



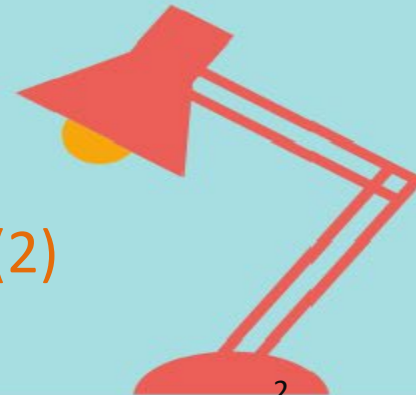
2023-Spring Advanced Computer Programming (11)

CSIE, Asia Univ.



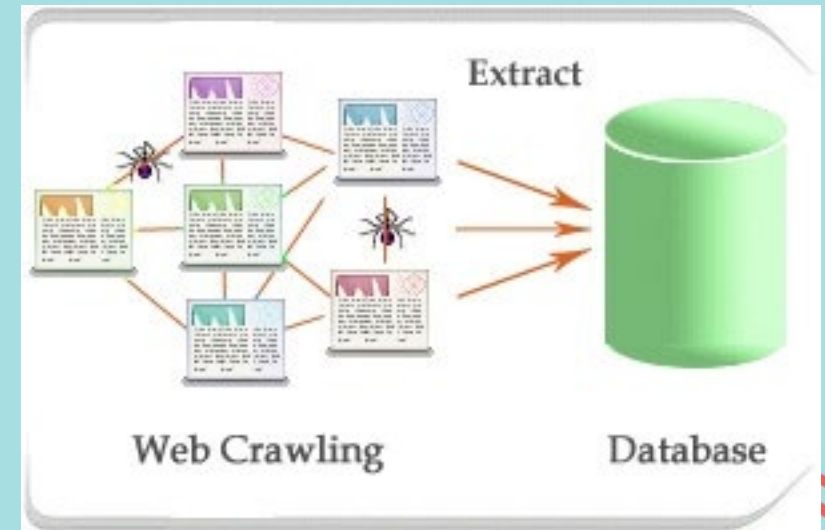
課程大綱

- W1-Introduction
- W2-Python libraries
- W3-BeautifulSoup(1)
- W4-BeautifulSoup(2)
- W5-
- W6-Scrapy(1)
- W7-Scrapy(2)
- W8-Storing Data
- W9-Midterm project
- W10-Web & HTTP
- W11-Flask
- W12-Flask Routes
- W13-Jinja template
- W14-Flask-form
- W15-Flask-mail
- W16-REST API
- W17-Project development(2)
- W18-Final presentation



Introduction to Web crawling

- What:
 - Web crawling is the technique for collecting data from different websites.
- How: (three major tools)
 - BeautifulSoup
 - Scrapy
 - Selenium



Beautiful Soup

 Beautiful Soup

latest

Search docs

Beautiful Soup Documentation

Quick Start

Installing Beautiful Soup

Making the soup

Kinds of objects

Navigating the tree

Searching the tree

Modifying the tree

Output

Specifying the parser to use

Encodings

Line numbers

[Docs](#) » Beautiful Soup Documentation

[View page source](#)

Beautiful Soup Documentation

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.


These instructions illustrate all major features of Beautiful Soup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

This document covers Beautiful Soup version 4.8.1. The examples in this documentation should work the same way in Python 2.7 and Python 3.2.





Scrapy



Scrapy


An open source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.

Maintained by [Zyte](#) (formerly Scrapinghub) and [many other contributors](#)

pypi **v2.8.0** wheel **yes** coverage **89%** Anaconda.org **2.8.0**

[Download](#) [Documentation](#) [Resources](#) [Community](#) [Commercial Support](#) [FAQ](#) [Fork on GitHub](#)

Install the latest version of Scrapy

 **Scrapy 2.8.0**

`$ pip install scrapy`

[PyPI](#) [Conda](#) [Release Notes](#)

Build and run your web spiders

```
$ pip install scrapy
$ cat > myspider.py <<EOF
import scrapy

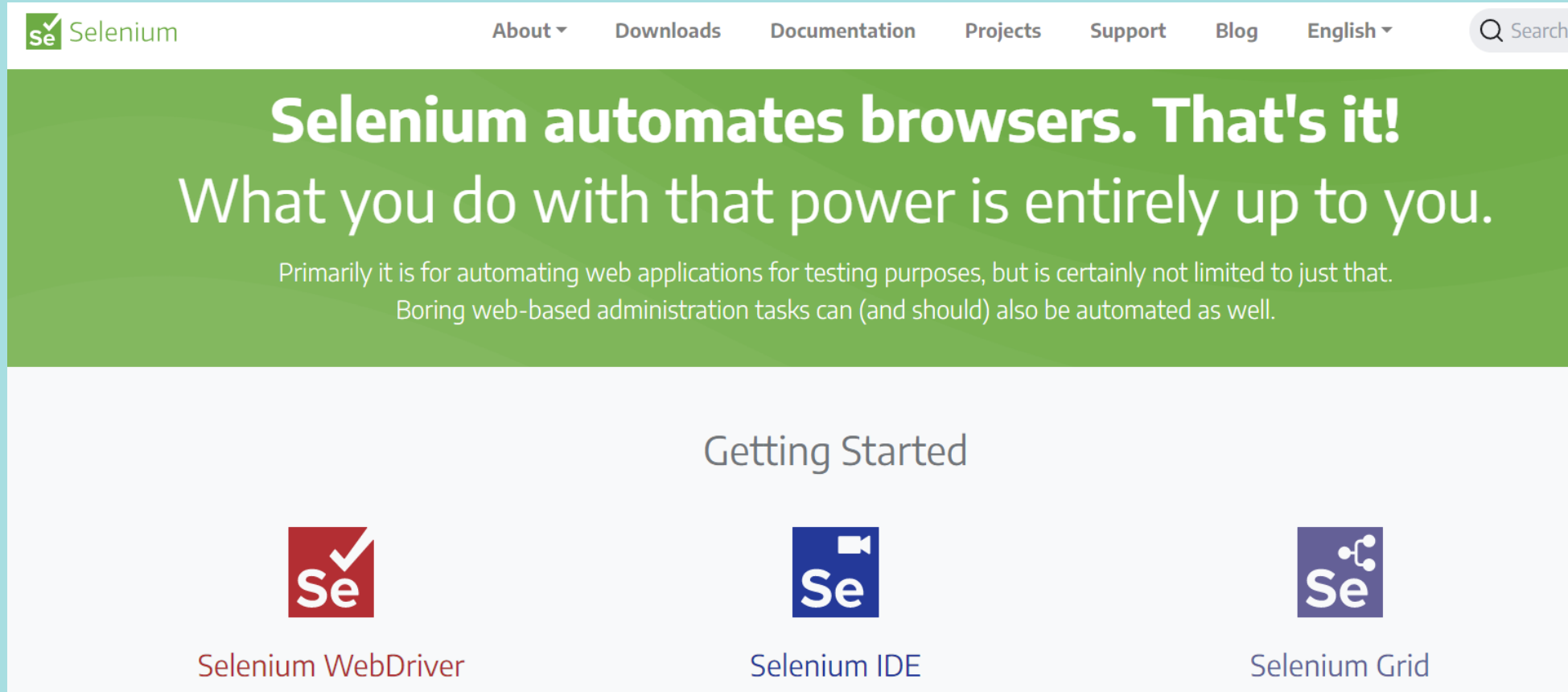
class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['https://www.zyte.com/blog/']

    def parse(self, response):
        for title in response.css('.oxy-post-title'):
            yield {'title': title.css('::text').get()}

        for next_page in response.css('a.next'):
            yield response.follow(next_page, self.parse)
EOF
$ scrapy runspider myspider.py
```



Selenium



The screenshot shows the Selenium website homepage. At the top is a navigation bar with links for About, Downloads, Documentation, Projects, Support, Blog, and English, along with a search bar. The main content area has a green header with the text "Selenium automates browsers. That's it! What you do with that power is entirely up to you." Below this is a paragraph explaining that Selenium is primarily for automating web applications for testing, but can also handle boring web-based administration tasks. The "Getting Started" section features three icons: Selenium WebDriver (red square with a white checkmark and 'Se'), Selenium IDE (blue square with a white video camera icon and 'Se'), and Selenium Grid (purple square with a white network icon and 'Se').

Selenium


About Downloads Documentation Projects Support Blog English Search

Selenium automates browsers. That's it!


What you do with that power is entirely up to you.

Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should) also be automated as well.


