2023-Spring
Advanced Computer Programming (11)

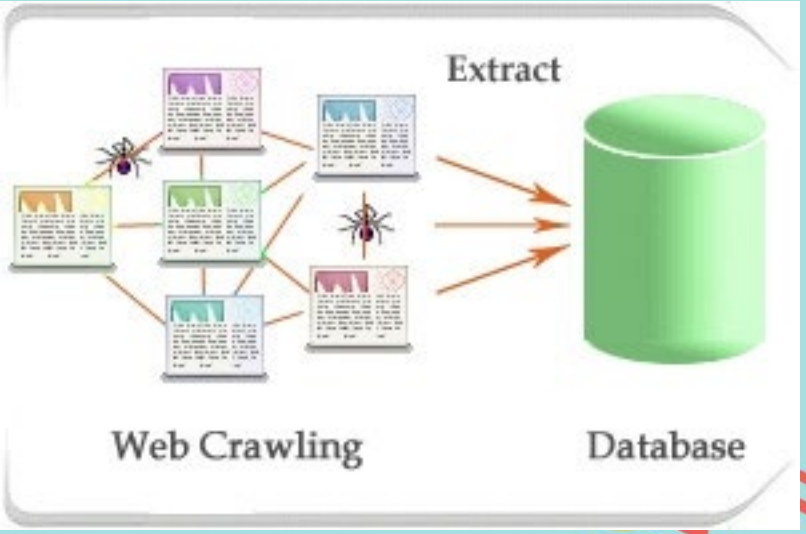
CSIE, Asia Univ.

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)
  - Beautiful Soup
  - Scrapy
  - Selenium

Beautiful Soup

Beautiful Soup

latest

Search docs

Docs » Beautiful Soup Documentation

View page source

Beautiful Soup Documentation

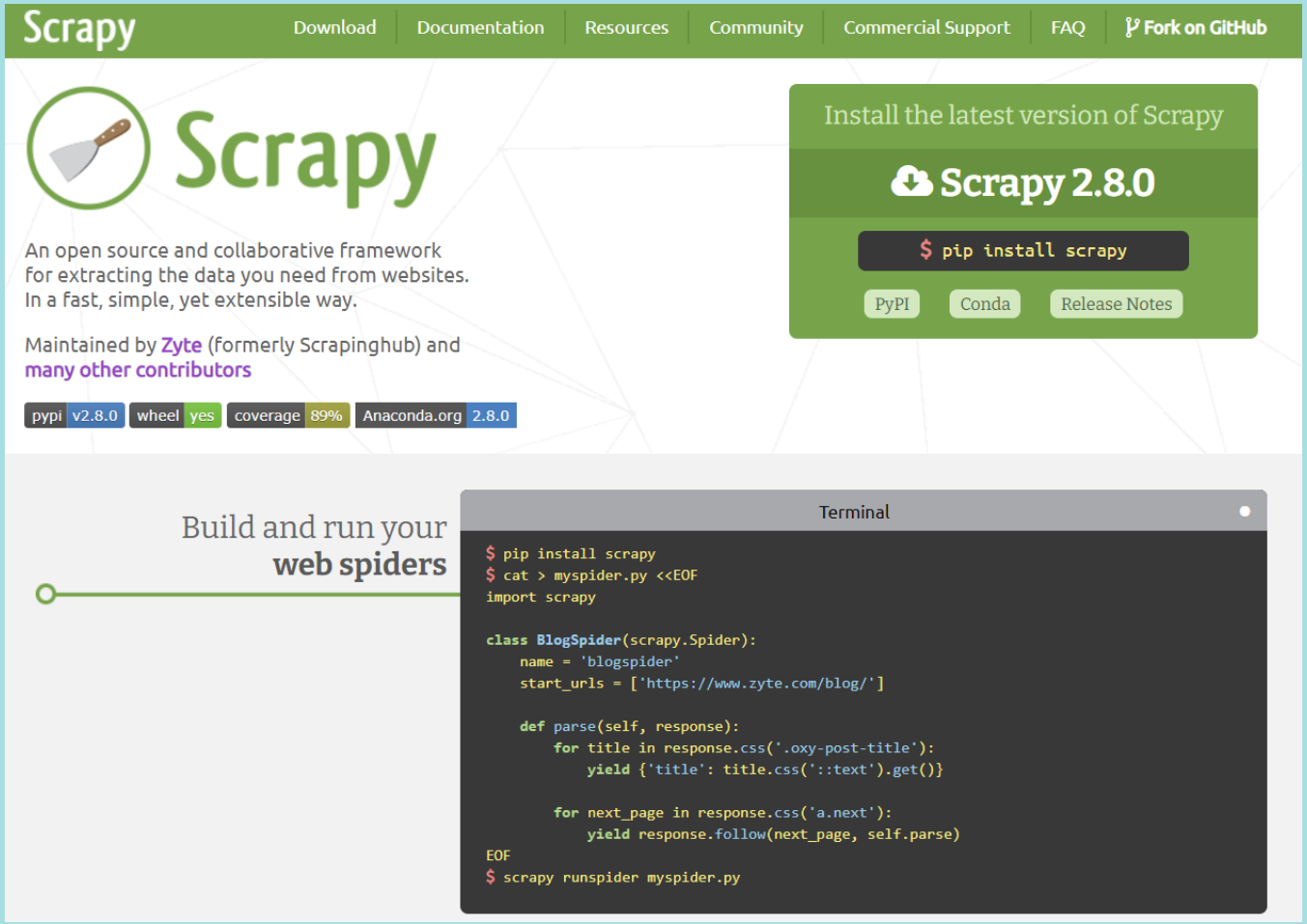
Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

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.

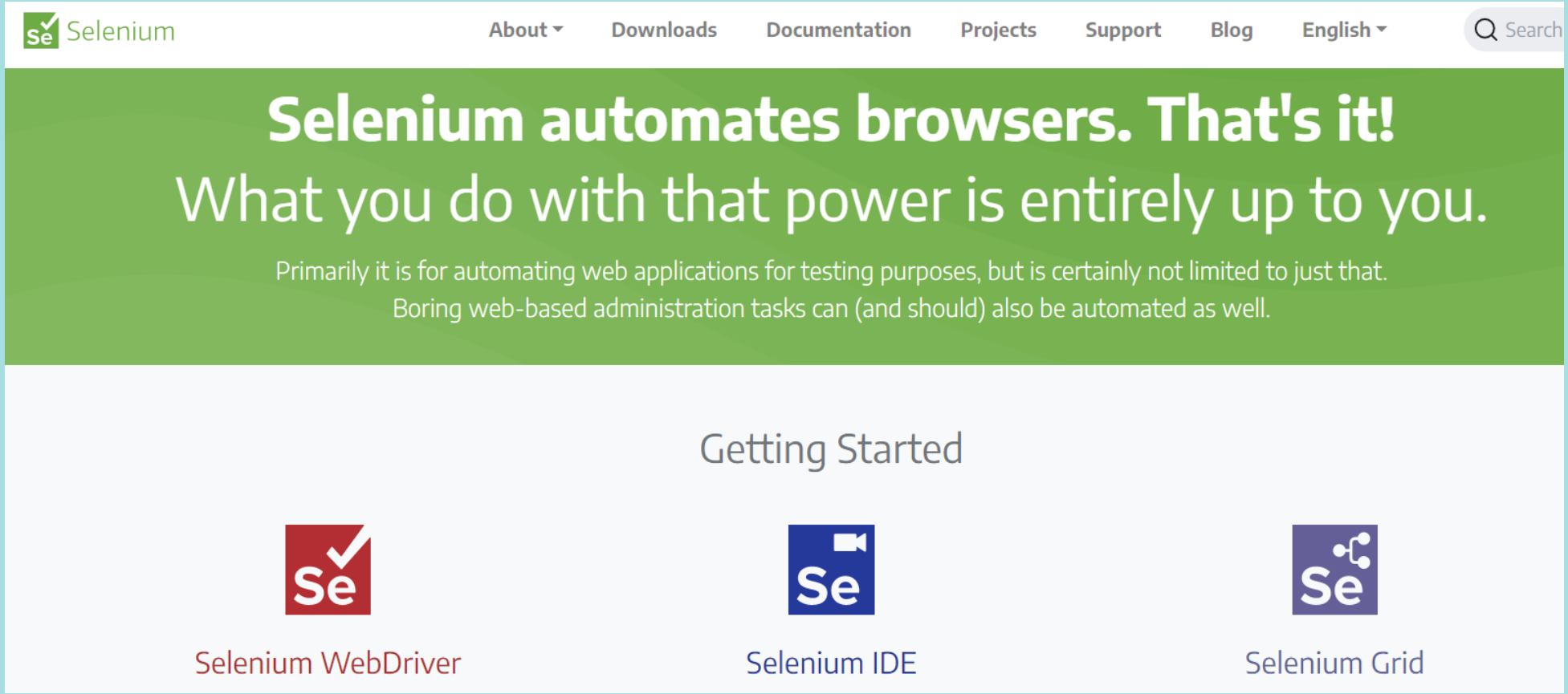
https://beautiful-soup-4.readthedocs.io/en/latest/

