

# Fastcampus

## 컴퓨터공학 입문 스쿨

### Python Basic\_Day5

# Recap

- function
- list comprehension
- dictionary comprehension
- file I/O

# Index

- Error Handling
- Shell Command
- SCM
- git, github

# Mini Project

- List comprehension 으로 FizzBuzz 한줄로 구현하기

```
["Fizz"*(not i%3) + "Buzz"*(not i%5) or i for i in  
range(1,100)]
```

# Ethiopian Multiplication

2로 나누고 곱하는 과정으로 두 수의 곱을 구현하는 방법

[https://en.wikipedia.org/wiki/Ancient\\_Egyptian\\_multiplication](https://en.wikipedia.org/wiki/Ancient_Egyptian_multiplication)

|     |    |      |        |      |
|-----|----|------|--------|------|
| 12  | *  | 7    | struck | ---- |
| 6   |    | 14   | struck | ---- |
| 3   |    | 28   | keep   | 28   |
| 1   |    | 56   | keep   | 56   |
| --> | 28 | + 56 | = 84   |      |

# Ethiopian Multiplication

```
numbers = str(input("two nums with space: ")).split()  
  
result = 0  
num1 = int(numbers[0])  
num2 = int(numbers[1])
```

# Ethiopian Multiplication

```
while num1 >= 1:  
    if num1 % 2 == 0:  
        print("%4d %7d struck" % (num1, num2))  
    else:  
        print("%4d %7d keep" % (num1, num2))  
        result += num2  
        # result = result + num2  
  
    num1 = num1 // 2  
    num2 = num2 * 2
```

# Ethiopian Multiplication

```
print("The result is ", result)
```

# Error Handle

by using `try, except`

필요한 만큼만 적절히 사용하셔야 합니다 by PEP 8

## Error Handle - Syntax

```
try:  
    실행문  
except:  
    실행문
```

## Error Handle - ValueError

```
try:  
    some_input = int(input("type some number: "))  
except ValueError:  
    print("I said type some NUMBER!!!!")
```

## Error Handle - ValueError

```
try:  
    some_input = int(input("type some number: "))  
except ValueError as e:  
    print("I said type some NUMBER!!!!")  
    print(e)
```

## Error Handle - FileNotFoundError

```
try:  
    f = open('error_example.txt', 'r')  
except FileNotFoundError as e:  
    print(e)  
else:  
    text = f.read()  
    f.close()
```

## Error Handle - Multiple Error

```
try:  
    ...  
except error type 1:  
    ...  
except error type 2:  
    ...
```

## Error Handle - Pass Error

```
try:  
    f = open('error_example.txt', 'r')  
except FileNotFoundError as e:  
    pass  
else:  
    text = f.read()  
    f.close()
```

# Shell Command

# Shell

| 운영체제(OS)에서 사용자의 명령을 해석하여 대신 실행해주는 프로그램

- Unix, Linux

bash(ubuntu, OSX default), sh, zsh, csh, ksh, ...

- Windows

explorer.exe(for GUI Windows)

cmd.exe(for CLI MS-DOS)

## 몇 가지 간단한 Shell 명령어

\$ : Shell 명령어의 시작

```
$ ls  
$ ls -al  
  
$ cd Documents  
$ mkdir css  
$ cd css  
$ mkdir python && cd python
```

## 몇가지 간단한 Shell 명령어

```
$ mkdir python – make directory python  
$ cd python – change directory  
$ cd .. – up to
```

```
$ touch hello.py – create hello.py  
$ exit – terminate shell  
$ mv hello.py /python  
$ cp hello.py /python
```

```
$ python --version  
$ python --help
```

