the PythonAnywhere web app settings page. The browser tabs include 'Web app setup - p1', 'Bash console 77604', 'Python Anywhere h...', 'Deploying static file...', 'Settings | Django do...', 'django', and 'Django 實戰 - 自己...'. The URL is [https://www.pythonanywhere.com/user/p179457/webapps/#tab\\_id\\_p179457\\_pythonanywhere\\_com](https://www.pythonanywhere.com/user/p179457/webapps/#tab_id_p179457_pythonanywhere_com).

Summary statistics:

This month (previous month)	8 (0)
Today (yesterday)	8 (0)
Hour (previous hour)	3 (5)

Want some more data? [Paying accounts](#) get pretty charts :)

**Code:**

What your site is running:

Source code:	<a href="#">Enter the path to your web app source code</a>	<a href="#">Go to directory</a>
Working directory:	<a href="#">/home/p179457/</a>	
WSGI configuration file:	<a href="#">/var/www/p179457_pythonanywhere_com_wsgi.py</a>	
Python version:	3.5	

**Virtualenv:**

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it, NB - will do nothing if the virtualenv does not exist.

[Enter path to a virtualenv, if desired](#)

**Log files:**

The first place to look if something goes wrong.

Access log:	<a href="#">p179457.pythonanywhere.com.access.log</a>
Error log:	<a href="#">p179457.pythonanywhere.com.error.log</a>
Server log:	<a href="#">p179457.pythonanywhere.com.server.log</a>

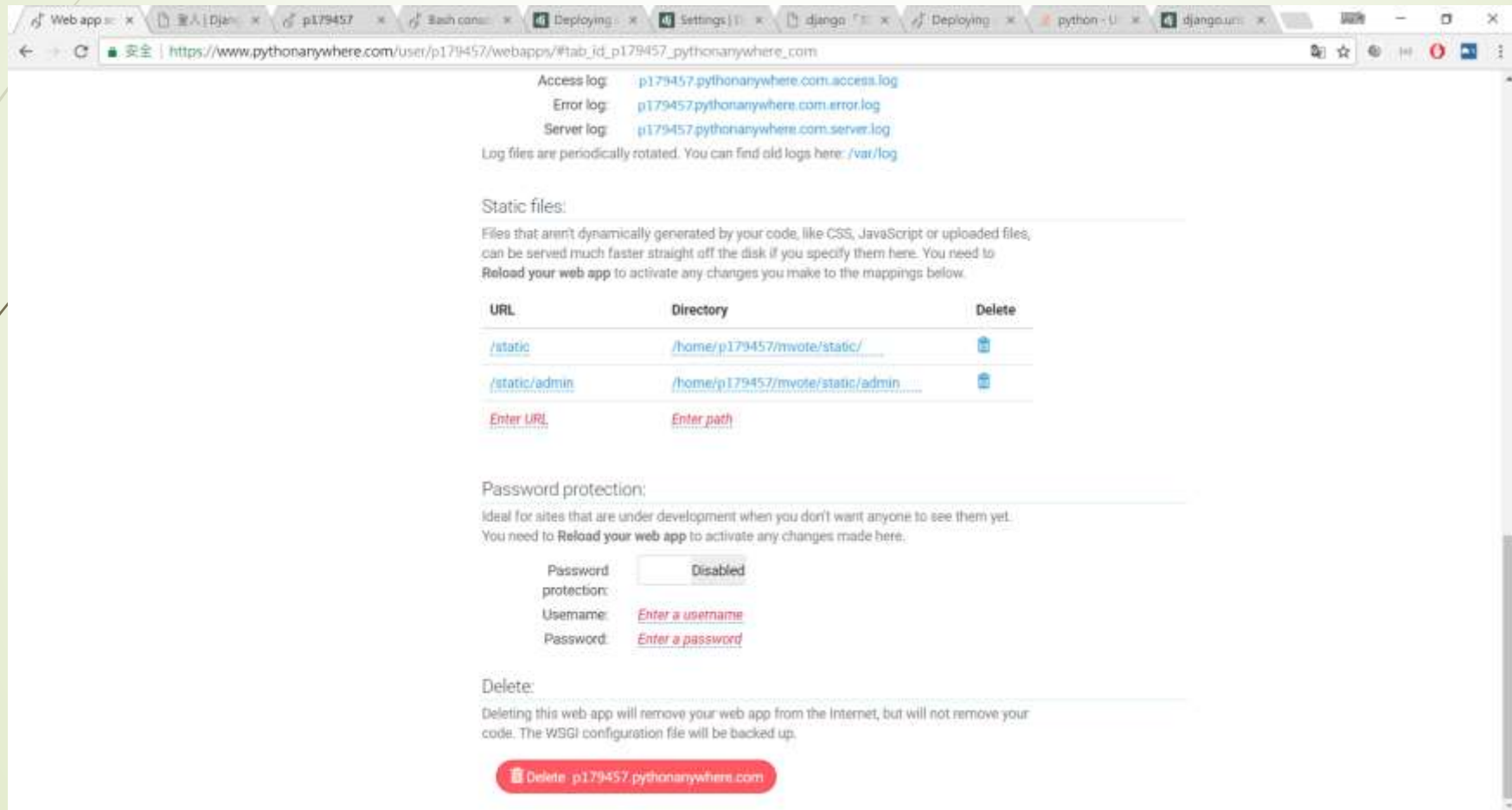
Log files are periodically rotated. You can find old logs here: [/var/log](#)

**Static files:**

Files that aren't dynamically generated by your code, like CSS, JavaScript or uploaded files.

- 設定靜態檔案所使用的位置，請參考下圖中的內容做修正。如果這個網站一開始需要設定瀏覽的帳號以及密碼，也可以在這裡設定。但是此帳號密碼是網站的帳號密碼，並不是admin後台管理網頁的密碼。admin後台管理網頁的帳號密碼一樣是在Bash的終端機介面中使用指令的方式加以設定

## 設定Web的相關參數之二



The screenshot shows the settings page for a Django web application on PythonAnywhere. The browser tabs include 'Web app', 'p179457', 'Deploying', 'Settings', 'django', and 'python'. The URL is [https://www.pythonanywhere.com/user/p179457/webapps/#tab\\_id\\_p179457\\_pythonanywhere\\_com](https://www.pythonanywhere.com/user/p179457/webapps/#tab_id_p179457_pythonanywhere_com).

**Log files:**

- Access log: [p179457.pythonanywhere.com.access.log](#)
- Error log: [p179457.pythonanywhere.com.error.log](#)
- Server log: [p179457.pythonanywhere.com.server.log](#)

Log files are periodically rotated. You can find old logs here: [/var/log](#)

**Static files:**

Files that aren't dynamically generated by your code, like CSS, JavaScript or uploaded files, can be served much faster straight off the disk if you specify them here. You need to **Reload your web app** to activate any changes you make to the mappings below.

URL	Directory	Delete
<a href="#">/static</a>	<a href="#">/home/p179457/mvote/static/</a>	
<a href="#">/static/admin</a>	<a href="#">/home/p179457/mvote/static/admin</a>	
<a href="#">Enter URL</a>	<a href="#">Enter path</a>	

**Password protection:**

Ideal for sites that are under development when you don't want anyone to see them yet. You need to **Reload your web app** to activate any changes made here.

Password protection:  Disabled

Username: [Enter a username](#)

Password: [Enter a password](#)

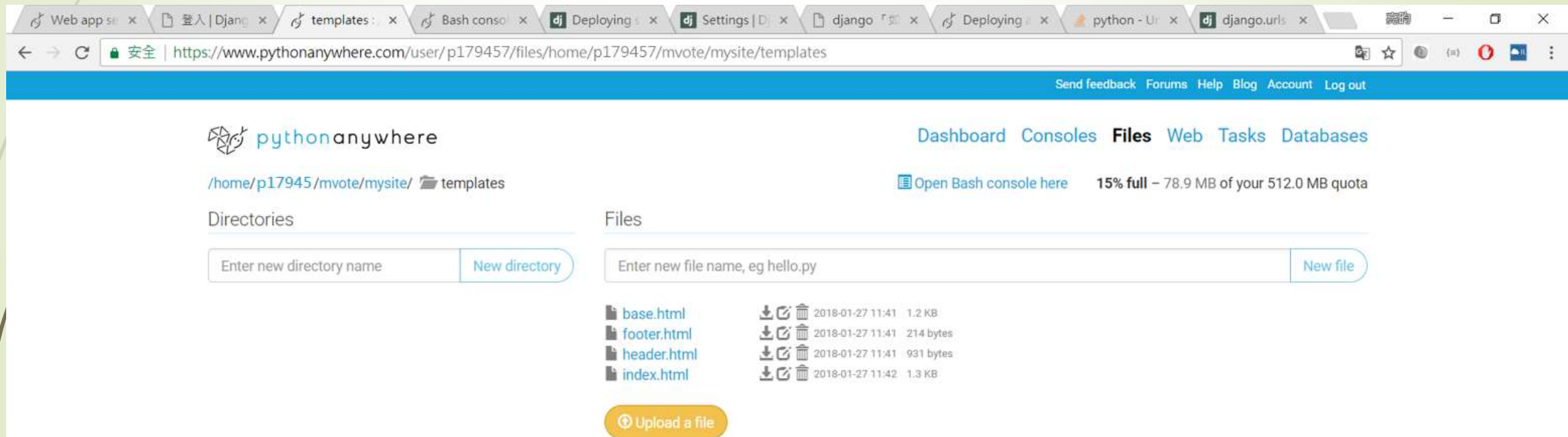
**Delete:**

Deleting this web app will remove your web app from the Internet, but will not remove your code. The WSGI configuration file will be backed up.