Scrapy



https://scrapy.org/

Selenium



https://www.selenium.dev/

Beautiful Soup Demo-1

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

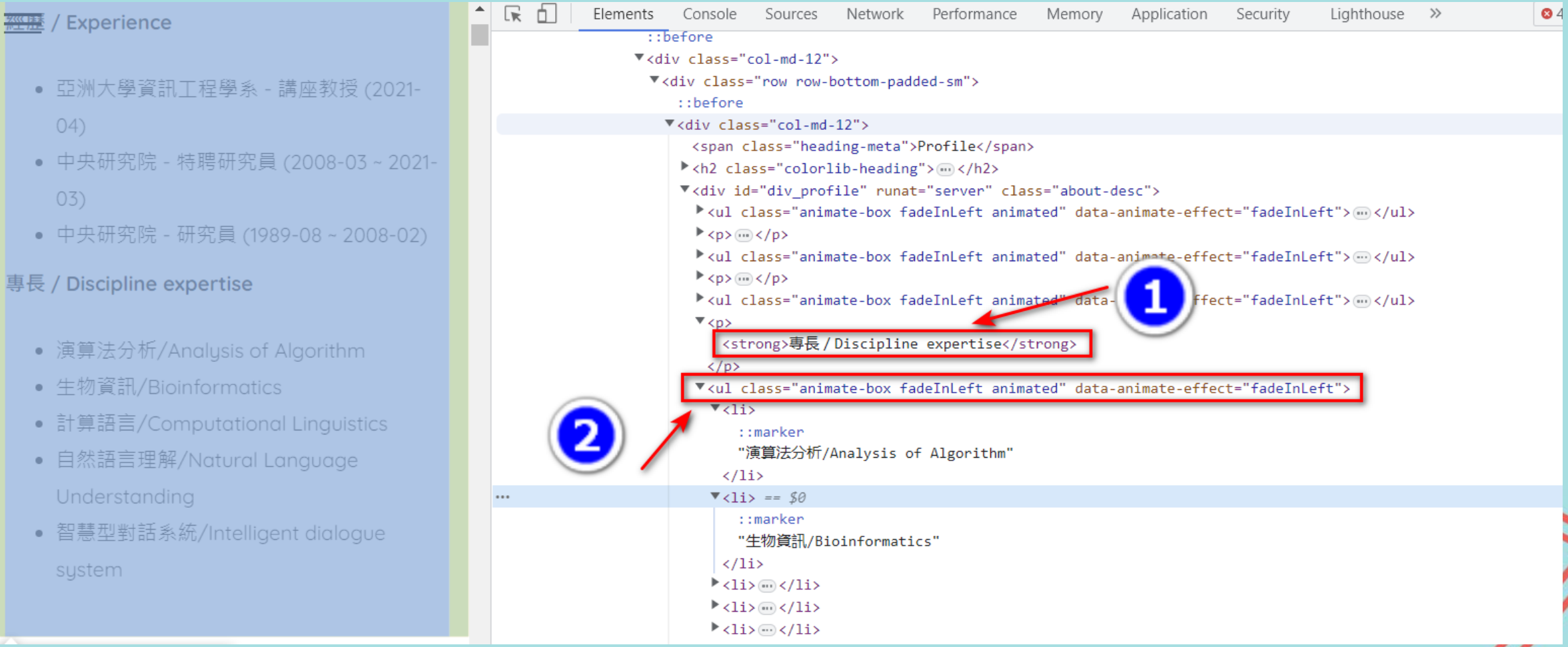
Crawle professors' disciplines from https://csie.asia.edu.tw/en/associate_professors_2

Step 1

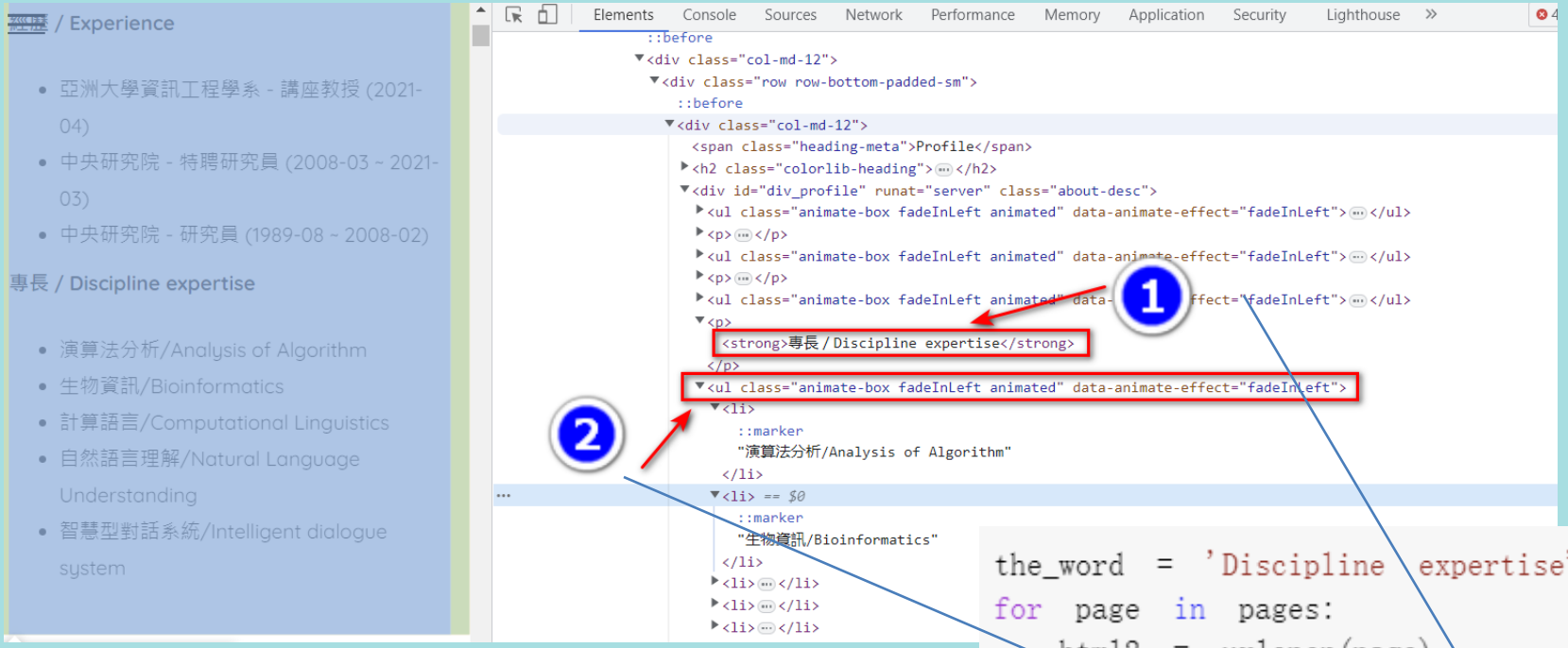
```
[ ] from urllib.request import urlopen
    from bs4 import BeautifulSoup
    import re
    pages = set()
    html = urlopen('https://csie.asia.edu.tw/en/associate_professors_2')
    bs = BeautifulSoup(html, 'html.parser')
    for link in bs.find_all('a', href=re.compile('^(http://research.asia.edu.tw/TchEportfolio/)')):
        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



Beautiful Soup Demo-3




```
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.jl
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

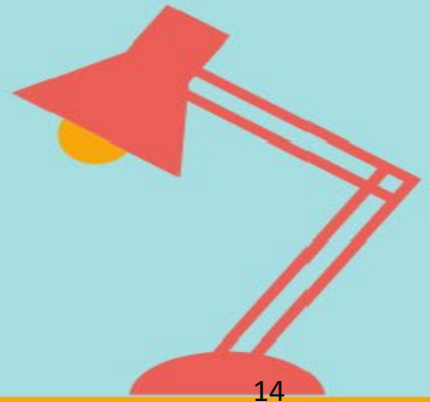
```python
from selenium import webdriver
from bs4 import BeautifulSoup
import re

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 obje
ct
  browser = webdriver.Chrome(options=op, executable_path = 'C:\ProgramData\chocolatey\lib\chromedriver\tools\chromedrive
r.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

ACP111En_Midterm.ipynb

檔案 編輯 檢視畫面 插入 執行階段 工具 說明 無法儲存變更

+ 程式碼 + 文字 複製到雲端硬碟

Demo-Web crawler

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)
```

