HW0: Khởi động (cài đặt các công cụ; sử dụng Jupyter Notebook, Python, và Numpy ở mức cơ bản)

Trần Trung Kiên (ttkien@fit.hcmus.edu.vn)

Cập nhật lần cuối: January 17, 2024

Cài đặt các công cụ

Terminal (cửa sổ dòng lệnh)

Với các bạn dùng Windows, mặc dù không bắt buộc nhưng mình khuyên là nên cài **WSL - Windows Subsytem for Linux** (bạn có thể cài theo video này) và cài đặt các công cụ mà sẽ nói ở dưới trên WSL. Mình muốn bạn làm quen dần với các câu lệnh Linux vì sau này nhiều khả năng là bạn sẽ đụng đến (ví dụ, khi bạn phải tương tác với OS trên Google Colab hoặc khi phải remote vào các server). Thêm nữa, so với cách tương tác với máy tính bằng giao diện đồ họa như bạn vẫn quen làm thì cách tương tác bằng các câu lệnh sẽ cho bạn nhiều sức mạnh điều khiển hơn (mặc dù ban đầu mới chuyển từ giao diện đồ họa qua các câu lệnh thì bạn sẽ thấy hơi khó khăn). Để sử dụng thành thạo các câu lệnh thì cần thực hành cho nhiều. Khi sử dụng máy tính hàng ngày, bạn cố gắng tập tương tác với máy tính thông qua các câu lệnh (trong WSL, để mở một file nào đó bằng chương trình mặc định của Windows thì bạn có thể gõ `explorer.exe tên-file`, hoặc để mở chương trình explorer của Windows tại thư mục hiện tại thì bạn gõ `explorer.exe .` vì `.` có ý nghĩa là đường dẫn đến thư mục hiện tại). Nếu bạn nào không biết những câu lệnh Linux cơ bản như `ls`, `cd`, `mv`, `cp`, … thì bạn có thể xem 2 video đầu trong series này.

Nếu không dùng WSL thì bạn có thể dùng các terminal có sẵn của Windows

như cmd hoặc PowerShell. Có lẽ bạn nên chọn PowerShell vì … có từ "power" ;-). Từ đây trở đi, khi nói về terminal của Windows là mình nói về PowerShell, không phải cmd.

Conda

Khi làm lập trình với Python, ta có nhu cầu cài đặt các công cụ / package như Jupyter Notebook, Python, … Việc cài đặt các package một cách thủ công rất phiền vì mỗi package thường sẽ cần tới các package khác; hơn nữa, phiên bản cụ thể của mỗi package thường sẽ yêu cầu các phiên bản cụ thể của các package khác. Ở các môn trước, bạn có thể đã được hướng dẫn cài Anaconda; khi cài Anaconda thật ra là ở bên dưới đã cài cho bạn rất nhiều package phổ biến và Ananconda đã giúp bạn giải quyết vấn đề phụ thuộc giữa các package mà mình đã nói ở trên. Việc cài Anaconda có ưu điểm là đơn giản, nhưng nhược điểm là cài đặt rất nhiều package mà bạn có thể không dùng đến (kích thước của thư mục Anaconda: ~5G).

Khi làm lập trình với Python, ta cũng có nhu cầu chia sẻ môi trường code (gồm các package nào, mỗi package có phiên bản bao nhiêu) để người khác có thể chạy ra được các kết quả giống như của ta. Ngoài ra, ta cũng có thể có nhu cầu có nhiều môi trường code khác nhau cho nhiều project khác nhau (ví dụ, một project dùng Python 2, một project dùng Python 3).

Một giải pháp cho tất cả các nhu cầu đã nêu là:

1. Cài Miniconda. Miniconda là tập con của Anaconda, và chỉ bao gồm các package cơ bản, trong đó quan trọng nhất là **Conda** - chương trình vừa quản lý việc cài đặt / xóa các package, vừa quản lý việc tạo / xóa các môi trường code. Nếu bạn nào đã cài Anaconda ở các môn trước thì bạn có thể xóa Anaconda trước rồi sau đó cài Miniconda; nếu bạn vẫn muốn giữ lại Anaconda (ví dụ, bạn vẫn đang cần dùng cho một môn/việc khác) thì bạn không cần cài Miniconda, vì trong Anaconda đã có Conda rồi. Đối với những bạn cài Miniconda thì bạn download bản ứng với Python 3.x. Nếu bạn dùng Windows và muốn cài trên WSL thì bạn download bản của Linux; bạn cài bằng cách mở terminal của WSL, `cd` đến file cài vừa download (đuôi .sh), rồi gõ `bash tên-file.sh`, trong quá trình cài nếu có hỏi gì thì bạn cứ chọn đồng ý.
2. Dùng Conda để tạo các môi trường code, cài đặt các package cần thiết

trên mỗi môi trường code, chia sẻ môi trường code với người khác. Dưới đây mình sẽ hướng dẫn chi tiết các câu lệnh trên terminal của Conda.