[Delete p179457.pythonanywhere.com](#)

# 建立投票網站基本架構

- 把前一節使用的模板檔案base.html, index.html, header.html, footer.html等上傳到templates資料夾之下，如下圖



The screenshot shows the PythonAnywhere file manager interface. The browser address bar displays the URL: `https://www.pythonanywhere.com/user/p179457/files/home/p179457/mvote/mysite/templates`. The page header includes navigation links: [Send feedback](#), [Forums](#), [Help](#), [Blog](#), [Account](#), and [Log out](#). The main content area shows the current directory path: `/home/p17945/mvote/mysite/ templates`. Below this, there are two sections: "Directories" and "Files". The "Directories" section has a text input field for "Enter new directory name" and a "New directory" button. The "Files" section has a text input field for "Enter new file name, eg hello.py" and a "New file" button. Below the input fields, there is a list of files:

File Name	Download	Share	Delete	Date	Size
<a href="#">base.html</a>				2018-01-27 11:41	1.2 KB
<a href="#">footer.html</a>				2018-01-27 11:41	214 bytes
<a href="#">header.html</a>				2018-01-27 11:41	931 bytes
<a href="#">index.html</a>				2018-01-27 11:42	1.3 KB

At the bottom of the file list, there is an "Upload a file" button.

## 建立投票網站基本架構

- 在 `mysite/static` 之下建立 `images` 以及上傳 `logo.png` 檔案，並到 Console 中執行 `python manage.py collectstatic` 讓靜態檔案可以在已部署的網站中生效
- 設定 `urls.py` 的內容如下：
  - `from django.contrib import admin`
  - `from django.urls import path`
  - `from mysite import views`
  
  - `urlpatterns = [`
  - `path('admin/', admin.site.urls),`
  - `path("", views.index),`
  - `]`

## 建立投票網站基本架構

- ▶ 在 `views.py` 中加入以下的內容：
  - ▶ `from django.shortcuts import render`
  - ▶
  - ▶ `def index(request):`
  - ▶  `return render(request, 'index.html', locals())`



## 建立投票網站基本架構

- ▶ `header.html`的內容要修改為如下：
  - ▶ `<!-- header.html (Registration01) -->`
  - ▶ `<nav class='navbar navbar-default'>`
  - ▶ `<div class='container-fluid'>`
  - ▶ `<div class='navbar-header'>`
  - ▶ `<div class='navbar-brand' align=center>`
  - ▶ 投票趣
  - ▶ `</div>`
  - ▶ `</div>`

## 建立投票網站基本架構

```
➤ <ul class='nav navbar-nav'>
➤     <li class='active'><a href='/'>Home</a></li>
➤     <li><a href='/accounts/signup'>註冊</a></li>
➤     <li><a href='/accounts/login'>登入</a></li>
➤     <li><a href='/accounts/logout'>登出</a></li>
➤     <li><a href='/admin'>後台管理</a></li>
➤ </ul>
➤ </div>
➤ </nav>
```

## 建立投票網站基本架構

- 在Pythonanywhere.com中順利啟用Django網站



## 建立投票網站基本架構

- 為了能夠讓網站可以接受使用者投票，建立2張資料表
  - 用來儲存投票議題的Poll
  - 以及每一個議題的內容項目PollItem
- 一個Poll可以有許多個PollItem
- 每一個PollItem只屬於一個Poll

## 建立投票網站基本架構

- ▶ 此投票網站的 `models.py` 內容可設計如下：
  - ▶ `#_*_ coding: utf-8 *_*`
  - ▶ `from django.db import models class`
  - ▶ `Poll(models.Model):`
    - ▶ `name = models.CharField(max_length=200, null=False)`
    - ▶ `created_at = models.DateField(auto_now_add=True)`
    - ▶ `enabled = models.BooleanField(default=False)`
    - ▶ `def __str__(self):`
      - ▶ `return self.name`

## 建立投票網站基本架構

- `class PollItem(models.Model):`
- `poll = models.ForeignKey(Poll, on_delete=models.CASCADE)`
- `name = models.CharField(max_length=200, null=False)`
- `image_url = models.CharField(max_length=200, null=True, blank=True)`
- `vote = models.PositiveIntegerField(default=0)`
- `def __str__(self):`
- `return self.name`



# 建立投票網站基本架構

- 簡化示範網站的複雜度，在這裡先把所有輸入投票項目的工作交給admin網頁管理，因此在admin.py中要models.py中的類別註冊到Admin中，如下所示：
  - `from django.contrib import admin`
  - `from mysite import models`
  - 
  - `class PollAdmin(admin.ModelAdmin):`
  - `list_display = ('name', 'created_at', 'enabled')`
  - `ordering = ('-created_at',)`
  - 
  - `class PollItemAdmin(admin.ModelAdmin):`
  - `list_display = ('poll', 'name', 'vote', 'image_url')`
  - `ordering = ('poll',)`
  - 
  - `admin.site.register(models.Poll, PollAdmin)`
  - `admin.site.register(models.PollItem, PollItemAdmin)`

## 建立投票網站基本架構

- ▶ 此網站至少要有
  - ▶ 顯示投票項目
  - ▶ 顯示投票網頁的能力
- ▶ 修正後的`urls.py`如下所示：
  - ▶ `from django.contrib import admin`
  - ▶ `from django.urls import path`
  - ▶ `from mysite import views`
  
  - ▶ `urlpatterns = [`
  - ▶ `path('admin/', admin.site.urls),`
  - ▶ `path("", views.index),`
  - ▶ `path('poll/<int:pollid>', views.poll, name='poll-url'),`
  - ▶ `path('vote/<int:pollid>/<int:pollitemid>', views.vote, name='vote-url'),`
  - ▶
  - ▶ `]`

## 建立投票網站基本架構

- ▶ views.py內容如下所示：
  - ▶ `#_*_ coding: utf-8 *_*`
  - ▶ `from django.shortcuts import render`
  - ▶ `from django.shortcuts import redirect`
  - ▶ `from mysite import models`
  - ▶ `# Create your views here.`
  - ▶
  - ▶ `def index(request):`
    - ▶ `polls = models.Poll.objects.all()`
    - ▶ `return render(request, 'index.html', locals())`

# 建立投票網站基本架構

```
def poll(request, pollid):
    try:
        poll = models.Poll.objects.get(id = pollid)
    except:
        poll = None
    if poll is not None:
        pollitems = models.PollItem.objects.filter(poll=poll).order_by('-vote')
    return render(request, 'poll.html', locals())

def vote(request, pollid, pollitemid):
    try:
        pollitem = models.PollItem.objects.get(id = pollitemid)
    except:
        pollitem = None
    if pollitem is not None:
        pollitem.vote = pollitem.vote + 1
        pollitem.save()
    target_url = '/poll/{}'.format( pollid)
    return redirect(target_url)
```

## 建立投票網站基本架構

- ▶ index.html的內容則為如下所示：
  - ▶ <!-- index.html (Registration01) -->
  - ▶ {% extends "base.html" %}
  - ▶ {% block title %}投票趣{% endblock %}
  - ▶ {% block content %}
  - ▶ <div class='container'>
  - ▶ {% for message in messages %}
  - ▶ <div class='alert alert-  
{{message.tags}}'>{{ message }}</div>
  - ▶ {% endfor %}

# 建立投票網站基本架構

```
➤ <div class='row'>
➤   <div class='col-md-12'>
➤     <div class='panel panel-default'>
➤       <div class='panel-heading' align=center>
➤         <h3>歡迎光臨投票趣</h3>
➤         <p>歡迎使用Facebook註冊/登入你的帳號，以擁有投票和製作投票的功能。</p>
➤       </div>
➤     </div>
➤   </div>
➤ </div>
➤ <div class='row'>
➤   {% for poll in polls %}
➤     {% if forloop.first %}
➤       <div class='list-group'>
➤         {% endif %}
➤         <a href='{% url "poll-url" poll.id %}' class='list-group-item'>{{ poll.name }}</a>
➤       {% if forloop.last %}
➤         </div>
➤       {% endif %}
➤     {% endif %}
➤   {% endif %}
```



## 建立投票網站基本架構

- {% empty %}
- <center><h3>目前並沒有活躍中的投票項目</h3></center>
- {% endfor %}
  
- <div class='list-group'>
  
- </div>
- </div>
- </div>
- {% endblock %}

# 建立投票網站基本架構

➡ index.html 首頁的顯示結果如下圖



# 建立投票網站基本架構

- ▶ poll.html的內容如下所示：
  - ▶ `<!-- poll.html (mvote project) -->`
  - ▶ `{% extends "base.html" %}`
  - ▶ `{% block title %}投票趣{% endblock %}`
  - ▶ `{% block content %}`
  - ▶ `<div class='container'>`
  - ▶ `{% for message in messages %}`
  - ▶ `<div class='alert alert-{{message.tags}}'>{{ message }}</div>`
  - ▶ `{% endfor %}`
  - ▶ `<div class='row'>`
  - ▶ `<div class='col-md-12'>`
  - ▶ `<div class='panel panel-default'>`
  - ▶ `<div class='panel-heading' align=center>`
  - ▶ `<h3>{{ poll.name }}</h3>`
  - ▶ `</div>`
  - ▶ `</div>`
  - ▶ `</div>`
  - ▶ `</div>`

