DAT12SYD

Session 2
May 2018

Agenda

1. Recap
2. Python Fundamentals
3. Terminal Ninja
4. Github How-to

Recap

WiFi
GA-Guest
yellowpencil

Recap

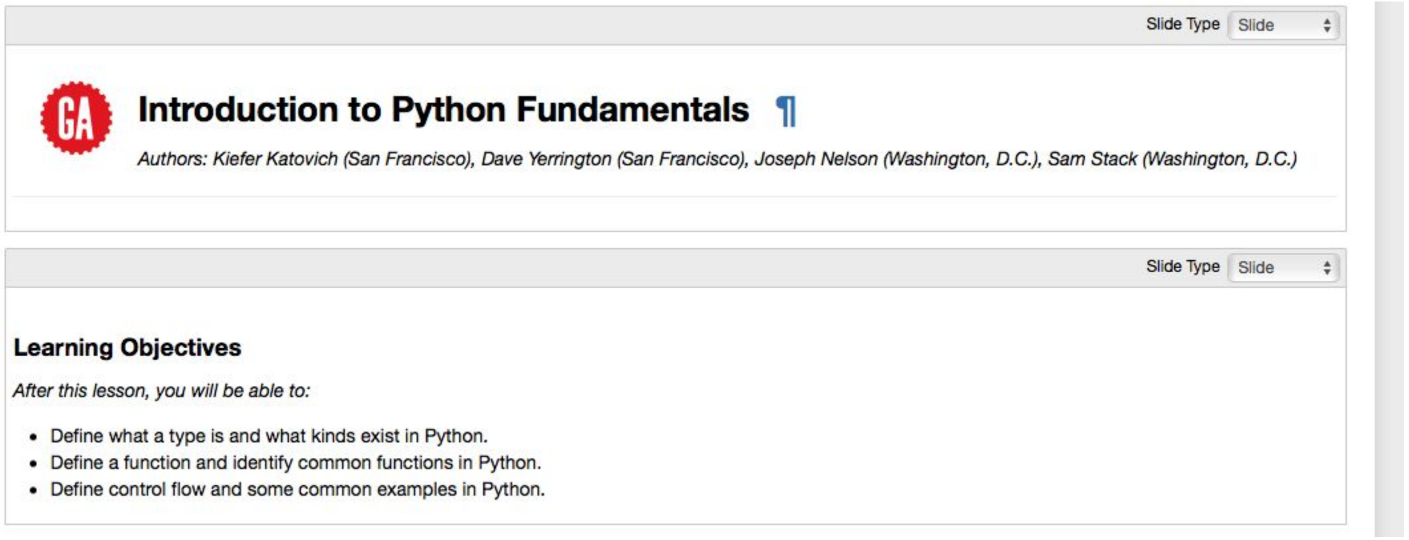
Install the following software for your OS:

- Install Ananconda 3.6 (https://www.anaconda.com/download/)
- Install Git (https://git-scm.com/downloads)
- Install sublime text 3 (https://www.sublimetext.com/3)
- Make a github account (https://github.com)
- Make a Slack account (https://slack.com/, @dat12syd.slack.com)

Python Fundamentals

Python Fundamentals

intro_to_python_fundamentals_v2.ipynb



Slide Type: Slide

Introduction to Python Fundamentals ¶

Authors: Kiefer Katovich (San Francisco), Dave Yerrington (San Francisco), Joseph Nelson (Washington, D.C.), Sam Stack (Washington, D.C.)

