

**Fastcampus**

**Computer Science School**

**Python Basic\_Day4**



# Recursive

```
times = int(input("How many times want to curse the beast??: "))
def recurse_beast(a):
    if a == 0:
        print("curse complete!")
    else:
        print("Fusion!!!(%d times left)" % a - 1)
        recurse_beast(a-1)

recurse_beast(times)
```

# List Comprehension

존재하는 리스트를 활용하여 새로운 리스트를 생성하는 방법

비슷한 표현들

- Set Comprehension
- Dictionary Comprehension
- Parallel list Comprehension

# List Comprehension

```
old_list = [1, 2, 3, 4, 5,]  
  
doubled_list = []  
for i in old_list:  
    doubled_list.append(i * 2)
```

# List Comprehension

```
old_list = [1, 2, 3, 4, 5,]  
  
doubled_list = []  
for i in old_list:  
    doubled_list.append(i * 2)
```

```
doubled_list = []
```

# List Comprehension

```
old_list = [1, 2, 3, 4, 5,]  
  
doubled_list = []  
for i in old_list:  
    doubled_list.append(i * 2)
```

```
doubled_list = [i * 2]
```

# List Comprehension

```
old_list = [1, 2, 3, 4, 5,]  
  
doubled_list = []  
for i in old_list:  
    doubled_list.append(i * 2)
```

```
doubled_list = [i * 2 for i in old_list]
```



## List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
    if i % 2 == 0:
        doubled_list.append(i * 2)
```

## List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
    if i % 2 == 0:
        doubled_list.append(i * 2)
```

```
doubled_list = []
```

## List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
    if i % 2 == 0:
        doubled_list.append(i * 2)
```

```
doubled_list = [i * 2]
```

## List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
    if i % 2 == 0:
        doubled_list.append(i * 2)
```

```
doubled_list = [i * 2 for i in old_list]
```

## List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
    if i % 2 == 0:
        doubled_list.append(i * 2)
```

```
doubled_list = [i * 2 for i in old_list if i % 2 == 0]
```

## Mini Project

- List comprehension 으로 FizzBuzz 한줄로 구현하기

# File I/O

# File I/O

```
f = open(filename, mode)
f.close()
```

mode

r - 읽기모드

w - 쓰기모드

a - 추가모드(파일의 마지막에 새로운 내용을 추가)



## Create New File

```
f = open("Newfile.txt", 'w')  
f.close()
```

## Write text

```
f = open("Newfile.txt", 'a')
for i in range(1,11):
    text = "line %d. \n" % i
    f.write(text)
f.close()
```

## Read text

```
f = open("Newfile.txt", 'r')
text = f.readline()
print(text)
f.close()
```

## Read All text

```
f = open("Newfile.txt", 'r')
while True:
    text = f.readline()
    if not text: break
    print(text)
f.close()
```

## Read All text using readlines

```
f = open("Newfile.txt", 'r')
texts = f.readlines()
for text in texts:
    print(texts)
f.close()
```

## Add text

```
f = open("Newfile.txt", 'a')
for i in range(11, 20):
    text = "New line %d \n" % i
    f.write(text)
f.close()
```

## Get rid of f.close()

```
with open("foo.txt", 'w') as f:  
    f.write("foo is text dummy")
```

# 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()
```

