CSC17104 – PROGRAMMING FOR DATA SCIENCE

HOMEWORK 0: WARMUP – BASIC PYTHON, JUPYTER NOTEBOOK/ JUPYTER LAB

Lab Instructor: Lê Nhựt Nam

Email: lenam.fithcmus@gmail.com

1. Setup the Linux environment

To prepare for the laboratory, the students who are using Windows need to set up a Linux environment. On this environment, you must also install various useful tools such as Git, Jupyter Notebook, Python, etc. You have a variety of alternatives regarding how to handle this requirement.

- **Option 1**: You can remove all Window and install Linux. In this option, you will learn to use Linux in the fastest way. However, you need to stay highly motivated due to at the first time there are many difficulties that you must face and tackle them. And then once you have used Linux proficiently, you probably will not want to go back to Windows. If you choose this option, you must backup all your data first. Because if you are not careful when installing Linux, you may choose the wrong option that will erase all data on your computer and then install it.
  You can install Linux Mint. Linux Mint is a Linux distribution that is easy to use and provides a high survival probability for you when moving from a window to a Linux environment. In addition, when using Linux, if you need Windows, you can also install a Windows virtual machine on Linux (you can watch this video).
- **Option 2**: You can install Windows and Linux in a dual way. If option 1 strictly makes you focus on Linux to learn and use it, this option gives you the flexibility to use both of them. Similar to option 1, you also must backup all your data first.
- **Option 3**: You can install and use Linux inside Windows through a tool called WSL—Windows Subsystem for Linux (up to now, there are two versions of WSL: WSL Version 1 and WSL Version 2, depending on the Windows version that you

are using). You will be given experience in using Linux at the lowest level in the three listed options. The most significant advantages here may be the ease of installation and the method with the lowest risk of data loss.

2. Install necessary tools

To prepare for the Jupyter Notebook and Python labs, you will do this section ("Install necessary tools") and the next two ("Using Jupyter Notebook at a Basic Level", "Using Python at a Basic Level"). In this section, you will install the tools in the Linux environment that you set up above.

a. Conda

When working on Data Science, we have the need to install tools/packages such as Jupyter Notebook, Python, etc. Installing packages manually is very annoying because each package will often need other packages. Furthermore, specific versions of each package will often require specific versions of other packages. In previous courses, you may have been instructed to install Anaconda. When installing Anaconda, it installs for you a lot of packages needed for Data Science, and Anaconda helps you deal with the problem of dependencies between packages. Installing Anaconda has the advantage of being simple, but the disadvantage is that it installs a lot of packages that you may not use (the size of the Anaconda folder is ~5G).

When working on Data Science, we also have the need to share the coding environment (including which packages and how many versions each package has) so that others can run the same results as us. In addition, we may also have the need to have different coding environments for different projects (for example, one project using Python 2 and one project using Python 3).

The following is one possible solution to all the issues mentioned:

i) Install Miniconda: Miniconda is a subset of Anaconda and includes only the basic packages, the most important of which is **Conda**—the program that both manages the installation and removal of packages and manages the initialization and destruction of programming environments. If you have already installed Anaconda in previous courses on a *Linux environment*, you can remove Anaconda first and then install Miniconda; if you still want to keep Anaconda

(for example, you still need to use it for another subject/work), then you do not need to install Miniconda because in Anaconda you already have Conda. For those of you who install Miniconda, you can download the Python 3.x version and for Linux. You install by opening the terminal, `cd` to the downloaded installation file (.sh extension), then typing `bash filename.sh`. During the installation process, if the system asks anything, you just choose to agree.

ii) Use Conda to initialize programming environments, install the necessary packages for each environment, and share the programming environment with others. Here are some detailed instructions on Conda's terminal commands.

When installing Anaconda/Miniconda, by default there will be a programming environment named "`base`". This environment will usually be automatically activated when you turn on the terminal (you will see the word "`base`" on the left side of the prompt in the terminal).

To check the available environments and the directory in which each environment is installed:

```
conda env list
```

To retrieve the information of a Conda command, for example, `conda env list`:

```
conda env list --help
```

To shutdown/ disable the current environment:

```
conda deactivate
```

To activate a Conda environment, for example, the base environment

```
conda activate base
```

A Conda environment is just a directory where packages are installed and enabling or disabling an environment is essentially adding or removing the path to the environment's directory, which is stored in the PATH variable. In an operating system, the PATH variable contains paths to directories. In the terminal, when you type the name of a program, such as python, the system will search for the executable file named python in the directories in the PATH variable and if the file is found, it will be executed.

To show the content of PATH variable:

```
echo $PATH
```

To know the directory of the executable programming in terminal, such as python

```
which python
```

In the current Conda environment, to show the list of all installed packages:

```
conda list
```

In the current Conda environment, to install packages

```
conda install package_name_1 package_name_2 …
```

To install a specific version of the package

```
conda install package_name_1=version …
```

The command conda install will go to the online repository (named channel) of Conda (in the default way, there is only one repository named default), then download and install the necessary dependency packages.

To retrieve the information about current version of any package in the online repository

```
conda search package_name
```

To remove/ delete a package in current environment

```
conda remove package_name_1 package_name_2 …
```

To initialize an environment with name x with package p1, p2:

conda create -n x p1=version_p1 p2=version_p2

We can also write package information to a file and create an environment from this file:

```
conda env create --file filename
```

Thus, we can share this environment configuration file with other people/friends to make sure that everyone in your group is using the same environment for programming.