Getting Started



Selenium WebDriver



Selenium IDE



Selenium Grid



Beautiful Soup Demo-1

CO PRO ACP111En_Midterm.ipynb 檔案 編輯 檢視畫面 插入 執行階段 工具 說明

+ 程式碼 + 文字 複製到雲端硬碟

BeautifulSoup

Target

Crawle professors' disciplines from https://csie.asia.edu.tw/en/associate_professors_2

Step 1

```
[ ] from urllib.request import urlopen
    from bs4 import BeautifulSoup
    import re
    pages = set()
    html = urlopen('https://csie.asia.edu.tw/en/associate_professors_2')
    bs = BeautifulSoup(html, 'html.parser')
    for link in bs.find_all('a', href=re.compile('^(http://research.asia.edu.tw/TchEportfolio/)')):
        if link.attrs['href'] not in pages:
            #We have encountered a new page
            newPage = link.attrs['href']
            pages.add(newPage)
            print(newPage)
```



<https://reurl.cc/8qMAY7>



Beautiful Soup Demo-2

/ Experience

- 亞洲大學資訊工程學系 - 講座教授 (2021-04)
- 中央研究院 - 特聘研究員 (2008-03 ~ 2021-03)
- 中央研究院 - 研究員 (1989-08 ~ 2008-02)

專長 / Discipline expertise

- 演算法分析/Analysis of Algorithm
- 生物資訊/Bioinformatics
- 計算語言/Computational Linguistics
- 自然語言理解/Natural Language Understanding
- 智慧型對話系統/Intelligent dialogue system

```

::before
  <div class="col-md-12">
    <div class="row row-bottom-padded-sm">
      ::before
        <div class="col-md-12">
          <span class="heading-meta">Profile</span>
          <h2 class="colorlib-heading"></h2>
          <div id="div_profile" runat="server" class="about-desc">
            <ul class="animate-box fadeInLeft animated" data-animate-effect="fadeInLeft"></ul>
            <p></p>
            <ul class="animate-box fadeInLeft animated" data-animate-effect="fadeInLeft"></ul>
            <p></p>
            <ul class="animate-box fadeInLeft animated" data-animate-effect="fadeInLeft"></ul>
            <p>
              <strong>專長 / Discipline expertise</strong>
            </p>
            <ul class="animate-box fadeInLeft animated" data-animate-effect="fadeInLeft">
              <li>
                ::marker
                "演算法分析/Analysis of Algorithm"
              </li>
              <li> == $0
                ::marker
                "生物資訊/Bioinformatics"
              </li>
              <li></li>
              <li></li>
              <li></li>
            </ul>
          </div>
        </div>
      </div>
    </div>
  </div>

```


Beautiful Soup Demo-3

/ Experience

- 亞洲大學資訊工程學系 - 講座教授 (2021-04)
- 中央研究院 - 特聘研究員 (2008-03 ~ 2021-03)
- 中央研究院 - 研究員 (1989-08 ~ 2008-02)

專長 / Discipline expertise

- 演算法分析/Analysis of Algorithm
- 生物資訊/Bioinformatics
- 計算語言/Computational Linguistics
- 自然語言理解/Natural Language Understanding
- 智慧型對話系統/Intelligent dialogue system

HTML structure snippet:

```
...  
<div class="col-md-12">  
  <span class="heading-meta">Profile</span>  
  <h2 class="colorlib-heading"></h2>  
  <div id="div_profile" runat="server" class="about-desc">  
    <ul class="animate-box fadeInLeft animated" data-animate-effect="fadeInLeft"></ul>  
    <p></p>  
    <ul class="animate-box fadeInLeft animated" data-animate-effect="fadeInLeft"></ul>  
    <p></p>  
    <ul class="animate-box fadeInLeft animated" data-animate-effect="fadeInLeft"></ul>  
    <p></p>  
    <strong>專長 / Discipline expertise</strong>  
    <ul class="animate-box fadeInLeft animated" data-animate-effect="fadeInLeft">  
      <li></li>  
      <li></li>  
      <li></li>  
      <li></li>  
      <li></li>  
    </ul>  
  </div>  
</div>  
...
```

```
the_word = 'Discipline expertise'  
for page in pages:  
    html2 = urlopen(page)  
    bs2 = BeautifulSoup(html2, 'html.parser')  
    professor_name = bs2.find("h1", {"id": "colorlib-logo"})  
    professor_discipline = bs2.find(string=re.compile(the_word))  
    if professor_name:  
        print(professor_name.text.strip())  
    if professor_discipline:  
        print(professor_discipline.parent.find_next_sibling("ul").text)
```

Scrapy Demo-1

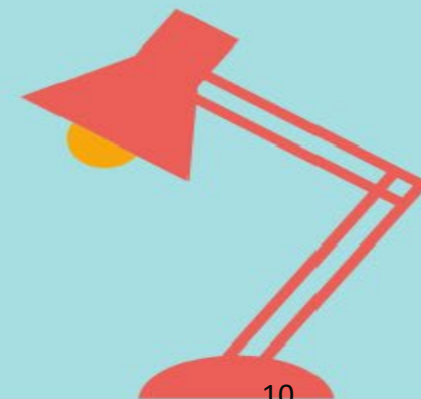
```
scrapy startproject csie
cd csie/csie/spiders
code professor_spider.py
scrapy crawl professor
scrapy crawl professor -o professors.json
scrapy crawl professor -o professors.json
scrapy crawl professor -o professors.csv
```

```
[ ] #professor_spider.py
import scrapy
class ProfessorSpider(scrapy.Spider):
    name = "professor"
    start_urls = [
        "https://csie.asia.edu.tw/zh_tw/associate_professors_2",
    ]

    def parse(self, response):
        for professor in response.xpath('//span[@class="i-member-value member-data-value-name"]/text()'):
            yield {
                'name': professor.get(),
            }
```



<https://reurl.cc/8qMAY7>



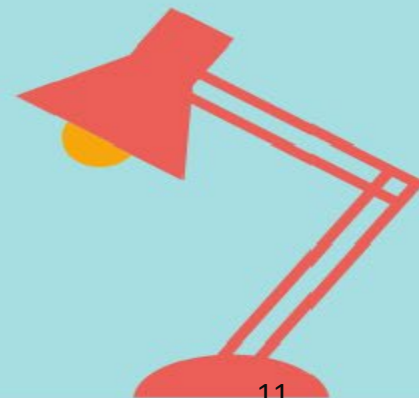
Scrapy Demo-2

```
#discipline_spider.py
import scrapy
from bs4 import BeautifulSoup
import re

class DisciplineSpider(scrapy.Spider):
    name = "discipline"
    def start_requests(self):
        urls = [
            'https://csie.asia.edu.tw/zh_tw/associate_professors_2',
        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)
            self.log(f'resuest url {url}')

    def parse(self, response):
        bs = BeautifulSoup(response.text, 'lxml')
        for link in bs.find_all('a', href=re.compile('^(http://research.asia.edu.tw/TchEportfolio/)')):
            sub_url = link.attrs['href']
            yield scrapy.Request(url=sub_url, callback=self.parse_discipline)

    def parse_discipline(self, response):
        bs2 = BeautifulSoup(response.body, 'html.parser')
        the_word = 'Discipline expertise'
        professor_name = bs2.find("h1", {"id": "colorlib-logo"})
        professor_discipline = bs2.find(string=re.compile(the_word))
        yield{
            'name': professor_name.text.strip(),
            'discipline': professor_discipline.parent.find_next_sibling("ul").text,
        }
```



Scrapy Demo-3

```
#discipline2_spider.py
import scrapy
class Discipline2Spider(scrapy.Spider):
    name = "discipline2"
    def start_requests(self):
        urls = [
            'https://csie.asia.edu.tw/zh_tw/associate_professors_2',
        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        for link in response.xpath('//a[contains(@href, "research.asia.edu.tw")]/@href').extract():
            yield scrapy.Request(link, callback=self.parse_discipline)

    def parse_discipline(self, response):
        professor_name = response.xpath('//h1[@id="colorlib-logo"]/a/text()').get()
        professor_disciplines = response.xpath('//strong[contains(text(), "Discipline expertise")]/following::ul[1]//li/text()').extract()
        if professor_name:
            yield {
                'url': response.request.url,
                'name': professor_name.strip(),
                'discipline': professor_disciplines,
            }
```

Selenium Demo

```
from selenium import webdriver
from bs4 import BeautifulSoup
import re

browser = webdriver.Chrome(executable_path = 'C:\ProgramData\chocolatey\lib\chromedriver\tools\chromedriver.exe')
browser.get("https://csie.asia.edu.tw/en/associate_professors_2")
bs = BeautifulSoup(browser.page_source, 'html.parser')
browser.close()
pages = set()

for link in bs.find_all('a', href=re.compile('^(http://research.asia.edu.tw/TchEportfolio/)')):
    if link.attrs['href'] not in pages:
        #We have encountered a new page
        newPage = link.attrs['href']
        pages.add(newPage)
        print(newPage)
```

