DATA SCIENCE DEVELOPMENT ENVIRONMENT¶

Peyman Hesami

Data Science FoundationsGeneral Assembly

August 1st, 2017

LEARNING OBJECTIVES- PART 1

- **Using the Command Line**

After this lesson, you will be able to:

- Create folders and files and execute commands using the command line (mkdir, touch, cd, ls, …)
- Get familiar with Python development environment

PRE-WORK

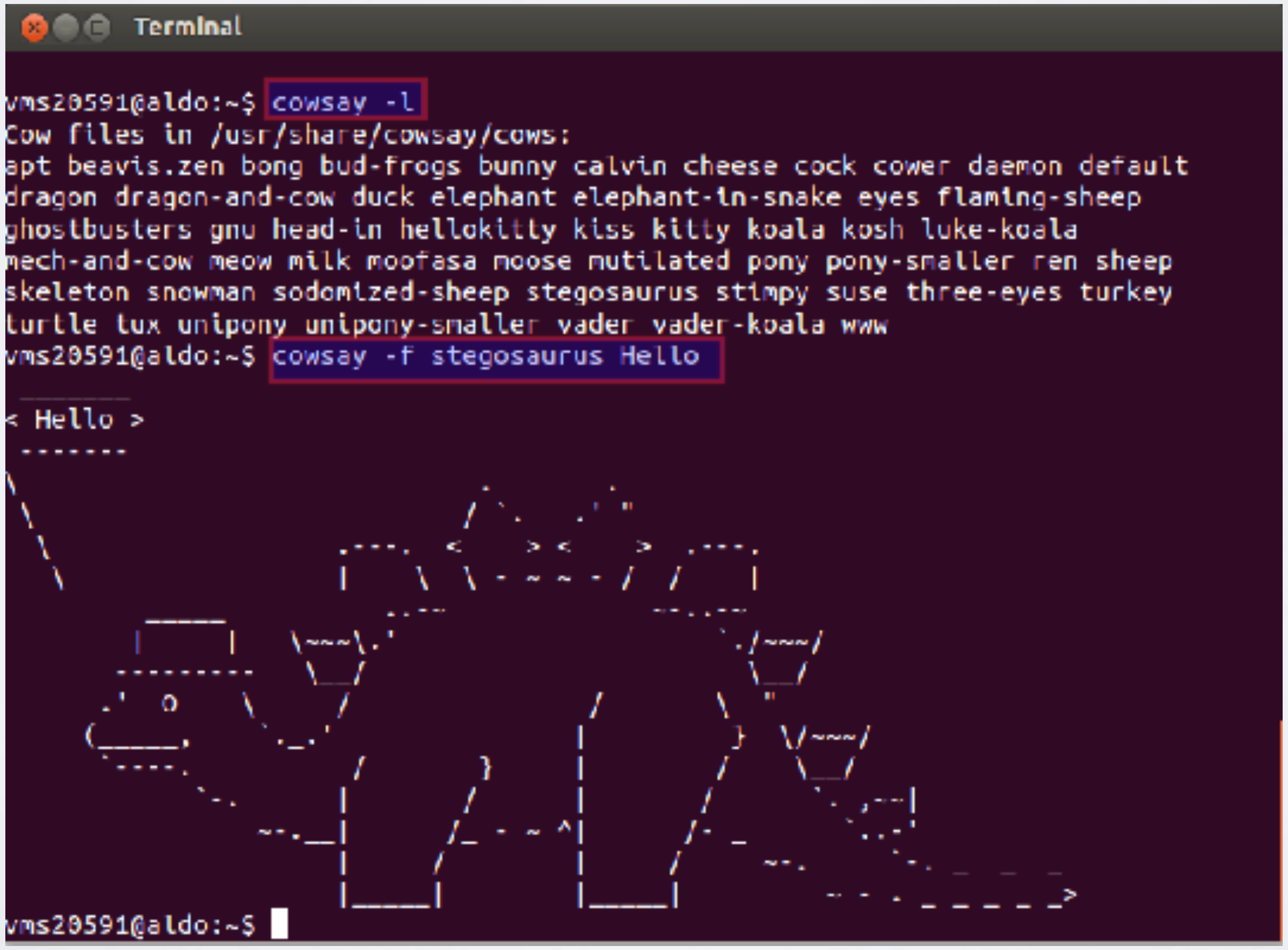
- **Mac**
  - Install Homebrew: https://brew.sh
  - Install Git (after installing Homebrew, type "brew install git").
- **Windows**
  - Install Git Bash: https://git-for-windows.github.io

