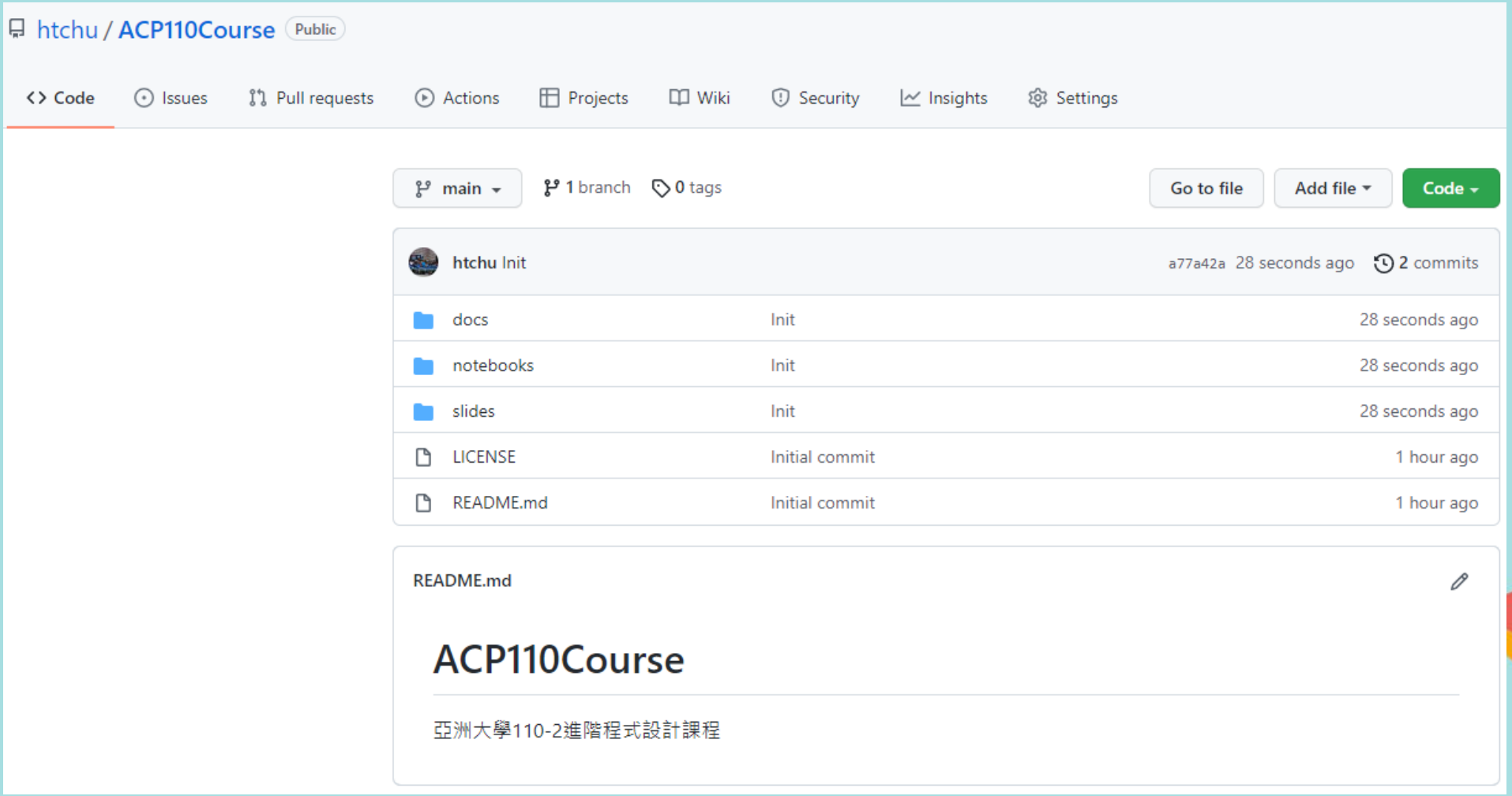
111-2進階程式設計課程(4)
Advanced Computer Programming

亞大資工系

課程大綱

- W1-課程介紹/Introduction
- W2-Python libraries
- W3-BeautifulSoup(1)
- W4-BeautifulSoup(2)
- W5-Scrapy(1)
- W6-Scrapy(2)
- W7-Storing Data
- W8-Project development(1)
- W9-Midterm presentation
- 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