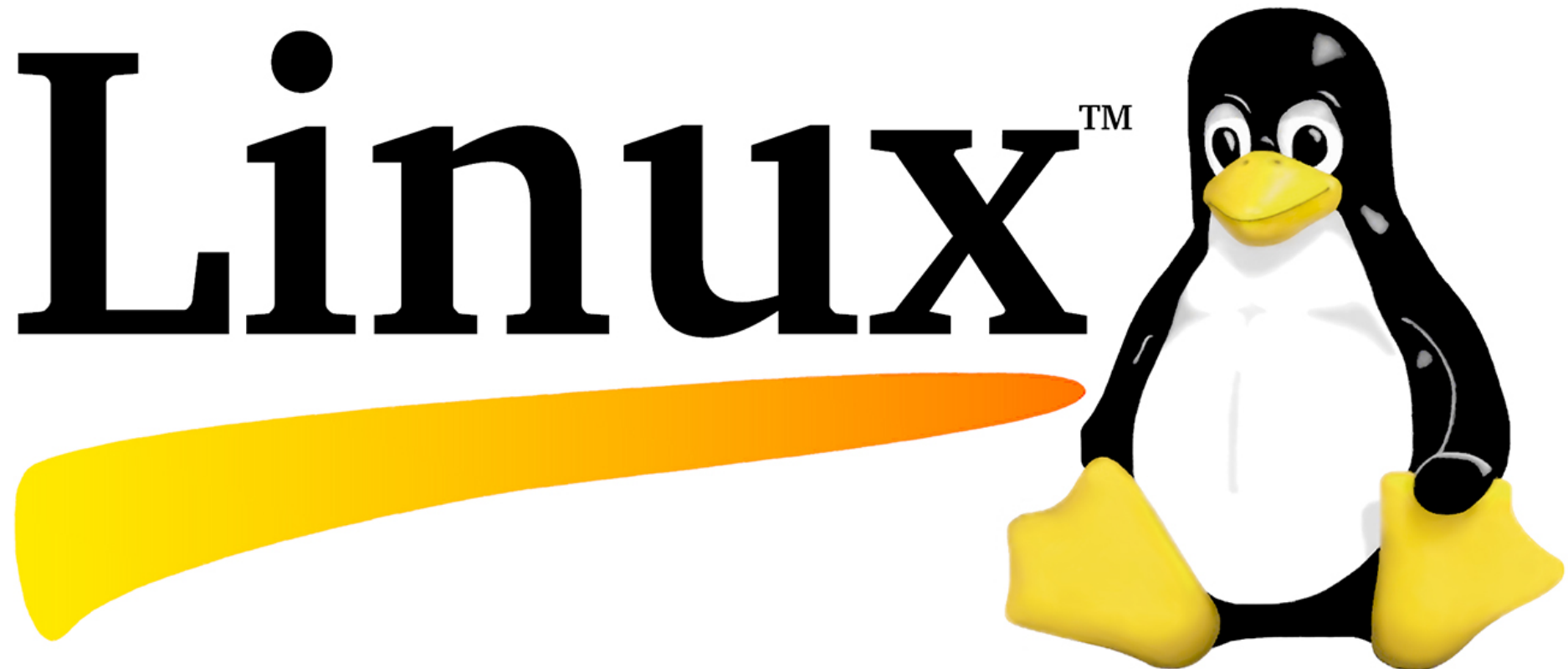
## Error Handle - Multiple Error

```
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()
```

