IEMS 5703
Network Programming and System Design

Lecture 1 - Course Introduction

Albert Au Yeung
11th January, 2018

Agenda

Course Administration

- Course details
- Course schedule
- Assessment Schemes
- Policies and Rules

Course Content Overview

- Computer networking and data communication
- Network programming (socket programming)
- Client-server architecture
- Network applications
- Programming in Python

Course Instructors

Lecturer: Albert Au Yeung

- Email: cmauyeung@ie.cuhk.edu.hk
- For lecture content, materials, details of assignments, project arrangements, reference materials, etc.

TA: Fenghao XU

- Email: xf016@ie.cuhk.edu.hk
- Contact Fenghao if you need specific help when working on your assignments and project

Lectures

Venue & Time

- Yasumoto International Academic Park (YIA) **LT7**
- Thursday 7:00pm – 9:30pm
- **Lecture dates (12 lectures)**:
  - 11th, 18th, 25th January
  - 1st, 8th, 22nd February
  - 1st, 8th, 15th, 22nd, 29th March
  - 12th April
- Refer to the course Website for the most up-to-date schedule of the course
- **Final Examination**:
  - 26th April

Assessment Scheme

- **10%** - Attendance (Lecture 2 to Lecture 12)
- **60%** - Programming Tasks
  - **35%**: Assignment 0 to Assignment 3
  - **25%**: Mini Project
- **30%** - Final Examination

Programming Assignments

- A total of 4 programming assignments
- All should be finished using Python
- Late submission will **NOT** be marked
- Topics of each assignment:
    1. Python programming basics
    2. Socket programming
    3. Threading, multiprocessing and asyncio
    4. Asynchronous tasks and message queue

Mini Project

- Create a **network application** with certain functions
- Examples:
  - Instant messaging
  - Web-based multiplayer game
  - News subscription system
  - File Synchronization across computers
  - ...
- Each student should work on his/her own project (NO group project)
- Criteria: system design, complexity, creativity, robustness, etc.
- More details will be provided later
- Feel free to discuss with me if you have any idea

What should you expect?

Take this course if you:

- Have background in computer networks and related concepts
- Have basic understanding or willing to learn the Python Programming Language
- Would like to challenge yourself with interesting programming and system design problems

Approach of this Course

- Network programming and system architecture is a HUGE topic
- **Focus** of this course:
  - Some theories and background of computer networks
  - More practical knowledge and skills in network programming (in Python)
- What you will **learn** after taking this course?
  - Network programming in Python
  - Concurrent programming (threads and processes)
  - Various ways to enable communications between clients and servers
  - How to build a distributed network application

Some Rules

What you should do in this course?

- Attend the lectures, and raise questions whenever you have any
- Seek help as **early** as possible (e.g. if you have difficulties in picking up Python programming, or if you cannot set up the development environment)
- Feel free to make **suggestions** to the course and/or lectures
- Do your own assignments, and do NOT make your work publicly available before the deadline

Honesty in Academic Work

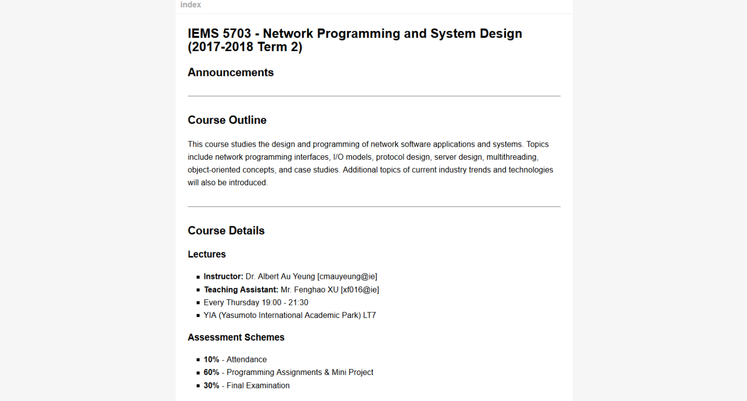
- Zero tolerance on cheating and plagiarism
- Read: http://www.cuhk.edu.hk/policy/academichonesty/
- Cite references whenever you use materials from any other sources
- It will be considered plagiarism no matter you copy other's work or allow others to copy your work

Online Resources

- Assignments will be released and collected on the CUHK E-Learning System: https://elearn.cuhk.edu.hk/
- You will submit your assignments there

Online Resources

- Course Website: https://course.ie.cuhk.edu.hk/~iems5703/ or http://iems5703.albertauyeung.com
- **Lecture slides**, **assignments**, **project details**, **references** will be available there



IEMS 5703 - Network Programming and System Design (2017-2018 Term 2)

Announcements

Course Outline

This course studies the design and programming of network software applications and systems. Topics include network programming interfaces, I/O models, protocol design, server design, multithreading, object-oriented concepts, and case studies. Additional topics of current industry trends and technologies will also be introduced.