教材Github

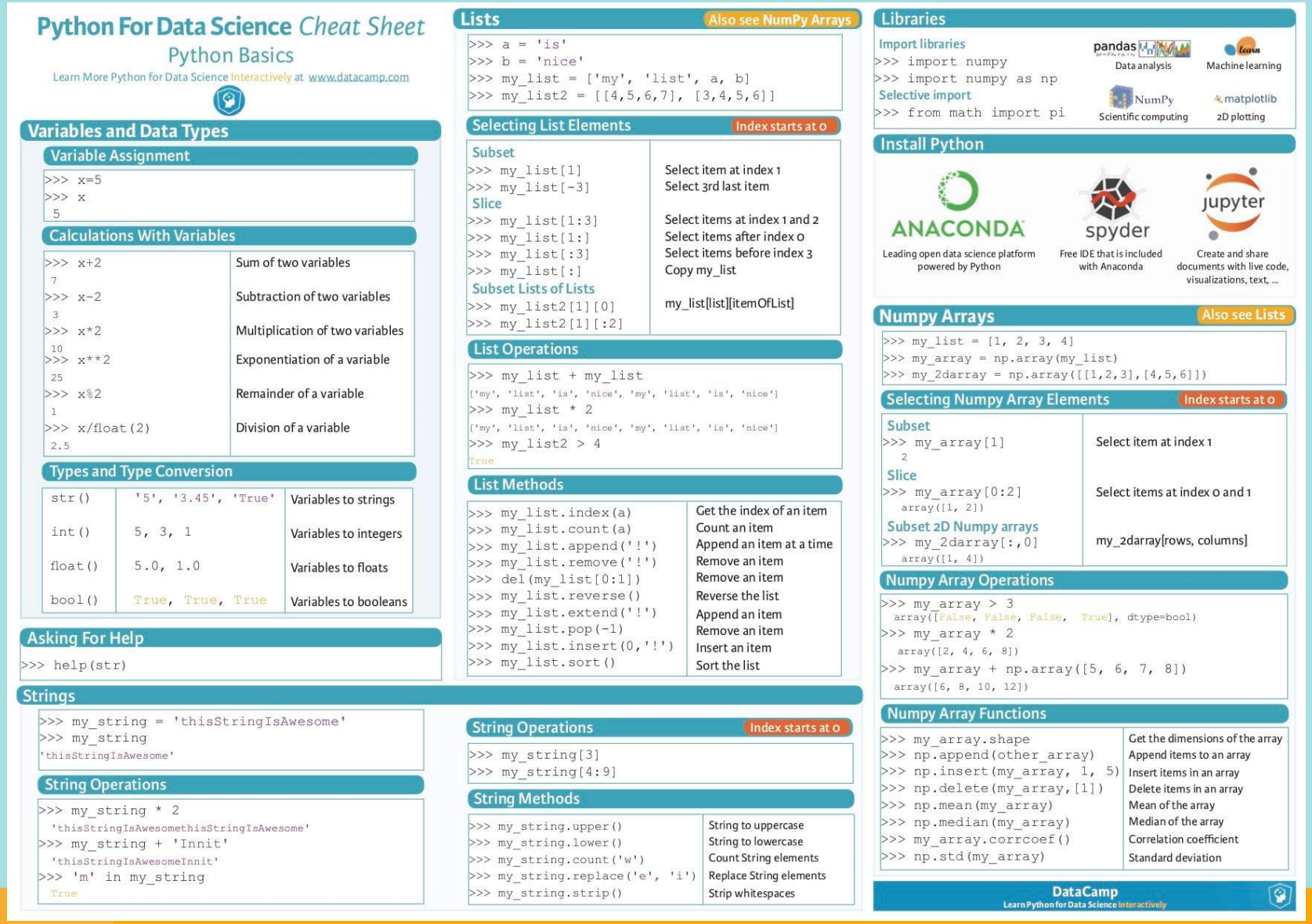


https://github.com/htchu/ACP110Course

大綱

- Python Review
- Building Scrapers
- robots.txt
  - https://developers.google.com/search/docs/advanced/robots/robots_txt
- 作業 1(Assignment 1):
  - 抓取 https://www.cna.com.tw/ 的首頁
  - 查找和解釋 https://www.cna.com.tw/robots.txt

Python For Data Science Cheat Sheet



Python For Data Science *Cheat Sheet*

Python Basics

Learn More Python for Data Science Interactively at www.datacamp.com

Variables and Data Types

Variable Assignment

```
>>> x=5
>>> x
 5
```

Calculations With Variables

```
>>> x+2
 7
>>> x-2
 3
>>> x*2
 10
>>> x**2
 25
>>> x%2
 1
>>> x/float(2)
 2.5
```