# Linux



## Before Linux



- 1965년 데니스 리치, 켄 톰슨 외 x명이 AT&T Bell 연구소에서 PDP-7 기반 어셈블리어로 작성한 UNIX를 개발

# Before Linux



- 1973년 데니스 리치와 켄 톰슨이 C를 개발한 뒤, C 기반 UNIX 재작성



# Before Linux

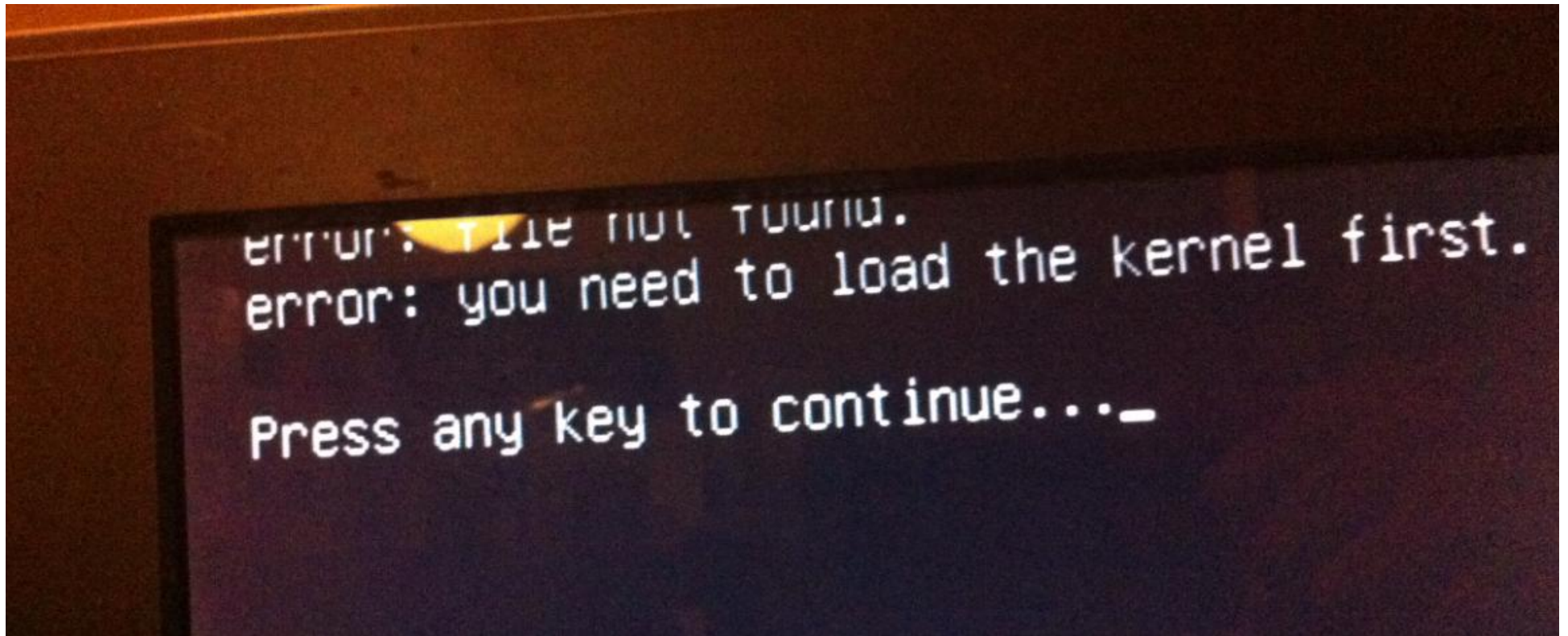


- 1984년 리처드 스톨먼이 오픈 소프트웨어 자유성 확보를 위한 GNU 프로젝트 돌입

## Meaning of GNU

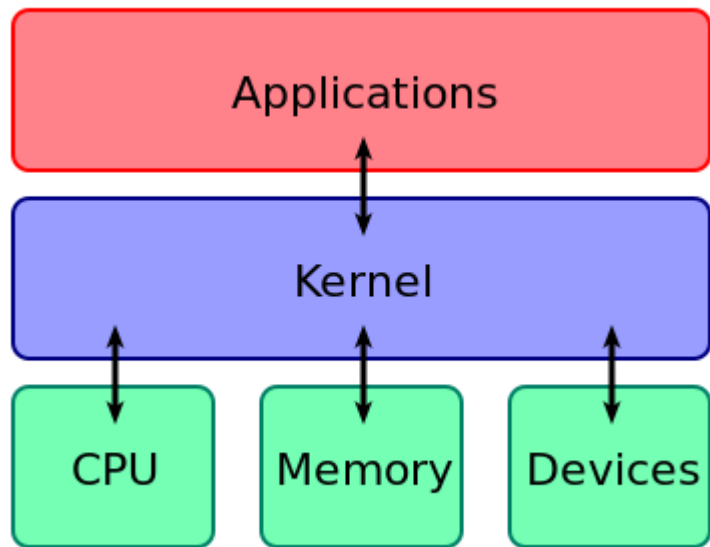
GNU == G NU is N ot U nix

## Before Linux



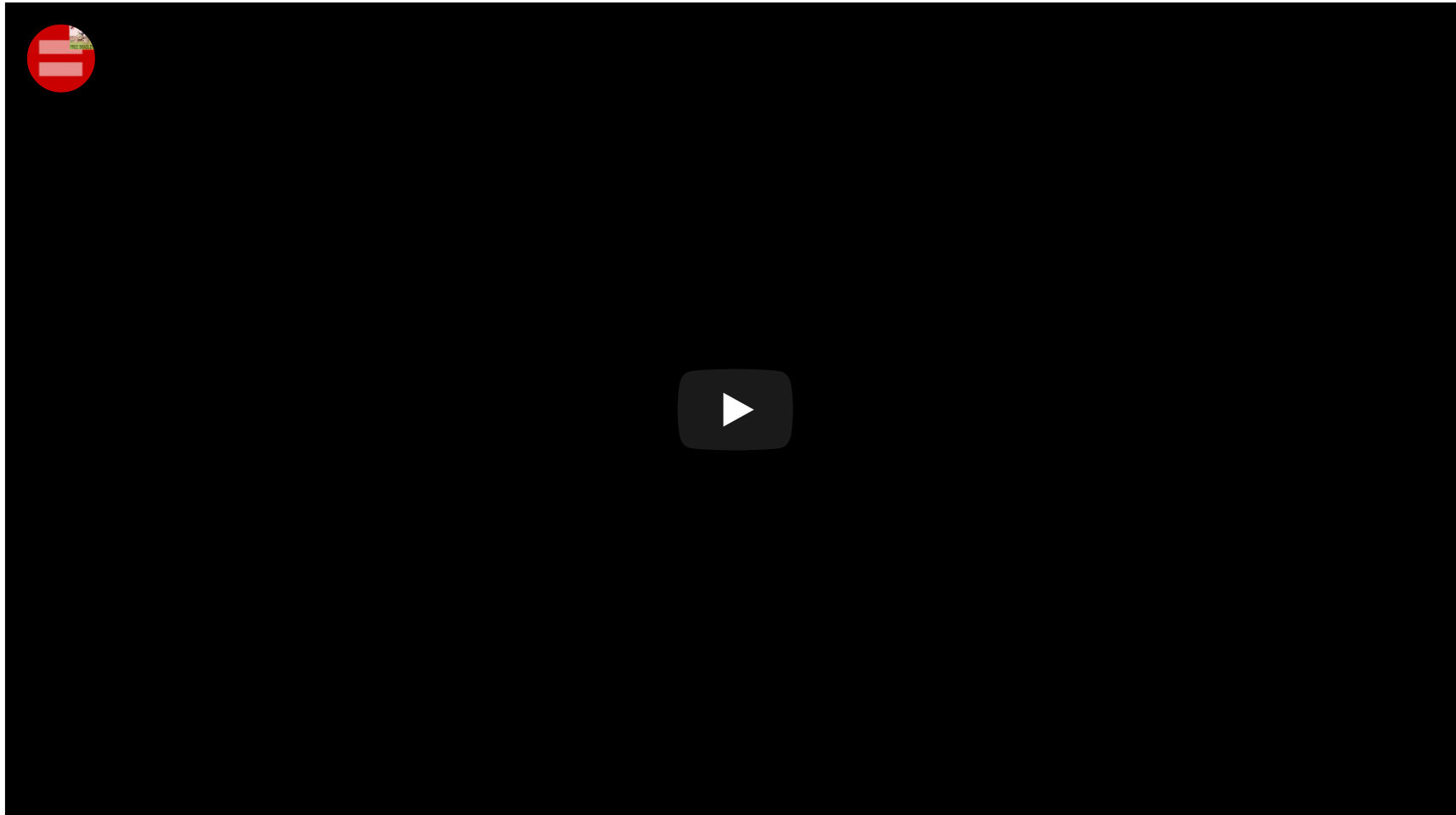
- But, GNU 프로젝트에는 커널이 없었고..

# Kernel



- 하드웨어와 응용프로그램을 이어주는 운영체제의 핵심 시스템소프트웨어

# Linus Torvalds



- 헬싱키 대학생이던 리누스 토발즈는 앤디 타넨바움의 MINIX를 개조한 Linux를 발표
- 0.1 - bash(GNU Bourne Again SHell), gcc(UNIX 기반 C 컴파일러)

# Linux

- 리누스 토발즈가 작성한 커널 혹은 GNU 프로젝트의 라이브러리와 도구가 포함된 운영체제
- PC와 모바일, 서버, 임베디드 시스템 등 다양한 분야에서 활용
- Redhat, Debian, Ubuntu, Android 등 다양한 배포판이 존재

# Shell

- 운영체제의 커널과 사용자를 이어주는 소프트웨어
- sh(Bourne Shell): AT&T Bell 연구소의 Steve Bourne이 작성한 유닉스 셸
- csh: 버클리의 Bill Joy가 작성한 유닉스 셸(C언어랑 비슷한 모양)
- bash(Bourne Again Shell): Brian Fox가 작성한 유닉스 셸
  - 다양한 운영체제에서 기본 셸로 채택
- zsh: Paul Falstad가 작성한 유닉스 셸
  - sh 확장형 셸
  - 현재까지 가장 완벽한 셸