# Vim

Vi IMproved의 약자로 vi 호환 텍스트 에디터

# vim Basic

```
$ vim hello.py
```

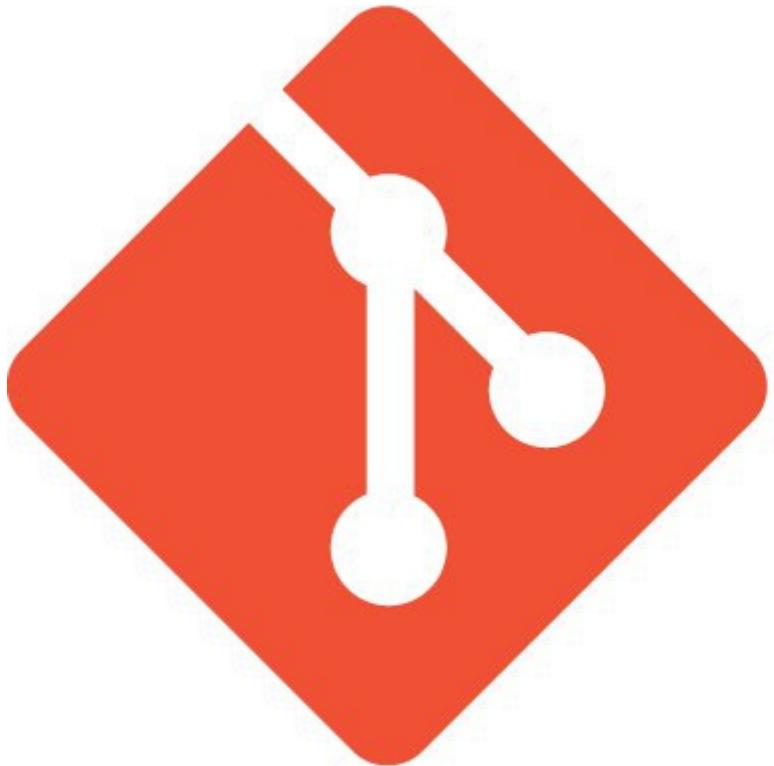
## Command

```
h,j,k,l - cursor
i - insert
v - visual
d - delete
y - yank
p - paste
u - undo
r - replace
$ - move end of line
^ - move start of line

:q - quit
:q! - quit(no warning)
:wq - write and quit

:{number} - move to {number}th line
```