Khi cài Anaconda/Miniconda thì mặc định sẽ có một môi trường code tên là base; môi trường này thường sẽ được tự động kích hoạt khi bật terminal (bạn sẽ thấy phía trái của dòng prompt ở terminal có từ base). Các bạn dùng Windows và không dùng WSL lưu ý: sau khi cài Miniconda thì bạn sẽ thấy có 2 terminal mới là Anaconda Prompt (ứng với cmd) và Anaconda PowerShell Prompt (ứng với PowerShell), bạn sẽ dùng một trong 2 terminal này. Ở dưới đây, khi nói về các câu lệnh, nếu mình không nói gì thì có nghĩa là chạy được trên cả terminal của Linux lẫn Windows. Khi mình nói terminal của Linux thì có nghĩa là WSL trên Windows hoặc terminal của Linux/Mac. Còn terminal của Windows thì có nghĩa là Anaconda PowerShell Prompt.

Để xem các môi trường code hiện có và thư mục mà mỗi môi trường được cài đặt:
`conda env list`

Để tra cứu thông tin của một câu lệnh conda, chẳng hạn `conda env list`:
`conda env list --help`

Để tắt môi trường code hiện tại:
`conda deactivate`

Để bật một môi trường code nào đó, chẳng hạn base:
`conda activate base`
Một môi trường code chẳng qua là một thư mục chứa các package được cài đặt, và việc bật/tắt một môi trường về cơ bản là thêm/xóa đường dẫn đến thư mục của môi trường vào biến PATH (trong hệ điều hành, biến PATH chứa đường dẫn đến các thư mục; ở terminal, khi ta gõ tên một chương trình, chẳng hạn python, thì hệ thống sẽ tìm kiếm file chạy tên là python trong các thư mục ở biến PATH, nếu tìm thấy thì sẽ chạy file này).
Để in ra nội dung biến PATH: `echo $PATH` (Linux) hoặc `$Env:path` (Windows).
Để biết khi gõ tên một chương trình ở terminal, chẳng hạn python, thì file chạy python ở thư mục nào sẽ được thực thi: `which python` (Linux) hoặc `gcm python` (Windows).

Ở môi trường code hiện tại, để xem danh sách các package đã được cài đặt:
`conda list`

Ở môi trường code hiện tại, để cài đặt các package:
`conda install tên-package-1 tên-package-2 ...`

Để chỉ định thêm thông tin về phiên bản:
`conda install tên-package-1=phiên-bản-mấy ...`
Câu lệnh `conda install` sẽ lên các thùng chứa package trên mạng (còn gọi là channel) của Conda (mặc định thì chỉ có một thùng chứa tên là default), rồi download và cài đặt các package cần thiết.

Để tra cứu thông tin về các phiên bản hiện có của một package nào đó ở thùng chứa trên mạng:
`conda search tên-package`

Ở môi trường code hiện tại, để xóa các package:
`conda remove tên-package-1 tên-package-2 ...`

Để tạo một môi trường code tên x với các package p1, p2:
`conda create -n x p1=ver-mấy p2=ver-mấy`

Ta cũng có thể ghi thông tin các package vào file (ghi như thế nào thì một lát mình sẽ nói) rồi tạo môi trường từ file này:
`conda env create --file tên-file`

Ta có thể chia sẻ file này với những người khác để đảm bảo mọi người đều có cùng một môi trường code.

