

Python for Data Science 2

Lab 1

Amir Farbin

Plan for next few labs

- Lab 1 Session: Setup WSL/MacOS/Linux + Python + Jupyter
 - We will check you on Wednesday.
- Lab 2 Assignment:
 - Given out today
 - Due Friday 1/31 11:59pm
- Lab 2 Session (1/24):
 - Working session. Lab 3 given out (Due 2/7)
 - I'll give you a tour of Unix Shell.
- Lab 3 Session (1/31): GitHub + Get setup for submission. Submit Lab 2.

Operating System

- Software system that manages the computer hardware and software and provides common services for programs.
 - Sharing of resource between programs: processor, memory, storage, ...
 - Intermediary between programs and hardware.
 - Provides Application Program Interface (API) / Software Development Kit (SDK) for building programs and interfacing them with OS.
- Examples: Windows, MacOS, Linux, iOS, Android, ...
- Modern OSs are Multi-tasking: allow multiple programs to simultaneously run.
 - Each program ~ a process.
 - Pre-emptive multitasking: the OS gives slice of CPU time to each process.

Unix

- Multitasking, Multi-user, OS originally developed in 1970 by AT&T Bell Labs to run on mainframes with many connected terminals.
 - Written in C programming language.
- Many modern operating systems, including MacOS and Linux, implement Unix standards.
- “Unix Philosophy”
 - Plain text data storage.
 - Hierarchical file system.
 - Devices and inter-process communication via files
 - Main program that runs is the *kernel*.
 - Primary user interface is a *command-line* interpreter, called a shell.
 - Modular:
 - lots of small programs serve as tools
 - strung together via the *command-line* interpreter
 - passing information between each other via pipes

Operating Systems

- **Linux**- implementation of a Unix Kernel
 - Distributions- Packaging Linux Kernel with rest of OS software (mostly from GNU project)
 - Examples: RedHat, Debian, Ubuntu, CentOS, ...
- **BSD**- implementation of Unix Kernel + full OS
- **MacOS**- Officially certified as Unix
 - Built on BSD + Apple Kernel + Custom Apple Modules + Mac Interface
- **iOS**- Shares a lot with MacOS
- **Windows**
 - Windows Subsystem for Linux

Mixing OSs

- Each operation system has different API/SDK that allow programs to interact with the computer and its components.
- All the Unix-base OS can compile and execute any basic Unix program that do not rely on any additional APIs provided by the OS.
 - One of the reasons that Unix is widely used is that it makes it easy to make your software work on different architectures, since almost all architectures have some Unix-like OS.
- How can you run a program from OS A (e.g. Linux) to run on OS B (Windows)?
 - Two things to think about:
 - Difference in OS API/SDK
 - Difference in Architecture (e.g. x86 vs ARM)

Approaches to Mix OSs

- **Wrapper**- Thin layer that presents OS A's API to programs, but wraps OS B's APIs.
 - **WSL 1**: used this approach to enable running Linux programs in Windows
 - **Wine/SteamOS**: similar approach enabling running Windows in Unix
- **Translation**- In case of same OS but different Architecture, convert machine code from A to B.
 - Rosetta (Mac OS transition from PowerPC → Intel) did on the fly translation.
 - Rosetta 2 (Mac OS transition from Intel → Apple M1) translates Intel code to ARM first time you run.
 - Windows on ARM does something similar to run Intel x86 Apps.
- **Virtual Machine**- An application that simulates (usually with help from underlying OS and hardware) a computer and its components
 - “Image” files on host machine appear as storage devices
 - For example: Create image file, boot VM, install OS into image file.
 - **WSL 2**: Uses VMs to run an actual Linux kernel.
 - You can use VM software to run one OS inside of another (e.g. Parallels or VMWare for running Windows programs on the Mac)
- **Hypervisor**- Separates computer resources and simultaneously runs several VMs
 - For example in the cloud
- **Container**- Encapsulates everything needed to run specific software (including OS), but shares same Kernel and Resources between containers (and possibly host OS).
 - Much less resource intensive than a VM

Why not Anaconda?

- While it appears to make things easy to install on on Windows...
 - It isn't necessary. All the packages we need can be installed in one command using python's pip installer.
 - It “fakes” a unix like environment, with many limitations.
 - I want you to be exposed to Unix ASAP.
 - Its difficult for me to support.
- If you are using a windows computer, I require you to use WSL.

Windows Subsystem for Linux

+ Data Science Stack

1. Enable Windows Linux Subsystem

- Settings → Apps → Programs and Features → Turn Features on or off → Windows Subsystem for Linux

2. Install Ubuntu 24 LTS

- Windows App Store

3. Startup Ubuntu (starts a shell)

- First time will ask for username/password to make an account
- Remember this username / password

4. Update package list (the rest of these commands you type into the terminal/shell window)

- `sudo apt-get update`

5. `sudo apt-get install <package_name>`

- python3, python3-pip

6. Close and re-open the Ubuntu Shell (sets up path correctly)

7. `sudo pip3 install <package_name>`

- numpy, matplotlib, pandas, jupyter

8. run: `jupyter notebook`

- Copy/paste link into browser

WSL Disk vs Windows Disk

- The WSL virtual machine emulates a storage volume where your install of linux and everything else you do will sit.
- This is distinct from the usual storage volumes in Windows.
- The WSL virtual volume is just an “image” file on your windows volume which WSL uses to emulate a storage disk.
- But Windows “mounts” the WSL volume and WSL mounts the Windows volume, meaning
 - You can get to the Windows volume from WSL (for example the Ubuntu terminal or jupyter notebook)
 - You can get to the WSL volume from Windows (for example your browser, Explorer, or any Windows App)
- Your browsers are running in Windows, so your downloads are on the Windows Volumes. You’ll have to copy them into the WSL volumes.
- How? From Google:
 - To access WSL volumes from Windows, navigate to the path "\\wsl\\" in your File Explorer, where you will see a list of your installed Linux distributions, allowing you to browse their file systems as if they were network shares; for example, to access the Ubuntu distribution, go to “\\wsl\\Ubuntu”
 - To access Windows volumes from within WSL (Windows Subsystem for Linux), navigate to the /mnt directory within your Linux distribution, where each Windows drive letter will be represented as a folder, like /mnt/c for your C: drive; essentially, you can access your Windows files by using the path /mnt/<drive letter> within your WSL terminal.

Getting Lab 2

- Option 1: Download it from the web and copy it into your WSL volume
- Option 2: Clone repository:
 - `git clone https://github.com/UTA-DataScience/DATA3402.Spring.2025.git`

For Wednesday's Lecture

- Bring your laptops.
- Make an attempt to get WSL setup (if on Windows), install python, Jupyter.
- Have jupyter running with Lab 2 loaded.
- The TAs will go around and check your laptops.
- Don't panic if you have trouble.
 - Most of you won't have any issues.
 - If you are having issues, TA and I will help resolve them.