Sum of two variables

Subtraction of two variables

Multiplication of two variables

Exponentiation of a variable

Remainder of a variable

Division of a variable

Types and Type Conversion

`str()`	`'5', '3.45', 'True'`	Variables to strings
`int()`	`5, 3, 1`	Variables to integers
`float()`	`5.0, 1.0`	Variables to floats
`bool()`	`True, True, True`	Variables to booleans

Asking For Help

```
>>> help(str)
```

Lists

Also see NumPy Arrays

```
>>> a = 'is'
>>> b = 'nice'
>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Selecting List Elements

Index starts at 0

Subset

```
>>> my_list[1]
>>> my_list[-3]
```

Select item at index 1

Select 3rd last item

Slice

```
>>> my_list[1:3]
>>> my_list[1:]
>>> my_list[:3]
>>> my_list[:]
```

Select items at index 1 and 2

Select items after index 0

Select items before index 3

Copy my_list

Subset Lists of Lists

```
>>> my_list2[1][0]
>>> my_list2[1][:2]
```

my_list[list][itemOfList]

List Operations

```
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list2 > 4
True
```

List Methods

`>>> my_list.index(a)`	Get the index of an item
`>>> my_list.count(a)`	Count an item
`>>> my_list.append('!')`	Append an item at a time
`>>> my_list.remove('!')`	Remove an item
`>>> del(my_list[0:1])`	Remove an item
`>>> my_list.reverse()`	Reverse the list
`>>> my_list.extend('!')`	Append an item
`>>> my_list.pop(-1)`	Remove an item
`>>> my_list.insert(0,'!')`	Insert an item
`>>> my_list.sort()`	Sort the list

Libraries

Import libraries

```
>>> import numpy
>>> import numpy as np
```

Selective import

```
>>> from math import pi
```

pandas Data analysis

learn Machine learning

NumPy Scientific computing

matplotlib 2D plotting

Install Python

ANACONDA — Leading open data science platform powered by Python

spyder — Free IDE that is included with Anaconda

jupyter — Create and share documents with live code, visualizations, text, ...

Numpy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]
>>> my_array = np.array(my_list)
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

Selecting Numpy Array Elements

Index starts at 0

Subset

```
>>> my_array[1]
  2
```

Select item at index 1

Slice

```
>>> my_array[0:2]
  array([1, 2])
```

Select items at index 0 and 1

Subset 2D Numpy arrays

```
>>> my_2darray[:,0]
  array([1, 4])
```

my_2darray[rows, columns]

Numpy Array Operations

```
>>> my_array > 3
  array([False, False, False,  True], dtype=bool)
>>> my_array * 2
  array([2, 4, 6, 8])
>>> my_array + np.array([5, 6, 7, 8])
  array([6, 8, 10, 12])
```

Numpy Array Functions

`>>> my_array.shape`	Get the dimensions of the array
`>>> np.append(other_array)`	Append items to an array
`>>> np.insert(my_array, 1, 5)`	Insert items in an array
`>>> np.delete(my_array,[1])`	Delete items in an array
`>>> np.mean(my_array)`	Mean of the array
`>>> np.median(my_array)`	Median of the array
`>>> my_array.corrcoef()`	Correlation coefficient
`>>> np.std(my_array)`	Standard deviation

DataCamp — Learn Python for Data Science Interactively

Strings

```
>>> my_string = 'thisStringIsAwesome'
>>> my_string
'thisStringIsAwesome'
```

String Operations

```
>>> my_string * 2
  'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
  'thisStringIsAwesomeInnit'
>>> 'm' in my_string
  True
```

String Operations

Index starts at 0

```
>>> my_string[3]
>>> my_string[4:9]
```

String Methods

`>>> my_string.upper()`	String to uppercase
`>>> my_string.lower()`	String to lowercase
`>>> my_string.count('w')`	Count String elements
`>>> my_string.replace('e', 'i')`	Replace String elements
`>>> my_string.strip()`	Strip whitespaces

Beginner's Python Cheat Sheet

Variables and Strings

Variables are used to store values. A string is a series of characters, surrounded by single or double quotes.

Hello world

```
print("Hello world!")
```

Hello world with a variable

```
msg = "Hello world!"
print(msg)
```

Concatenation (combining strings)

```
first_name = 'albert'
last_name = 'einstein'
full_name = first_name + ' ' + last_name
print(full_name)
```

Lists

A list stores a series of items in a particular order. You access items using an index, or within a loop.

Make a list

