Fastcampus

Computer Science Extension School

Python Basic_Day4

Recap

- function
- list comprehension
- File I/O
- Error Handle

Index

- Shell Command
- vim command
- git, github
- lambda function
- map, filter

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 쉘
- csh: 버클리의 Bill Joy가 작성한 유닉스 쉘(C언어랑 비슷한 모양)
- bash(Bourne Again Shell): Brian Fox가 작성한 유닉스 쉘
    - 다양한 운영체제에서 기본 쉘로 채택
- zsh: Paul Falstad가 작성한 유닉스 쉘
    - sh 확장형 쉘
    - 현재까지 가장 완벽한 쉘

Let's learn bash

> for windows: https://git-scm.com/

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

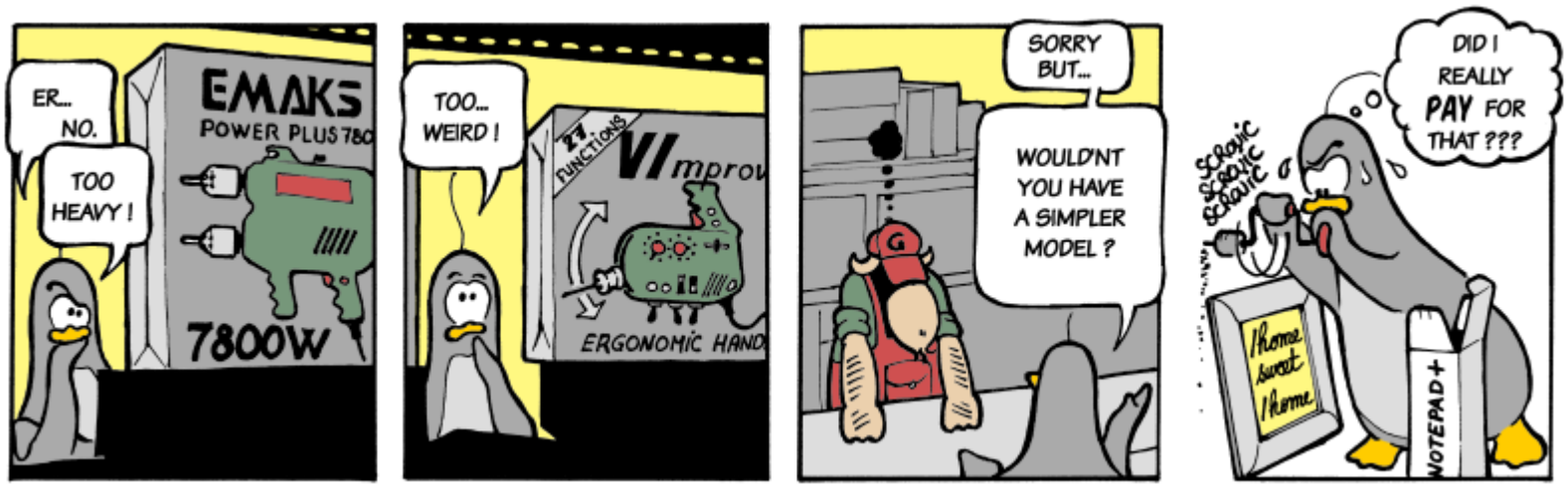
```
$ mv hello.py python
$ cp hello.py python

$ 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(or arrow key) - 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
```

copy & paste

```
$ vim hello.py
```

`v`

`-- visual --`

블록지정 후 `y`

`p`

press `esc` to escape

`:wq`

```
$ python hello.py
```

Use macro with Vim

`$ vim hello.py`

`qa` - a라는 매크로를 생성

`--recording--` 이 보이면 매크로 작성

`q` - 매크로 작성 종료

`@a` - a 매크로 실행

`10@a` - a 매크로 10회 실행


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??_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