Step 2

Product Solutions Open Source Pricing

htchu / ACP111CourseEn Public

Code Issues Pull requests Actions Projects Security

main 1 branch 0 tags

Notebook

LICENSE

README.md

README.md

ACP111CourseEn

Advanced Computer Programming@Asia Univ.

https://colab.research.google.com/github/htchu/ACP111CourseEn/blob/main/Notebook/ACP111En_Midterm.ipynb

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

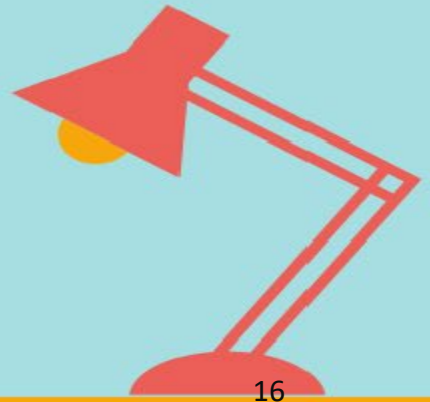
README.md

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)

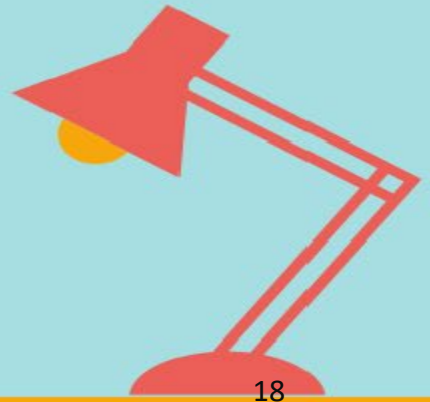


https://github.com/jackjyq/fullstack_tutorial


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.

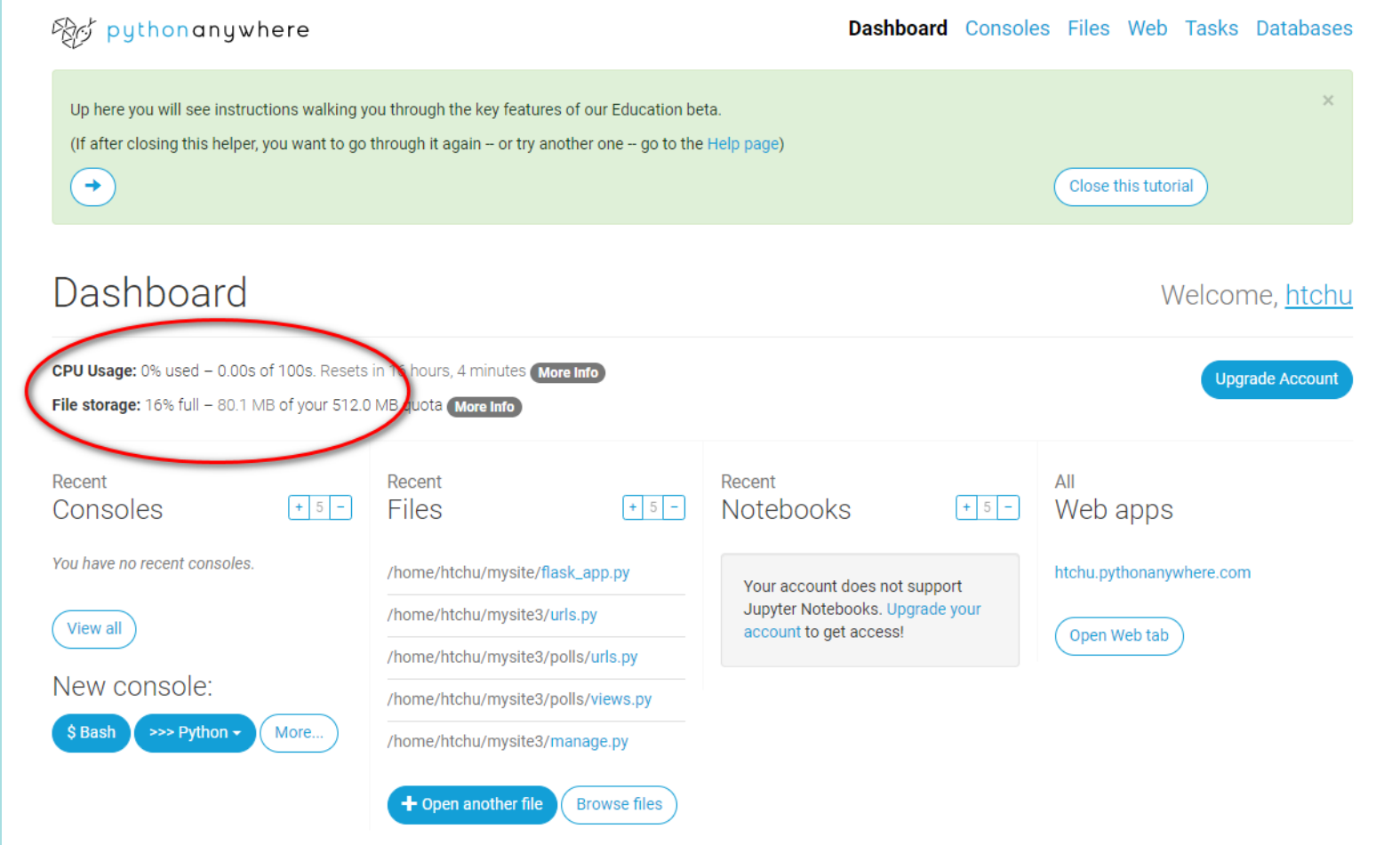
(Big) **Frameworks**

(Micro μ) **Frameworks**

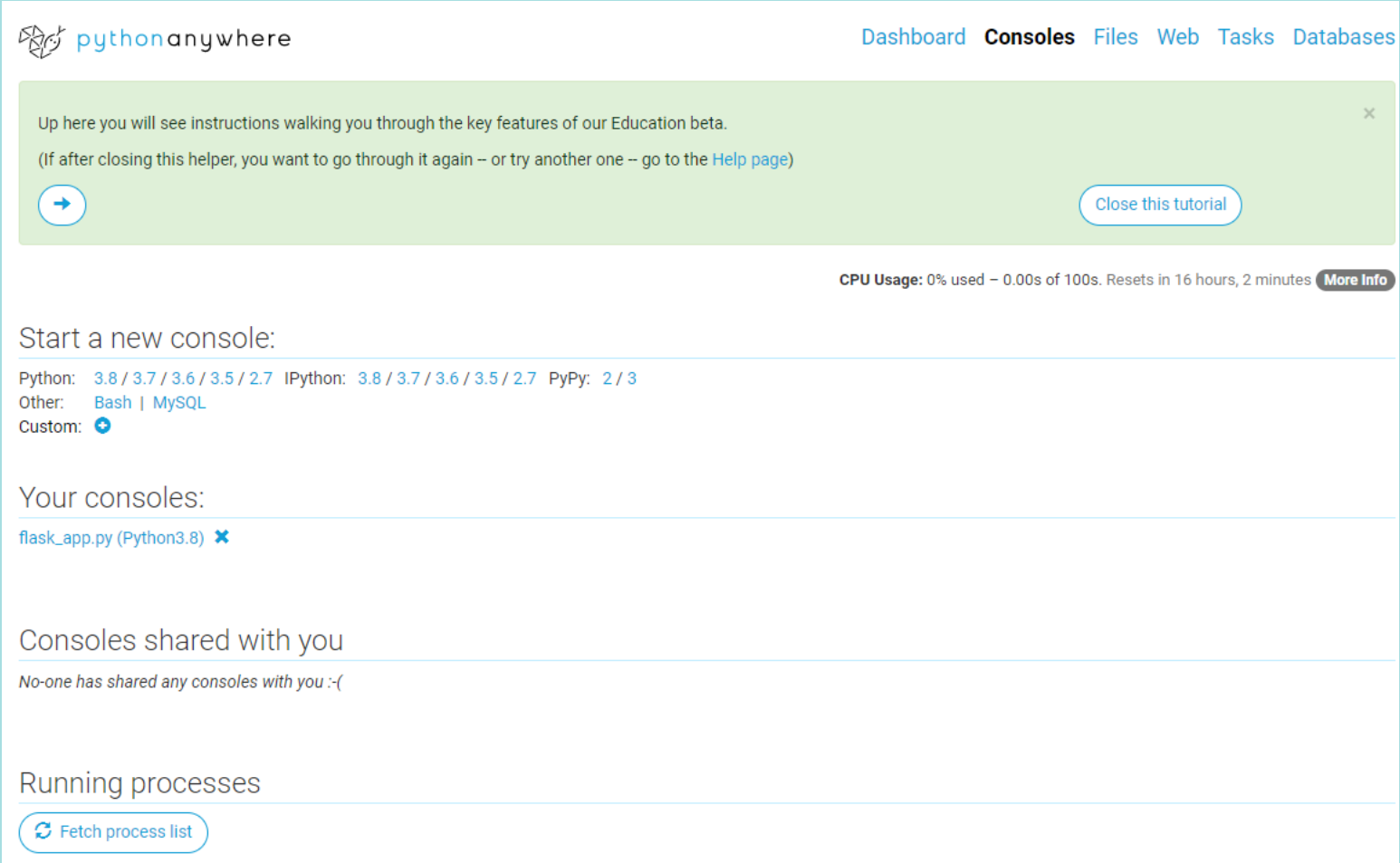
PythonAnywhere – Free online Python execution environment