# git Basic



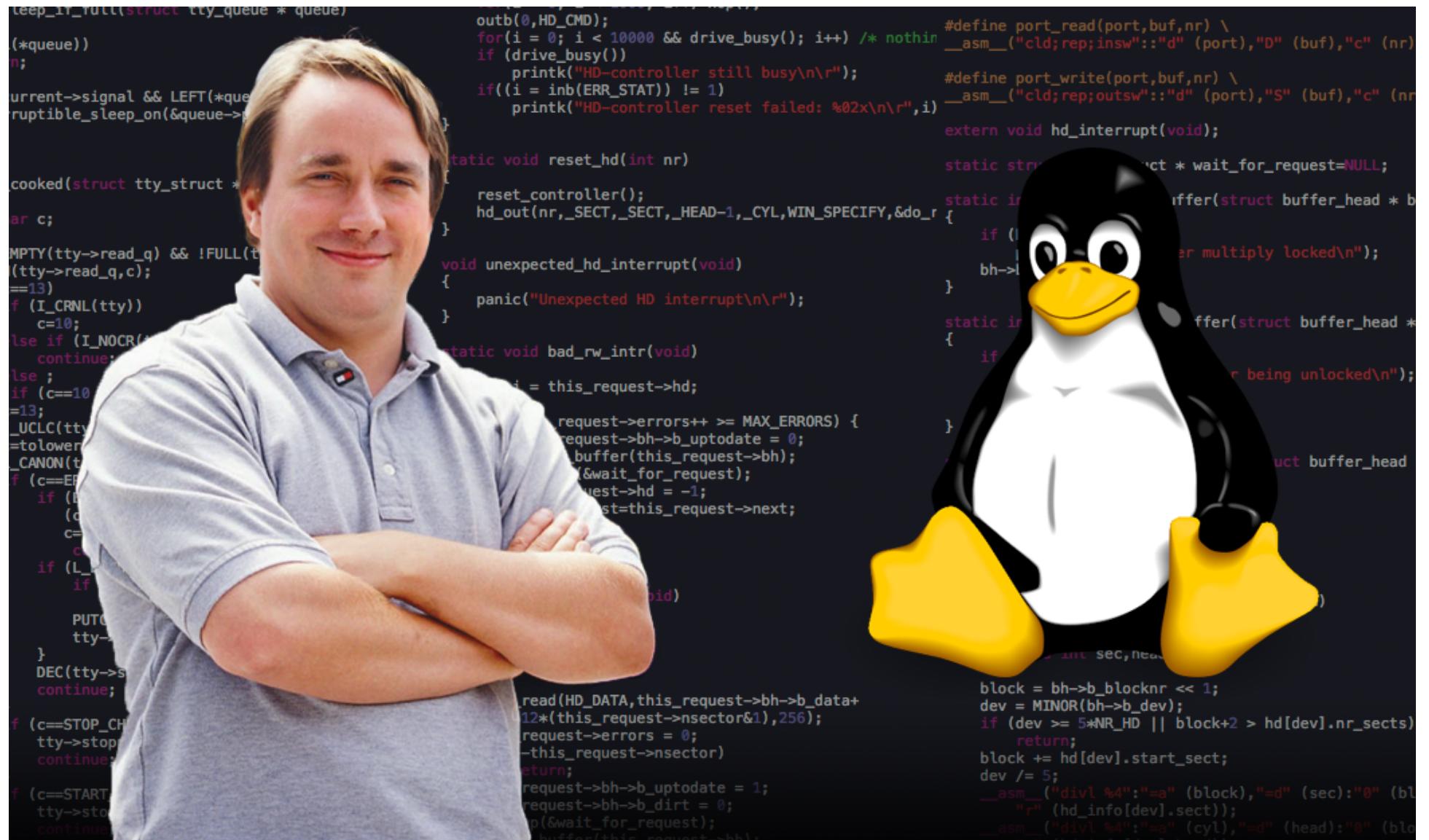
git

# VCS (Version Control System)

== SCM (Source Code Management)

< SCM (Software Configuration Management: 형상관리)

# chronicle of git



## chronicle of git

- Linux Kernel을 만들기 위해 Subversion을 쓰다 화가 난 리누스 토발즈는 2주만에 git이라는 버전관리 시스템을 만듦  
[git official repo](#)

# Characteristics of git

- 빠른속도, 단순한 구조
- 분산형 저장소 지원
- 비선형적 개발(수천개의 브랜치) 가능

## Pros of git

- 중간-발표자료\_최종\_진짜최종\_15-4(교수님이 맘에들어함)\_언제까지??\_이걸로 갑시다.ppt
- 소스코드 주고받기 없이 동시작업이 가능해져 생산성이 증가
- 수정내용은 commit 단위로 관리, 배포 뿐 아니라 원하는 시점으로 Checkout 가능
- 새로운 기능 추가는 Branch로 개발하여 편안한 실험이 가능하며, 성공적으로 개발이 완료되면 Merge하여 반영
- 인터넷이 연결되지 않아도 개발할 수 있음