# 建立投票網站基本架構

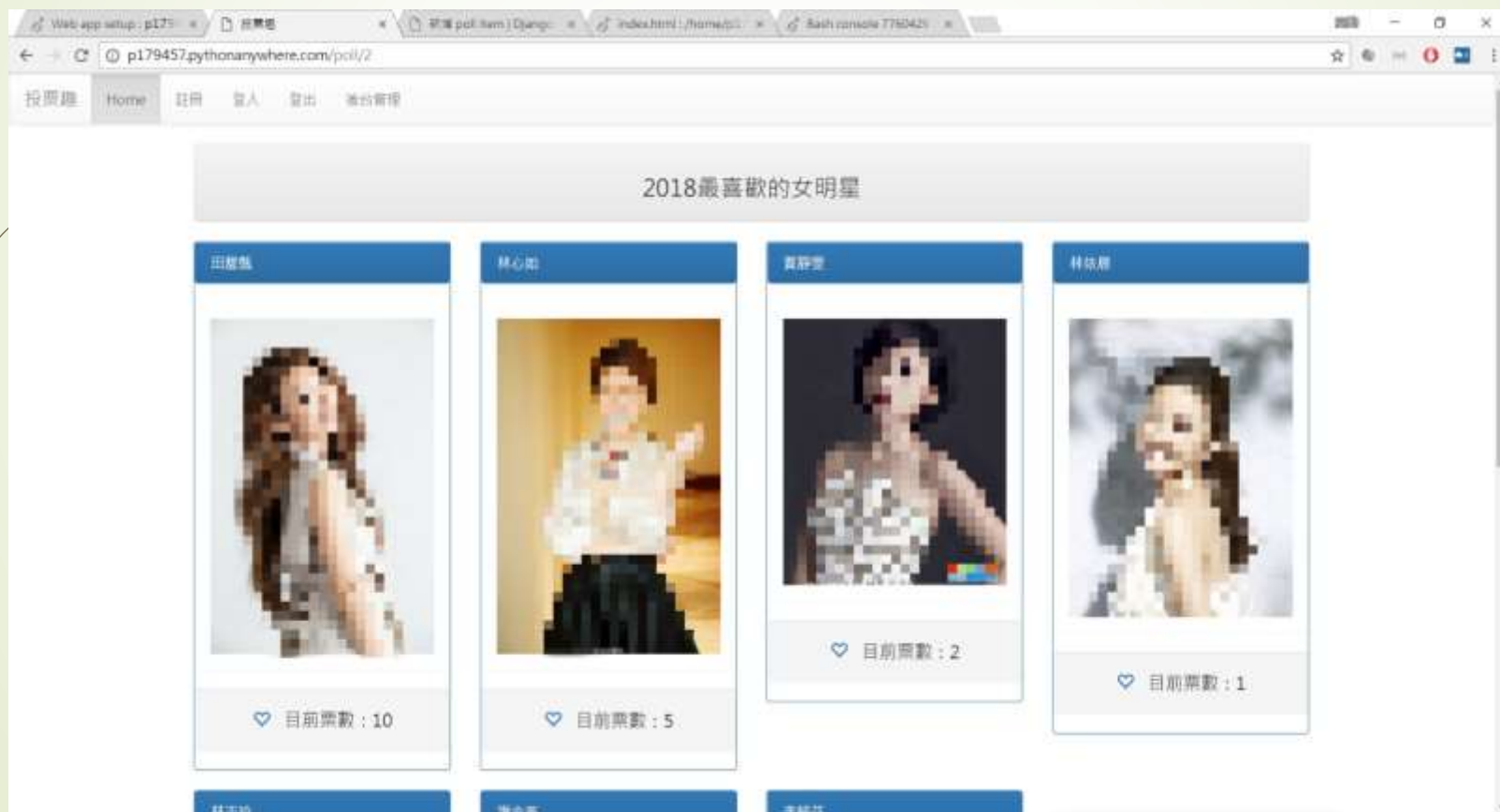
```
➤ {% for pollitem in pollitems %}
➤   {% cycle "<div class='row'>" "" "" "" "" %}
➤   <div class='col-sm-3'>
➤     <div class='panel panel-primary'>
➤       <div class='panel panel-heading'>
➤         {{ pollitem.name }}
➤       </div>
➤       <div class='panel panel-body'>
➤         {% if pollitem.image_url %}
➤           <img src='{{ pollitem.image_url }}' width='100%'>
➤         {% else %}
➤           <img src='http://i.imgur.com/Ous4iGB.png' width='100%'>
➤         {% endif %}
➤       </div>
➤       <div class='panel panel-footer' align=center>
➤         <h4>
➤         <a href='/vote/{{poll.id}}/{{pollitem.id}}' title='投票'>
➤           <span class='glyphicon glyphicon-heart-empty'>
➤         </span>
➤         </a>
```

## 建立投票網站基本架構

- &nbsp;
- 目前票數：{{ pollitem.vote }}</h4>
- </div>
- </div>
- </div>
- {% cycle "" "" "" "" "</div>" %}
- {% endfor %}
- </div>

# 建立投票網站基本架構

## ➔ poll.html的投票網頁



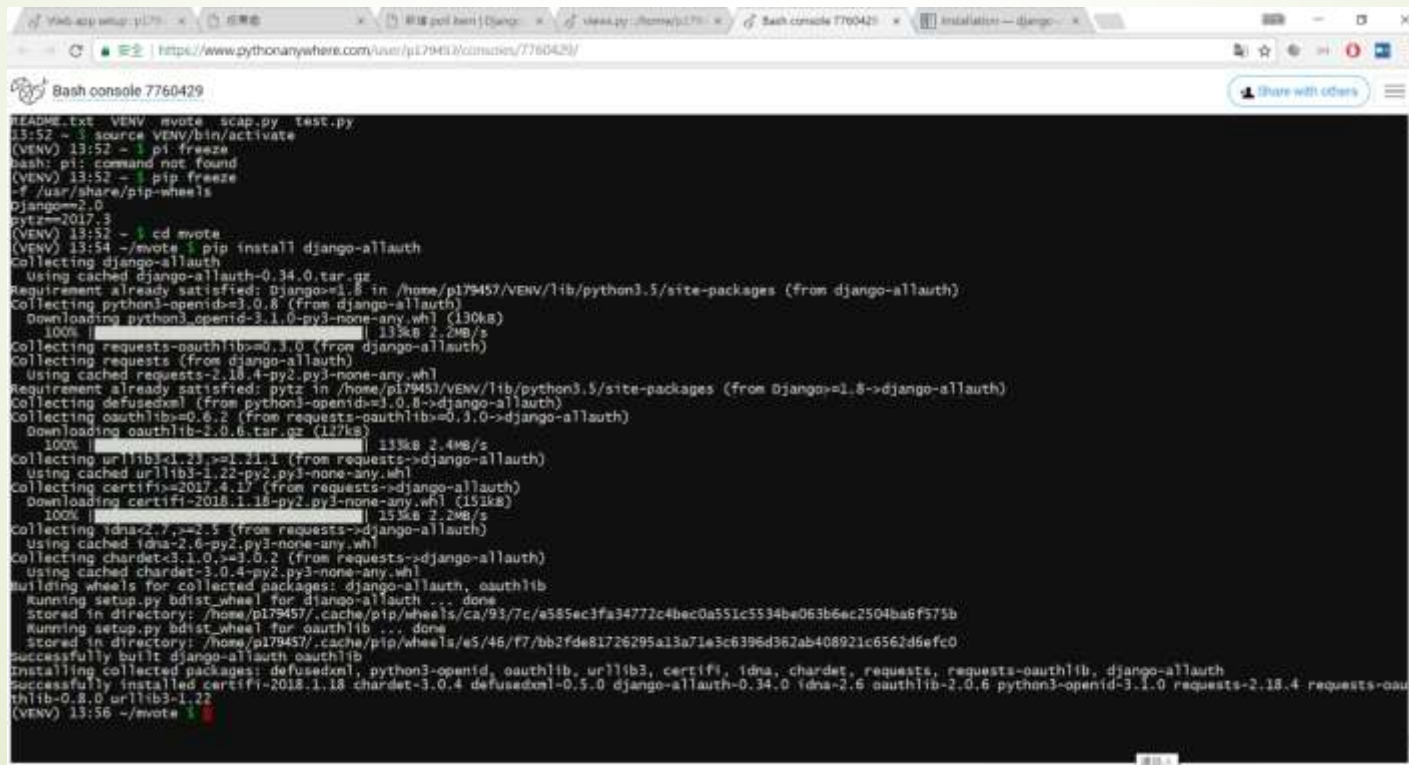


## 使用Facebook驗證帳號操作實務

- 將以django-allauth來做為網站帳號驗證的框架。
- 之所以說django-allauth是一個框架，主要的原因是它具備了所有帳號驗證的功能，除了我們在前一節中介紹的django-registration的所有功能之外，還包括了幾乎包含了市面上所有主要會員網站的第三方驗證功能，而更特別的是它用資料庫的方式簡化新增網站驗證的操作流程。
- 如果你打算做一個全方位功能的第三方驗證帳號之會員網站，django-allauth絕對是值得試的模組。