```
bikes = ['trek', 'redline', 'giant']
```

Get the first item in a list

```
first_bike = bikes[0]
```

Get the last item in a list

```
last_bike = bikes[-1]
```

Looping through a list

```
for bike in bikes:
    print(bike)
```

Adding items to a list

```
bikes = []
bikes.append('trek')
bikes.append('redline')
bikes.append('giant')
```

Making numerical lists

```
squares = []
for x in range(1, 11):
    squares.append(x**2)
```

Lists (cont.)

List comprehensions

```
squares = [x**2 for x in range(1, 11)]
```

Slicing a list

```
finishers = ['sam', 'bob', 'ada', 'bea']
first_two = finishers[:2]
```

Copying a list

```
copy_of_bikes = bikes[:]
```

Tuples

Tuples are similar to lists, but the items in a tuple can't be modified.

Making a tuple

```
dimensions = (1920, 1080)
```

If statements

If statements are used to test for particular conditions and respond appropriately.

Conditional tests

```
equals            x == 42
not equal         x != 42
greater than      x > 42
  or equal to     x >= 42
less than         x < 42
  or equal to     x <= 42
```

Conditional test with lists

```
'trek' in bikes
'surly' not in bikes
```

Assigning boolean values

```
game_active = True
can_edit = False
```

A simple if test

```
if age >= 18:
    print("You can vote!")
```

If-elif-else statements

```
if age < 4:
    ticket_price = 0
elif age < 18:
    ticket_price = 10
else:
    ticket_price = 15
```

Dictionaries

Dictionaries store connections between pieces of information. Each item in a dictionary is a key-value pair.

A simple dictionary

```
alien = {'color': 'green', 'points': 5}
```

Accessing a value

```
print("The alien's color is " + alien['color'])
```

Adding a new key-value pair

```
alien['x_position'] = 0
```

Looping through all key-value pairs

```
fav_numbers = {'eric': 17, 'ever': 4}
for name, number in fav_numbers.items():
    print(name + ' loves ' + str(number))
```

Looping through all keys

```
fav_numbers = {'eric': 17, 'ever': 4}
for name in fav_numbers.keys():
    print(name + ' loves a number')
```

Looping through all the values

```
fav_numbers = {'eric': 17, 'ever': 4}
for number in fav_numbers.values():
    print(str(number) + ' is a favorite')
```

User input

Your programs can prompt the user for input. All input is stored as a string.

Prompting for a value

```
name = input("What's your name? ")
print("Hello, " + name + "!")
```

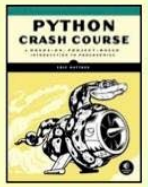
Prompting for numerical input

```
age = input("How old are you? ")
age = int(age)