HOMEBREW

- Package manager for MacOS
- “Homebrew installs **the stuff you need** that Apple didn’t”

COMMAND LINE VS GUI

- There was a time when computers didn't come with a graphical user interface (GUI)
- Interaction using text commands through command line interface (CLI)



```
vms20591@aldo:~$ cowsay -l
Cow files in /usr/share/cowsay/cows:
apt beavis.zen bong bud-frogs bunny calvin cheese cock cower daemon default
dragon dragon-and-cow duck elephant elephant-in-snake eyes flaming-sheep
ghostbusters gnu head-in hellokitty kiss kitty koala kosh luke-koala
mech-and-cow meow milk moofasa moose mutilated pony pony-smaller ren sheep
skeleton snowman sodomized-sheep stegosaurus stimpy suse three-eyes turkey
turtle tux unipony unipony-smaller vader vader-koala www
vms20591@aldo:~$ cowsay -f stegosaurus Hello
 _______
< Hello >
 -------
\                             .       .
 \                           / `.   .' "
  \                  .---.  <    > <    >  .---.
   \                 |    \  \ - ~ ~ - /  /    |
         _____          ..-~             ~-..-~
        |     |   \~~~\.'                    `./~~~/
       ---------   \__/                        \__/
      .'  O    \     /               /       \  "
     (_____,    `._.'               |         }  \/~~~/
      `----.          /       }     |        /    \__/
            `-.      |       /      |       /      `. ,~~|
                ~-.__|      /_ - ~ ^|      /- _      `..-'
                     |     /        |     /     ~-.     `-. _  _  _
                     |_____|        |_____|         ~ - . _ _ _ _ _>
vms20591@aldo:~$
```

WHY COMMAND LINE?

- Everything you can do in a windowed environment, you can do in the terminal, FASTER!
- Finding files, installing packages, web browsing (for example lynx package)

WHAT IS SHELL?

- A type of command-line program that contains a simple, text-based user interface
- Accepts text as an input and translates it into the appropriate functions you want your computer to run
- Mac shell (terminal) is unix based
- Windows equivalents: *Cygwin*, *Git Bash*
- Just for fun: http://hackertyper.com

GIT BASH

- Provides a BASH emulation used to run Git from command line
- Bourne again shell (Bash) is a free Unix shell
- A Shell that allows you to run Unix commands on a Windows device

WHY UNIX?

- A family of multitasking, multiuser computer operating systems
- Developed in 1960 in AT&T Bell Labs
- Written in C and Assembly
- Flexible and more efficient
- Popular within programmer communities

COMMAND LINE (TERMINAL) COMMON TERMINOLOGIES

ABSOLUTE PATH

- **Absolute path:** specific location of a file or folder as accessed from the root directory
- **Root directory:** starting point from which all other folders are defined (typically shown as **/.**)
- **Home directory:** Usually not the same as your root directory (/Users/[Your Username])
- **Example:** */Users/phesami/Documents/General_Assemply_Teach/your-development-environment*

UNIX COMMANDS AND FILE PATHS

- **cd** — a command for "change directory" — with no parameters takes us to our home directory
- **pwd** — a command for "print working directory" — gives you the absolute path of your current location

RELATIVE PATH

- A reference to a file or folder **relative** to your current position

- If we are in the folder `/a/b/` and we want to open the file that has the absolute path `/a/b/c/file.txt`, we can simply type:

```
$open c/file.txt
```

- Absolute or Relative? **./c/file.txt**

GENERAL FORMAT FOR COMMANDS

<command> -<options> <arguments>

- `<command>` is the action we want the computer to take
- `<options>` (or "flags") modify the behavior of the command
- `<arguments>` are the things we want the command to act on

- Example: **open** **c/file.txt**

HANDS ON PRACTICE WITH COMMAND LINE

- Open your command line: terminal **(MAC) and GIT BASH (windows)**
- Go to your home directory: **cd ~ (or cd)**
- Navigate to your desktop: **cd ~/Desktop**

HANDS ON PRACTICE WITH COMMAND LINE

- List all files and directories in the current folder: **ls**
- List all the files and directories ion your desktop: **ls ~/Desktop**
- Navigate to your Desktop: **cd ~/Desktop**
- Creating a new directory called session2: **mkdir session2**
- Navigate to session2: **cd session2**
- Create a new text file called file1: **touch file1.txt**
- Remove file1: **rm file1.txt**
- Remove directory session2?