# 在Pythonanywhere中安裝django-allauth與設定

- 使用 `pip install django-allauth` 安裝
- 在Pythonanywhere中要安裝此模組，只要到Bash Console



```
README.txt VENV mvote scap.py test.py
13:52 ~ source VENV/bin/activate
(VENV) 13:52 ~ pi freeze
bash: pi: command not found
(VENV) 13:52 ~ pi freeze
~ # /usr/share/pip-wheels
django=2.0
pytz=2017.3
(VENV) 13:52 ~ cd mvote
(VENV) 13:54 ~/mvote # pip install django-allauth
Collecting django-allauth
  Using cached django-allauth-0.34.0.tar.gz
Requirement already satisfied: Django<=1.8 in /home/pi79457/ENV/lib/python3.5/site-packages (from django-allauth)
Collecting python3-openid<=3.0.8 (from django-allauth)
  Downloading python3_openid-3.1.0-py3-none-any.whl (130kB)
    100% |#####| 13kB 2.2MB/s
Collecting requests-oauthlib<=0.3.0 (from django-allauth)
Collecting requests (from django-allauth)
  Using cached requests-2.18.4-py2.py3-none-any.whl
Requirement already satisfied: pytz in /home/pi79457/ENV/lib/python3.5/site-packages (from Django==1.8->django-allauth)
Collecting defusedxml (from python3-openid<=3.0.8->django-allauth)
Collecting oauthlib<=0.6.2 (from requests-oauthlib<=0.3.0->django-allauth)
  Downloading oauthlib-2.0.6.tar.gz (127kB)
    100% |#####| 13kB 2.4MB/s
Collecting urllib3<1.23,=>1.21.1 (from requests->django-allauth)
  Using cached urllib3-1.22-py2.py3-none-any.whl
Collecting certifi<=2017.4.17 (from requests->django-allauth)
  Downloading certifi-2018.1.18-py2.py3-none-any.whl (151kB)
    100% |#####| 15kB 2.2MB/s
Collecting idna<2.7,=>2.5 (from requests->django-allauth)
  Using cached idna-2.6-py2.py3-none-any.whl
Collecting chardet<3.1.0,=>3.0.2 (from requests->django-allauth)
  Using cached chardet-3.0.4-py2.py3-none-any.whl
Building wheels for collected packages: django-allauth, oauthlib
  Running setup.py bdist_wheel for django-allauth ... done
  Stored in directory: /home/pi79457/.cache/pip/wheels/ca/93/7c/e585ec3fa34772c4bec0a551c5534be063b6ec2504ba6f575b
  Running setup.py bdist_wheel for oauthlib ... done
  Stored in directory: /home/pi79457/.cache/pip/wheels/e5/46/f7/bb2fde81726295a13a71e3c6396d367ab408921c6562d6efc0
Successfully built django-allauth oauthlib
Installing collected packages: defusedxml, python3-openid, oauthlib, urllib3, certifi, idna, chardet, requests, requests-oauthlib, django-allauth
Successfully installed certifi-2018.1.18 chardet-3.0.4 defusedxml-0.5.0 django-allauth-0.34.0 idna-2.6 oauthlib-2.0.6 python3-openid-3.1.0 requests-2.18.4 requests-ou
thlib-0.8.0 urllib3-1.22
(VENV) 13:56 ~/mvote #
```

# 在Pythonanywhere中安裝django-allauth與設定

- ▶ 在settings.py中要加入：
  - ▶ INSTALLED\_APPS = (
    - ▶ 'django.contrib.admin',
    - ▶ 'django.contrib.auth',
    - ▶ 'django.contrib.contenttypes',
    - ▶ 'django.contrib.sessions',
    - ▶ 'django.contrib.messages',
    - ▶ 'django.contrib.staticfiles',
    - ▶ 'mysite',
    - ▶ 'django.contrib.sites',
    - ▶ 'allauth',
    - ▶ 'allauth.account',
    - ▶ 'allauth.socialaccount',
    - ▶ 'allauth.socialaccount.providers.facebook',
    - ▶ )

# 使用Mailgun(改mail anywhere)

82

- `SITE_ID = 1`
- `LOGIN_REDIRECT_URL = '/'`
- `EMAIL_BACKEND = 'django_mailgun.MailgunBackend'`
- `MAILGUN_ACCESS_KEY = 'key-5*****227e7'`
- `MAILGUN_SERVER_NAME = 'my.Django.server.tw'`
- `AUTHENTICATION_BACKENDS = (`
- `'django.contrib.auth.backends.ModelBackend',`
- `'allauth.account.auth_backends.AuthenticationBackend',`
- `)`

# 在Pythonanywhere中安裝django-allauth與設定

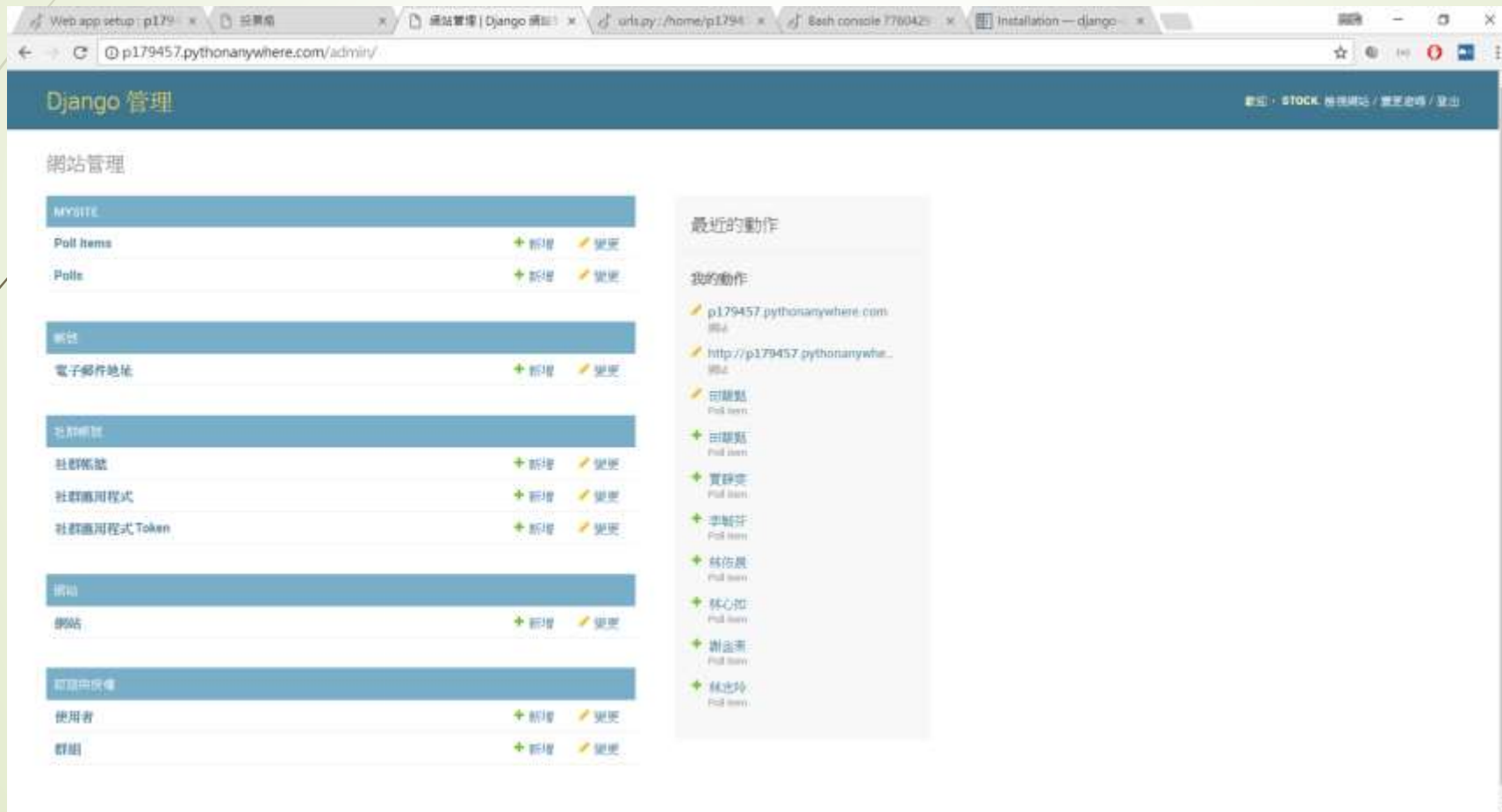
83

- 在urls.py中也要加入下面這一行，表示所有在/accounts之後的網址要先到allauth中去確定一下有沒有處理的函數（包括login, logout, 以及signup等等）：
  - `path('accounts/', include('allauth.urls'))`,