pi = input("What's the value of pi? ")
pi = float(pi)
```


Python Crash Course

Covers Python 3 and Python 2

nostarchpress.com/pythoncrashcourse



Building Scrapers

Adobe Digital Editions - Web Scraping with Python

檔案(F) 編輯(E) 閱讀(R) 說明(H)

◀ 圖書館

目錄

- Preface
- I. Building Scrapers
- 1. Your First Web Scraper
- 2. Advanced HTML Parsing
- 3. Writing Web Crawlers
- 4. Web Crawling Models
- 5. Scrapy
- 6. Storing Data
- II. Advanced Scraping
- 7. Reading Documents
- 8. Cleaning Your Dirty Data
- 9. Reading and Writing Natural Languages
- 10. Crawling Through Forms and Logins
- 11. Scraping JavaScript
- 12. Crawling Through APIs
- 13. Image Processing and Text Recognition
- 14. Avoiding Scraping Traps
- 15. Testing Your Website with Scrapers
- 16. Web Crawling in Parallel
- 17. Scraping Remotely
- 18. The Legalities and Ethics of Web Scraping
- Index

Chapter 1. Your First Web Scraper

Once you start web scraping, you start to appreciate all the little things that browsers do for you. The web, without a layer of HTML formatting, CSS styling, JavaScript execution, and image rendering, can lo first, but in this chapter, as well as the next one, we'll cover how to format and interpret data without the help of a browser.

This chapter starts with the basics of sending a GET request (a request to fetch, or "get," the content of a web page) to a web server for a specific page, reading the HTML output from that page, and doing som in order to isolate the content that you are looking for.

Connecting

If you haven't spent much time in networking or network security, the mechanics of the internet might seem a little mysterious. You don't want to think about what, exactly, the network is doing every time yo go to *http://google.com*, and, these days, you don't have to. In fact, I would argue that it's fantastic that computer interfaces have advanced to the point where most people who use the internet don't have the f works.

However, web scraping requires stripping away some of this shroud of interface—not just at the browser level (how it interprets all of this HTML, CSS, and JavaScript), but occasionally at the level of the net

To give you an idea of the infrastructure required to get information to your browser, let's use the following example. Alice owns a web server. Bob uses a desktop computer, which is trying to connect to Alic machine wants to talk to another machine, something like the following exchange takes place:

1. Bob's computer sends along a stream of 1 and 0 bits, indicated by high and low voltages on a wire. These bits form some information, containing a header and body. The header contains an immediat router's MAC address, with a final destination of Alice's IP address. The body contains his request for Alice's server application.
2. Bob's local router receives all these 1s and 0s and interprets them as a packet, from Bob's own MAC address, destined for Alice's IP address. His router stamps its own IP address on the packet as the sends it off across the internet.
3. Bob's packet traverses several intermediary servers, which direct his packet toward the correct physical/wired path, on to Alice's server.
4. Alice's server receives the packet at her IP address.
5. Alice's server reads the packet port destination in the header, and passes it off to the appropriate application—the web server application. (The packet port destination is almost always port 80 for web thought of as an apartment number for packet data, whereas the IP address is like the street address.)
6. The web server application receives a stream of data from the server processor. This data says something like the following:

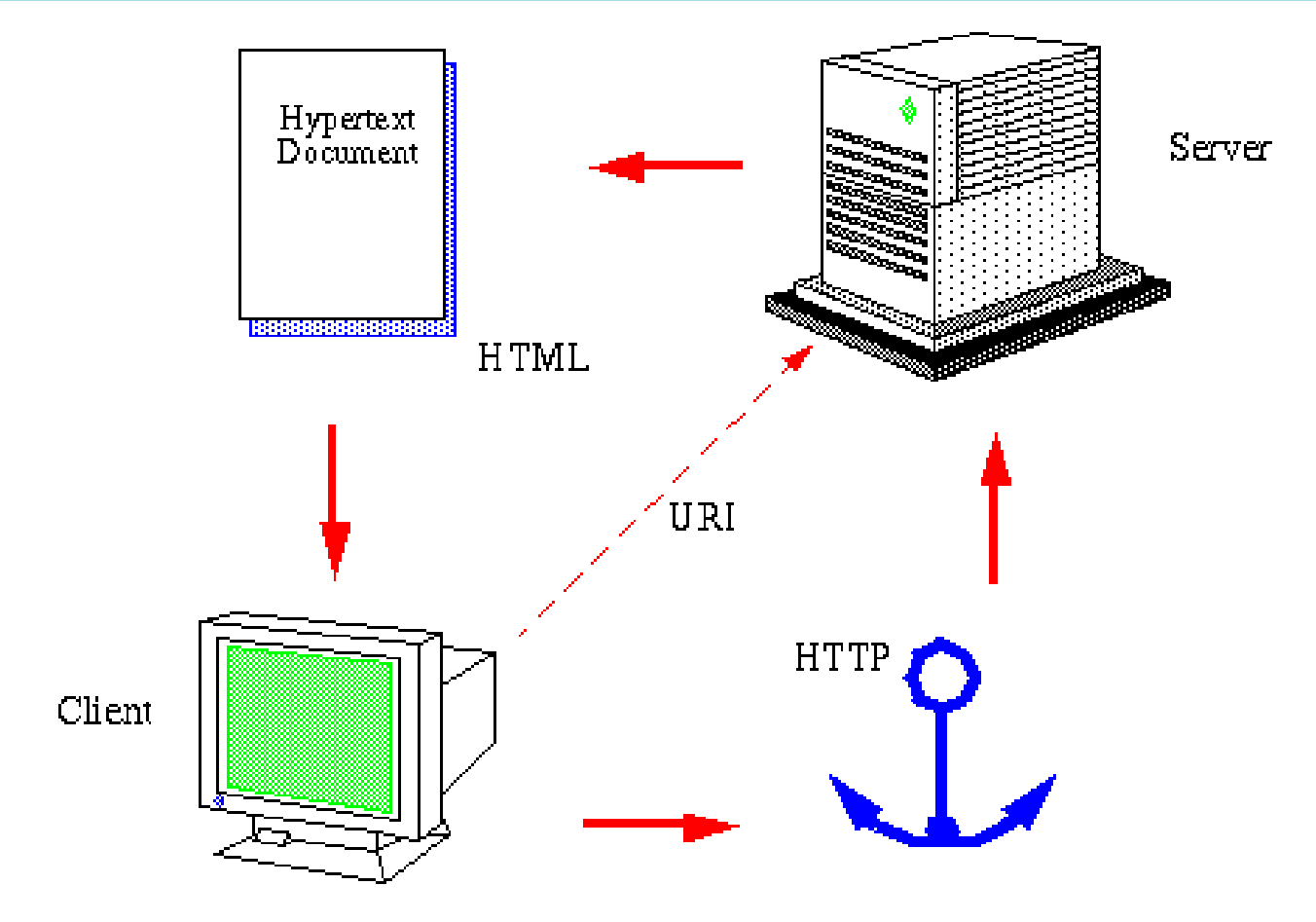
   - This is a GET request.

   - The following file is requested: *index.html*.
7. The web server locates the correct HTML file, bundles it up into a new packet to send to Bob, and sends it through to its local router, for transport back to Bob's machine, through the same process.

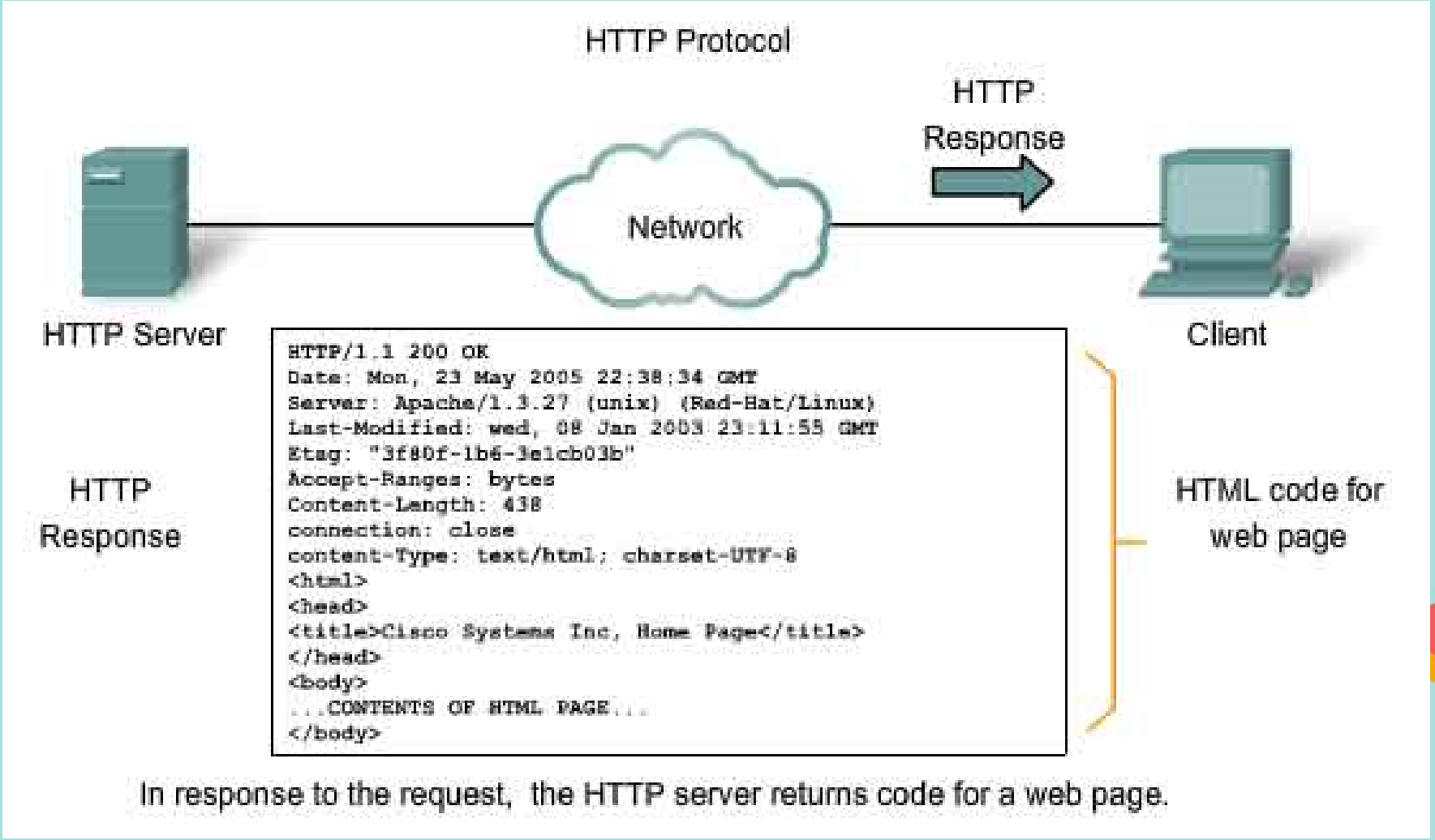
HTTP, URI, HTML