Course Details

Lectures

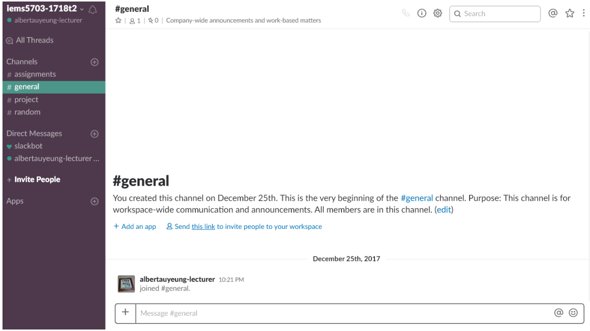
- **Instructor:** Dr. Albert Au Yeung [cmauyeung@ie]
- **Teaching Assistant:** Mr. Fenghao XU [xf016@ie]
- Every Thursday 19:00 - 21:30
- YIA (Yasumoto International Academic Park) LT7

Assessment Schemes

- **10%** - Attendance
- **60%** - Programming Assignments & Mini Project
- **30%** - Final Examination

Online Resources

- For more convenient communication among us and discussions among yourselves, we will use Slack in this course: https://iems5703-1718t2.slack.com/
- Sign up for an account on slack and join the above team
- NOTE: **DO NOT** post any solution of assignments on Slack or any other public channels



#general

You created this channel on December 25th. This is the very beginning of the #general channel. Purpose: This channel is for workspace-wide communication and announcements. All members are in this channel. (edit)

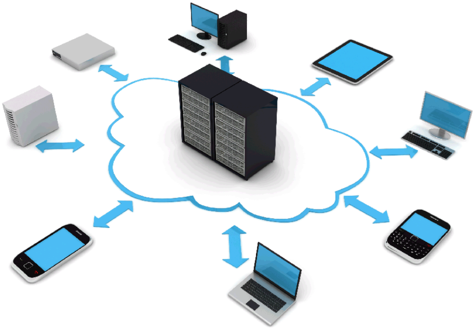
December 25th, 2017

albertauyeung-lecturer 10:21 PM
joined #general.

Course Overview

Computer Network

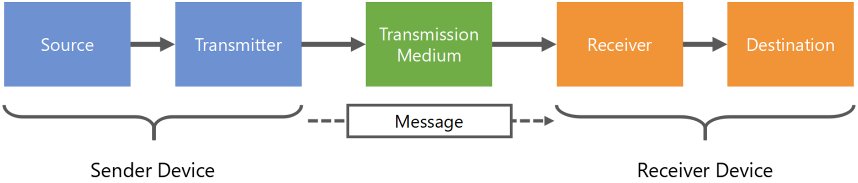
- A network that allows computers to perform data communication with one another



- The Internet is **a network of networks**. (Global Internet Traffic)

Data Communication

- Exchange of data between two devices using some form of transmission medium
- A simplified communication model:



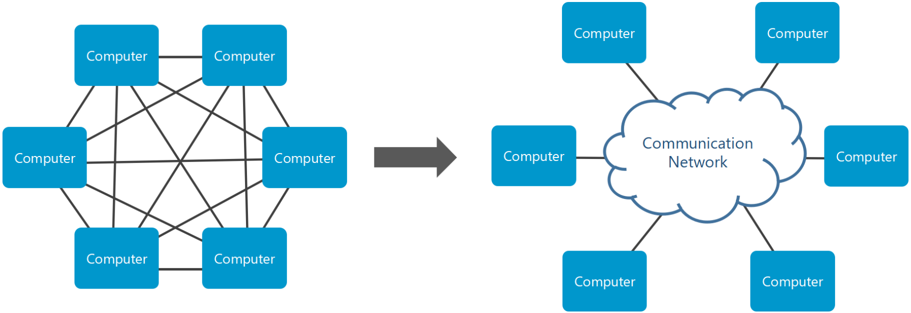
- When performing communication, we need **protocols**: rules that govern how data is transmitted in this system