- 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)

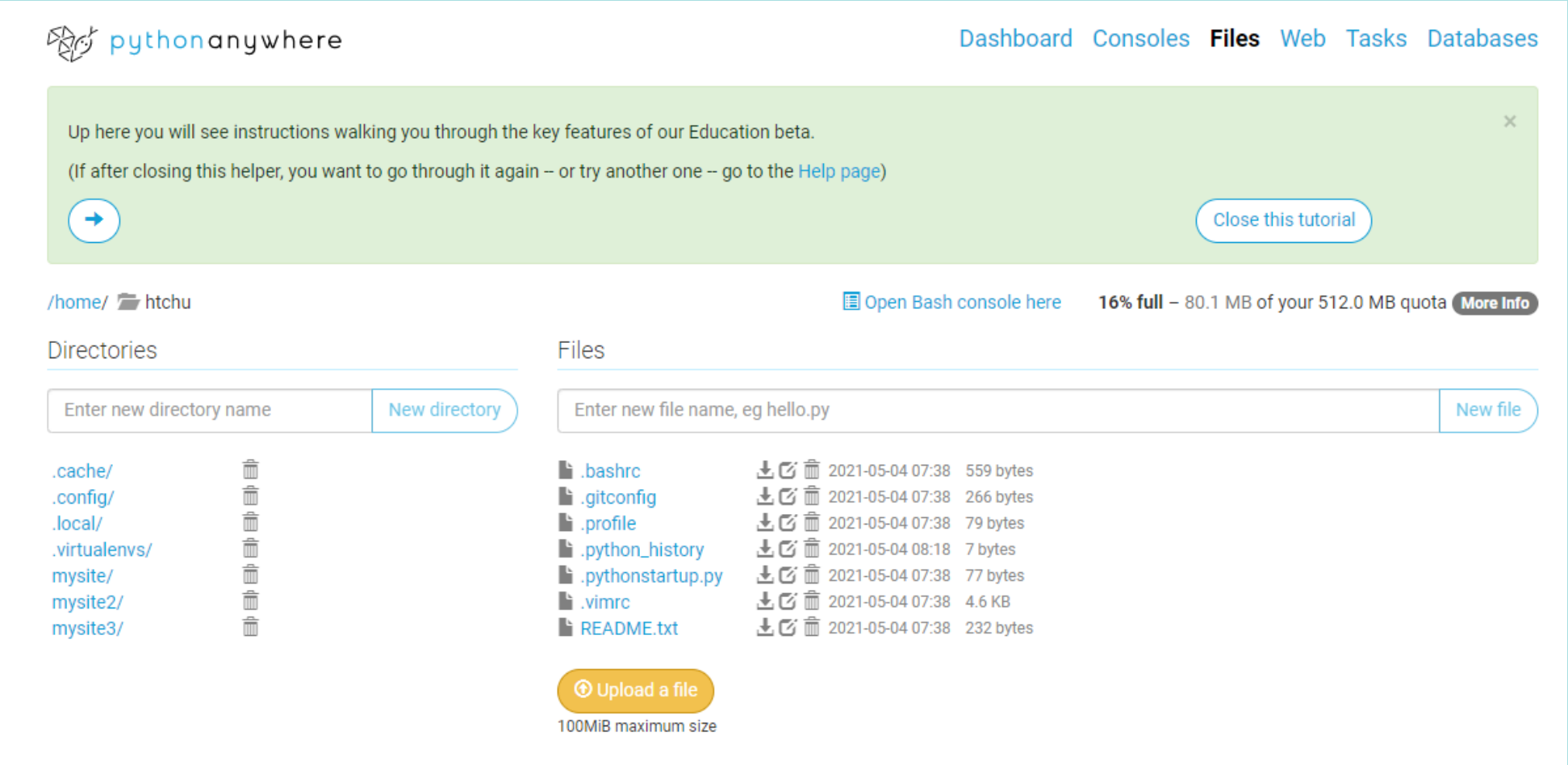
Dashboard儀表板



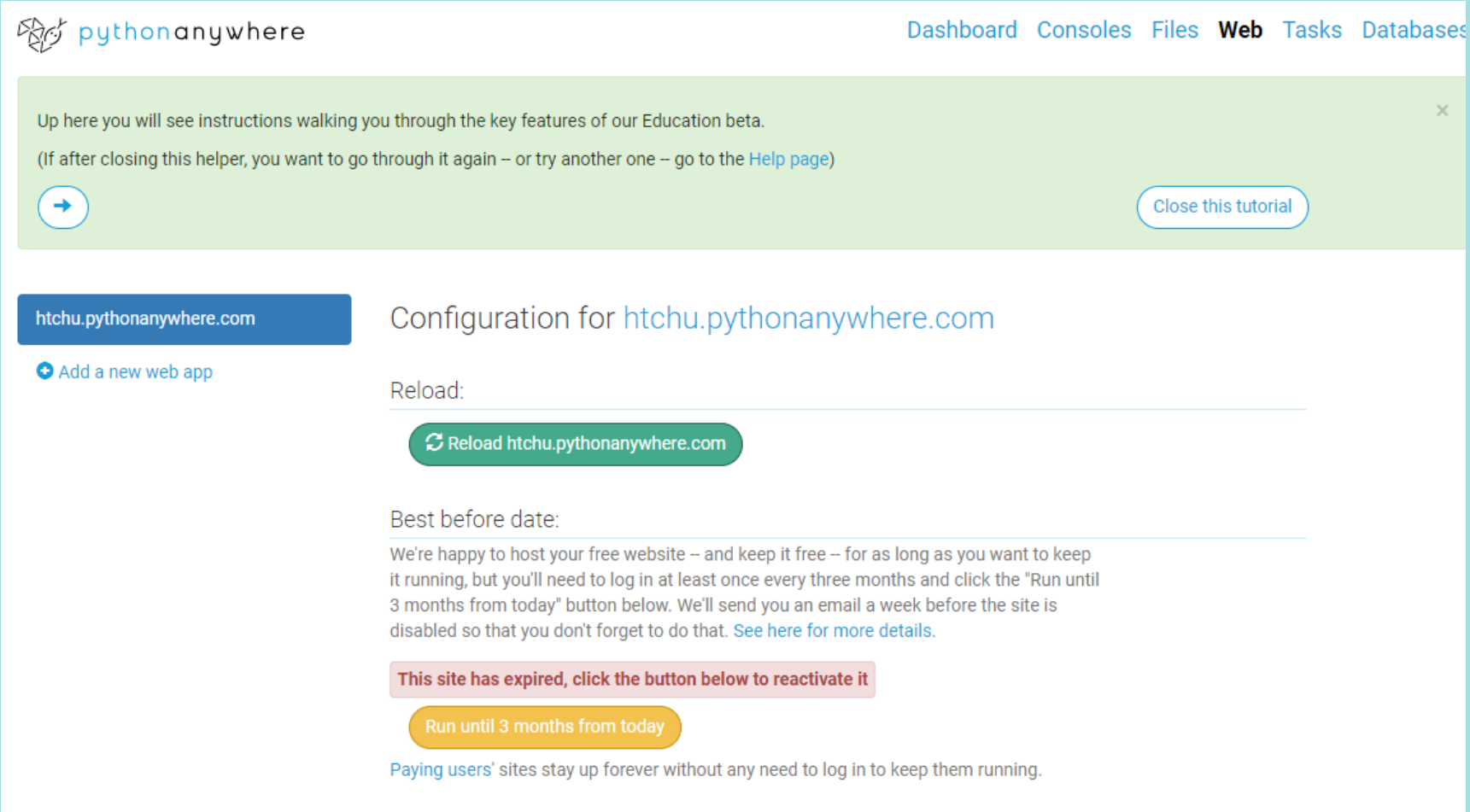
Command Console命令控制台



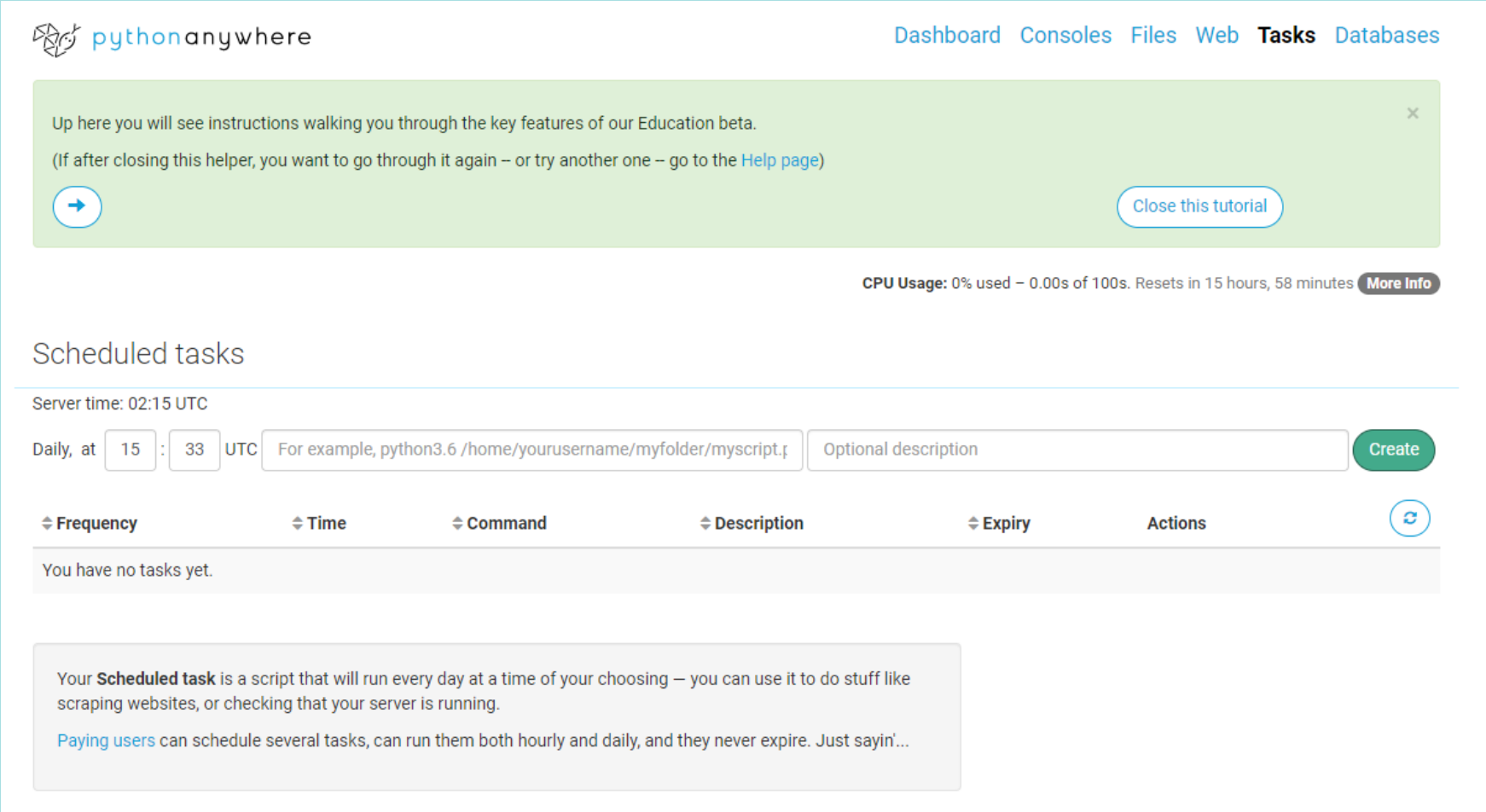
Files


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

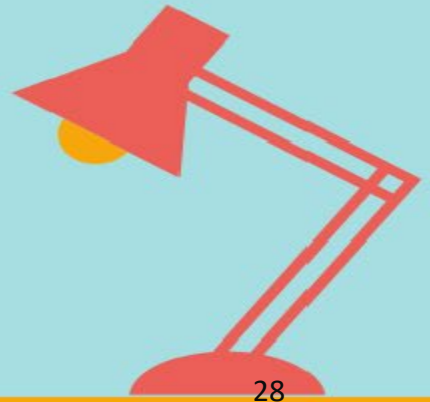

pythonanywhere
Dashboard
Consoles
Files
Web
Tasks
Databases
Up here you will see instructions walking you through the key features of our Education beta.
(If after closing this helper, you want to go through it again -- or try another one -- go to the Help page)
Close this tutorial
htchu.pythonanywhere.com
Add a new web app
Configuration for htchu.pythonanywhere.com
Reload:
Reload htchu.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 has expired, click the button below to reactivate it
Run until 3 months from today
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 15 : 33 UTC
For example, python3.6 /home/yourusername/myfolder/myscript.p
Optional description
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'...