```
<!DOCTYPE html>
<html>
<!-- created 2010-01-01 -->
 <head>
  <title>sample</title>
 </head>
 <body>
  <p>Voluptatem accusantium
   totam rem aperiam.</p>
 </body>
</html>
```

HTML



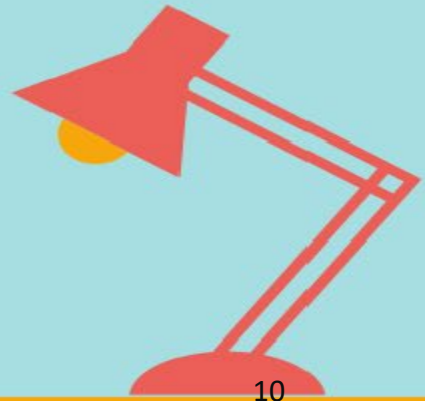
HTTP


HTTP Protocol
HTTP
Response
Network
HTTP Server
Client
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.27 (unix) (Red-Hat/Linux)
Last-Modified: wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
connection: close
content-Type: text/html; charset-UTF-8
<html>
<head>
<title>Cisco Systems Inc, Home Page</title>
</head>
<body>
...CONTENTS OF HTML PAGE...
</body>
HTTP
Response
HTML code for
web page
In response to the request, the HTTP server returns code for a web page.

robots.txt

- A robots.txt file is a simple text file containing rules about which crawlers may access which parts of a site. For example, the robots.txt file for example.com may look like this:

```
# This robots.txt file controls crawling of URLs under https://example.com.
# All crawlers are disallowed to crawl files in the "includes" directory, such
# as .css, .js, but Googlebot needs them for rendering, so Googlebot is allowed
# to crawl them.
User-agent: *
Disallow: /includes/

User-agent: Googlebot
Allow: /includes/

Sitemap: https://example.com/sitemap.xml
```



https://developers.google.com/search/docs/advanced/robots/robots_txt

Regular Expressions in Python

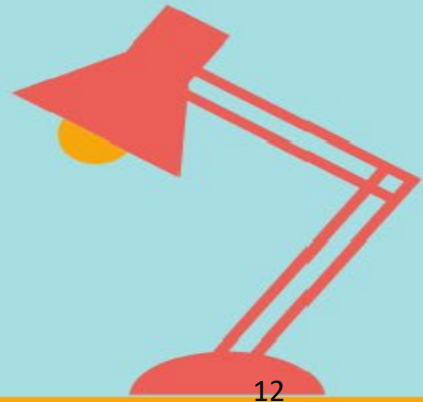
正規表示法 (Regex) 是用來處理字串的方法，Regex 用自己一套特殊的符號表示法，讓我們可以很方便的搜尋字串、取代字串、刪除字串或測試字串是否符合樣式規則。
Regex 它不是一個程式語言，他只是一種「字串樣式規則」的「表示法」，用來表達字元符號在字串中出現的規則，大部分的程式語言都有支援 Regex 的用法，而任何工具只要支援這種表示法，你就可以在這工具上用 Regex 來處理字串。

Regex 表示式	說明
\d{4}-\d{2}-\d{2}	從文本裡找出 YYYY-MM-DD 格式的日期字串
cat|dog	從文本裡找出 cat 或 dog 字串
[A-Z]\w+	從文本裡找出所有字首是大寫的英文字
^[A-Za-z]\d{9}$	驗證字串是否是台灣身份證字號

https://www.dataquest.io/blog/regex-cheatsheet/

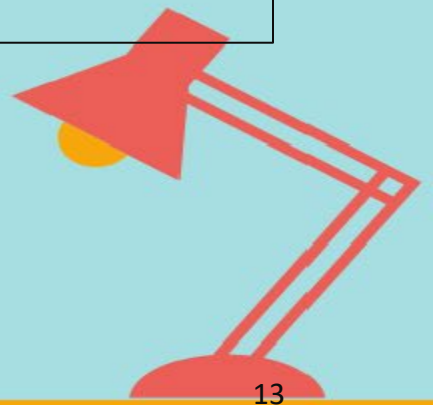
Regex 正規表示法
基本語法

- 或 (or)：| 管線符號，用來將所有可能的選擇條件分隔開。
- 群組 (grouping)：() 小括號，用來表示作用範圍或優先順序。。
- 量詞 (quantifier)：quantifier 用來接在字符串或群組後面，表示某個條件應該出現「幾次」。常見的量詞有：
  - ?：表示連續出現 0 次或 1 次。例如 colou?r 可以用來匹配 color 或 colour。
  - *：表示連續出現 0 次或多次。例如 ab*c 可以用來匹配 ac, abc, abbc, abbbc 或 abbbbbbc。
  - +：表示連續出現 1 次或多次。例如 ab+c 可以用來匹配 abc, abbc, abbbc, abbbbbbc，但 ac 不符合。
  - {min,max}：表示至少連續出現 min 次，但最多連續出現 max 次。
- **字元**
  - \d 用來匹配所有阿拉伯數字 0-9
  - \s 用來匹配所有的空白字元
  - \w所有非空白字元,意思同等於 [A-Za-z0-9_]
  - .(dot or period) 用來匹配除了換行符號 (line breaks) \n \r 之外的任何一個字元



Regular Expressions and BeautifulSoup

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re

html = urlopen('http://www.pythonscraping.com/pages/page3.html')
bs = BeautifulSoup(html, 'html.parser')
images = bs.find_all('img', {'src':re.compile('\.\.\/img\/gifts/img.*\.jpg')})
for image in images:
    print(image['src'])
```