Slide Type: Slide

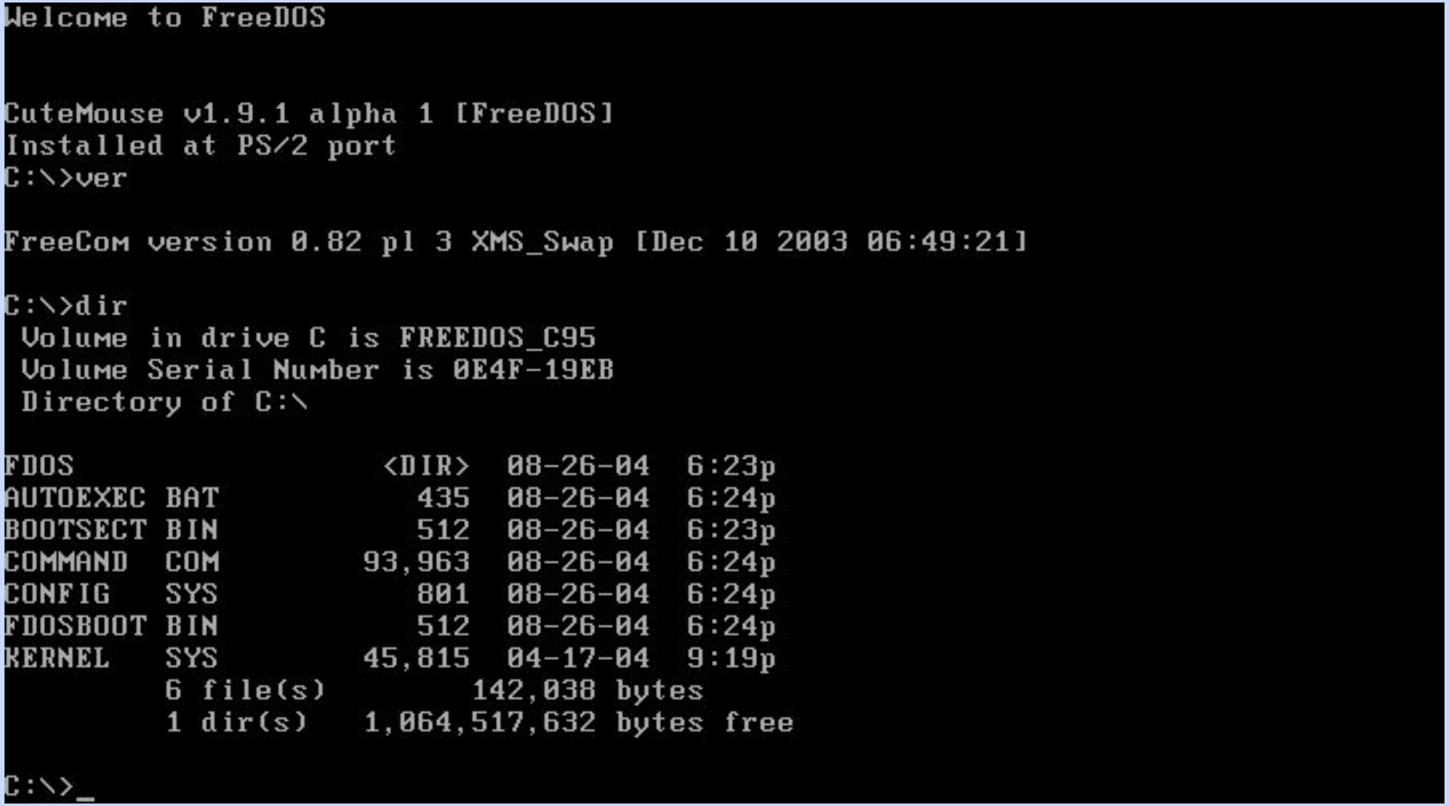
Learning Objectives

After this lesson, you will be able to:

- Define what a type is and what kinds exist in Python.
- Define a function and identify common functions in Python.
- Define control flow and some common examples in Python.

Terminal 101

What is a GUI?



```
Welcome to FreeDOS


CuteMouse v1.9.1 alpha 1 [FreeDOS]
Installed at PS/2 port
C:\>ver

FreeCom version 0.82 pl 3 XMS_Swap [Dec 10 2003 06:49:21]

C:\>dir
 Volume in drive C is FREEDOS_C95
 Volume Serial Number is 0E4F-19EB
 Directory of C:\

FDOS                 <DIR>  08-26-04  6:23p
AUTOEXEC BAT           435  08-26-04  6:24p
BOOTSECT BIN           512  08-26-04  6:23p
COMMAND  COM        93,963  08-26-04  6:24p
CONFIG   SYS           801  08-26-04  6:24p
FDOSBOOT BIN           512  08-26-04  6:24p
KERNEL   SYS        45,815  04-17-04  9:19p
         6 file(s)        142,038 bytes
         1 dir(s)   1,064,517,632 bytes free

C:\>_
```

Why Terminal?

Lightening *FAST*

- Running processes
- Finding files
- Substring match of file contents
- Assessing performance
- Remote operations
- Web browsing
- Installing packages
- Managing your development environment

What is a Shell?

A command line / terminal that contains a simple text-based interface that allows us to access the systems services.

*Windows Users

Git Bash

Windows command prompt

Windows PowerShell

Windows Subsystem for Linux

Why? Since UNIX is the prefered operating system for programmers and developers alike, almost all of the lessons at GA have been written for interaction with UNIX.