# 在Pythonanywhere中安裝django-allauth與設定

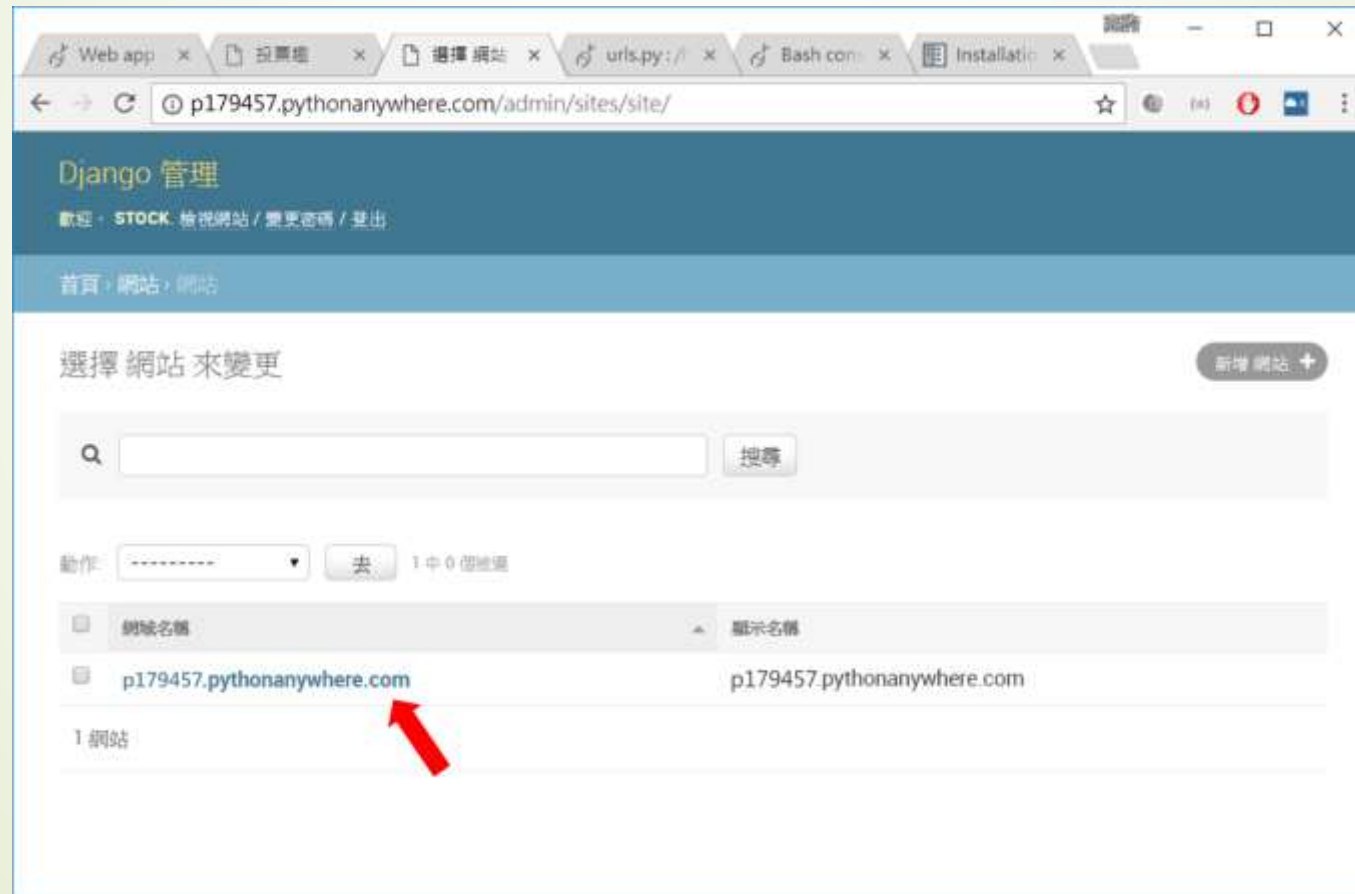
➔ django-allauth用來建立**第三方帳號驗證**的資料表





# 在Pythonanywhere中安裝django-allauth與設定

## ➡ 設定網站名稱



## 到Facebook開發者網頁申請驗證機制

- 以Facebook為例，請先登入到Facebook，然後連結到 <https://developers.facebook.com/>，然後選擇新增應用程式



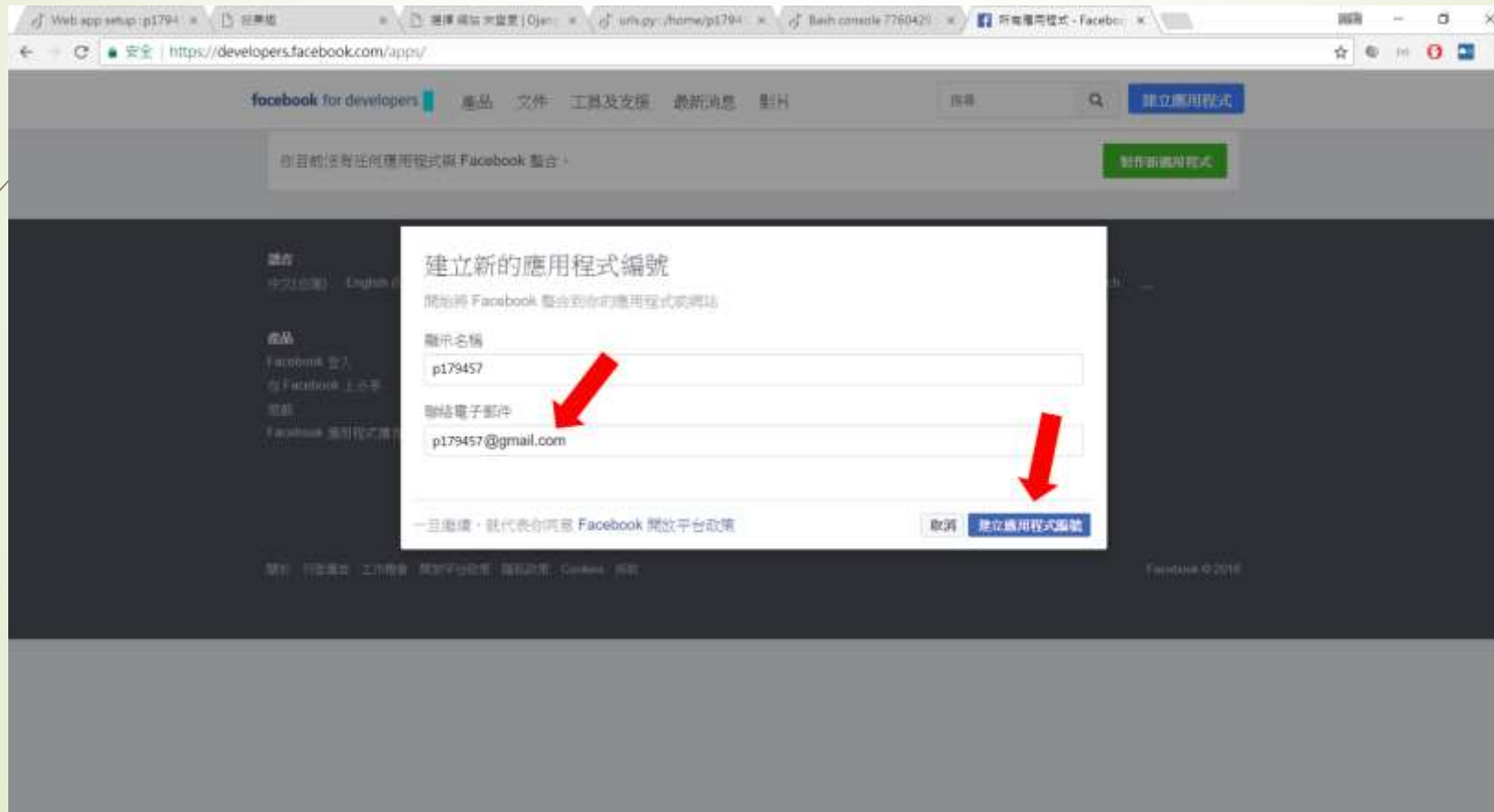
# 到Facebook開發者網頁申請驗證機制

- ➡ 為應用程式輸入名稱



# 到Facebook開發者網頁申請驗證機制

- 按下「建立應用程式編號」按鈕，然後選擇產品Facebook登入



# 到Facebook開發者網頁申請驗證機制

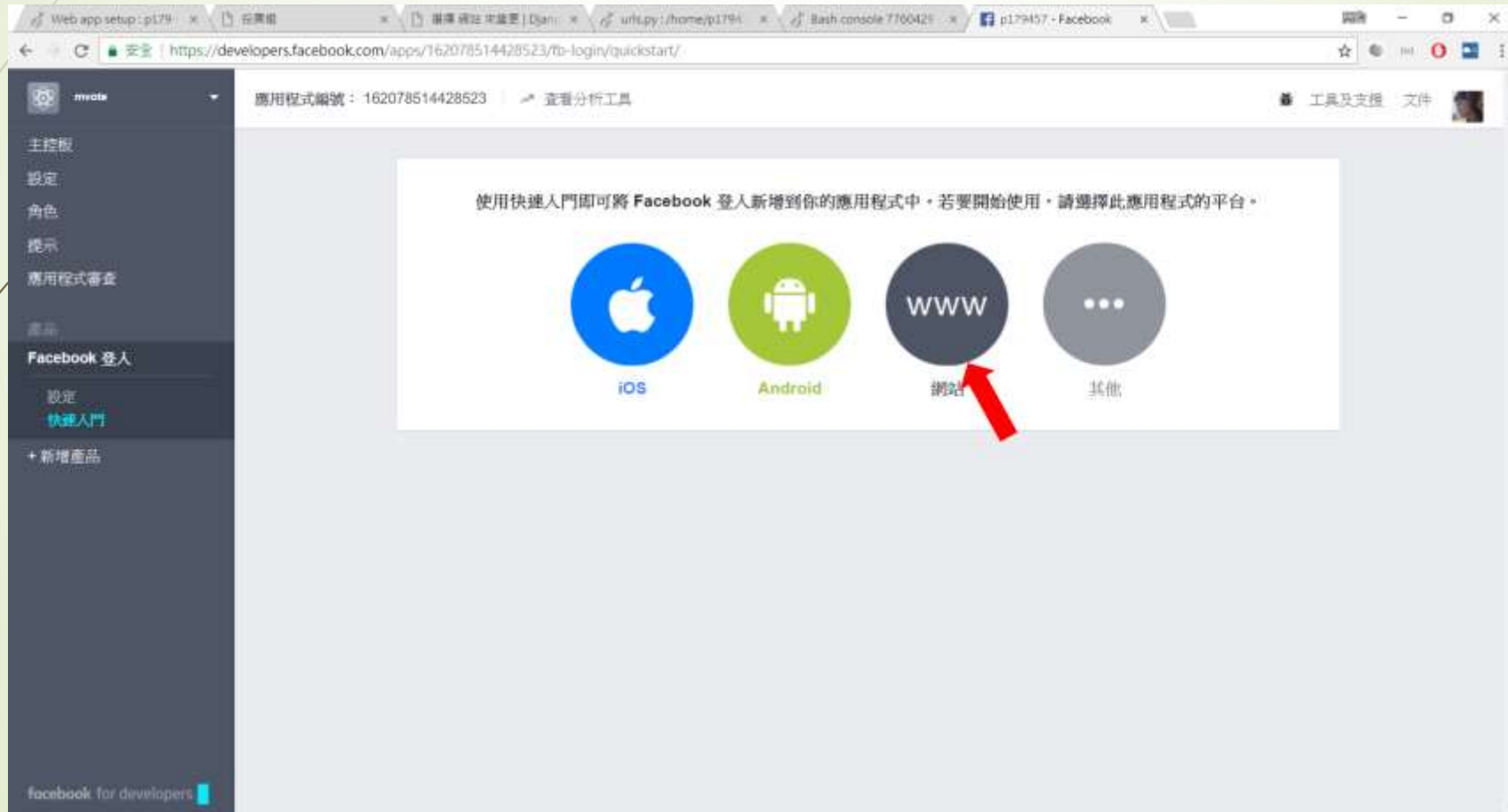
## ➤ Facebook應用程式建立完成之畫面





# 到Facebook開發者網頁申請驗證機制

➡ 選擇Facebook應用程式類別之畫面





# 到Facebook開發者網頁申請驗證機制

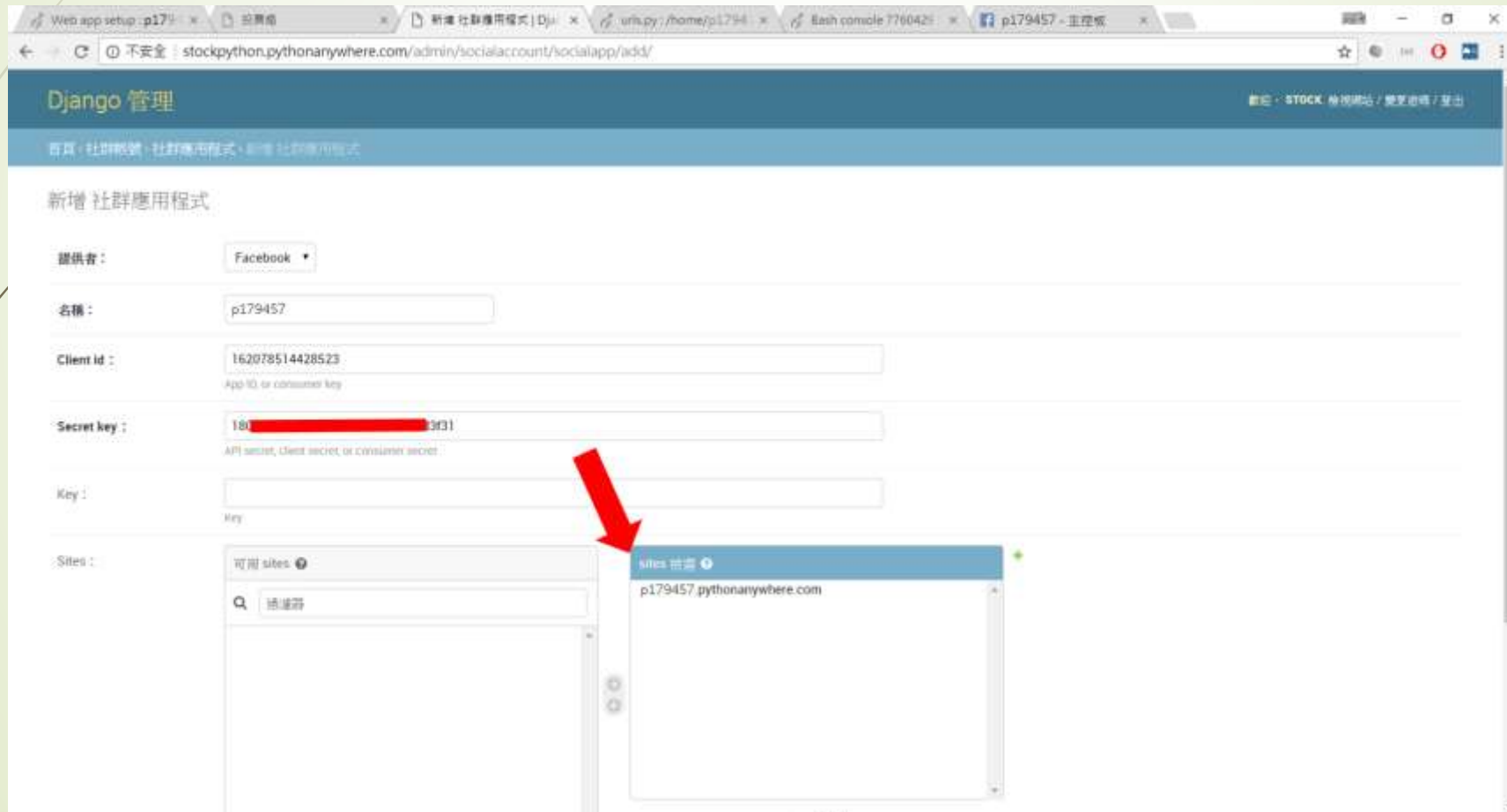
## ➡ 應用程式的主控板



The screenshot shows the Facebook Developer Dashboard for an application named 'p179457'. The URL in the browser is <https://developers.facebook.com/apps/162078514428523/dashboard/>. The application ID is 162078514428523. The dashboard includes a sidebar with navigation options: 主控板 (highlighted with a red arrow), 設定, 角色, 提示, 應用程式審查, 產品, Facebook 登入, and + 新增產品. The main content area shows the application name 'p179457', its API version 'v2.11', and the application ID. There is a section for 'Facebook Analytics' with a '設定分析工具' button and a '試用體驗版' button. Below that is a 'Facebook 登入' section with a '活躍登入用戶' button and a '總數' button. The bottom right corner has three checkboxes: '每日活躍用戶', '每週活躍用戶', and '每月活躍用戶', all of which are checked.