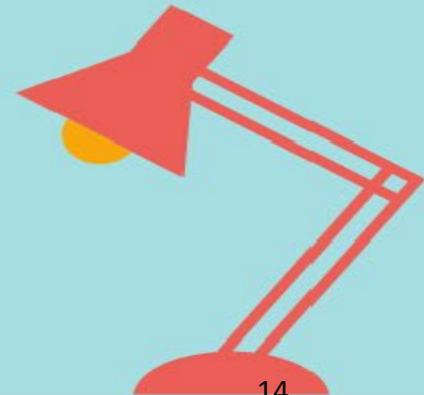


```
Anaconda Prompt (Anaconda3)

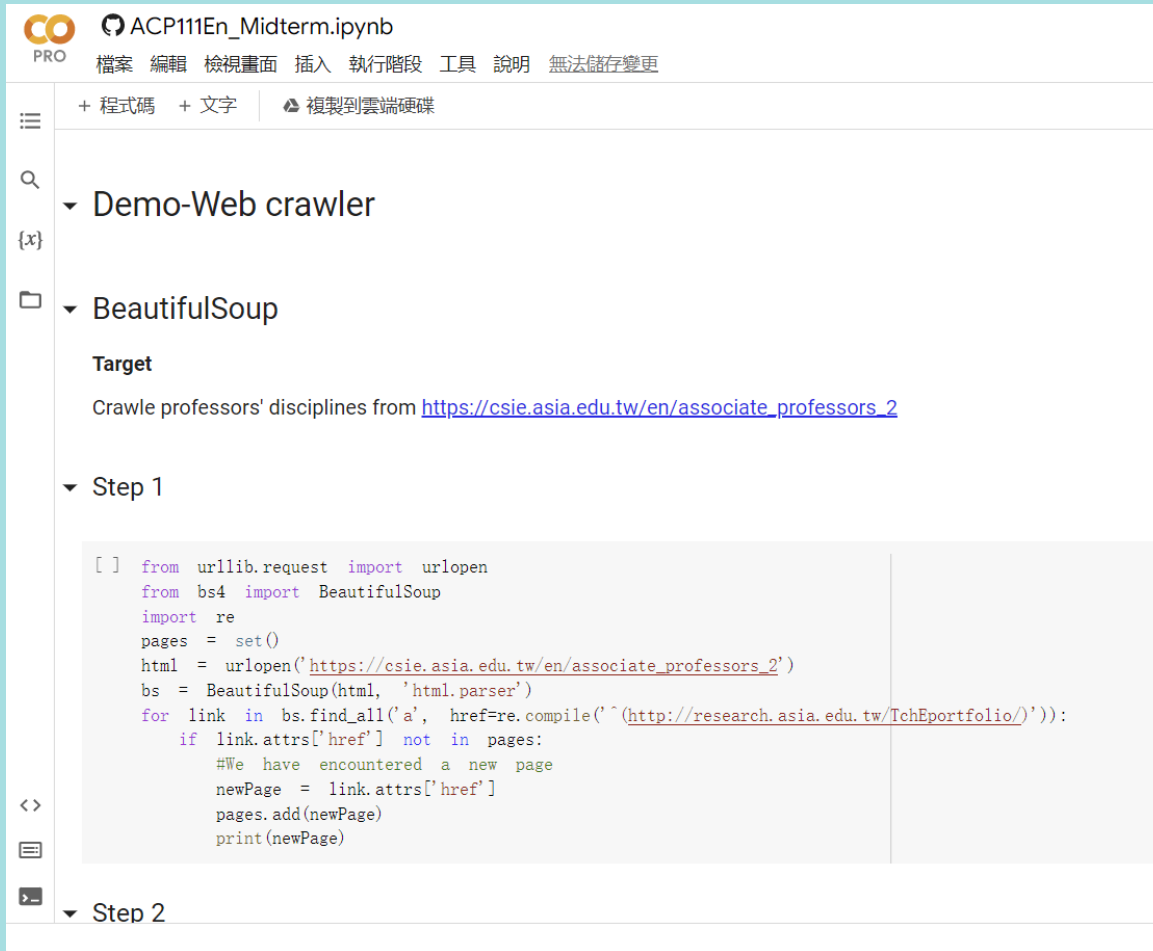
(base) D:\scrapy\selenium>python visitweb.py
D:\scrapy\selenium\visitweb.py:6: DeprecationWarning: executable_path has been deprecated, please pass in a Service object
    browser = webdriver.Chrome(options=op, executable_path = 'C:\ProgramData\chocolatey\lib\chromedriver\tools\chromedriver.exe')
DevTools listening on ws://127.0.0.1:63466/devtools/browser/893a21eb-1323-4507-a6fb-053c0ed22c71
```

Summary of Web Scraping Tools

- Beautiful Soup
 - A **library** that makes it easy to scrape information from web pages.
- Scrapy
 - A **framework** for extracting the data you need from websites in a fast, simple, yet extensible way.
- Selenium
 - Primarily it is for **automating browsers** for testing purposes.



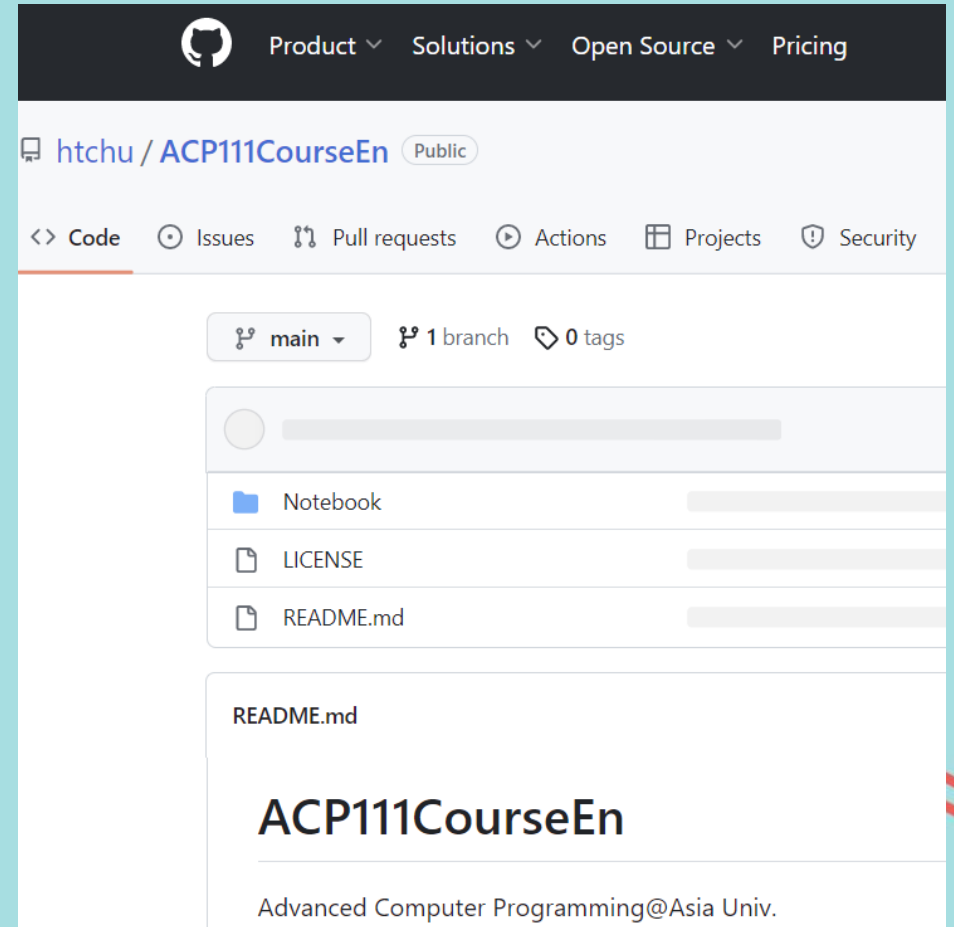
Colab notebook for the demos



The screenshot shows a Google Colab notebook interface. At the top, the title is 'ACP111En_Midterm.ipynb'. Below the title bar, there are tabs for '程式碼' (Code) and '文字' (Text), and a button for '複製到雲端硬碟' (Copy to Google Drive). The left sidebar shows a file explorer with a folder named 'Demo-Web crawler' and a file named 'BeautifulSoup'. The main area displays the code for 'Step 1', which is a web crawler using BeautifulSoup and urllib. The code imports the necessary libraries, sets up a set of pages to crawl, and uses BeautifulSoup to parse the HTML and find links to new pages. The code is as follows:

```
[ ] from urllib.request import urlopen
    from bs4 import BeautifulSoup
    import re
    pages = set()
    html = urlopen('https://csie.asia.edu.tw/en/associate_professors_2')
    bs = BeautifulSoup(html, 'html.parser')
    for link in bs.find_all('a', href=re.compile('^(http://research.asia.edu.tw/TchEportfolio/)')):
        if link.attrs['href'] not in pages:
            #We have encountered a new page
            newPage = link.attrs['href']
            pages.add(newPage)
            print(newPage)
```

Below the code, there is a section for 'Step 2'.



FullStack = Front-end + Back-end

jackjyq / fullstack_tutorial Public archive

<> Code Issues 1 Pull requests 19 Actions Security Insights

master 20 branches 0 tags Go to file <> Code

jackjyq fix typo defae2a on Apr 8, 2021 45 commits

backend	update backend	2 years ago
docs	update frontend	2 years ago
frontend	fix typo	2 years ago
.gitignore	add documentation	4 years ago
README.md	update backend	2 years ago

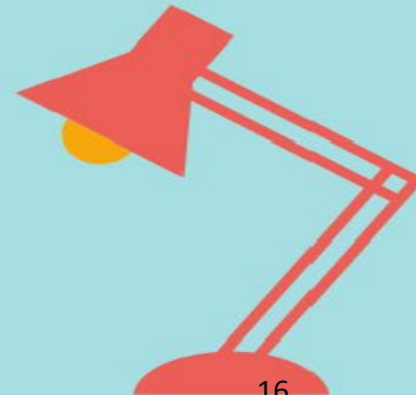
☰ README.md

FullStack Tutorial

managed by ppm

In this fullstack tutorial, I will implement a simple CRUD App, including:

- A Frontend using JavaScript(React)
- A Backend using Python(Flask)





python



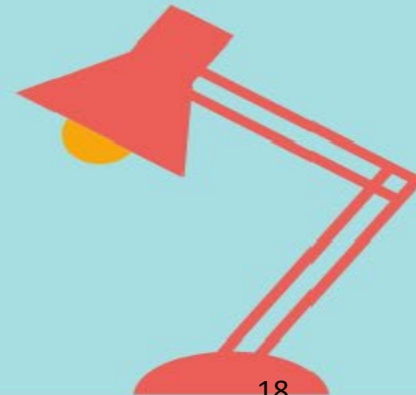
Flask

web development,
one drop at a time



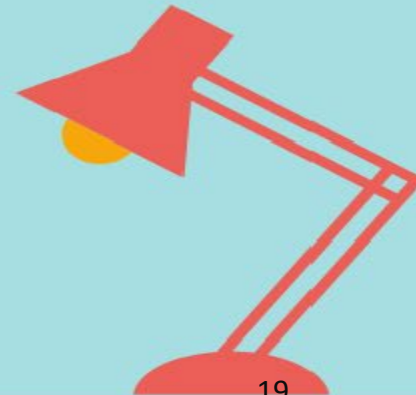
Front-end Design

- Front-end covers design and implementation of user interface, interaction, animation effects, etc., including:
 - Front-end Development: Use HTML, CSS, JavaScript and other technologies to realize front-end design, including converting design into web pages, web animation, interactive effects, etc.
 - Cross-platform Design: Designers need to consider different browsers, operating systems, and devices so that web pages can run normally on various platforms.
 - User Experience Design (UX Design): Through user research and testing, designers design the structure, process, and content of the user interface to enhance user experience.
 - Web Design: Convert the visual design draft provided by the designer into an actual web page, design web page layout, color matching, fonts, etc., and consider the adaptability of different devices (computers, mobile phones, tablets).
 - Styling: Design the style of web page elements, such as color, font, size, border, background, etc., and implement these styles using CSS technology.
 - Interactive Design: Designers design how users interact with web pages, such as buttons, forms, slides, animation effects, etc.



Back-end Design

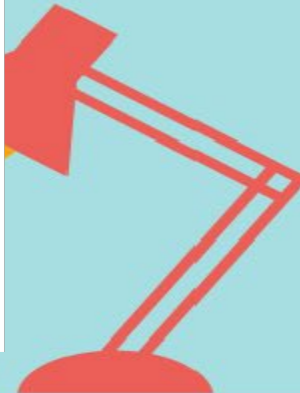
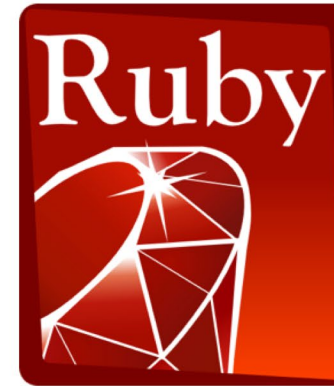
- Backend design refers to the process of designing and developing the server-side of a web application or software :
 1. Server-side programming languages: The choice of programming languages used to develop the backend logic, such as Java, Python, PHP, Ruby, or Node.js.
 2. Databases: The choice of database technology and design of database schema to manage the application's data.
 3. APIs: The design and implementation of APIs that allow the frontend of the application to communicate with the backend.
 4. Authentication and Authorization: Designing the backend to handle user authentication and authorization, such as password management, permissions, and access control.
 5. Server architecture: The design of the server architecture, including load balancing, caching, and other techniques to ensure scalability and high performance.



(Big) Frameworks



python

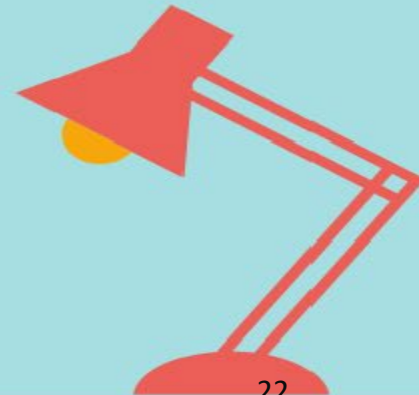


(Micro μ) Frameworks




PythonAnywhere – Free online Python execution environment

- Free account is limited as follows:
 - Only one App (Application) can be created
 - Off-net access to the Internet is limited
 - CPU and storage are limited (100 seconds of CPU time a day, 512MB of storage)
 - Does not provide Jupyter (but does have IPython)
 - There can only be two Consoles (Bash and Python)



Dashboard儀表板

 pythonanywhere

Dashboard Consoles Files Web Tasks Databases

Up here you will see instructions walking you through the key features of our Education beta.

(If after closing this helper, you want to go through it again – or try another one – go to the [Help page](#))

→

Close this tutorial

Dashboard

Welcome, [htchu](#)

CPU Usage: 0% used – 0.00s of 100s. Resets in 10 hours, 4 minutes [More Info](#)

File storage: 16% full – 80.1 MB of your 512.0 MB quota [More Info](#)

[Upgrade Account](#)

Recent Consoles [+](#) [5](#) [-](#)

You have no recent consoles.

[View all](#)

New console:

[\\$ Bash](#) [>>> Python](#) [More...](#)

Recent Files [+](#) [5](#) [-](#)

[/home/htchu/mysite/flask_app.py](#)

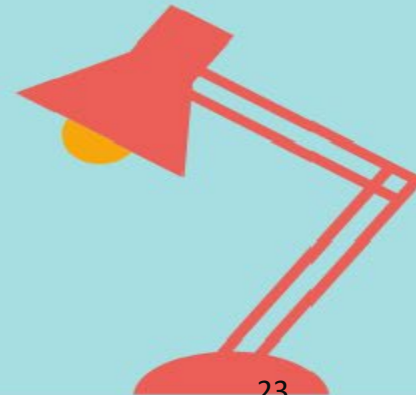
[/home/htchu/mysite3/urls.py](#)

[/home/htchu/mysite3/polls/urls.py](#)


[/home/htchu/mysite3/polls/views.py](#)

[/home/htchu/mysite3/manage.py](#)

[+ Open another file](#) [Browse files](#)

Recent Notebooks [+](#) [5](#) [-](#)Your account does not support Jupyter Notebooks. [Upgrade your account](#) to get access!**All Web apps**[htchu.pythonanywhere.com](#)[Open Web tab](#)


Command Console 命令控制台

 pythonanywhere

Dashboard **Consoles** Files Web Tasks Databases

Up here you will see instructions walking you through the key features of our Education beta.


(If after closing this helper, you want to go through it again – or try another one – go to the [Help page](#))




Close this tutorial

CPU Usage: 0% used – 0.00s of 100s. Resets in 16 hours, 2 minutes [More Info](#)

Start a new console:

Python: [3.8](#) / [3.7](#) / [3.6](#) / [3.5](#) / [2.7](#) IPython: [3.8](#) / [3.7](#) / [3.6](#) / [3.5](#) / [2.7](#) PyPy: [2](#) / [3](#)
Other: [Bash](#) | [MySQL](#)
Custom: 


Your consoles:

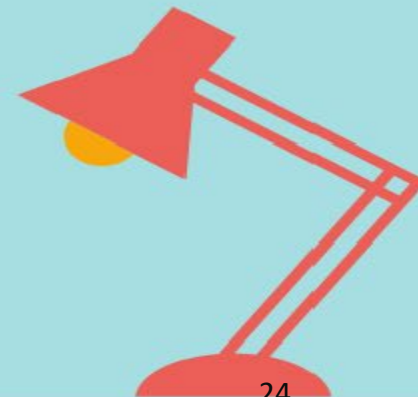
[flask_app.py \(Python3.8\)](#) 

Consoles shared with you


No-one has shared any consoles with you :-(

Running processes

 [Fetch process list](#)



Files

 pythonanywhere


Dashboard Consoles **Files** Web Tasks Databases

Up here you will see instructions walking you through the key features of our Education beta.

(If after closing this helper, you want to go through it again – or try another one – go to the [Help page](#))








→

Close this tutorial

/home/  htchu [Open Bash console here](#) **16% full** – 80.1 MB of your 512.0 MB quota [More Info](#)





























Directories

Enter new directory name [New directory](#)

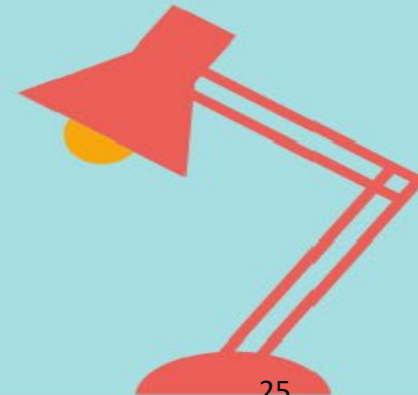
.cache/	
.config/	
.local/	
.virtualenvs/	
mysite/	
mysite2/	
mysite3/	

Files


Enter new file name, eg hello.py [New file](#)

 .bashrc	  	2021-05-04 07:38	559 bytes
 .gitconfig	  	2021-05-04 07:38	266 bytes
 .profile	  	2021-05-04 07:38	79 bytes
 .python_history	  	2021-05-04 08:18	7 bytes
 .pythonstartup.py	  	2021-05-04 07:38	77 bytes
 .vimrc	  	2021-05-04 07:38	4.6 KB
 README.txt	  	2021-05-04 07:38	232 bytes

[Upload a file](#)
100MiB maximum size




Web

 pythonanywhere

Dashboard Consoles Files **Web** Tasks Databases


Up here you will see instructions walking you through the key features of our Education beta.