MacOS

Use spotlight search to open terminal

⌘ (Command) + Space

Type “Term…”

Push Enter

Paths

Absolute Paths

Relative Paths

Working with Paths

```
$ cd

$ pwd
```

Navigating using Terminal

Changing directories

Listing files

Creating directories and files

Removing files

Navigating using Terminal

cd ~

Or just cd + enter

Used for shortening long absolute paths

Try cd ~/Desktop

Navigating using Terminal

$ ls lists files and directories in current working directory

$ ls /Applications (MacOS)

Navigating using Terminal

Make a new directory with $ mkdir

Create a new file $ touch myfile.txt

Remove file $ rm myfile.txt

Navigating using Terminal

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.

Using Wildcards in the Command Prompt

The wildcard symbol (*) is used to operate on multiple files. Example: create a folder on your desktop and add some files.

```
mkdir ~/Desktop/example_folder
cd ~/Desktop/example_folder
touch cat.txt
touch dog.txt
touch bird.txt
touch fish.txt
```

You can then use the wildcard * to operate on subsets of files. List any file with "i" in the file name, for example:

```
ls *i*
```

Or, remove any file with "d":

```
rm *d*
ls
```

Hidden Directories Can Also Be Found

There are hidden directories all over your file system — mainly to save you from youself. Using the parameters -lha to ls, we can find these directories.

```
ls -lha
```

-l:	One entry per line.
-h:	When used with the -l option, use unit suffixes: byte, kilobyte, megabyte, etc.
-a:	Include directory entries whose names begin with a dot (.).

Editing and Examining Files

At times it's helpful to edit files in a pinch. We can accomplish this by using the terminal editor nano.

Use the following syntax to edit files from the terminal with nano:

```
nano [filename]
```

These hotkeys are available:

- ctrl-w: Search within file.
- ctrl-o: Save file as [filename].
- ctrl-x: Exit editor.

The bottom of the editor contains the most common operations.

Echo File Content to the Terminal

$ Cat, head and tail

$ cat is used to quickly display short files.
Also to concatenate files

```
cat /etc/passwd
```

Echo File Content to the Terminal

```
$ head -n 10 myfile.txt

$ tail -n 2 myfile.txt
```

```
# $ tail -f myfile.txt
import time

for i in range(0,1000):
        with open('myfile.txt', 'a+') as myfile:
        myfile.write('hello'+str(i)+'\n')

        time.sleep(0.5)
```

Grep

Searching Inside Files: grep

The grep command will 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, but at the expense of efficiency.

Finding Files

$ locate finds files all over your file system.
$ find command will find files relative to the current working directory, needs to be used with a pipe.
Finding All Notebook Files Within Subdirectories of the Current Working Directory

```
$ find . | grep ipynb
```

Finding Specific File(s) Within the Entire System

```
$ locate nanorc
```

Finding Specific File(s) With a Substring Match

```
$ locate log
```

Counting Lines in a File

wc -l

```
$ wc -l /etc/passwd
```

wc -w

```
$ wc -w /etc/passwd
```

Here's some optional (but highly recommended) reading about pipe and I/O redirection on the command line:

- I/O redirection
- Good examples of piping commands together

Terminal 101 - Commands

```
[tab]     # Autocomplete command
pwd       # Where am I? The programmer's "um"
ls        # List all files in the current directory
cd        # Change Directories
mkdir     # Make a Directory
rmdir     # Remove an empty directory
rm        # Remove a file or a directory [There is no undo]
touch     # Create a file
open      # Open a file in the default application
code      # Open the VSCode Editor (atom will open in Atom)
say       # Make your computer talk
```

Advanced Commands

```
ls -lsa                        # long format, system blocks, view hidden files (.)

Ls -lt                         # long format, sort by time modified (most recently
modified first)

ps aux | grep <keyword>        # process status, all users, usernames, even those
without controlling terminal, search for keyword e.g. “python”

nano, vim                      # editors
```