# Let's learn bash



# Shell Command Basic

```
$ cd documents

$ mkdir python - make directory python
$ cd python - change directory
$ cd .. - up to

$ ls
$ ls -al

$ touch hello.py - create hello.py
$ exit - terminate shell
```

# Shell Command Basic

```
$ mv hello.py python  
$ cp hello.py python
```

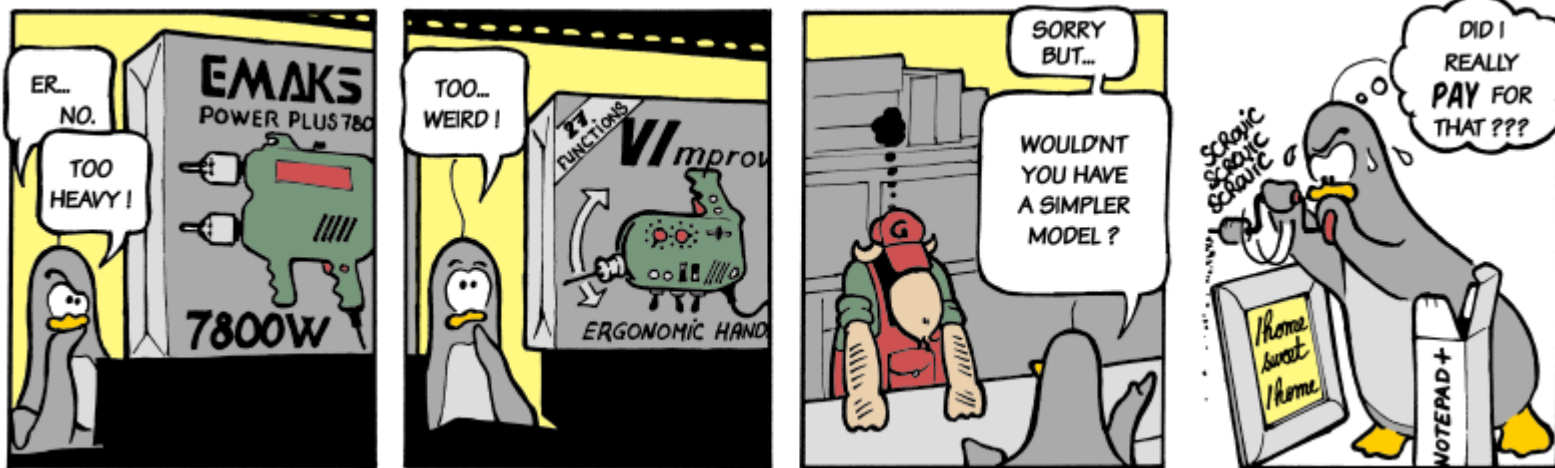
```
$ rm hello.py  
$ rm -rf python/
```

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

# Vim



# Vim



Copyright (c) 2007 Laurent Gregoire

- Vi improved Text Editor

# Vim Basic

## Command

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

:q – quit
:q! – quit w/o write(no warning)
:wq – write and quit

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

## write `hello.py` with Vim

```
$ vim
```

```
$ vim hello.py
```

```
i
```

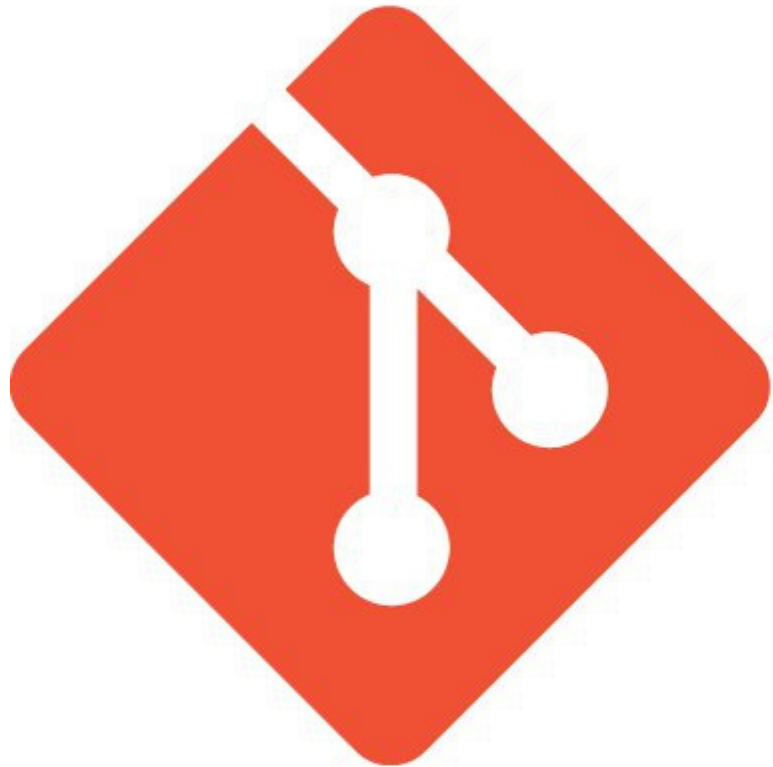
```
-- insert --
```

```
type print("hello python!")
```

```
press esc to escape
```

```
:wq
```

```
$ python hello.py
```