(If after closing this helper, you want to go through it again – or try another one – go to the [Help page](#))




Close this tutorial

htchu.pythonanywhere.com



Configuration for htchu.pythonanywhere.com


Reload:



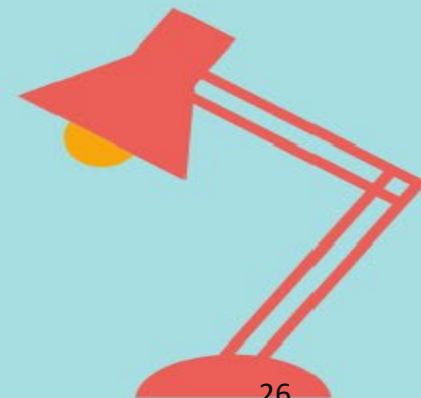
Best before date:

We're happy to host your free website – and keep it free – for as long as you want to keep it running, but you'll need to log in at least once every three months and click the "Run until 3 months from today" button below. We'll send you an email a week before the site is disabled so that you don't forget to do that. [See here for more details.](#)


This site has expired, click the button below to reactivate it



Paying users' sites stay up forever without any need to log in to keep them running.




Tasks任務

 pythonanywhere

Dashboard Consoles Files Web **Tasks** Databases

Up here you will see instructions walking you through the key features of our Education beta.

(If after closing this helper, you want to go through it again – or try another one – go to the [Help page](#))



Close this tutorial

CPU Usage: 0% used – 0.00s of 100s. Resets in 15 hours, 58 minutes [More Info](#)

Scheduled tasks

Server time: 02:15 UTC

Daily, at : UTC

Create

Frequency	Time	Command	Description	Expiry	Actions
You have no tasks yet.					

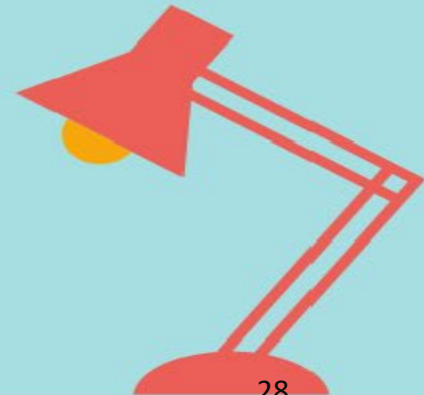
Your **Scheduled task** is a script that will run every day at a time of your choosing – you can use it to do stuff like scraping websites, or checking that your server is running.

[Paying users](#) can schedule several tasks, can run them both hourly and daily, and they never expire. Just sayin'...



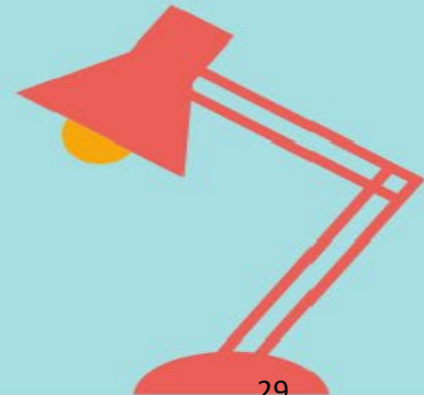
Web application frameworks

- JavaScript
 - Express.js, React.js, Angular.js
- PHP
 - Laravel, CodeIgniter
- Ruby
 - Rails
- Python
 - Django, Flask , FastAPI
- Java
 - Spring Boot



Flask framework

- Required
 - **Jinja**-template engine
 - **Werkzeug**-WSGI toolkit
- Optional
 - **sqlalchemy**-SQL toolkit
 - marshmallow: simplified object serialization
 - Celery-task queue



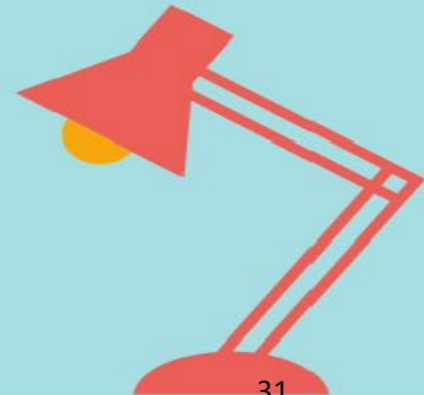
Flask extensions

- **Flask-Bootstrap:** Bootstrap
- **Flask-WTF:** WTForms including CSRF, file upload, and reCAPTCHA
- **Flask-Moment:** Localization of Dates and Times
- **Flask-Babel:** Internationalization and localization support
- **Flask-DebugToolbar:** In-browser debugging tools
- **Flask-Assets:** Integration of CSS and JavaScript assets
- **Flask-Session:** implementation of user sessions with server-side storage
- **Flask-SocketIO:** Socket.IO server implementation with support for WebSocket and long-polling



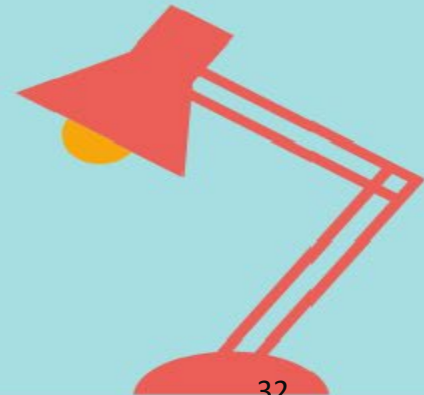
Flask

- Flask is a class with
 - run() function
 - route() functions
- Flask is a command
 - flask run
 - flask routes
 - flask shell



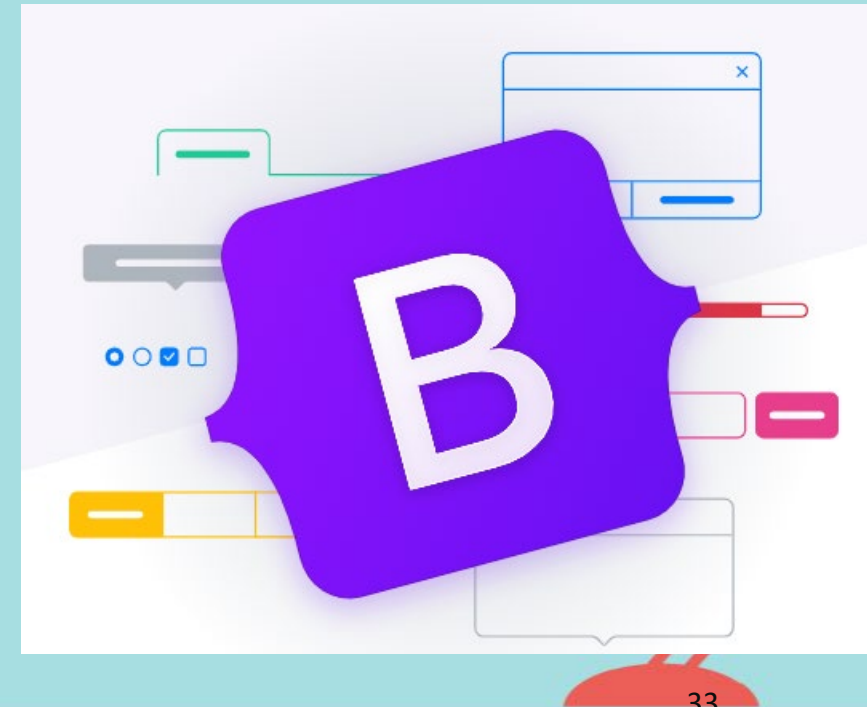
Flask environment

- Bash (Linux/Mac)
 - export FLASK_APP= appname
 - export FLASK_ENV=development
 - flask run
- Windows command
 - set FLASK_APP=appname
 - set FLASK_ENV=development
 - flask run



Bootstrap

- The most popular front-end toolkit in the world.
- Quickly design and customize responsive mobile-first sites
- Currently v5.1.3



Bootstrap-Flask

Bootstrap-Flask

license MIT pypi v2.0.2 build passing coverage 92%

Bootstrap-Flask is a collection of Jinja macros for Bootstrap 4 & 5 and Flask. It helps you to render Flask-related data and objects to Bootstrap markup HTML more easily:

- Render Flask-WTF/WTForms form object to Bootstrap Form.
- Render data objects (dict or class objects) to Bootstrap Table.
- Render Flask-SQLAlchemy `Pagination` object to Bootstrap Pagination.
- etc.

Installation

```
$ pip install -U bootstrap-flask
```

Example

Register the extension:

```
from flask import Flask
# To follow the naming rule of Flask extension, although
# this project's name is Bootstrap-Flask, the actual package
# installed is named `flask_bootstrap`.
from flask_bootstrap import Bootstrap5

app = Flask(__name__)
bootstrap = Bootstrap5(app)
```

Installation

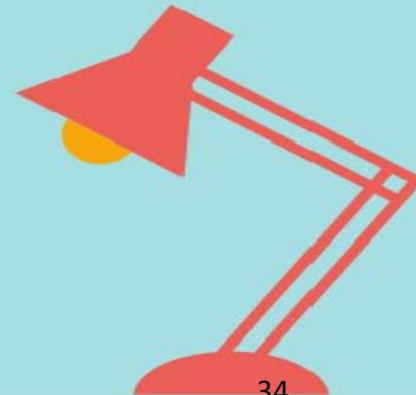
```
$ pip install -U bootstrap-flask
```

Example

Register the extension:

```
from flask import Flask
from flask_bootstrap import Bootstrap5
```

```
app = Flask(__name__)
bootstrap = Bootstrap5(app)
```



Jinja-the template language

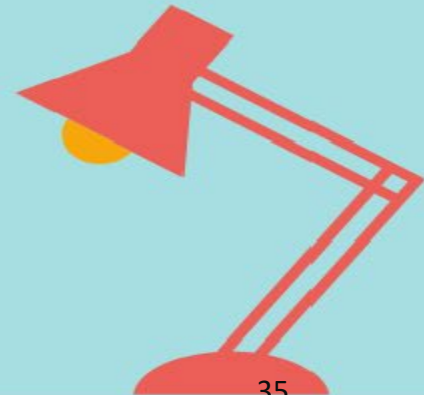
```
from jinja2 import Environment, PackageLoader, select_autoescape
env = Environment(
    loader=PackageLoader("yourapp"),
    autoescape=select_autoescape()
)
```

To load a template from this environment, call the `get_template()` method, which returns the loaded Template.