WILDCARDS

- Wildcard (symbol: *) is useful for operating on multiple files

- Execute the following commands:

  - **mkdir** ~/Desktop/session2
  - **cd ~/Desktop/session2**
  - **touch cat.txt**
  - **touch dog.txt**
  - **touch bird.txt**
  - **touch fish.txt**

WILDCARDS

- List any file with "i" in the file name: **ls *i***
- Remove any file with “d" in the file name: **rm *d***
- Validate your command: **ls**
- Practice: remove all .txt files?

HIDDEN DIRECTORIES

- There are hidden directories all over your file system — mainly to save you from yourself
- To see them: **ls -lha**

- You might need to modify some of hidden git files in future

EDITING AND EXAMINING FILES

- Minor changes to files (minor code change) can be accomplished through terminal using editor *nano*
- Download the file: SampleTextFile.txt from github/slack and copy it into **~/ Desktop/session2**
- Go to ~/Desktop/session2 (cd command)
- Open the file in terminal: **nano SampleTextFile.txt**
  - **ctrl-w**: Search within file
  - **ctrl-o:** Save file as [filename]
  - **ctrl-x**: Exit editor

ECHO FILE CONTENT TO THE TERMINAL

- To view the content of file as text: **cat SampleTextFile.txt (**Or: **cat /etc/passwd)**
- First few lines: **head SampleTextFile.txt**
- Last few lines: **tail SampleTextFile.txt**
- Specify number of lines: **head -n 12 SampleTextFile.txt**

SEARCHING INSIDE FILES: `GREP`

- Search within files and traverse within subdirectories
- Find all files with the word "the" inside: **`grep -r "the" *`**
- Omitting **`-r`** will cause **`grep`** to only look within the current subdirectory
- Using **`-i`** will make **`grep`** ignore the casing of characters

FINDING FILES

- The most useful operation from the terminal is finding files: **locate**
  - Finding Specific File(s) Within the Entire System: **locate** **`nanorc`**
- The **find** command will find files relative to the current working directory but needs to be used in conjunction with a pipe operation
  - Finding All Text Files Within Subdirectories of the Current Working Directory: **find** . | **grep** txt

