Python for Data Science 2

Lab 1

Amir Farbin

Review

- Please Review Early Lectures from 1401

  - Lecture 3: From Transistor to iPhone

  - Lecture 4: Storage, Filesystem, Firmware, ...

  - Lecture 5: Operating Systems, Programming Languages, ...

  - Lecture 6: History of Programming Languages

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.

- 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:

    - *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*: 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.

    - *Virtual Machine*- An application that simulates (usually with help from underlying OS and hardware) a computer and it's 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

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 20 LTS

   • Windows App Store

3. Startup Ubuntu (starts a shell)

   • First time will ask for username/password to make an account

4. Update package list

   • `sudo apt-get update`

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

   • python3, python3-pip

6. Close are open 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

Questions…

- Whats the difference between the Internet and the World Wide Web?

    - Explain the relations between Web server, HTTP (Hypertext Transfer Protocol), browser?

    - What is a Denial of Service attack?

- What are the components of a Data Center?

    - power (UPS) / cooling

    - network

    - storage

    - compute

    - services, for example database

- What is Cloud computing?

    - What is Virtualization?

        - Explain difference between a Virtual Image, a Virtual Machine, Virtualization, and a Hypervisor?

    - What is Containerization?

- Whats the difference between a Data Center and a Supercomputer (aka High Performance Computing)?

- What's the difference between a CPU and an accelerator like a GPU?