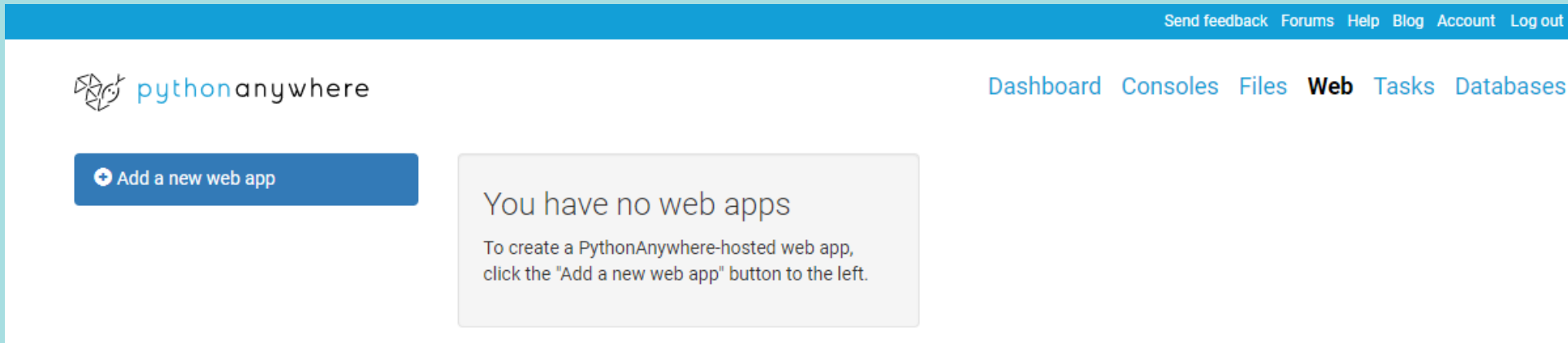
```
template = env.get_template("mytemplate.html")
```

To render it with some variables, call the `render()` method.

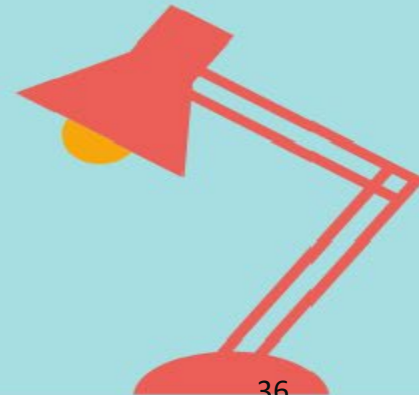
```
print(template.render(the="variables", go="here"))
```



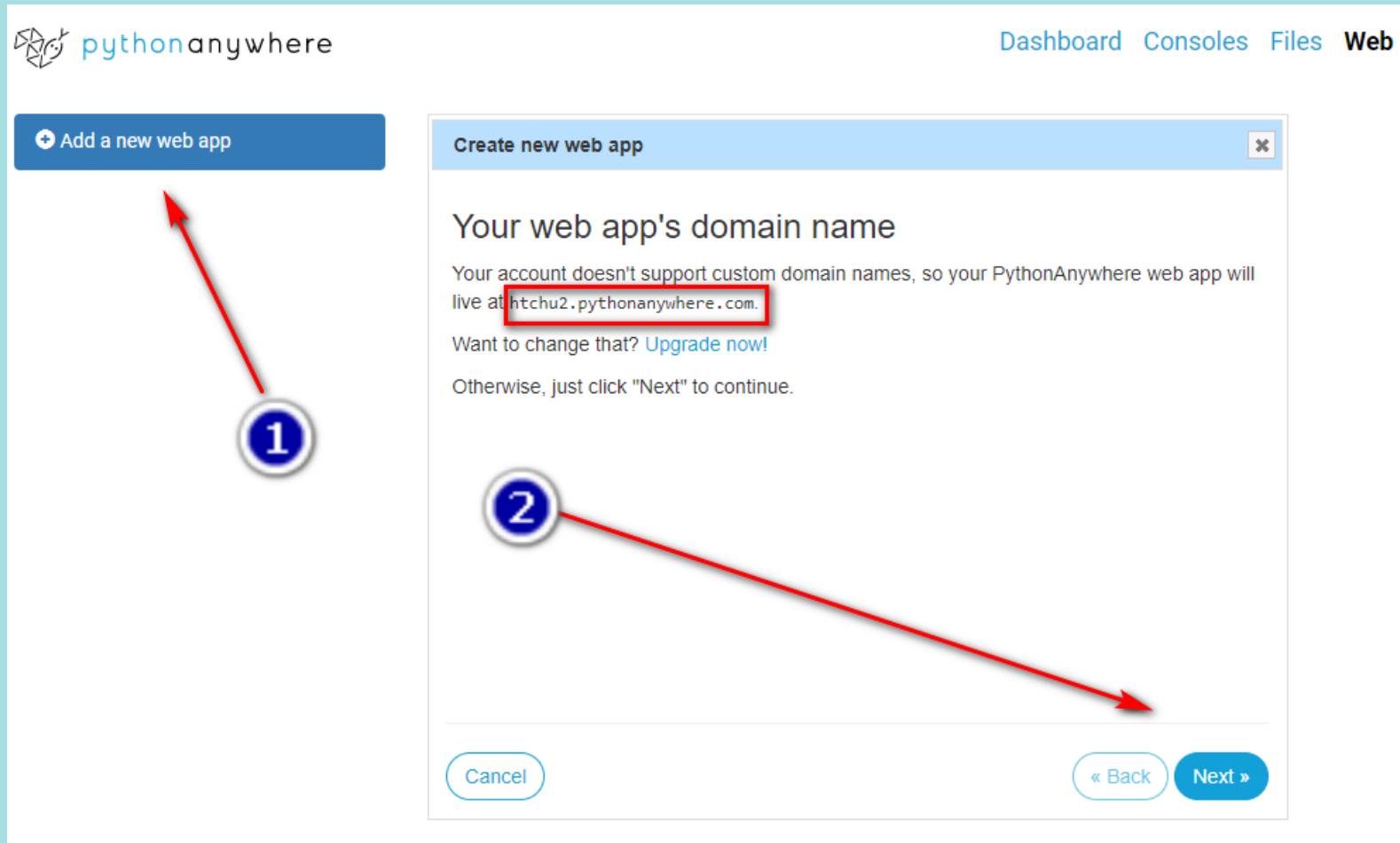
Step 1: Go to Web tab



The screenshot shows the PythonAnywhere dashboard interface. At the top, a blue navigation bar contains links for "Send feedback", "Forums", "Help", "Blog", "Account", and "Log out". Below this, the "pythonanywhere" logo is on the left, and a horizontal menu on the right includes "Dashboard", "Consoles", "Files", "Web" (which is bolded and underlined), "Tasks", and "Databases". On the left side of the main content area, there is a blue button with a plus icon and the text "Add a new web app". To the right of this button is a light gray box with the heading "You have no web apps" and the text "To create a PythonAnywhere-hosted web app, click the 'Add a new web app' button to the left."



Step 2: Add a new web app



The screenshot shows the PythonAnywhere dashboard with the 'Web' tab selected. A blue button labeled 'Add a new web app' is highlighted with a red arrow and a blue circle containing the number 1. A modal window titled 'Create new web app' is open, showing the domain name 'htchu2.pythonanywhere.com' highlighted with a red box and a red arrow pointing to it from a blue circle containing the number 2. The modal also includes a 'Next »' button.

pythonanywhere

Dashboard Consoles Files Web

+ Add a new web app

Create new web app

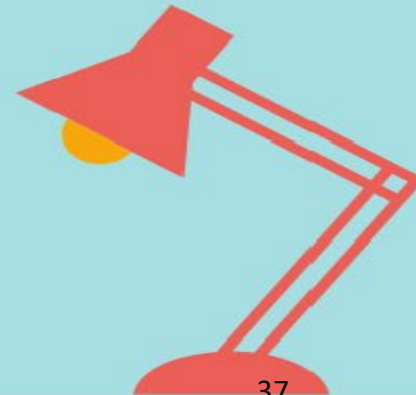
Your web app's domain name

Your account doesn't support custom domain names, so your PythonAnywhere web app will live at `htchu2.pythonanywhere.com`.

Want to change that? [Upgrade now!](#)

Otherwise, just click "Next" to continue.

Cancel « Back Next »



Step 3: Select a Python Web framework and a Python version

Select a Python Web framework

...or select "Manual configuration" if you want detailed control.