TRICK: COUNTING THE NUMBER OF LINES IN A FILE

- Find the number of lines in a file:¶

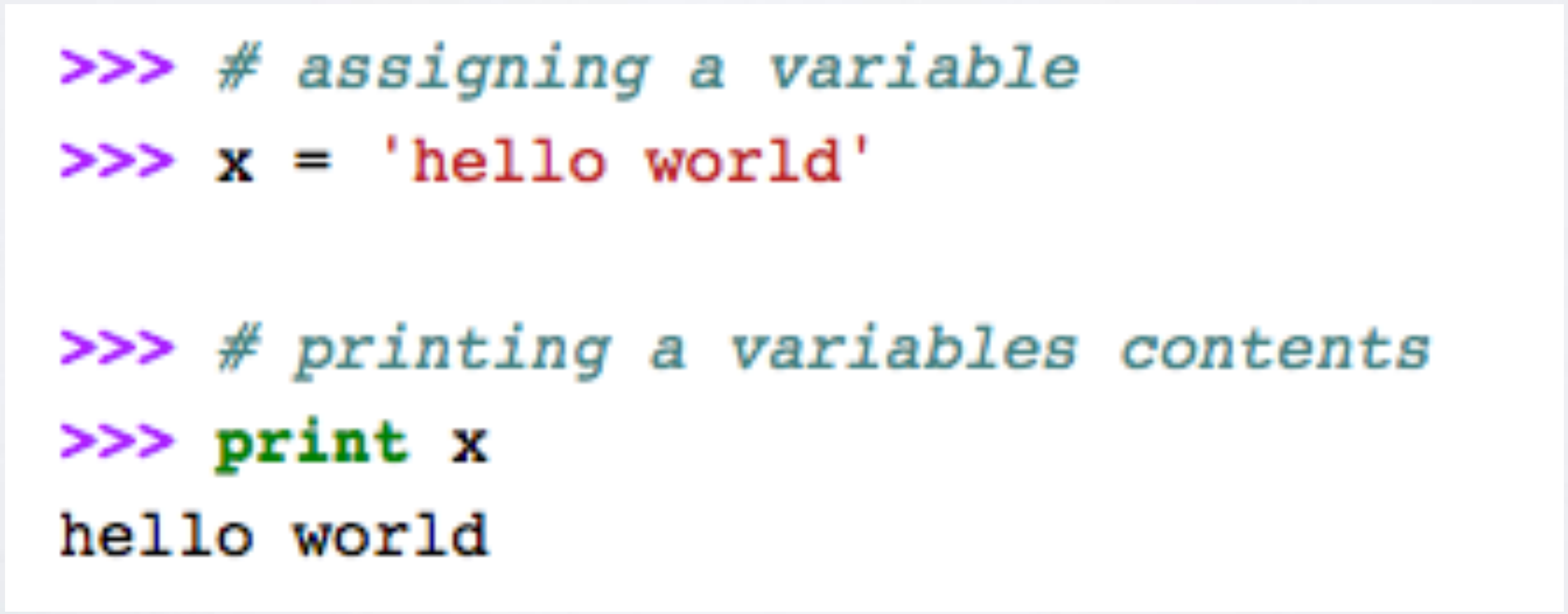
```
cat /etc/passwd | wc -l
```

- Find the number of words in a file:¶

```
cat /etc/test.txt | wc -w
```

INTRO TO DEVELOPMENT ENVIRONMENTS

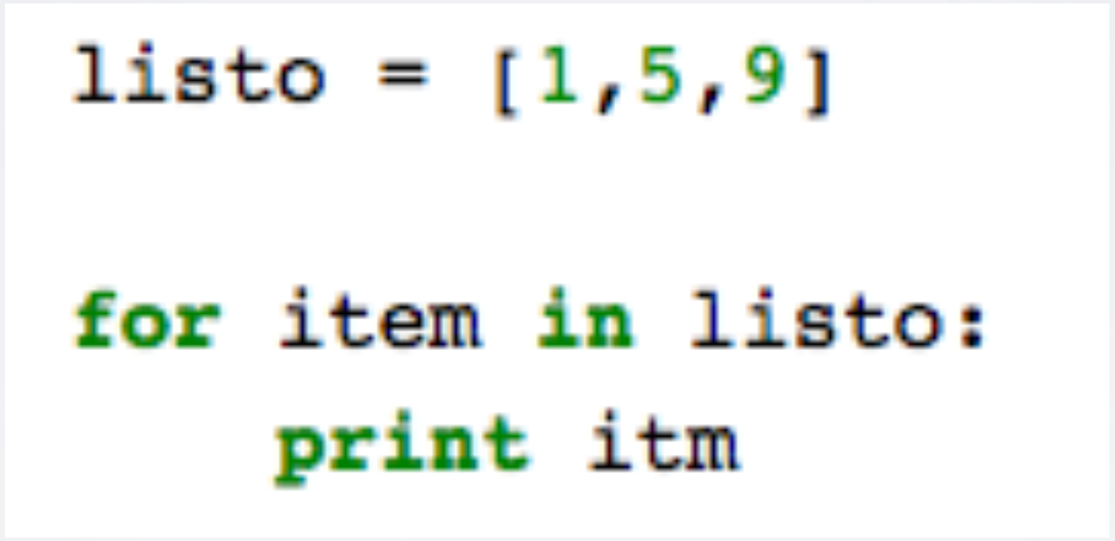
- In addition to command line, we can also execute commands in a variety of languages like python, Java and Git in terminal/command line as well
- In your terminal type: **python**
- Now execute the following commands:



```
>>> # assigning a variable
>>> x = 'hello world'

>>> # printing a variables contents
>>> print x
hello world
```

INTRO TO DEVELOPMENT ENVIRONMENTS

- No developer ever writes scripts in the command line. Why?
- Writing and trouble shooting a lot of code in the terminal can be tedious



```
listo = [1,5,9]

for item in listo:
    print itm
```

- An error in the second line of the `for loop!` But we still have to rewrite the entire loop and we can't go back and just edit out mistake inline

INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

- A program that provides an all-in-one environment to programmers
- Instead of writing your code in a text editor, and executing it in a command line window

COMMON ENVIRONMENTS FOR DATA SCIENCE

- The Anaconda package manager we installed earlier comes with two useful Python-based development environments (IDE): **`Spyder`** and **`Jupyter`**.