Protocols

- **Network protocols** defines how computers talk to each other, including:
  - How to start a communication
  - The format of a message
  - What should be done when the data is corrupted during transmission
  - What should be done when the connection is broken during transmission
  - ...
- Examples: **TCP/IP**, **HTTP**, **FTP**
- Internet protocols are specified in documents called **Requests for Comment (RFC)**, such as:
  - RFC 793 - Transmission Control Protocol (TCP)
  - RFC 1180 - A TCP/IP Tutorial
  - RFC 6455 - The WebSocket Protocol

Computer Network

When we have many computers that want to talk to one another, point-to-point links become not practical, especially when the distance is too far



The History of Internet in 3 Minutes

Problems and Challenges in Computer Networking

Challenges in Networking:

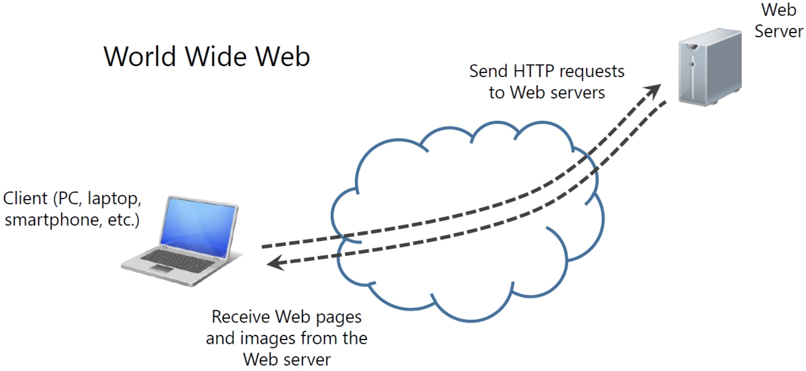
- How can data be transmitted from one node to another through the network?
  (e.g. routing/switching)
- How can we address the computers?
  (e.g. IP Address)
- How can we identify which applications on the computers the data should be delivered to?
  (port and socket)
- How to handle error or missing data?
  (e.g. the TCP protocol)
- What if a large amount of data is transmitted at the same time?
- How to **coordinate** a large number of applications over a network?

Applications

Common Applications on the Internet

- The World Wide Web (Web servers and browsers)
- File transfer (FTP servers and clients)
- Instant messaging & video conferencing (e.g. Skype, Whatsapp, Wechat)
- Peer-to-peer file sharing
- Video and audio streaming
- Cloud storage (Sync files across machines)
- ...

Example 1: The World Wide Web



World Wide Web

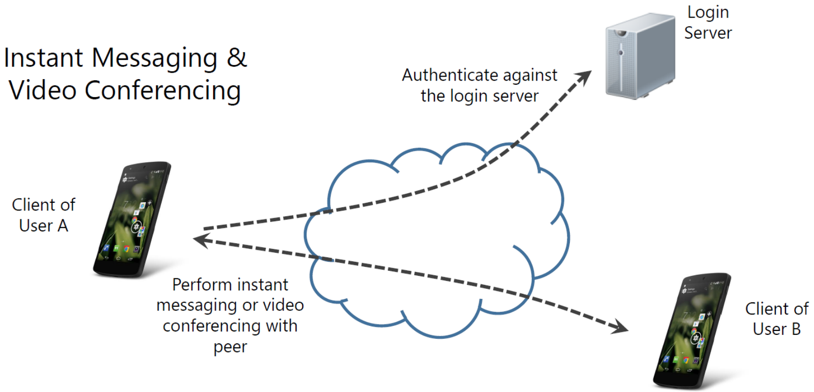
Web Server

Send HTTP requests to Web servers

Client (PC, laptop, smartphone, etc.)