In the case that this configuration is updated through this file, we can update our own environment by using the command

```
conda env create --file filename –force
```

And, to remove a Conda environment

```
conda remove -n environment_name --all
```

b. Jupyter Notebook, Python, and Python libraries

The packages used in the course (Jupyter Notebook, Python, ...) along with the version of each package into the file "`min_ds-env.yml`". To ensure that there are no problems when the teacher marks the student's work, everyone will use the same programming environment created from this file:

```
conda env create --file min_ds-env.yml
```

More about the channel named conda-forge in the file "`min_ds-env.yml`":

- This is a repository for online packages created by the user community, and Conda's official repository is default.
- Compared to default, conda-forge has more packages and is more frequently updated.
- In a programming environment, you can install packages from many different containers, but it is best to install only from one container (because packages in a container will work best); if you must choose between default and conda-forge, it's entirely up to you, but it should be coda-forge.

Test the environment created from the file "`min_ds-env.yml`":

- In the terminal, type conda activate min_ds-env to activate the environment "`min_ds-env`"
- We expect that when typing python, the system will run the python file corresponding to the "`min_ds-env`" environment. To test this, one way is to type `python`. Alternatively, you would type python to get to the Python command line, then type `import sys`, then `sys.executable`.

We will talk more about the Python command line. In the Python command line, you can execute Python commands; for example, you could type `1 + 1`. To exit the Python command line, type `quit()`. After exiting, the entered commands will not be saved. Therefore, it is common to only use Python's command line to quickly test commands. In the course assignments, you will code in Jupyter Notebook (discussed below), not in this command line.

3. Using Jupyter Notebook at Basic level

First, you download the file written in this Jupyter Notebook (right click and then "save as", you leave the file extension ipynb; or you can use the `wget link-download command`). To open this file, in the terminal you `cd` to the directory containing this file, then type `jupyter notebook`. You should see a local server created and the notebook's home page opened in your web browser. If the notebook home page is not automatically opened in the web browser (for example, if you use WSL on Windows), then you will need to copy the URL you see in the command line window (`http://localhost:8888/...`) and paste it into the web browser. On the notebook's home page, you will see the .ipynb file you just downloaded. You can open the ipynb file by clicking on it. The notebook corresponding to this file will be opened in a new tab.

A notebook will be made up of cells. You can use the up and down arrow keys to move between cells. There are two types of cells: markdown cells and code cells. Markdown cells allow editing and displaying text by writing code according to Markdown syntax and then executing it. The code cell allows you to write and execute Python code. The sign to identify the code cell is on the left with "In [...]," and the markdown cell does not.

If you can use the up and down arrow keys to move between cells (but not between lines in a cell), then you are in command mode of the notebook. To be surer of this, you can look at the current cell: if there is no blinking cursor and the cell border is not green, then you are in command mode. In command mode, you can execute commands at the level outside the cell, such as using the up/down arrow keys (or more conveniently j/k) to move between cells, or you can press the `a` key (above) or `b` key (below) to insert a cell above or below the current cell.

From command mode, to enter the current cell and add/edit content, you need to switch to edit mode by pressing the `enter` key (from edit mode to jump out of command mode, press the `esc` key). When in edit mode, you will see the cell border is green and the cursor is blinking. Here, you can add or edit the contents of the cell normally.

You can view the keys to execute commands in command mode and edit mode by selecting "Help" – "Keyboard Shortcuts" (or pressing the h key while in command mode). If you want to use your notebook effectively, then you should learn these keys. What key do you think helps to switch from code cell to markdown cell and vice versa?

Thus, to write code in a cell, you need to enter edit mode. To execute the code in that cell, you can press `Ctrl + Enter` or `Shift + Enter` (in any mode). With the markdown cell, you will write code according to Markdown syntax (you can see the common Markdown syntax here). With the code cell, you will write code in Python syntax (which will be guided below).

To create a new notebook, on the home page of the notebook, select "New"—"Python 3".

4. Using Python at basic level

You will watch the videos for this course from Coursera-Learn to Program: The Fundamentals (you will need to register an account on this platform in order to access this course). This is an excellent introduction to Python programming course that takes place in a list of "the best online courses of all-time" voted by Class Central.

This course has a total of 7 weeks (total video length is about 4 hours). You just have to watch the video, follow the code, and do the mini quizzes that will appear when you watch the video. You do not need to do the reading and exercises (quiz and coding exercises) at the end of each week (of course, if you have the time and want to do it, all the better). It will be better if you divide your time, doing a little bit each day (instead of doing all 7 weeks at a time). In addition, watching English videos will help you practice more of your English ability. If necessary, you can turn on the video's subtitles.

The course uses “IDLE” to code, but you will use Jupyter Notebook to get used to the tool. First, you will create a notebook file, and use the heading of the markdown cell to organize the content. For example:

- With “Week 1 - Python, Variables, and Functions” you will create a markdown cell with the text “# Week 1 - Python, Variables, and Functions” (“#” means heading 1)
- For each entry in Week 1 (Welcome to the Course, Getting Started, Variables and Functions), you will create a corresponding markdown cell and type heading 2

Then you can watch Week 1's video and code along by inserting additional code cells under the corresponding section in Week 1. Those who are familiar with Python and Jupyter Notebook can just watch the video and do the quiz in the video without the need to code, or just code when needed.

Sets, a Python data structure (similar to a set in Math) that is commonly used in Data Science, are discussed in the Python course on Coursera. You will watch and code according to the tutorial about sets: Python Tutorial: Sets-Set Methods and Operations to Solve Common Problems (video length is about 15 minutes). If you use Jupyter Notebook to code, if you find it difficult to hear, you can turn on Youtube's subtitles.)