- A common third-party environment is **`PyCharm`**.

JUPYTER NOTEBOOKS

- Jupyter uses cell based execution —> you can run all the code in a cell simultaneously
- It also has markdown and slide show integration (publishable results)
- Jupyter Notebooks open in your default browser from the command line by executing **jupyter notebook**

DEMO

SPYDER IDE

- Spyder has a selection-based execution —> you can run all the code that you have *selected* simultaneously

- Very similar to R studio

- It is a desktop software that opens in its own window. It can be opened from the command line by executing **spyder**

DEMO

PYCHARM

- An excellent fully-featured commercial IDE for writing Python code files
- Free community edition
- Features: debugging capabilities, intelligent code refactoring, and integration with Git

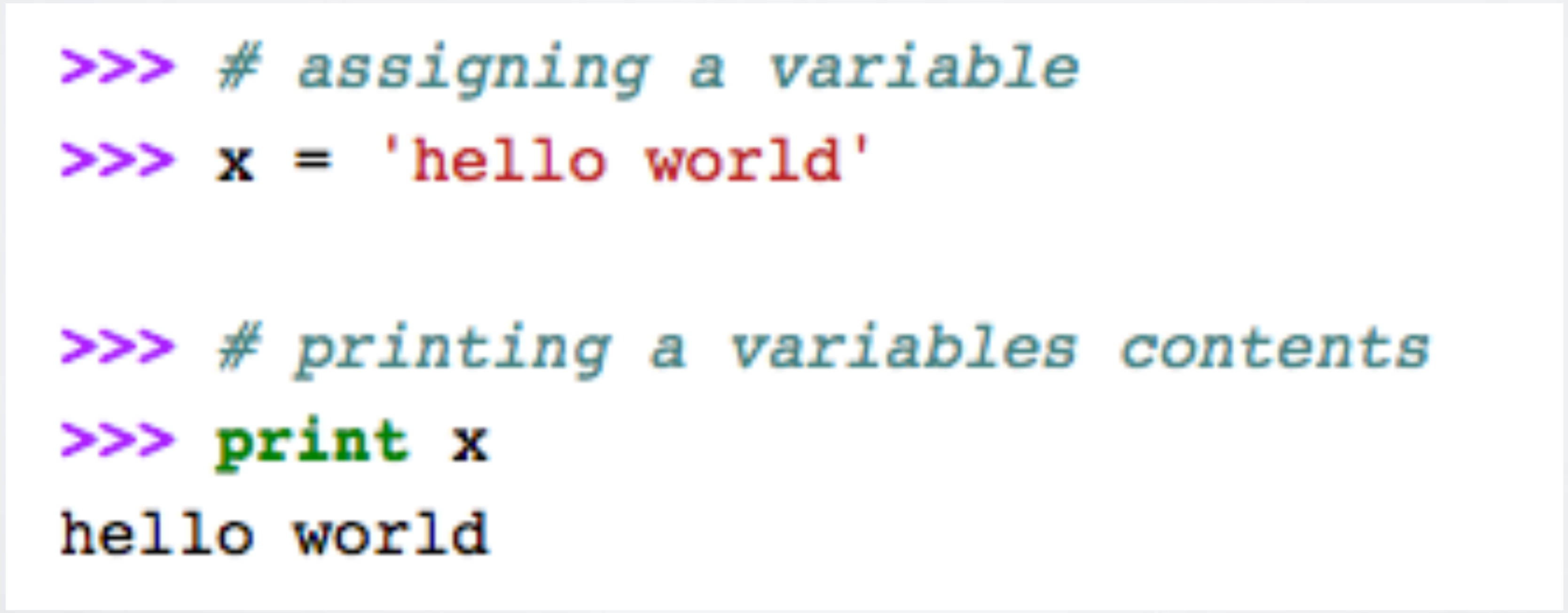
DEMO

TEXT EDITOR BASED IDE

- In addition to IDEs, some developers also use text editors to create or edit code and files
- More commonly used for files that are executed via the command line
- Some common text editors that you may see or use include (install at least one)
    - Sublime
    - Atom
    - Notepad++ (Windows)
    - Vim

PRACTICE

- Open up a Jupyter Notebook: **Jupyter notebook**
- Execute the following codes in two different cells:



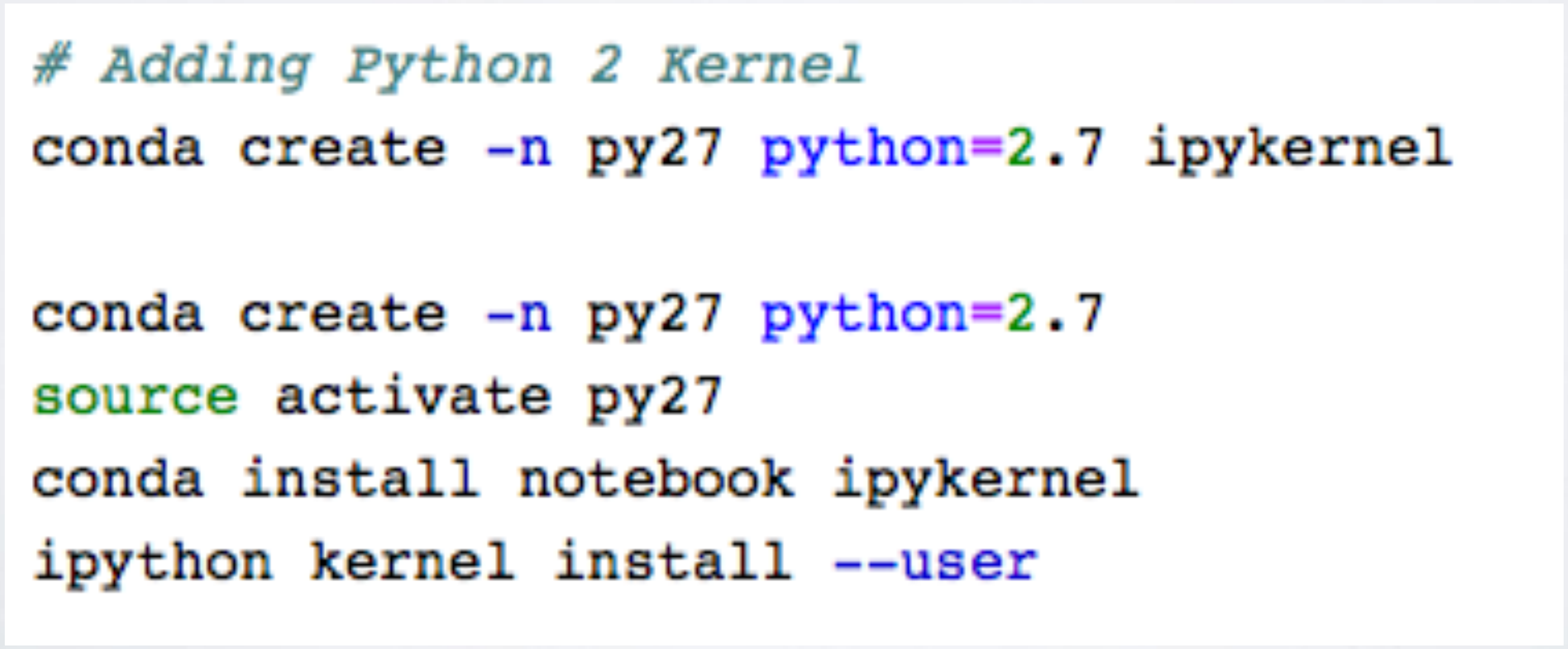
```
>>> # assigning a variable
>>> x = 'hello world'

>>> # printing a variables contents
>>> print x
hello world
```

PYTHON VERSION

- In your terminal type : **python -V**
- Check the upper right hand corner of your Jupyter Notebook as well
- We will be using Python 2.7

PYTHON VERSION

- Only if you have both python 2 and python 3 and python 3 is not showing up in your Jupiter notebook:
  - Create a python 2 kernel which you can select when opening a notebook



```
# Adding Python 2 Kernel
conda create -n py27 python=2.7 ipykernel

conda create -n py27 python=2.7
source activate py27
conda install notebook ipykernel
ipython kernel install --user
```

SUMMARY - PART I

- What is shell, command line and how to execute commands through them
- Python in command line
- Python IDEs

PRACTICE

- Download ipynb_practice1.ipynb from slack/github
- Copy it into your desired directory
- Launch jupyter notebook from that directory and open the .ipynb file