Advanced Commands

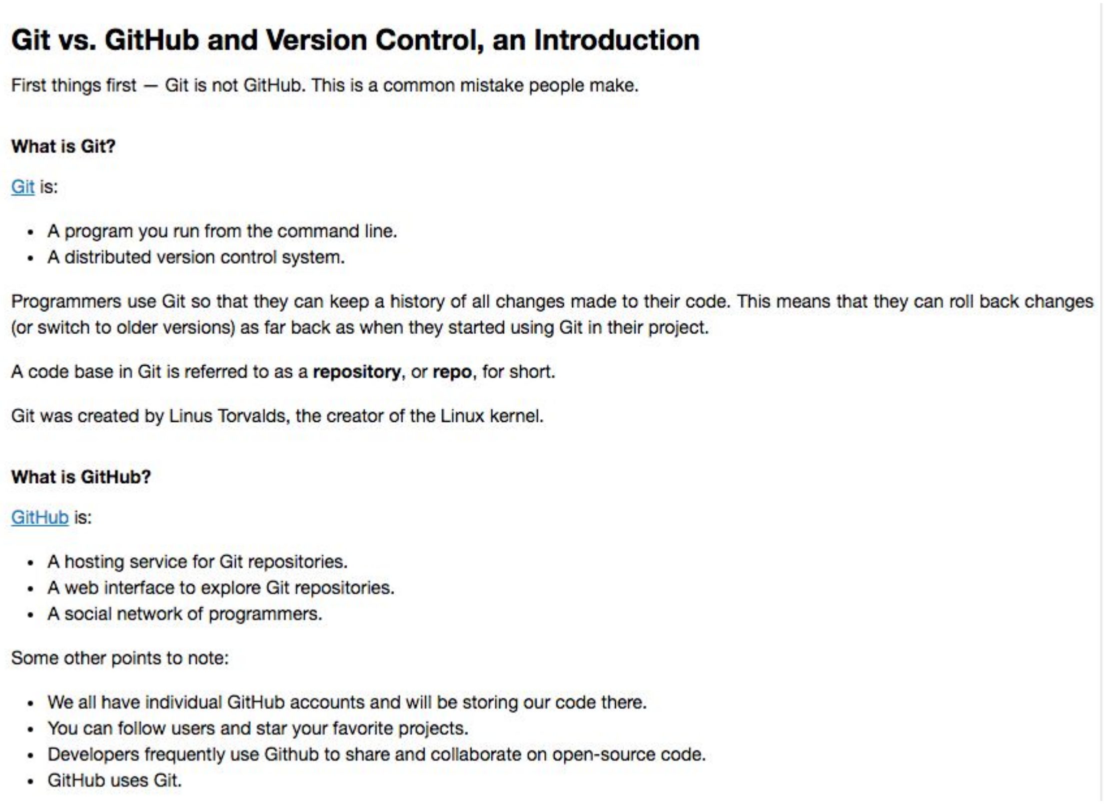
```
wc -l                         # count number of lines in file e.g. a csv

tar czvf, gzip, zip           # compress into archive

tar xzvf, unzip               # decompress file

ssh, scp                      # Secure shell, secure copy

whoami, which <keyword>       # list username, which python
```

Activity - Terminal 101

1. Navigate to your home directory with **cd ~**
2. Use **pwd** to discover its name
3. Use **ls** to see what is in your home directory
4. Use **cd ~** to navigate back down to your home directory
5. Create a new directory with **mkdir** called **sandbox**
6. Navigate to your downloads with **cd ..** or cd **~/Downloads**
7. Create a file in **Downloads** with **touch** called file.txt
8. Copy file.txt to your sandbox with **cp file.txt ~/sandbox/**
9. Rename **file.txt** to **hello.py** with **mv file.txt hello.py**
10. Change directory to **cd ~/sandbox**
11. Make a new file called **fake.py** using **nano fake.py**
12. Inside the file type `print(“hello world!”)`, then push Ctrl+o, Enter, Ctrl+x to save and exit
13. Make a directory called ‘crash_course’ using **mkdir crash_course**
14. Remove **fake.py** and **crash_course** with **rm,** you will need **-f** for one of the removals
15. Well done you’ve finished!

Intro to Git

Intro to Git



Git vs. GitHub and Version Control, an Introduction

First things first — Git is not GitHub. This is a common mistake people make.

What is Git?

Git is:

- A program you run from the command line.
- A distributed version control system.

Programmers use Git so that they can keep a history of all changes made to their code. This means that they can roll back changes (or switch to older versions) as far back as when they started using Git in their project.

A code base in Git is referred to as a **repository**, or **repo**, for short.

Git was created by Linus Torvalds, the creator of the Linux kernel.

What is GitHub?

GitHub is:

- A hosting service for Git repositories.
- A web interface to explore Git repositories.
- A social network of programmers.

Some other points to note:

- We all have individual GitHub accounts and will be storing our code there.
- You can follow users and star your favorite projects.
- Developers frequently use Github to share and collaborate on open-source code.
- GitHub uses Git.

Q&A

EXIT TICKETS

DON’T FORGET TO FILL OUT YOUR EXIT TICKET

Appendix

Fun Activities

Online game http://www.pythonchallenge.com

https://selfdrivingcars.mit.edu/deeptraffic/

http://playground.tensorflow.org/