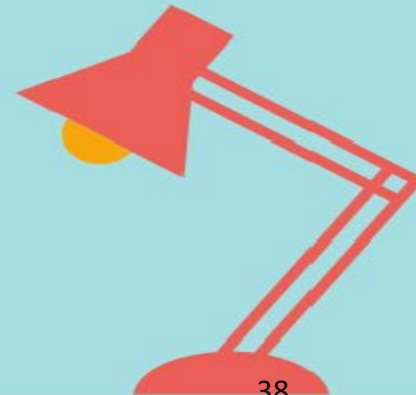
- » Django
- » web2py
- » Flask
- » Bottle
- » **Manual configuration** (including virtualenvs)

What other frameworks should we have here? Send us some feedback using the link at the top of the page!

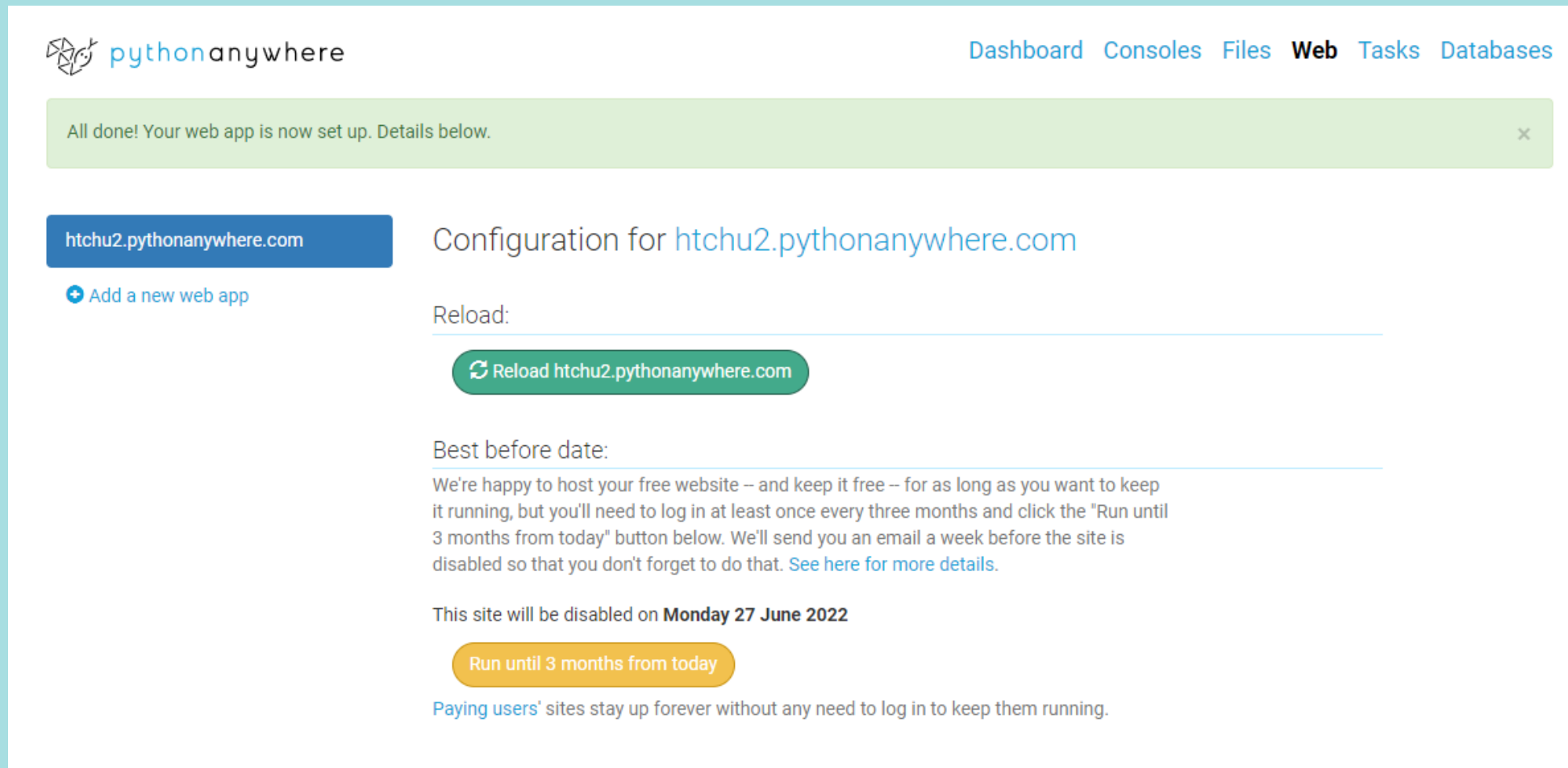
Select a Python version

- » Python 3.6 (Flask 2.0.0)
- » Python 3.7 (Flask 2.0.0)
- » Python 3.8 (Flask 2.0.0)
- » Python 3.9 (Flask 2.0.0)

Note: If you'd like to use a different version of Flask to the default version, you can use a virtualenv for your web app. There are [instructions here](#).



Step 4: Quick start new Flask project



The screenshot shows the PythonAnywhere web interface. At the top left is the PythonAnywhere logo. To the right are navigation links: Dashboard, Consoles, Files, Web (highlighted), Tasks, and Databases. A green success message at the top states: "All done! Your web app is now set up. Details below." Below this, on the left, is a blue button labeled "htchu2.pythonanywhere.com" and a link "+ Add a new web app". The main content area is titled "Configuration for htchu2.pythonanywhere.com". It contains a "Reload:" section with a green button "Reload htchu2.pythonanywhere.com". Below that is a "Best before date:" section with a paragraph explaining the free hosting policy and a yellow button "Run until 3 months from today". At the bottom, it states the site will be disabled on "Monday 27 June 2022" and includes a note for paying users.

pythonanywhere

Dashboard Consoles Files **Web** Tasks Databases

All done! Your web app is now set up. Details below. ×

htchu2.pythonanywhere.com

+ Add a new web app

Configuration for htchu2.pythonanywhere.com

Reload:

Reload htchu2.pythonanywhere.com

Best before date:

We're happy to host your free website – and keep it free – for as long as you want to keep it running, but you'll need to log in at least once every three months and click the "Run until 3 months from today" button below. We'll send you an email a week before the site is disabled so that you don't forget to do that. [See here for more details.](#)

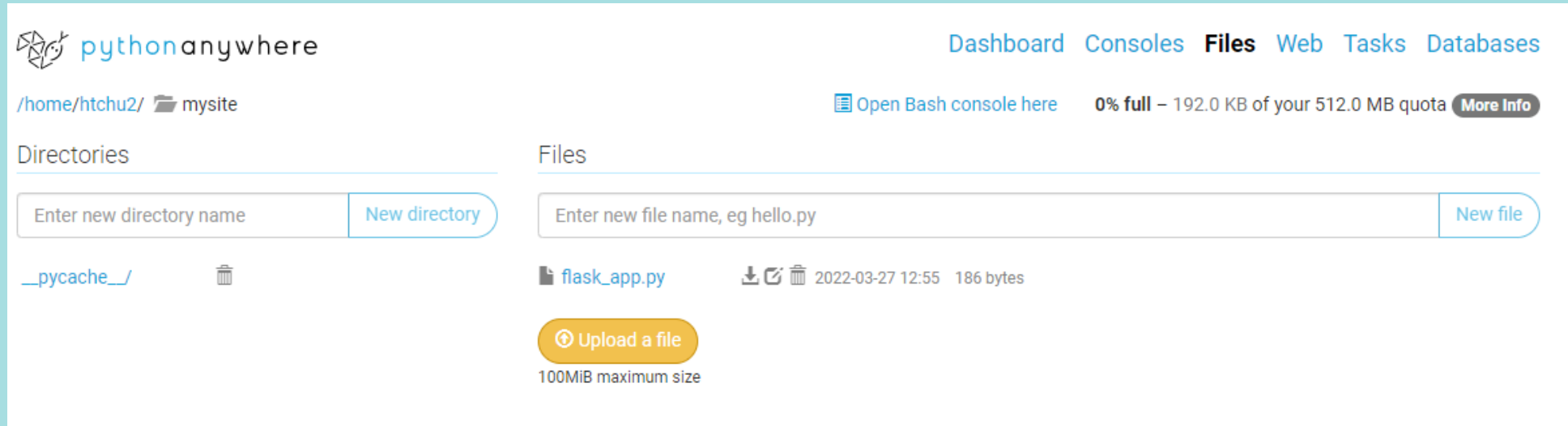
This site will be disabled on **Monday 27 June 2022**

Run until 3 months from today

[Paying users'](#) sites stay up forever without any need to log in to keep them running.



Step 5: check the files

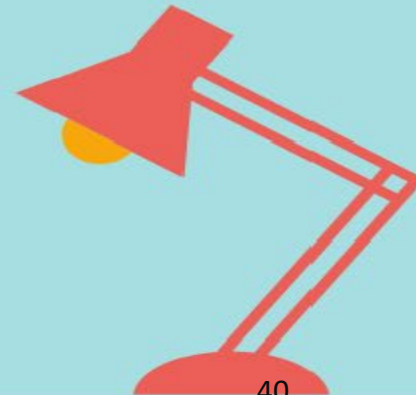


The screenshot shows the PythonAnywhere web interface. At the top left is the PythonAnywhere logo. To its right are navigation links: Dashboard, Consoles, **Files**, Web, Tasks, and Databases. Below the logo, the current path is shown as `/home/htchu2/` with a folder icon and the name `mysite`. On the right side of the header, there is a link to "Open Bash console here" and a storage status indicator: "0% full – 192.0 KB of your 512.0 MB quota" with a "More Info" button.