# git

# VCS (Version Control System)

== SCM (Source Code Management)

< SCM (Software Configuration Management: 형상관리)



# chronicle of git



## chronicle of git

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

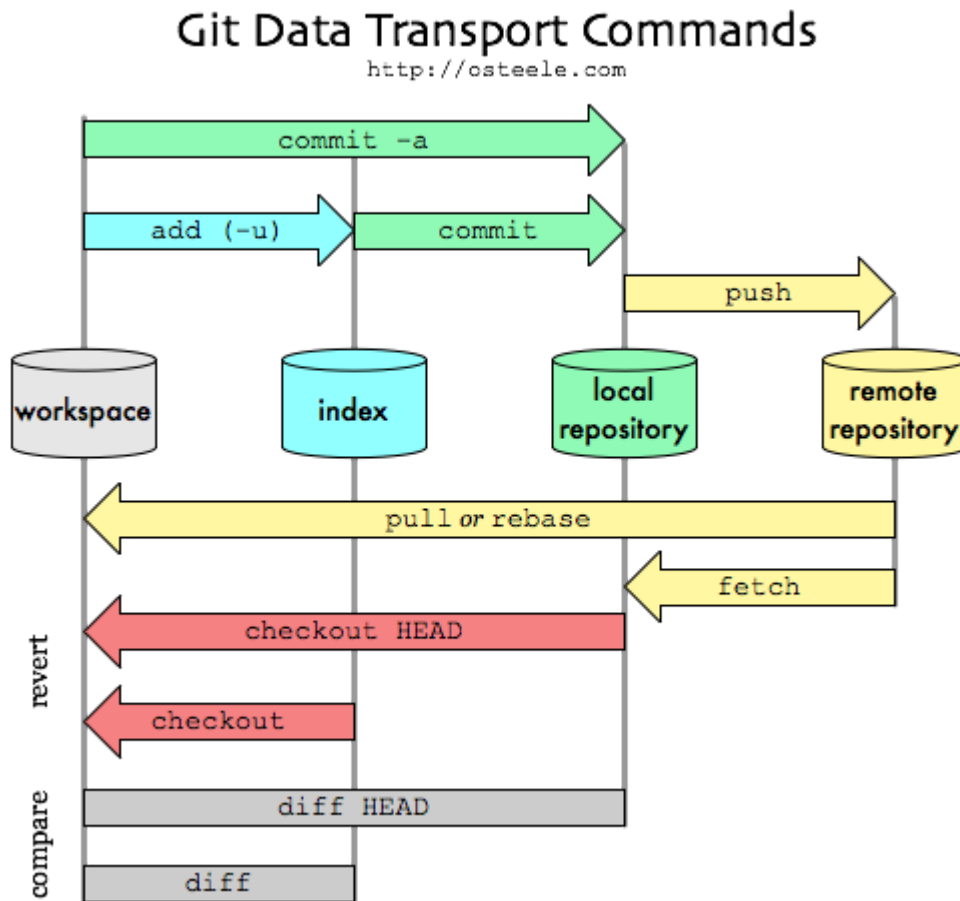
## Characteristics of git

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

## Pros of git

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

# git Process and Command



## Useful manager for mac

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

## install git

<https://gitforwindows.org/>

```
// MacOS  
$ 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

# 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
```

# TODO

- TIL repo 만들고 학습내용 정리하기