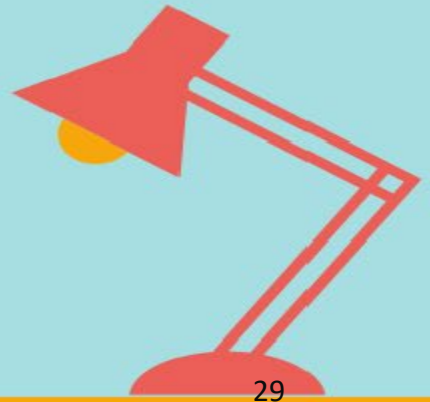
Web application frameworks

- 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

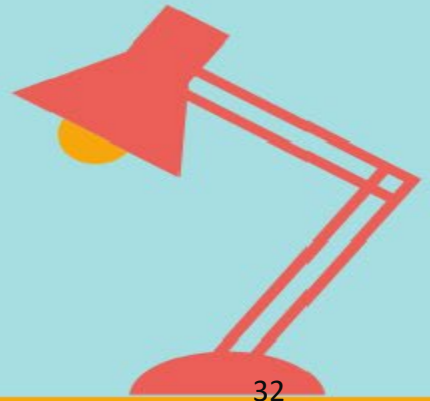
Flask

- Flask is a class with
  - run() function
  - route() functions

- Flask is a command
  - flask run
  - flask routes
  - flask shell

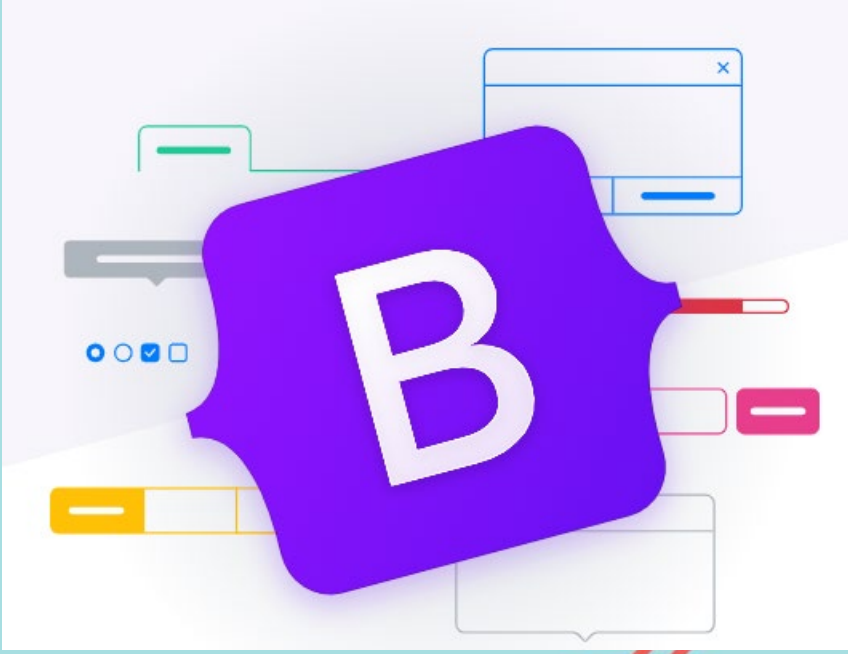
Flask environment

- 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**



https://getbootstrap.com/

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)
```


https://github.com/greyli/bootstrap-flask

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**.