The main content area is divided into two sections: "Directories" on the left and "Files" on the right.

Directories: It features a text input field "Enter new directory name" and a "New directory" button. Below this, a directory named `__pycache__` is listed with a trash icon next to it.

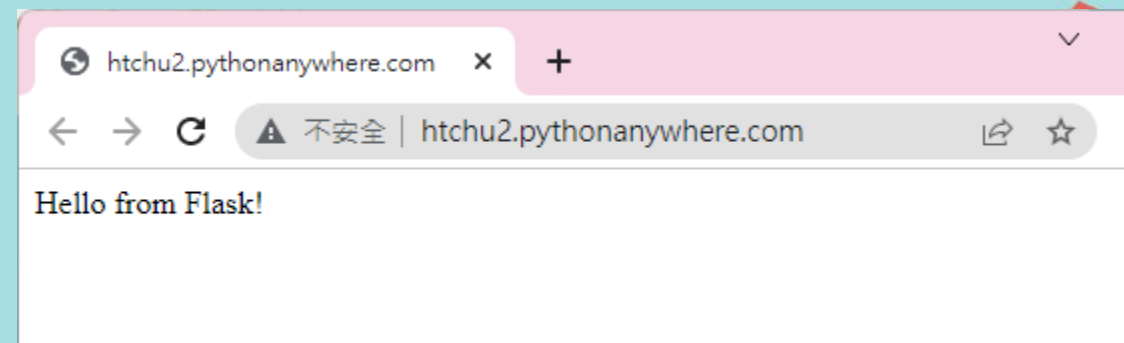
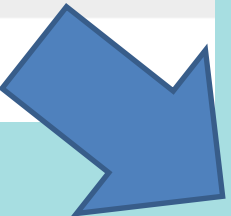
Files: It features a text input field "Enter new file name, eg hello.py" and a "New file" button. Below this, a file named `flask_app.py` is listed. To the left of the filename is a file icon. To the right are icons for download, copy, and delete, followed by the date and time "2022-03-27 12:55" and the size "186 bytes". Below the file list is an orange "Upload a file" button with a circular arrow icon, and the text "100MiB maximum size" below it.



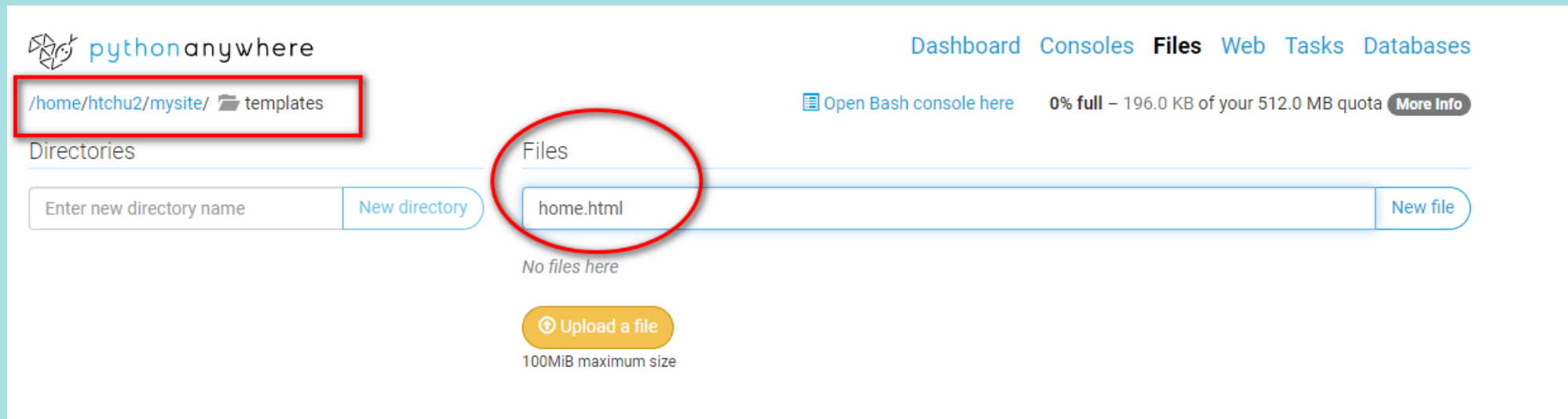
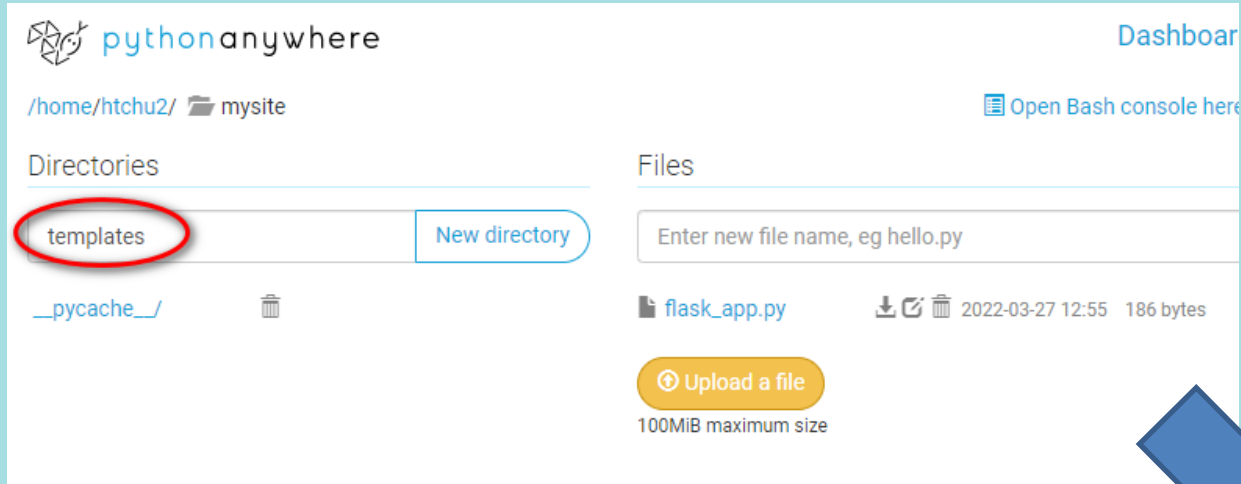
Step 6: check the program and the web app

 /home/htchu2/mysite/flask_app.py

```
1
2 # A very simple Flask Hello World app for you to get started with...
3
4 from flask import Flask
5
6 app = Flask(__name__)
7
8 @app.route('/')
9 def hello_world():
10     return 'Hello from Flask!'
11
12
```



Step 7: add html templates



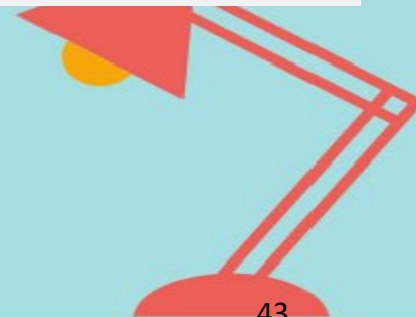
Step 8: edit home.html




- `<h1>Top 10 HTML tags</h1>`
`

`In this article we will explain our list of top 10 HTML tags.`

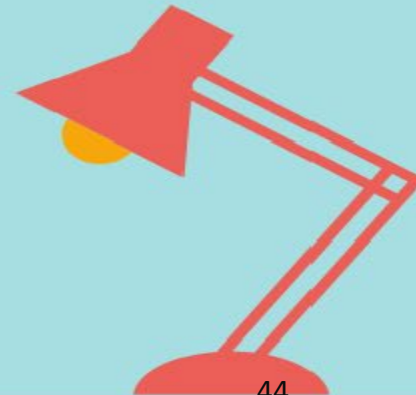
``<h2>First tag is`
```</h2>`Bold is all about making words more important.




Step 9: edit flask_app.py

 /home/htchu2/mysite/flask_app.py

```
1
2 # A very simple Flask Hello World app for you to get started with...
3
4 from flask import Flask, render_template
5
6 app = Flask(__name__)
7
8 @app.route('/')
9 def home():
10     return render_template("home.html")
11
```



Step 10: Reload web

 pythonanywhere

Dashboard Consoles Files Web

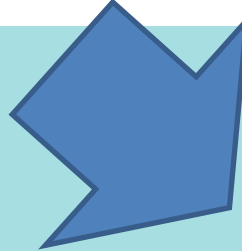
htchu2.pythonanywhere.com

+ Add a new web app

Configuration for htchu2.pythonanywhere.com

Reload:

Reload htchu2.pythonanywhere.com



htchu2.pythonanywhere.com x +

← → ↻ ⚠ 不安全 | htchu2.pythonanywhere.com

My First Web Page

Headings Are Great Fun

This is my first **paragraph** in my new *webpage*. This is going to be great. I am so excited I can hardly contain **myself**. Don't you just love paragraphs? I find them very useful.

Web Pages Are Exciting Too

Yes, that's right - web pages can be a lot of fun. Learning how to create web pages is easy and **entertaining**. This is my second *paragraph*. I hope you like it.

[Link to Google](#)

- Apples
- Bananas
- Pears
- Oranges
- Grapes

"A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away." - Saint Exupéry



Thanks!

Q&A