## Open-source project

<https://github.com/python/cpython>

<https://github.com/tensorflow/tensorflow>

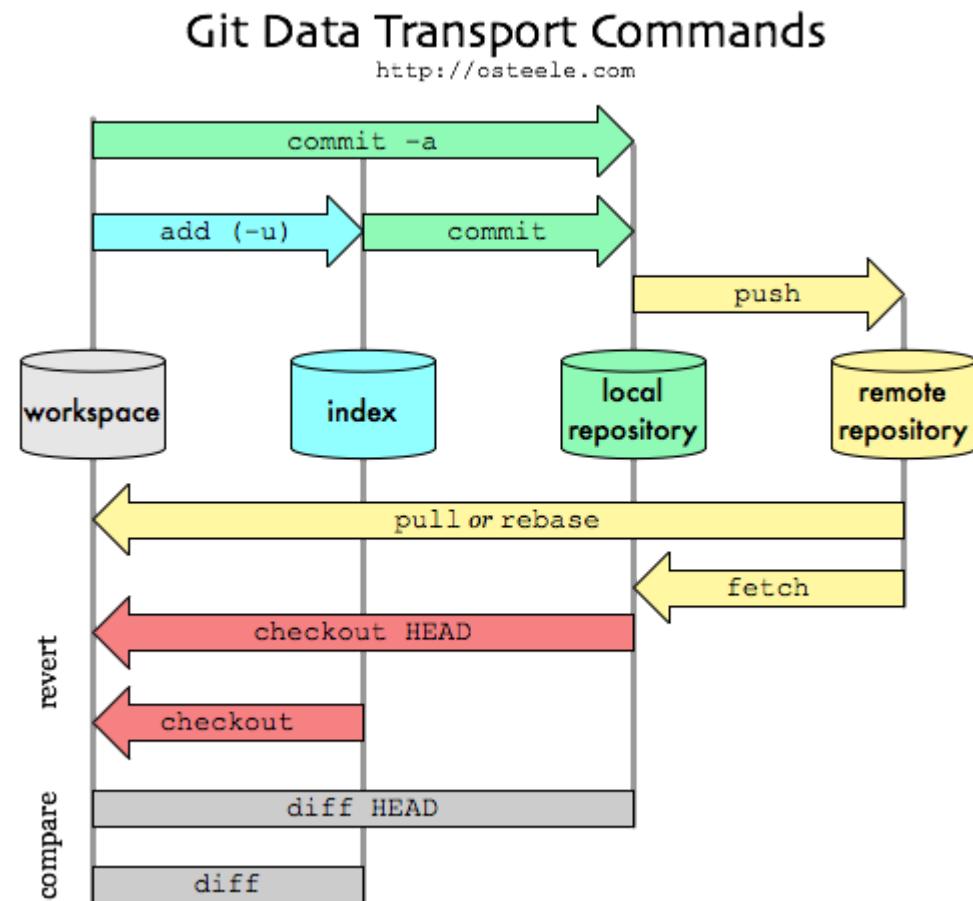
<https://github.com/JuliaLang/julia>

<https://github.com/golang/go>

## git inside

- Blob: 모든 파일이 Blob이라는 단위로 구성
- Tree: Blob(tree)들을 모은 것
- Commit: 파일에 대한 정보들을 모은 것

# git Process and Command



## Useful manager for mac

[http://brew.sh/index\\_ko.html](http://brew.sh/index_ko.html)

## install git

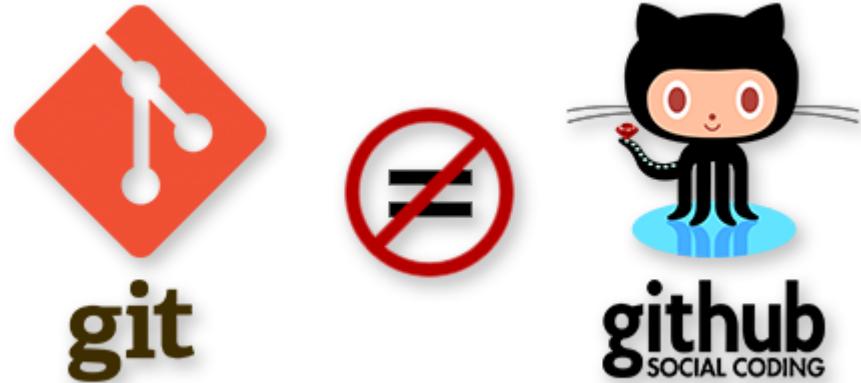
<https://git-scm.com/>

```
// Mac OS  
$ brew install git  
// Linux  
$ sudo apt-get install git
```

- Windows: [install git bash](#)

\$ git --version 으로 정상적으로 설치되었는지를 확인

# git is not equal to github



sign up github

<https://github.com/>

important!!

- 가입할 email 과 username 은 멋지게
- private repo를 원한다면 \$7/month

# Important github User Interface

Star



watch



# Set configuration

terminal

```
$ git config --global user.name "{{username}}"
$ git config --global user.email "{{github email address}}"
$ git config --list
```

# My First Repo

Let's make your first repo with github

# My First Repo

```
$ git init  
$ git add .  
$ git commit -m "some commit"
```

After create new repo through github,

```
$ git remote add origin  
https://github.com/{{username}}/{{repo}}.git  
$ git push origin master
```