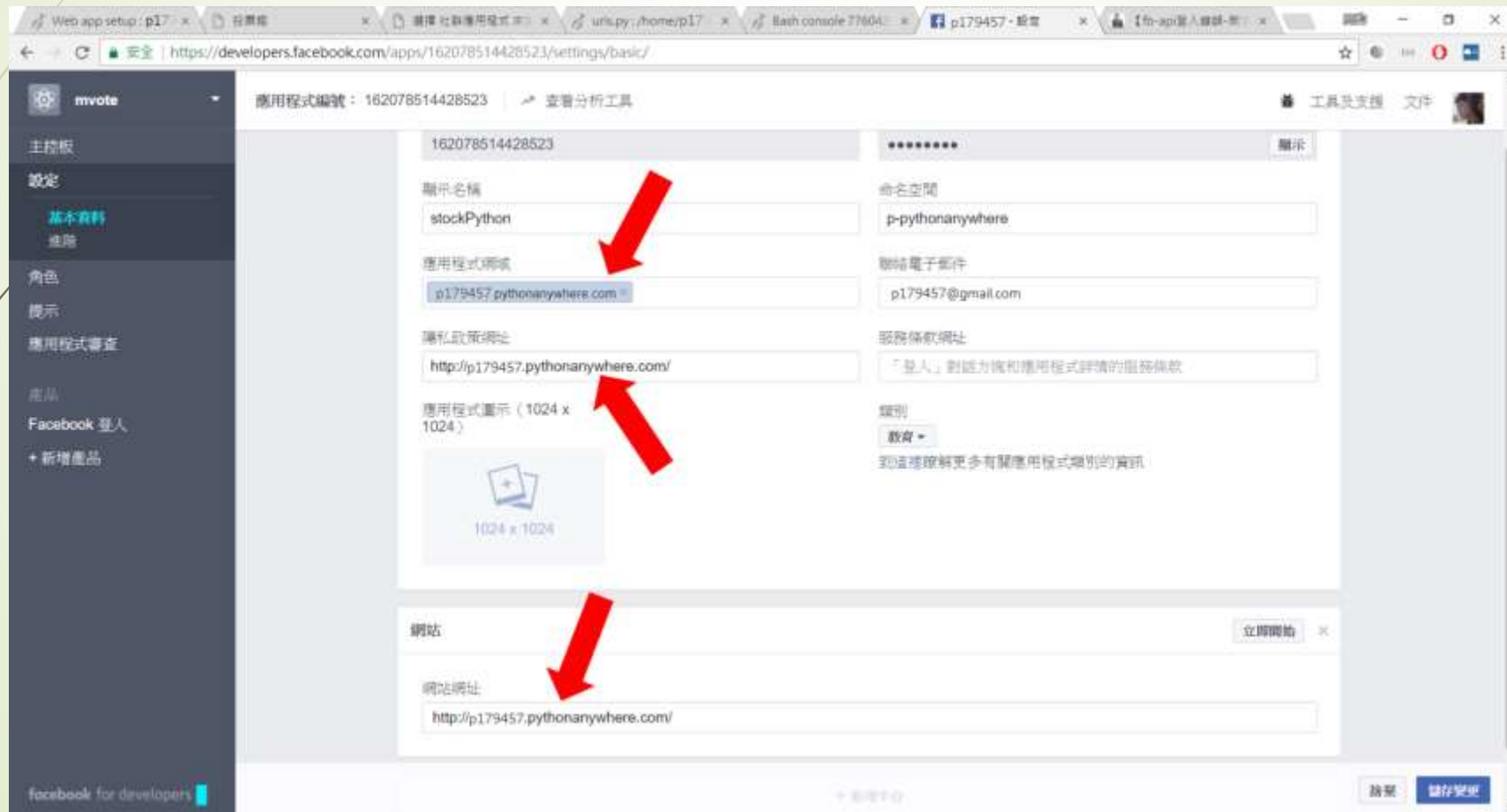
# 到Facebook開發者網頁申請驗證機制

- 在網站中建立allauth所需要的社群應用程式資料



# 到Facebook開發者網頁申請驗證機制

- 在Facebook的應用程式設定中，加入網站的網址

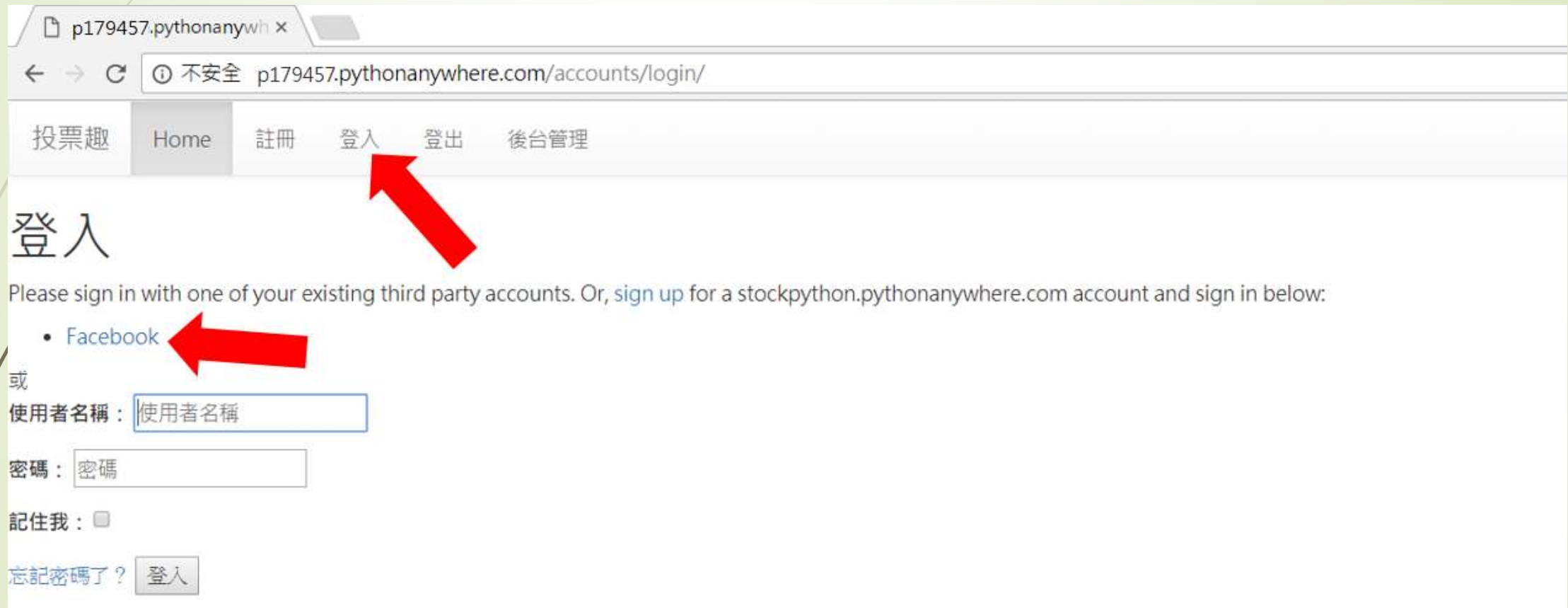


## 到Facebook開發者網頁申請驗證機制

- 完成以上的步驟之後，不需要多餘的程式碼，此時allauth即可正常運作
- 先以  
<http://p179457.pythonanywhere.com/accounts/logout>登出帳號
- 再以  
<http://p179457.pythonanywhere.com/accounts/signup>做註冊的動作
- 也可以使用  
<http://p179457.pythonanywhere.com/accounts/login>做登入的動作

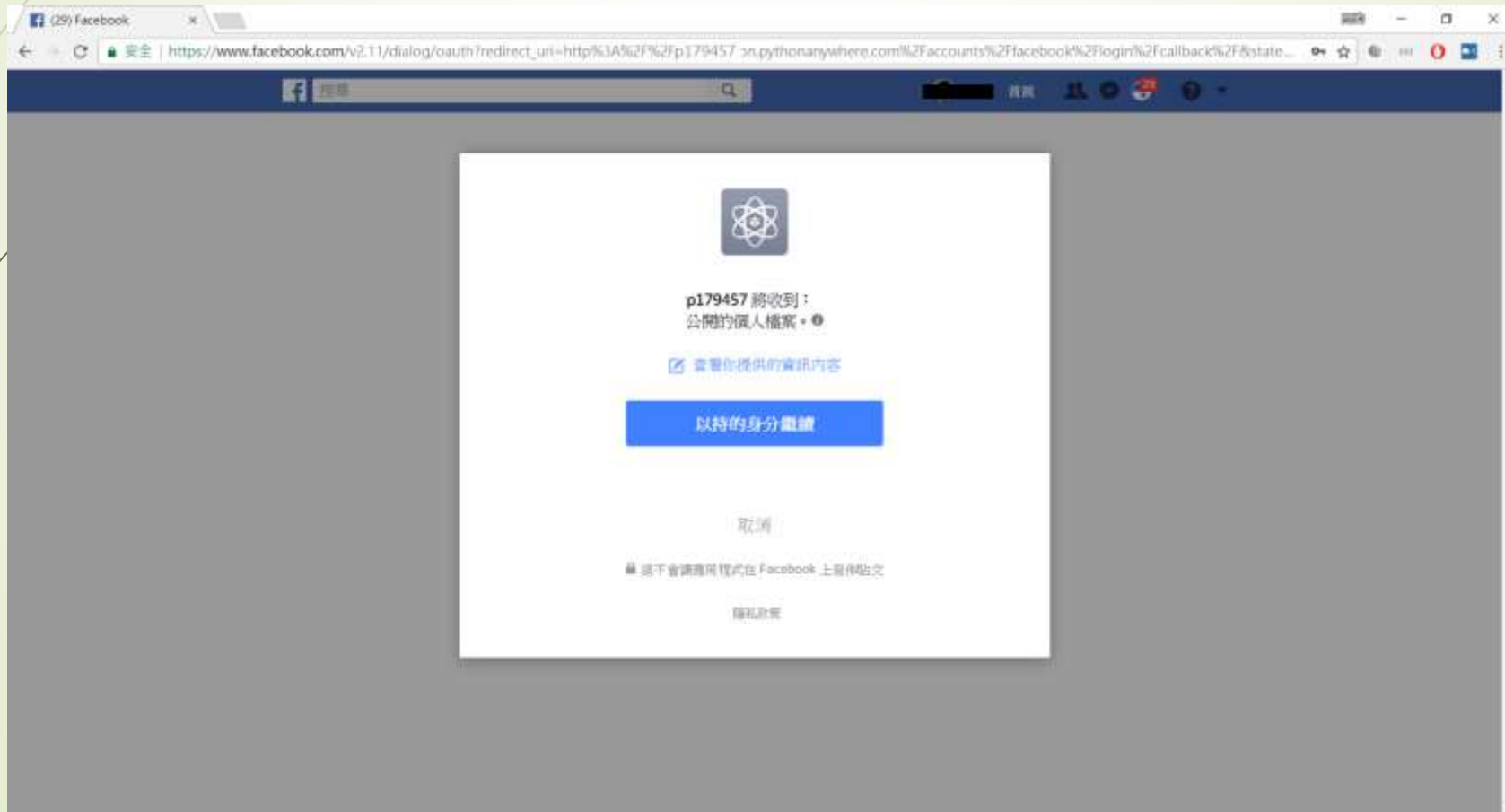
# 到Facebook開發者網頁申請驗證機制

➤ allauth的預設登入畫面



# 到Facebook開發者網頁申請驗證機制

## ➤ allauth的Facebook授權畫面





## 到Facebook開發者網頁申請驗證機制

- 目前為止Facebook的應用程式狀態是屬於測試階段，只有開發者自己的Facebook帳號可以使用，別人是沒有辦法使用Facebook登入的
- 回到Facebook的開發者網頁，把此**應用程式公開發行**

# 到Facebook開發者網頁申請驗證機制

- 在Facebook開發者網頁中發佈應用程式



## 在網站中識別使用者的登入狀態

- 由於django-allauth遵循Django原有的帳號驗證方法，所以原本在Django Authentication System中使用的內容都可以使用
- 在所有的Template模板中，使用 `user.is_authenticated` 判別使用者是否為已驗證完畢
- `user.is_active` 可以用來查詢使用者是否為有效的帳號
- 也可以使用 `user.username` 取得使用者名稱以及 `user.email` 來取得使用者的電子郵件帳號

# 在網站中識別使用者的登入狀態

- header.html · 修正為如下所示的樣子：

```
<!-- header.html (Registration01) -->
<nav class='navbar navbar-default'>
  <div class='container-fluid'>
    <div class='navbar-header'>
      <div class='navbar-brand' align=center>
        投票趣
      </div>
    </div>
  </div>
  {% load account %}
  <ul class='nav navbar-nav'>
    <li class='active'><a href='/>Home</a></li>
    {% if user.is_authenticated %}
    <li><a href='/accounts/logout'>登出</a></li>
    {% else %}
    <li><a href='/accounts/signup'>註冊</a></li>
    <li><a href='/accounts/login'>登入</a></li>
    {% endif %}
    <li><a href='/admin'>後台管理</a></li>
  </ul>
</div>
</nav>
```

# 在網站中識別使用者的登入狀態

- 在index.html中也要加以修正，如下所示：

- <!-- index.html (Registration01) -->
- {% extends "base.html" %}
- {% block title %}投票趣{% endblock %}
- {% block content %}
- <div class='container'>
- {% for message in messages %}
- <div class='alert alert-{{message.tags}}'>{{ message }}</div>
- {% endfor %}
- <div class='row'>
- <div class='col-md-12'>
- <div class='panel panel-default'>
- <div class='panel-heading' align=center>
- <h3>歡迎光臨投票趣</h3>
- <p>歡迎使用Facebook註冊/登入你的帳號，以擁有投票和製作投票的功能。</p>
- </div>
- </div>
- </div>
- </div>