Nếu file này được cập nhật thì ta có thể cập nhật môi trường code như sau:
`conda env create --file tên-file --force`

Để xóa một môi trường code:
`conda remove -n tên-môi-trường --all`

Jupyter Notebook, Python, và các thư viện của Python

Mình đã ghi các package được sử dụng trong môn học (Jupyter Notebook, Python, …) cùng với phiên bản của mỗi package vào file "ml-env.yml". Để đảm bảo không có vấn đề gì khi thầy cô chấm bài của trò thì mọi người sẽ cùng sử dụng môi trường code được tạo ra từ file này:
`conda env create --file ml-env.yml`

Nói thêm về channel tên là conda-forge trong file "ml-env.yml":

- Đây là thùng chứa các package trên mạng do cộng đồng người dùng tạo ra; còn thùng chứa chính thức của Conda tên là defaults
- So với defaults thì conda-forge có nhiều package hơn và thường được cập nhật hơn; do đó, mình thường ưu tiên dùng conda-forge hơn so với defaults

Test môi trường code tạo từ file "ml-env.yml":

- Ở terminal, gõ `conda activate ml-env` để kích hoạt môi trường "ml-env"
- Ta kỳ vọng khi gõ `python` thì hệ thống sẽ chạy file chạy python ứng với môi trường "ml-env". Để kiểm tra điều này, một cách là bạn gõ `which python` (Linux) hoặc `gcm python` (Windows). Một cách khác là bạn sẽ gõ `python` để vào cửa sổ dòng lệnh của Python, rồi gõ `import sys` rồi `sys.executable`.
  Nói thêm về cửa sổ dòng lệnh của Python. Ở cửa sổ dòng lệnh của Python, bạn có thể thực thi các câu lệnh của Python; ví dụ, bạn có thể gõ `1 + 1`. Để thoát ra khỏi cửa sổ dòng lệnh của Python, bạn gõ `quit()`. Sau khi thoát thì các câu lệnh đã gõ sẽ không được lưu lại; do đó, mình thường chỉ dùng cửa sổ dòng lệnh của Python để test nhanh các câu lệnh. Trong các bài tập môn học, bạn sẽ code trên Jupyter Notebook (sẽ nói ở dưới), chứ không dùng cửa sổ dòng lệnh này.

Sử dụng Jupyter Notebook ở mức cơ bản

Đầu tiên, bạn download file được viết bằng Jupyter Notebook này (chuột phải rồi "save as", bạn để đuôi của file là ipynb). Để mở file này thì ở terminal bạn `cd` đến thư mục chứa file này, rồi gõ `jupyter notebook`. Bạn sẽ thấy một server cục bộ được tạo ra, và trang chủ của notebook được mở ra ở web browser của bạn. Nếu trang chủ của notebook không được tự động mở ra ở web browser (ví dụ, nếu bạn dùng WSL trên Windows) thì bạn sẽ cần copy url mà bạn thấy ở cửa sổ dòng lệnh (có dạng: `http://localhost:8888 ...`) và paste vào web browser. Ở trang chủ của notebook, bạn sẽ thấy file .ipynb vừa download. Bạn có thể mở file ipynb bằng cách click vào file này. Notebook ứng với file này sẽ được mở ra ở một tab mới.

Một notebook sẽ được cấu thành từ các **cell**. Bạn có thể dùng phím mũi tên lên/xuống để di chuyển giữa các cell. Có hai loại cell là **markdown cell** và

code cell. Markdown cell cho phép soạn thảo và hiển thị văn bản bằng cách viết code theo cú pháp của Markdown rồi thực thi. Còn code cell cho phép bạn viết và thực thi Python code. Dấu hiệu để nhận biết code cell là ở bên trái có "In […]":, còn markdown cell thì không có.

Nếu bạn có thể dùng phím mũi tên lên/xuống để di chuyển giữa các cell (chứ không phải là di chuyển giữa các dòng trong cell) thì có nghĩa là bạn đang ở chế độ **command mode** của notebook. Để chắc chắn hơn về điều này, bạn có thể nhìn cell hiện tại: nếu không có con trỏ nhấp nháy và viền của cell không có màu xanh lá thì đúng là bạn đang ở command mode. Ở command mode, bạn có thể thực hiện các câu lệnh ở cấp độ bên ngoài cell như dùng phím mũi tên lên/xuống (hoặc tiện hơn là phím `j`/`k`) để di chuyển giữa các cell, hoặc bạn có thể ấn phím `a` (above) hoặc `b` (below) để chèn thêm một cell bên trên hoặc bên dưới cell hiện tại.

Từ command mode, để vào trong cell hiện tại và thêm/sửa nội dung, bạn cần chuyển sang **edit mode** bằng cách ấn phím `enter` (từ edit mode để nhảy ra ngoài command mode, bạn ấn phím `esc`). Khi ở edit mode, bạn sẽ thấy đường viền của cell có màu xanh lá và con trỏ nhấp nháy; tại đây, bạn có thể thêm/sửa nội dung của cell một cách bình thường.