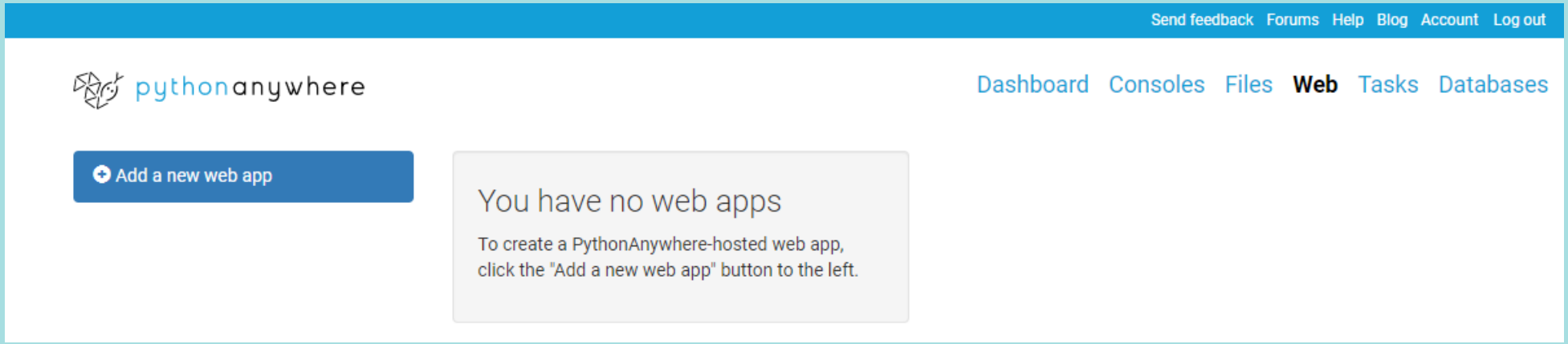
```
template = env.get_template("mytemplate.html")
```

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

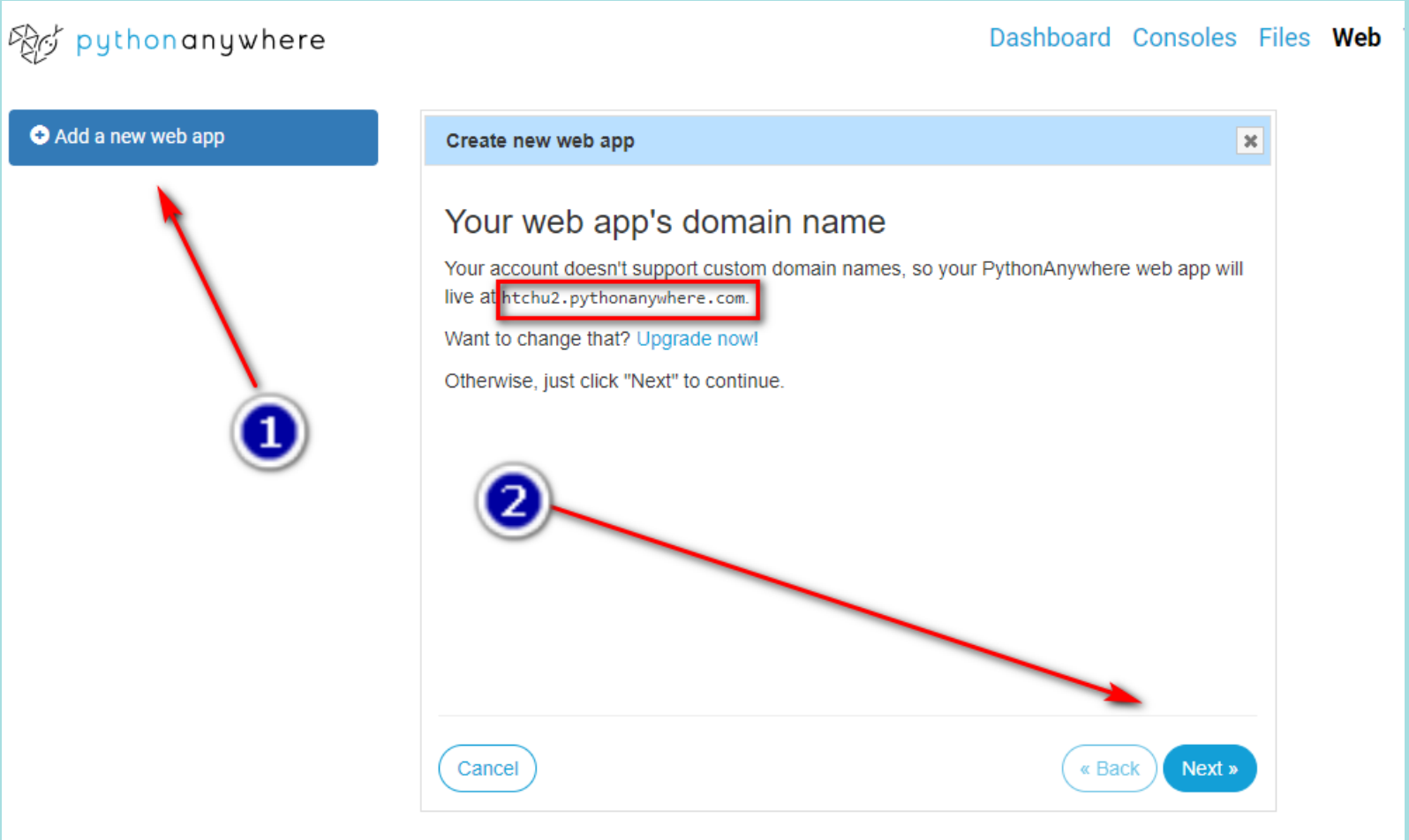
```
print(template.render(the="variables", go="here"))
```

https://jinja.palletsprojects.com/en/3.1.x/api/#basics

Step 1: Go to Web tab



Step 2: Add a new web app



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.

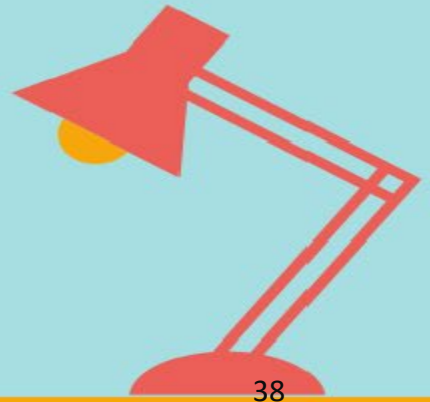
- » **Django**
- » **web2py**
- » **Flask**
- » **Bottle**
- » **Manual configuration** (including virtualenvs)

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