PART II- GIT AND GITHUB

LEARNING OBJECTIVES- PART II

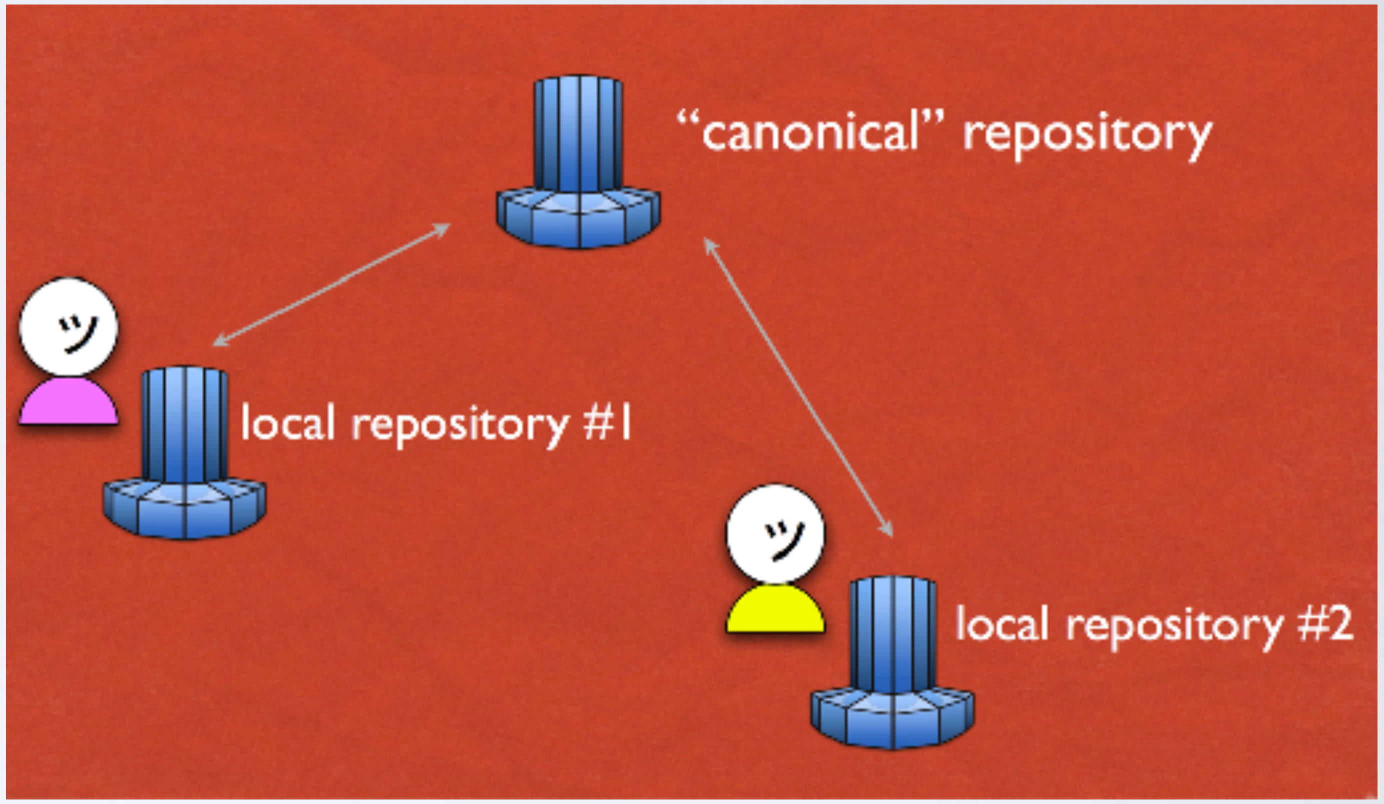
- **Introduction to Git**

After this lesson, you will be able to:

- Use and explain common Git commands, including init, add, commit, push, pull, and clone.
- Distinguish between local and remote repositories.
- Create, copy, and delete repositories locally or on GitHub.
- Clone remote repositories.
- Establish Secure Shell connections to remote repositories.

VERSION CONTROL SOFTWARES

- What is version control?
- Why version control SW?
    - Collaboration
    - Version tracking
- Examples: Git (distributed), SVN (centralized)
- Git/Github is popular, flexible, and cheap


"canonical" repository
local repository #1
local repository #2