Receive Web pages and images from the Web server

Example 2: Instant Messaging & Video Conferencing



Instant Messaging & Video Conferencing

Login Server

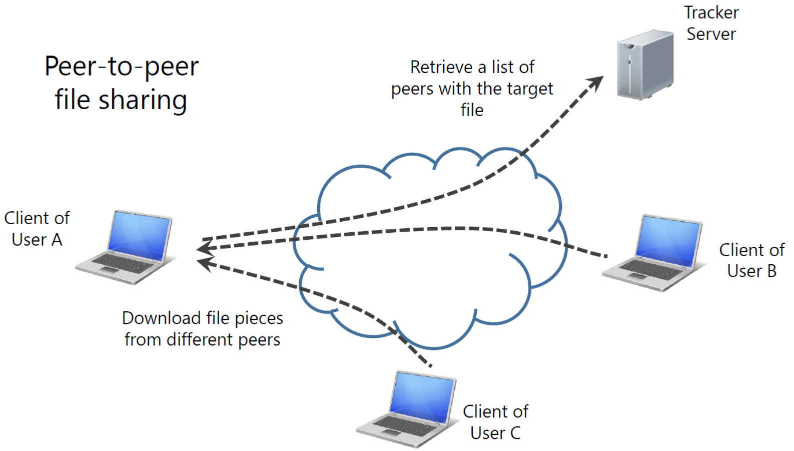
Authenticate against the login server

Client of User A

Perform instant messaging or video conferencing with peer

Client of User B

Example 3: P2P File Sharing



Major Topics

Network Programming

- How to make two or more computers talk to each other over a network?
- How to use common protocols to send and receive data?

Concurrent Programming

- How to simultaneously carry out different task in a program

Scalable architecture

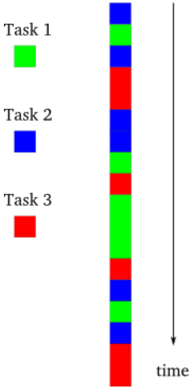
- How to make a system scalable when traffic increases and the system becomes more complex?
- How to make a network application robust?
- How to develop an efficient network application?

Network Programming

- Enable communications among computers using some protocols
- Our focus:
  - TCP/IP (TCP & UDP)
  - HTTP, Websockets
  - Develop your own servers and clients in Python
  - Data format for exchanging information (e.g. JSON, XML)
- Web scrapers / crawlers to collect information from the Internet

Concurrent Programming

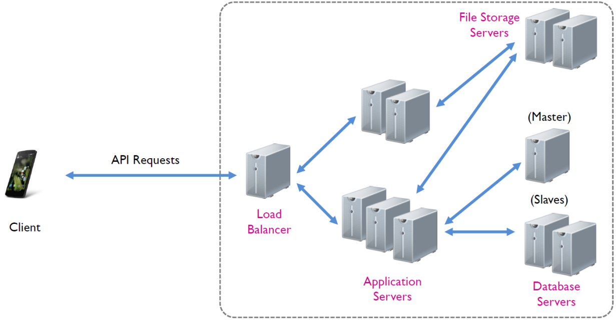
- How to perform tasks in parallel
- Our focus:
  - Threading and multiprocessing
  - Limitations of multithreading in Python
  - Aysnchronous model
  - Blocking and non-blocking calls



The Asynchronous model (Ref: Twisted Introduction - Part 1)

Scalable Architecture

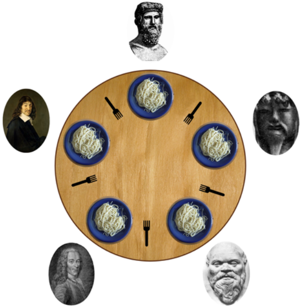
- How to design the architecture of a network application?
- How to coordinate different components in system when complexity increases
- Our focus:
  - Asynchronous tasks and message queues
  - Using databases



Challenges in Network and Concurrent Programming

The Dining Philosophers Problem

- Details: [Dining philosophers problem - Wikipedia]
- Five philosophers sit at a round table with bowls of spaghetti. Forks are placed between each pair of adjacent philosophers.
- Each philosopher must alternately think and eat.
- A philosopher can only eat when he has both left and right forks.
- Each fork can be held by only one philosopher.
- A proper solution should never arrive in a **deadlock** situation.



Challenges in Network and Concurrent Programming

The CAP Theorem

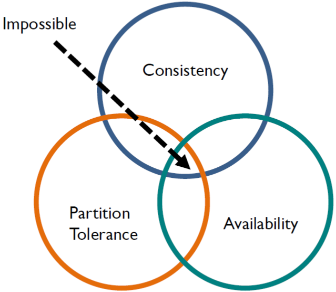
- Details: CAP theorem - Wikipedia
- In a distributed system, three properties are of particular interests:
  - C – Consistency
  - A – Availability
  - P – Partition Tolerance
- Recommended Reading: Kaushik Sathupadi. ‘A plain english introduction to CAP Theorem’ http://ksat.me/a-plain-english-introduction-to-cap-theorem/