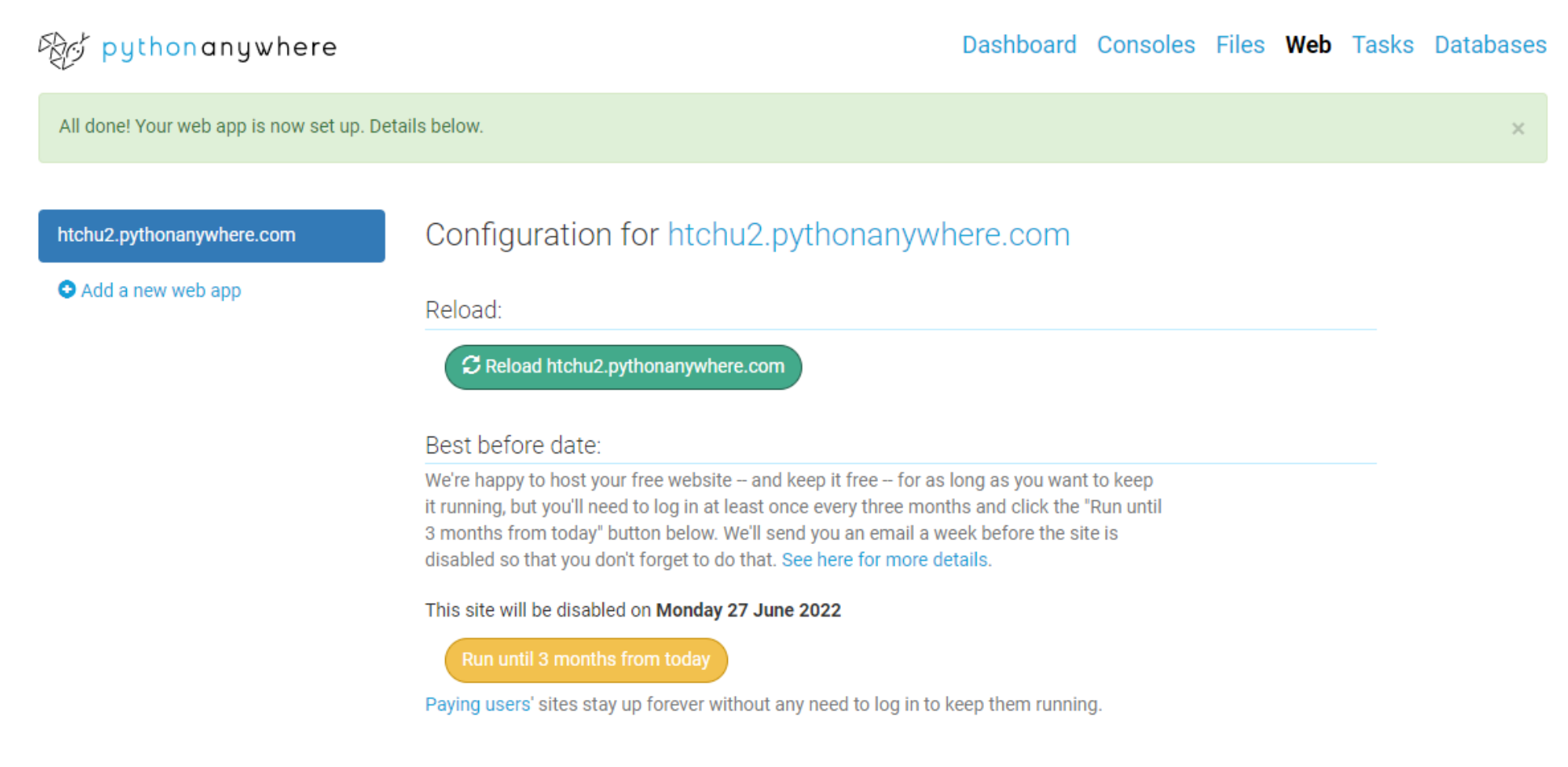
Select a Python version

- » **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)

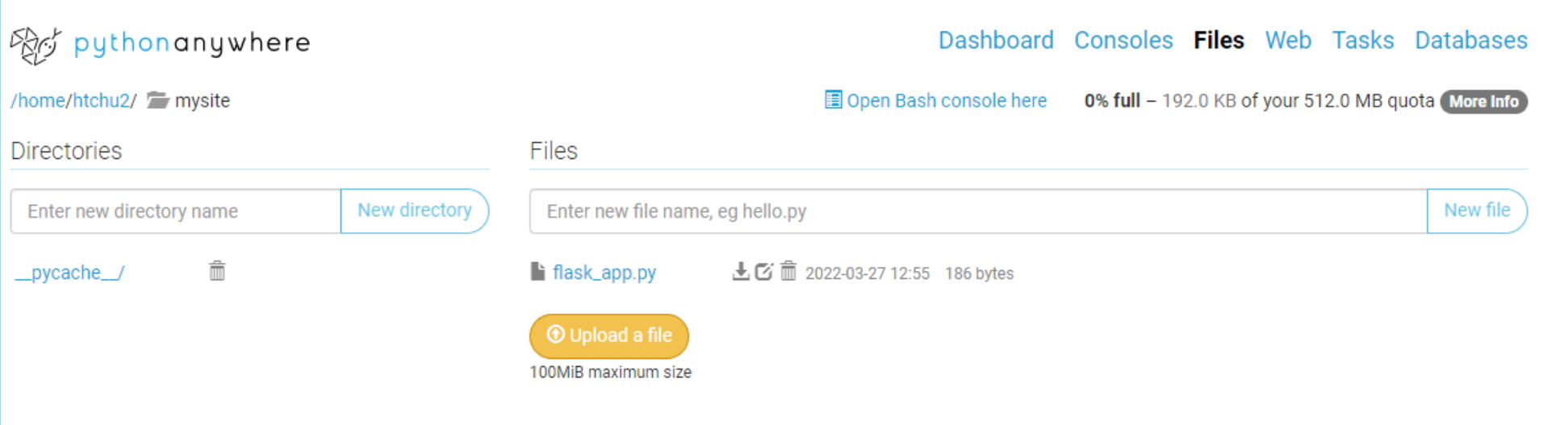
Note: If you'd like to use a different version of Flask to the default version, you can use a virtualenv for your web app. There are instructions here.



Step 4: Quick start new Flask project



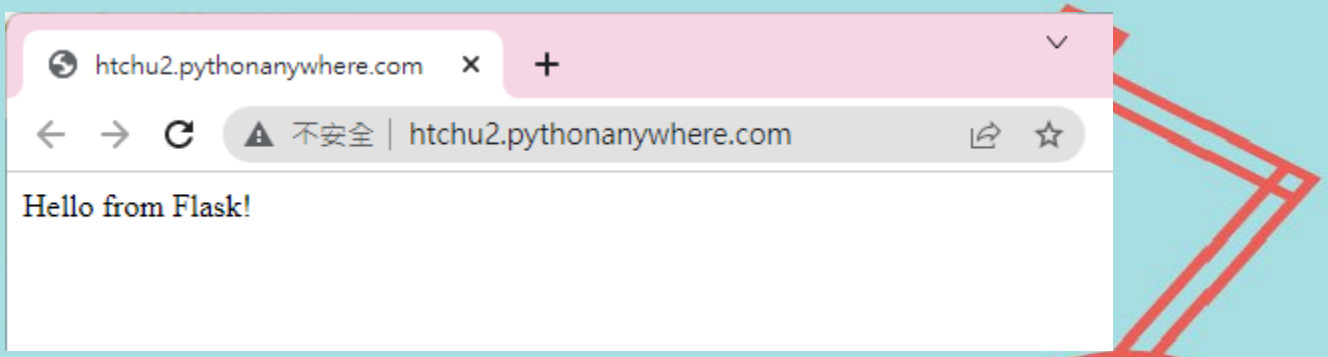
Step 5: check the files



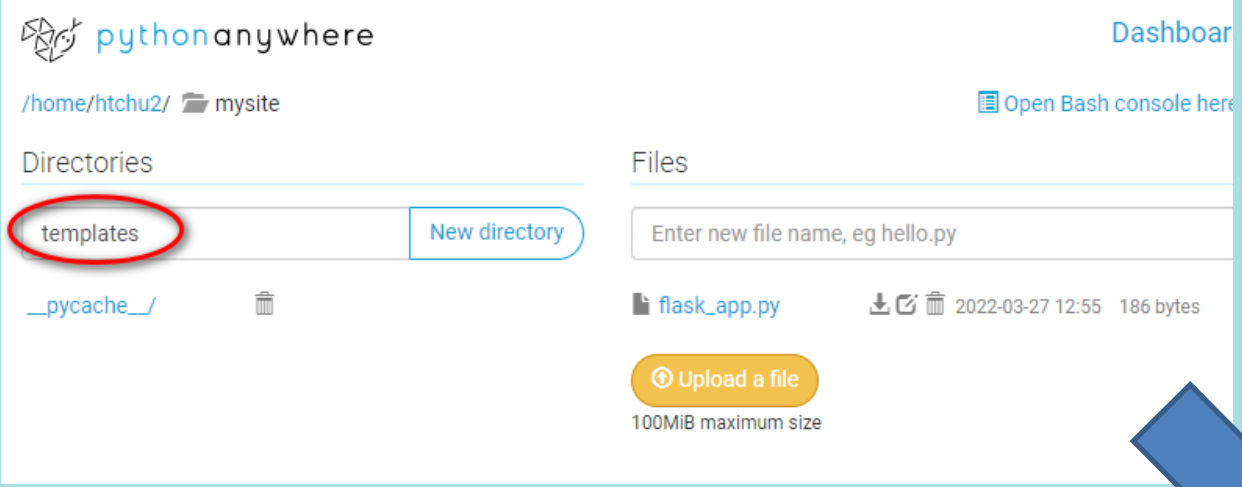
Step 6: check the program and the web app

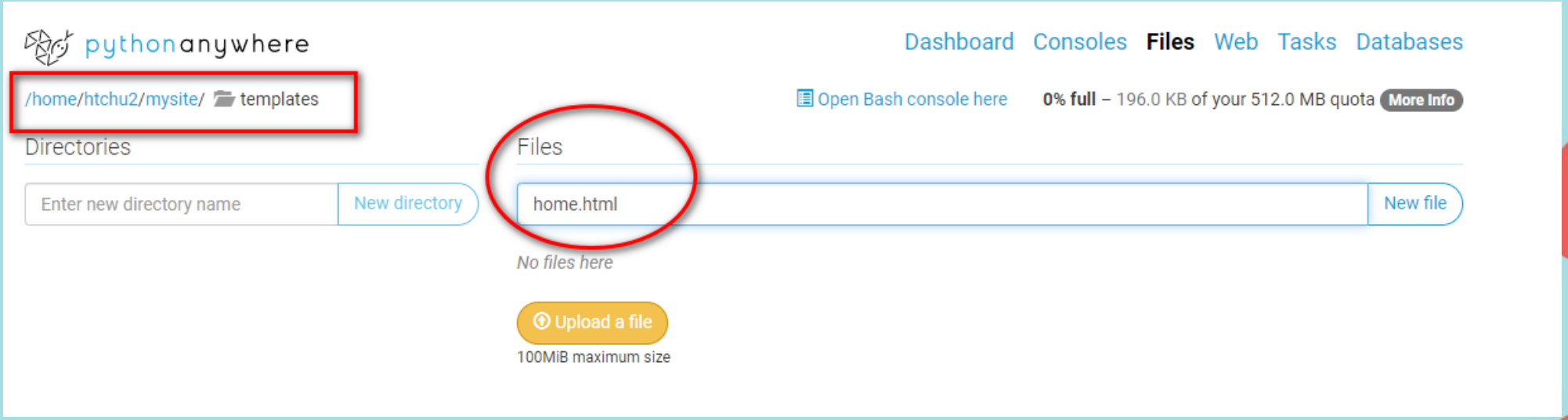
/home/htchu2/mysite/flask_app.py