# 在網站中識別使用者的登入狀態

```

- <div class='row'>
-   {% load account %}
-   {% for poll in polls %}
-     {% if forloop.first %}
-       <div class='list-group'>
-     {% endif %}
-     {% if user.is_authenticated %}
-       <a href='{% url "poll-url" poll.id %}' class='list-group-item'>{{ poll.name }}</a>
-     {% else %}
-       <a href='#' class='list-group-item' title='要登入之後才能夠前往投票！'>{{ poll.name }}</a>
-     {% endif %}
-   {% if forloop.last %}
-     </div>
-   {% endif %}
-   {% empty %}
-     <center><h3>目前並沒有活躍中的投票項目</h3></center>
-   {% endfor %}
-
-   <div class='list-group'>
-
- </div>
- </div>
- </div>
- {% endblock %}
```



## 在網站中識別使用者的登入狀態

➤ 修改後的views.py如下所示：

➤ from django.shortcuts import render

➤ from django.shortcuts import redirect

➤ from django.contrib.auth.decorators import login\_required

➤ from mysite import models

➤ def index(request):

➤ polls = models.Poll.objects.all()

➤ return render(request, 'index.html', locals())

➤

## 在網站中識別使用者的登入狀態

- `@login_required`
- `def poll(request, pollid):`
- `try:`
- `poll = models.Poll.objects.get(id = pollid)`
- `except:`
- `poll = None`
- `if poll is not None:`
- `pollitems =`  
`models.PollItem.objects.filter(poll=poll).order_by('-`  
`vote')`
- `return render(request, 'poll.html', locals())`

## 在網站中識別使用者的登入狀態

- `@login_required`
- `def vote(request, pollid, pollitemid):`
- `try:`
- `pollitem = models.PollItem.objects.get(id = pollitemid)`
- `except:`
- `pollitem = None`
- `if pollitem is not None:`
- `pollitem.vote = pollitem.vote + 1`
- `pollitem.save()`
- `target_url = '/poll/{}'.format( pollid)`
- `return redirect(target_url)`

## 在網站中識別使用者的登入狀態

- 希望註冊的使用者一定要**驗證完電子郵件**之後才能夠進行投票，那麼就可以在views.py的最前面加上：
  - `from allauth.account.decorators import verified_email_required`
  - 然後在`def poll`以及`def vote`之前加上**@verified\_email\_required**，如下所示：
    - `@login_required`
    - `@verified_email_required`
    - `def poll(request, pollid):`

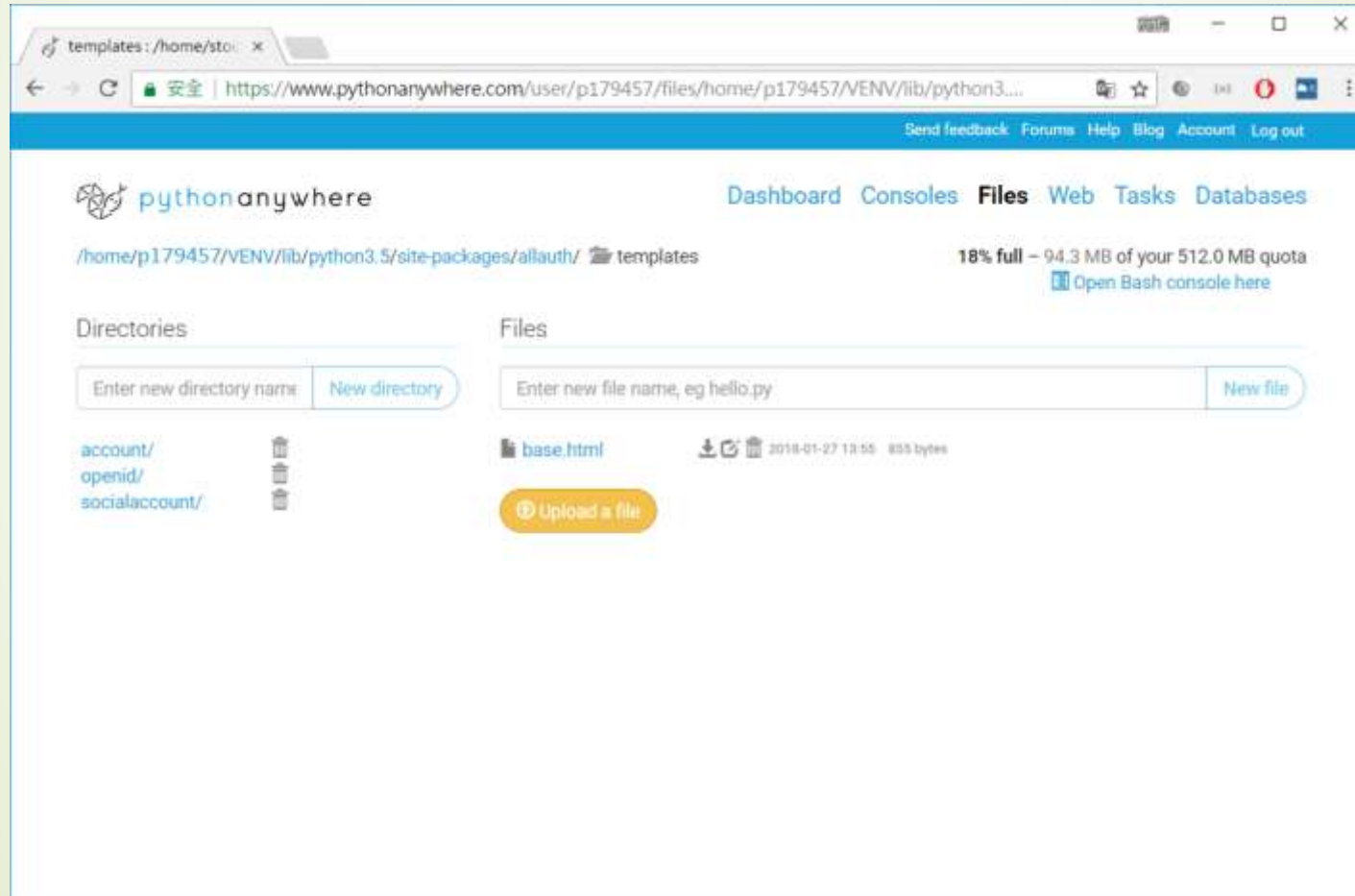
## 在網站中識別使用者的登入狀態

- 未驗證電子郵件的使用者打算進入投票頁面時，網站即會出現提醒要驗證電子郵件的頁面，如下圖



# 客製化django-allauth頁面

- 在Pythonanywhere中allauth預設的模板位置





## 客製化django-allauth頁面

- 第一步要做的動作就是把allauth底下templates內的模板複製到我們網站的templates底下
- 由於我們已經有共用的base.html模板了，所以此檔案就不用再複製了
- 只要把account、openid、以及socialaccount這3個資料夾複製到網站的templates底下即可

## 客製化django-allauth頁面

- 在Pythonanywhere中可以到Bash Console去操作複製的指令，如下所示：
  - (VENV) 08:54 ~ \$ `cd mvote/mysite`
  - (VENV) 08:54 ~/mvote \$ `cp -R ../../VENV/lib/python3.5/site-packages/allauth/templates/account/ templates`
  - (VENV) 08:55 ~/mvote \$ `cp -R ../../VENV/lib/python3.5/site-packages/allauth/templates/openid/ templates`
  - (VENV) 08:55 ~/mvote \$ `cp -R ../../VENV/lib/python3.5/site-packages/allauth/templates/socialaccount/ templates`

## 客製化django-allauth頁面

- ▶ 把網站的header.html修改如下：
  - ▶ `<!-- header.html (Registration01) -->`
  - ▶ `<nav class='navbar navbar-default'>`
  - ▶ `<div class='container-fluid'>`
  - ▶ `<div class='navbar-header'>`
  - ▶ `<div class='navbar-brand' align=center>`
  - ▶ `投票趣`
  - ▶ `</div>`
  - ▶ `</div>`
  - ▶ `{% load account %}`

# 客製化django-allauth頁面

```



```

## 客製化django-allauth頁面

- 引入我們的base.html，亦即表示在accounts底下的所有模板也都可以直接加以編輯，加入Bootstrap的格式設定。以logout.html為例，把它修改如下所示的樣子：

- `{% extends "account/base.html" %}`

- 

- `{% load i18n %}`

- 

- `{% block head_title %}{% trans "Sign Out" %}{% endblock %}`

- 

- `{% block content %}`

# 客製化django-allauth頁面

```

➤ <div class='container'>
➤   <div class='row'>
➤     <div class='panel panel-warning'>
➤       <div class='panel panel-heading'>
➤         <h1>{% trans "Sign Out" %}</h1>
➤       </div>
➤     <div class='panel panel-body'>
➤       <h3>{% trans 'Are you sure you want to sign out?' %}</h3>
➤       <form method="post" action="{% url 'account_logout' %}">
➤         {% csrf_token %}
➤         {% if redirect_field_value %}
➤         <input type="hidden" name="{{ redirect_field_name }}" value="{{ redirect_field_value }}"/>
➤         {% endif %}
➤         <button type="submit">{% trans 'Sign Out' %}</button>
➤       </form>
➤     </div>
➤   </div>
➤ </div>
➤ </div>
➤ {% endblock %}

```



# 客製化django-allauth頁面

- 原本allauth的登出畫面



# 客製化django-allauth頁面

- 加入Bootstrap標記指令的登出畫面