The CAP Theorem

- **C: (Atomic) Consistency**
  - A ‘read’ to the system will always reflect the latest ‘write’ action
  - To the rest of the system, a change occurs instantaneously, all node sees the same data at the same time
  - Example: Once you ‘liked’ a post, all users will see your action
- **A: Availability**
  - Every request received by a non-failing node must result in a response (the system is continuously available to the clients)
  - It does not guarantee that the response is given in a specific period of time, however there should be a response for ever request
- **P: Partition Tolerance**
  - A distributed system has multiple nodes, partition tolerance requires that the system continues to operate even when the network fails
  - When a network is partitioned, all messages sent from one node to another node will be lost

The CAP Theorem

- Also known as Brewer’s Theorem
- It states that it is **impossible** for a distributed system to have **all 3 properties** at the same time.
- Reference: Seth Gilbert and Nancy Lynch, “Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services”, ACM SIGACT News, Volume 33 Issue 2 (2002), pg. 51-59.



The CAP Theorem

Choosing between consistency and availability:

- **C + P**
  - When network is partitioned, partitioned nodes will not be able to return a response
  - Clients receive timeout or error
  - Preferred when **strict atom consistency is needed** (e.g. e-commerce site)
- **A + P**
  - A partitioned node will return the most recent version of the data it has, not guaranteed to be the same as the latest version
  - Opt for this if **availability** is important, and there is flexibility in returning the latest data to the clients

Python Programming

What is Python?

- An high-level interpreted programming language
- Created by Guido van Rossum in 1991
- Emphasizes code readability and flexibility (See Python's Design Philosophy)
- Current stable versions: Python 2.7 (Version 2), and Python 3.6 (Version 3)

Programming in Python

- Hello World in Python

```
$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>>
```

- Type **`python3`** (or simply **`python`**) to invoke the Python interpreter
- **`print()`** will output the arguments to standard output

Programming in Python

- Python programs (or scripts) are commonly named using the `.py` extension, and are called **modules**
- A "hello world" script **`hello.py`**:

```python
print("Hellow World!")
```

- Executing the script:

```
$ python3 hello.py
Hello World!
$
```

What do people use Python for?

Python is a general purpose programming language and are widely used in different domains. (See Python Success Stories)

- Web and Internet applications backend (e.g. Youtube, Dropbox, Reddit)
- Scientific computing
- Data science and machine learning (e.g. Tensorflow, Keras)
- Data visualization
- Financial Analysis
- ...

Installing Python

- Available on Linux / Mac / Windows (https://www.python.org/downloads/)
- Note: Download **Python 3.6** for this course
- IDEs recommended for Python programming:
  1. JetBrains PyCharm (Community Edition is free)
  2. MS Visual Studio Code (Free and open source)
- Python comes with some standard modules, other modules can be installed using `pip` (https://pypi.python.org/pypi). For example:

```
$ python3 -m pip install requests
```

Python Basics

```python
# Everything after a `#` is comment
# import modules using the import keyword
import math

# define functions using def
def power_three(x):
    return math.pow(x, 3)

if __name__ == "__main__":
    print(power_three(10))

# Executing this script prints 1000.0
```

- In Python, **indentation** is important: the statements in the same logical block should have the **same** indentation.
- Set your editor to use SPACES instead of TAB for indentation.
- You **do not** have to declare a variable before using it

Python Basics

```
# if-then-else statements
if x == 0:
    print("Zero!")
elif x > 0:
    print("Larger than Zero!")
else:
    print("Less than Zero")

# while loop
x = 0
while x < 10:
    x += 1

# for loop
for x in range(10):
    print(x)
```

Python Data Structures

Lists

- Lists are like arrays in other languages, but are more flexible

```python
cities = ["Hong Kong", "Macau", "Taipei", "Beijing"]
print(cities[0])     # prints "Hong Kong"
print(cities[2])     # prints "Taipei"

print(cities[-1])    # prints "Beijing"

print(len(cities))   # prints 4

print(cities[1:3])   # prints ["Macau", "Taipei"]
print(cities[:3])    # prints ["Hong Kong", "Macau", "Taipei"]
print(cities[2:])    # prints ["Taipei", "Beijing"]
print(cities[::-1])  # prints ["Beijing", "Taipei", "Macau", "Hong Kong"]
```

Python Data Structures

Using lists in for loops

- Lists are iterables, meaning that you can loop through each of its values as follows:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sum = 0
for n in numbers:
    sum += n

print(sum)  # prints 55
```

Python Data Structures

List methods

- List objects have a number of using methods:

```
x = [1, 2, 3]

x.append(4)             # x becomes [1, 2, 3, 4]
x.insert(0, 0)          # x becomes [0, 1, 2, 3]
x.extend([4, 5])        # x becomes [1, 2, 3, 4, 5]
x.sort(reverse=True)    # x becomes [3, 2, 1]
x.index(2)              # this returns 1, the index of the element 2
```

- For a comprehensive list of methods, see https://docs.python.org/3/tutorial/datastructures.html

Python Data Structures

Dictionaries

- Another commonly used data structure in Python is the **dictionary**
- It can be used to store **key-value pairs** (Similar to the "associative arrays" in PHP)
- Keys must be immutable types (e.g. Lists cannot be used as keys)

```
exam_scores = {"John": 70, "Mary": 80}

print(exam_scores["John"])   # prints 70
print(exam_scores["Mary"])   # prints 80

print(list(exam_scores.keys()))    # prints ["John", "Mary"]
print(list(exam_scores.values()))  # prints [70, 80]
```

Python Data Structures

Iterating over key-value pairs in a dictionary

- Given that a dictionary is used to store key-value pairs, you can iterate over all key-value pairs using a loop as follows:

```python
exam_scores = {"John": 70, "Mary": 80}

# exam_scores.items() actually returns [("John", 70), ("Mary", 80)]
# which is a list of 2-tuples
for name, score in exam_scores.items():
    print("%s scores %d in the exam." % (name, score))
```

- In the above `print` statement, `%s` is a placeholder for a string, and `%d` is a placeholder for an integer.
- More about dictionaries can be found at https://docs.python.org/3/tutorial/datastructures.html

Files in Python

```
f = open("file.txt", "r")   # open a file in read mode
for line in f:
    print(line)

f.close()  # Always close the file after use
```

- In practice, it is better to use the **codec** module to open a file, as it supports reading and writing unicode files (e.g. files with Chinese or Japanese characters encoded using UTF-8)

```
import codecs

f = codecs.open("file.txt", "r", "utf-8")
for line in f:
    print(line)

f.close()
```

Python Modules

- A **module** in Python is a file containing Python definitions and statements
- You can put your source codes in different modules to avoid having a huge single `.py` file if your project is large
- You can **import** class, functions and variables from other modules

```python
# This is in my_functions.py
def factorial(n):
    f = 1
    for i in range(n):
        f *= i + 1
    return f

# In another file main.py
from my_functions import factorial
print(factorial(5))  # prints 120
```

- Reference: https://docs.python.org/3/tutorial/modules.html

More about Python Programming

Documentations and Tutorials

- Read about the history of Python at https://en.wikipedia.org/wiki/Python_(programming_language)
- Read Python tutorials at https://docs.python.org/3/tutorial/
- Consult the documentation at https://docs.python.org/3/

Coding Convention and Styles

- Python's development is based on the Python Enhancement Proposals (PEP), which is a list of proposals of new features
- PEP 8 describes coding conventions or style guides for Python programming.

Others

- Explore Python packages and projects online: https://github.com/vinta/awesome-python

Using Virtualenv

Dependencies

- When working on a Python project, it is common that you will use modules outside of the standard library (e.g. requests, BeautifulSoup, numpy, pandas)
- These are called the project's dependencies
- Different projects may have different dependencies (on **different modules**, or even **different versions** of the modules)

Project isolation

- Virtualenv is a software that allows you to create an isolated environment for a project
- Install virtualenv by:

```
$ python3 -m pip install virtualenv
```

Using Virtualenv

- Once installed, you can create a virtual environment using the following command (`venv` is the name of the environment, which you can choose as you like):

```
$ virtualenv venv
```

- To activate the environemnt, use the following command:

```
$ source venv/bin/activate
(venv) $
```

- Once you see the `(venv)` prefix, it means that the virtual environment is successfully activated.
- From this point onwards, all `pip install` command will only install packages **within this environment**
- To exit the environment, type `deactivate`

Assignment 0

Python Programming Exercises

- http://iems5703.albertauyeung.com/assignment-0.html
- Refer to the instructions on the course Web site
- Submit your files in the format described in the instruction
- Late submissions will **NOT** be marked
- Make sure that your program can be executed under **Python 3.5**
- Search for "Python exercises" if you think this is not enough

End of Lecture 1