```python

# A very simple Flask Hello World app for you to get started with...

from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello from Flask!'


```

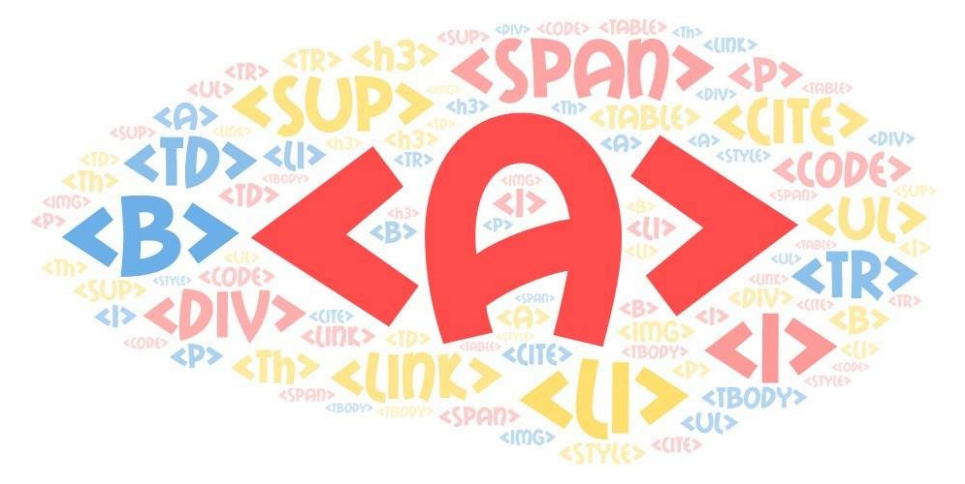


Step 7: add html templates





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.

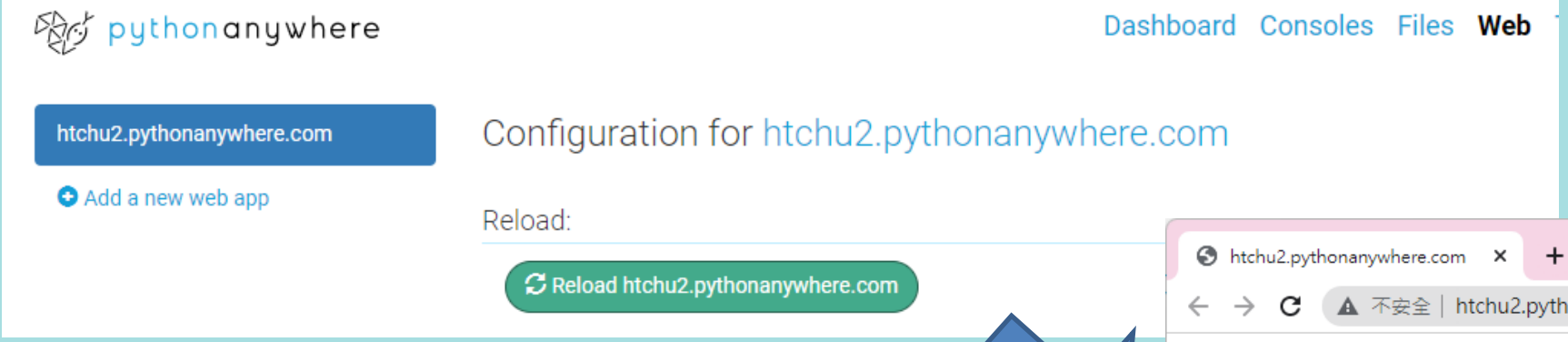
https://www.codebrainer.com/blog/top-10-html-tags

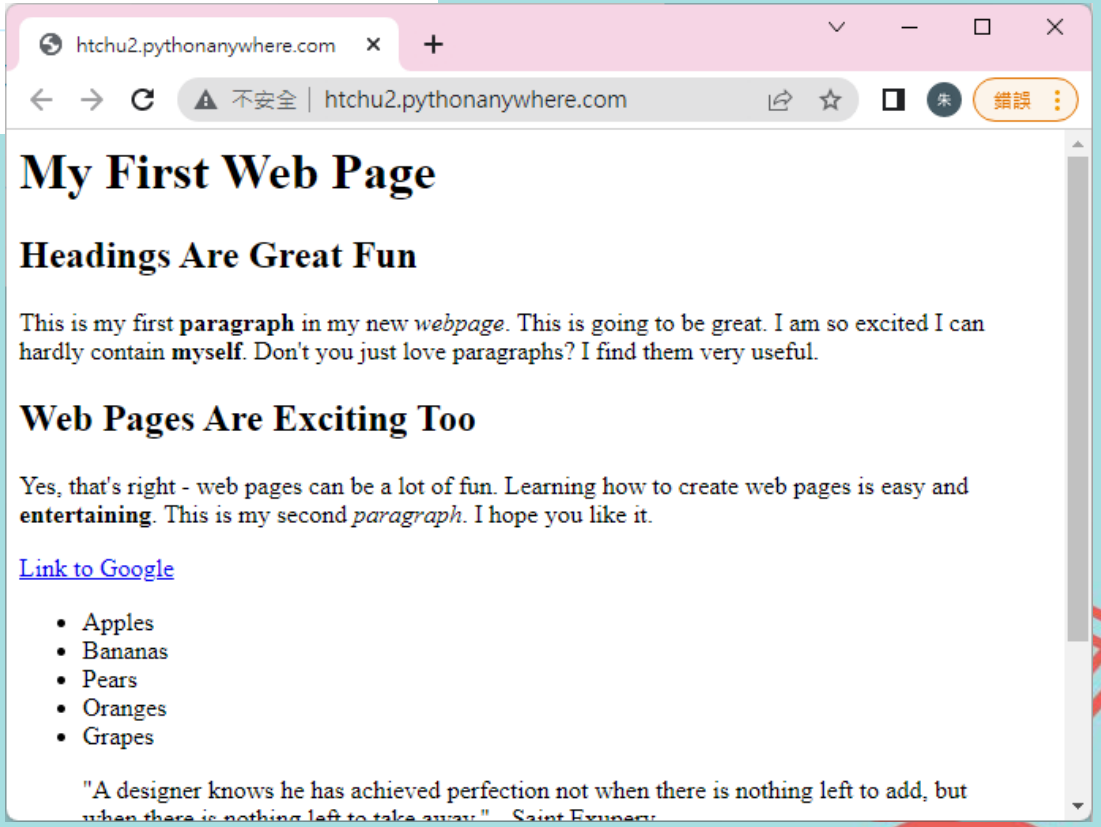
Step 9: edit flask_app.py

/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





Thanks! Q&A