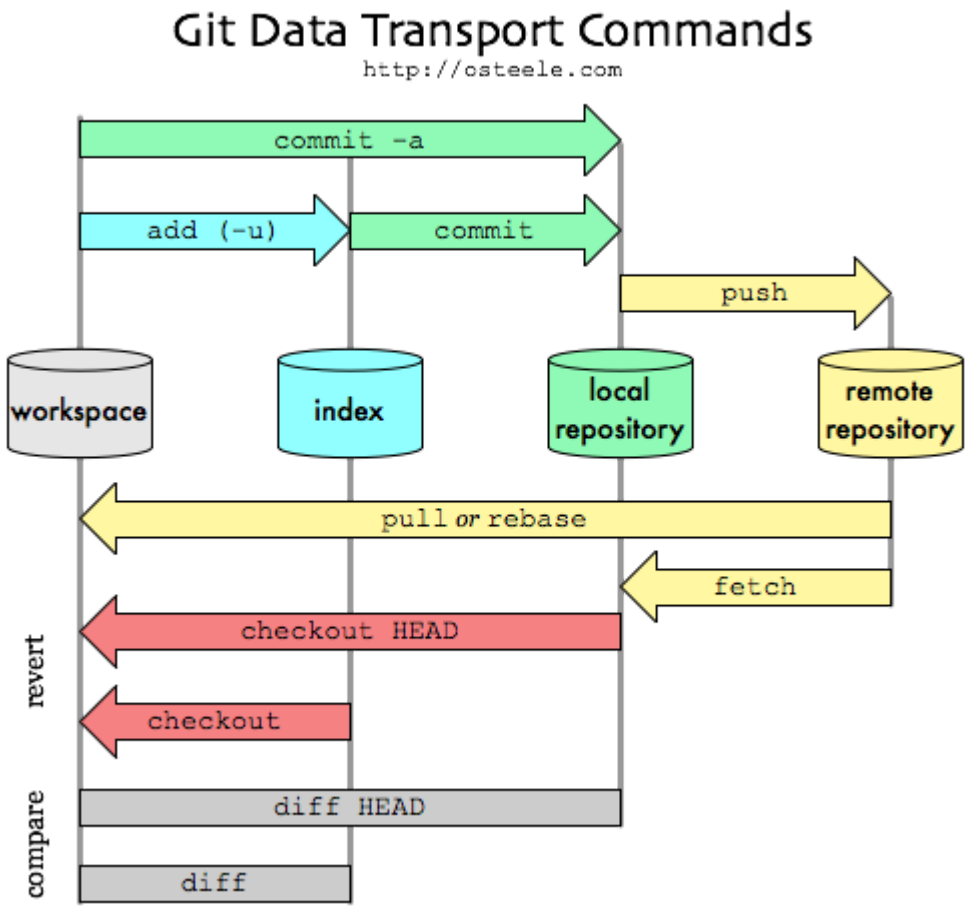
https://github.com/JuliaLang/julia

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

install git

https://git-scm.com/

```
// 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

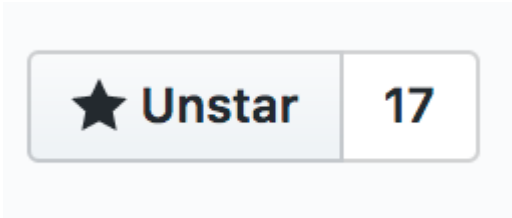
https://github.com/

important!!

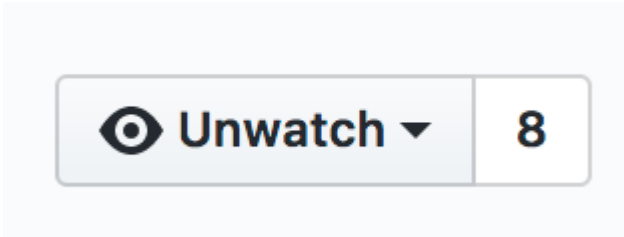
- 가입할 `email` 과 `username` 은 멋지게
- private repo를 원한다면 $7/month

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 --global core.editor "vim"
$ 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
```

Mini Project

- List comprehension 으로 FizzBuzz 한줄로 구현하기

Mini Project - hint

- `"Fast"*True + "Campus"*False`
- `elif ==> else if`

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

```
["fizzbuzz" if i%15==0 else "fizz" if i%3==0 else "buzz" if i%5==0 else i for i in range(1,100+1)]
```

lambda

lambda

- 익명함수(이름이 없는 함수)
- 간단한 수식을 함수로 지정해 한 두번 쓸 용도로 사용할 때
- 두 줄 이상 실행될 함수는 그냥 함수로 정의하는게 나음!

lambda

- python은 모든 것이 객체로 존재
- 간단한 연산 함수 조차 객체로 존재하여 리소스를 점유

lambda - traditional function

```
def get_next_integer(a):
        return a + 1
```

lambda - lambda function

```
lambda a: a+1
```

lambda - usual example

```
>>> (lambda a: a+1)(12)
13
```

lambda

- can't assign to lambda

```
(lambda a,b: a*b)(10,20)
```

- only expression, not statement

map, reduce, filter

map

- list의 각 element에 대해 특정한 함수를 적용

map - example

```
def get_squared(num_list):
    squared = []
    for num in num_list:
        squared.append(num**2)
    return squared
```

map - example

```
def squared_lambda(x):
    return x ** 2

list(map(squared_lambda, [1,2,3,4]))
```

map with lambda - example

```
list(map(lambda x: x**2, [1,2,3,4]))
```

map is rather than for

```
def print_with_sleep(x):
    time.sleep(1)
    return x ** 2
```

```
m = map(print_with_sleep, [1,2,3])
next(m)
next(m)
next(m)
..
```

```
for i in range(1,3+1):
    print_with_sleep(i)
```

timing is perfect!

```
m = map(print_with_sleep, [1,2,3])
for i in m:
    print(i)
```

```
m = map(print_with_sleep, [1,2,3])
list(m)
```

map is rather than for

- map은 제너레이터를 생성해 함수와 인자를 바인딩만 하고, 필요할 때 순회하며 값을 처리
- for는 한번에 처리하므로 for 수행 중 다른 일을 할 수 없음

filter

- 특정함수를 만족하는 요소만 남기는 필터

filter - example

```
def even_selector(x):
    if x % 2 == 0:
        return True
    else:
        return False

filter(even_selector, range(1,10+1))
```

filter with lambda - example

```
filter(lambda x: x%2==0, range(1,10+1))
```

Mini Project

```
recycle_bin = [1,2, "Fastcampus", [], 5,4, 5.6, "패스트캠퍼스"]
```

위 리스트의 요소 중 정수만 제곱하여 출력하기

map, filter, lambda를 모두 사용

> Hint: isinstance(1, int)