Lambda Expressions

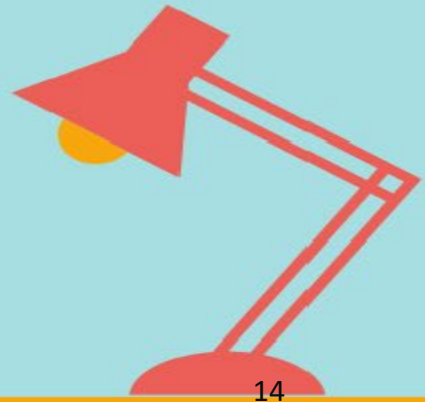
普通函數定義

```
def max(m, n):
    return m if m > n else n

print(max(10, 3)) # 顯示 10
```

Lambda函數定義

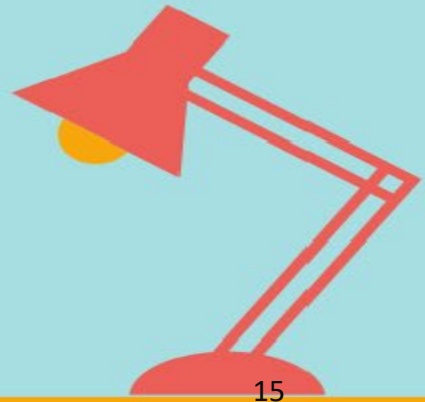
```
max = lambda m, n: m if m > n else n
print(max(10, 3)) # 顯示 10
```

```
bs.find_all(lambda tag: len(tag.attrs) == 2)

bs.find_all(lambda tag: tag.get_text() == 'Or maybe he\'s only resting?')
```



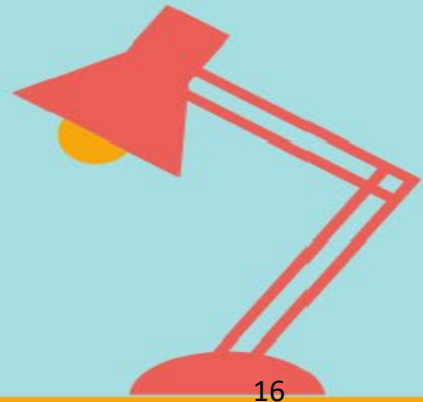
Writing Web Crawlers

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen('http://en.wikipedia.org/wiki/Kevin_Bacon')
bs = BeautifulSoup(html, 'html.parser')
for link in bs.find_all('a'):
    if 'href' in link.attrs:
        print(link.attrs['href'])
```



作業1/Assignment 1

- 概述：
  - 在本次作業中，我們將使用基本的 Python 網頁抓取工具 urllib 和 BeautifulSoup 從 cna 網站抓取數據。 請列出抓取的新聞內容並將其提交到 Tronclass 的作業條目。
- 目標：
  - 了解如何使用網頁抓取獲取網頁內容。
  - 探索真正的 html 文件。
  - 反思網絡抓取功能在數據科學中的可能用途。
- 指示：
  - 使用任何瀏覽器訪問 focustaiwan 網站。 www.cna.com.tw
  - 檢查 html 內容中的標籤。



Thanks!

Q&A