Bạn có thể xem các phím để thực hiện các câu lệnh ở command mode và edit mode bằng cách chọn "Help" – "Keyboard Shortcuts" (hoặc ấn phím `h` khi đang ở command mode). Nếu bạn muốn sử dụng notebook hiệu quả thì bạn nên học các phím này. Bạn hãy xem thử phím nào giúp chuyển từ code cell sang markdown cell và ngược lại?

Như vậy, để viết code ở một cell thì bạn cần phải vào edit mode; để thực thi code của cell đó thì bạn có thể ấn `Ctrl`+`Enter` hoặc `Shift`+`Enter` (ở mode nào cũng được). Với markdown cell thì bạn sẽ viết code theo cú pháp của Markdown (bạn xem các cú pháp thông dụng của Markdown ở đây, rất dễ), với code cell thì bạn sẽ viết code theo cú pháp của Python (sẽ được hướng dẫn bên dưới).

Để tạo ra một notebook mới thì ở trang chủ của notebook, bạn chọn "New" - "Python 3".

Jupyter Lab

Jupyter Lab là phiên bản cải tiến của Jupyter Notebook. Hai cái này về cơ bản là khác nhau ở giao diện sử dụng; Jupyter Lab cho khả năng tương

tác cao hơn (ví dụ, có thể phân chia ra thành nhiều cửa sổ con). Bạn dùng Jupyter Notebook hay Jupyter Lab thì đều được; file notebook đều chạy được trên cả hai. Để mở Jupyter Lab thì bạn gõ `jupyter lab` ở terminal (thay vì gõ `jupyter notebook`).

Sử dụng Python ở mức cơ bản

Bạn sẽ xem các video của khóa học này trên coursera (bạn sẽ cần phải đăng ký một tài khoản trên coursera). Đây là khóa học nhập môn Python rất tốt (được nằm trong danh sách "the best online courses of all-time" do trang Class Central bình chọn).

Khóa này có tất cả 7 tuần (tổng thời lượng video là khoảng 4 tiếng). Bạn chỉ cần phải xem video, code theo và làm các quiz nhỏ nhỏ sẽ hiện lên khi bạn xem video, bạn không cần phải làm phần reading và các bài tập (quiz và bài tập lập trình) ở cuối mỗi tuần (tất nhiên, nếu bạn có thời gian và muốn làm thì càng tốt). Mình nghĩ sẽ tốt hơn nếu bạn phân thời gian ra, mỗi ngày làm một ít (thay vì một lúc làm tất cả 7 tuần). Ngoài ra, việc xem video Tiếng Anh sẽ giúp bạn rèn luyện thêm về khả năng Tiếng Anh; nếu cần thì bạn có thể bật phụ đề của video.

Khóa học dùng "IDLE" để code, nhưng bạn sẽ dùng Jupyter Notebook để quen hơn với công cụ này. Đầu tiên, bạn sẽ tạo ra một file notebook, và dùng heading của markdown cell để tổ chức nội dung. Ví dụ:

- Với "Week 1 - Python, Variables, and Functions" thì bạn sẽ tạo một markdown cell với nội dung "# Week 1 - Python, Variables, and Functions" ("#" nghĩa là đánh heading 1)
- Với mỗi mục trong Week 1 (Welcome to the Course, Getting Started, Variables and Functions), bạn sẽ tạo ra một markdown cell tương ứng và đánh heading 2

Sau đó, bạn có thể xem video của Week 1 và code theo bằng cách chèn thêm các code cell ở dưới mục tương ứng trong Week 1.

Bạn nào mà đã rành Python và Jupyter Notebook thì có thể chỉ cần xem video và làm các quiz trong video thôi, chứ không cần code theo hoặc chỉ code khi cần.

Cấu trúc dữ liệu `set` của Python

Trong khóa học về Python trên coursera chưa nói đến `set` - một cấu trúc dữ liệu của Python (giống như tập hợp trong Toán) thường được dùng trong Khoa Học Dữ Liệu. Bạn sẽ xem và code theo tutorial về `set` này (thời lượng video khoảng 15 phút). (Bạn dùng Jupyter Notebook để code theo, nếu thấy khó nghe thì bạn có thể bật phụ đề của Youtube.)

Sử dụng Numpy ở mức cơ bản

Numpy là một thư viện quan trọng mà bạn sẽ sử dụng để làm các HW (homework) sắp tới. Bạn có thể đọc tutorial này (bỏ qua phần "Installing Numpy") và code theo (dùng Jupyter Notebook).