

# Linux commands & Numpy

Linux

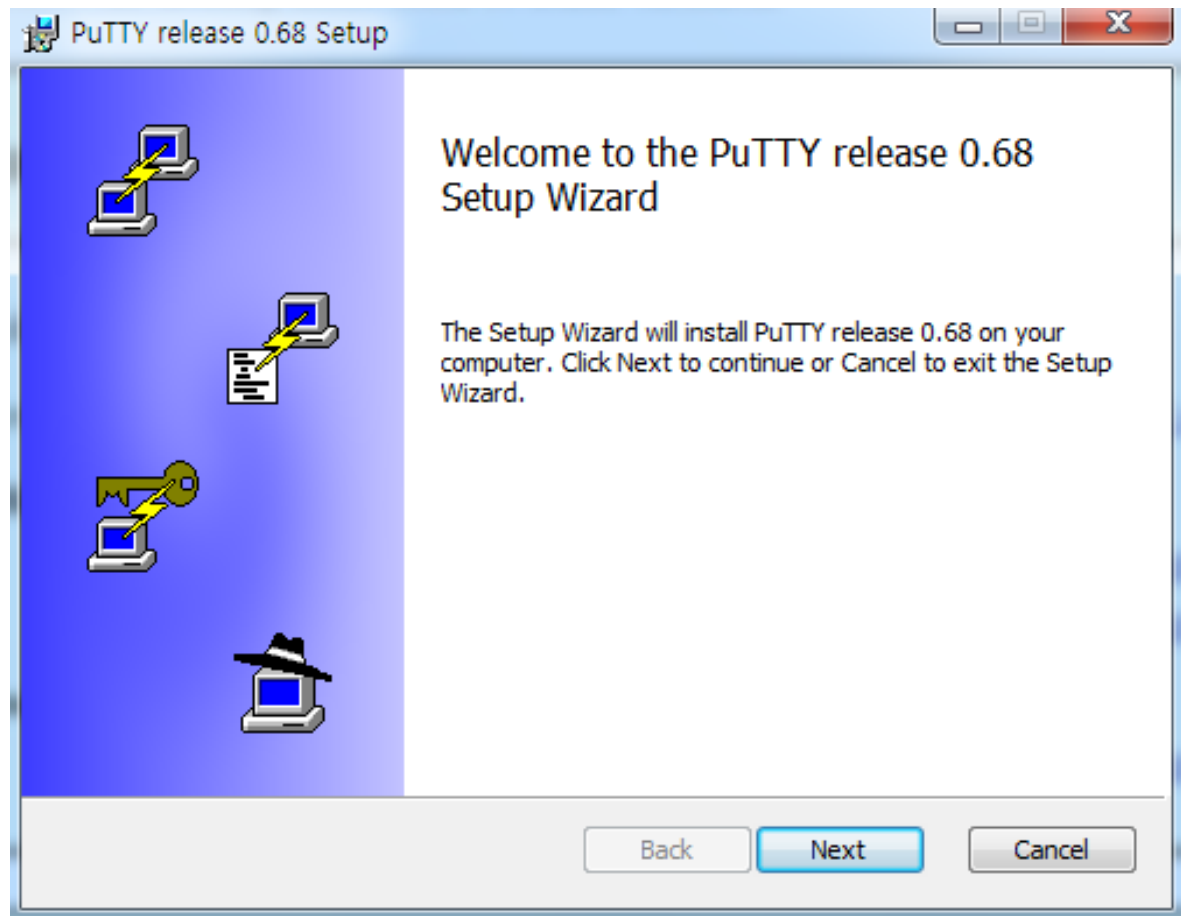
# Why Linux?

- 무료
- 오픈소스
  - 빠른 업데이트
  - 많은 무료 소프트웨어
    - Python
    - Numpy
    - Tensorflow
    - Scipy
    - matplotlib
- 편한? 콘솔 환경
  - 원격접속

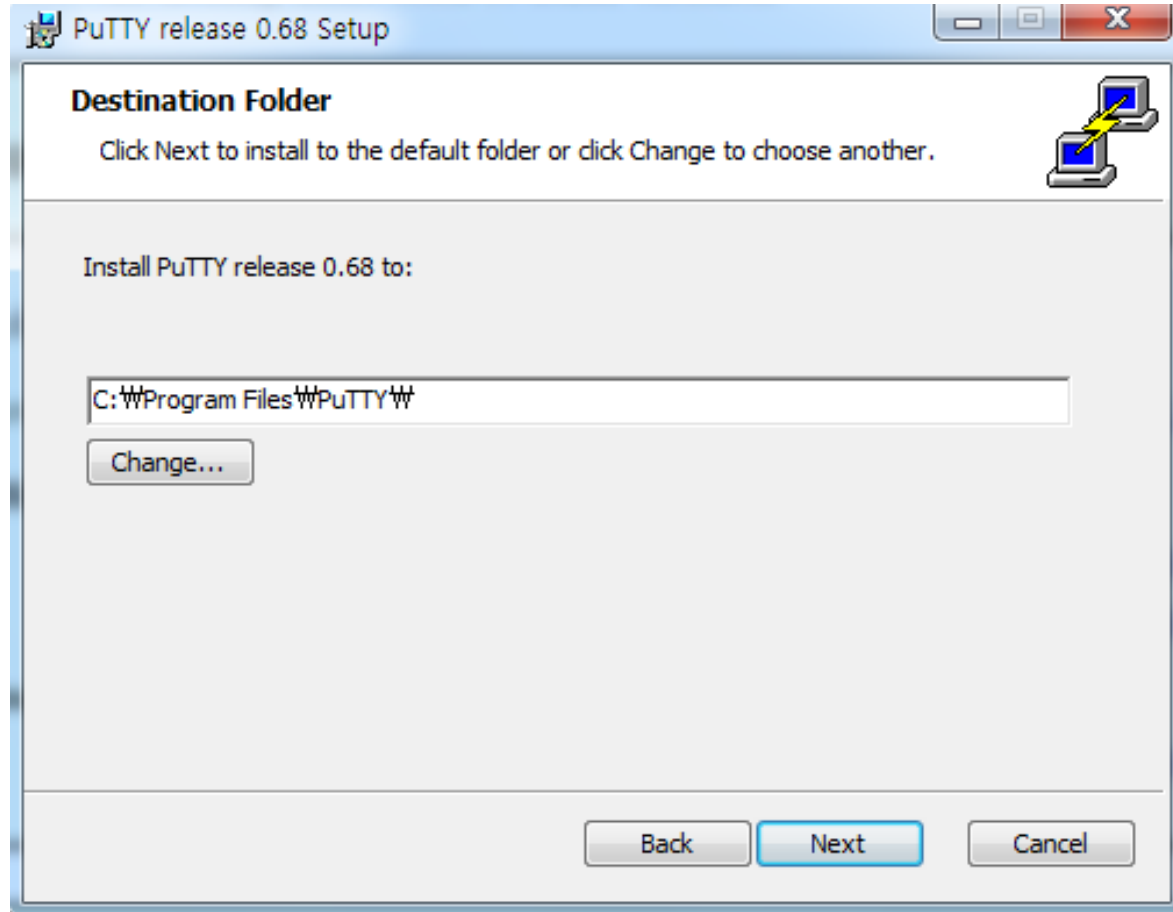
# 리눅스 원격접속

- 윈도우에서 리눅스로 원격접속하기 위해서는 putty가 필요
- Putty: ssh 클라이언트
  - ssh: 원격접속을 안전하게 할 수 있게 해주는 프로토콜
- Putty 다운로드:
  - <http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

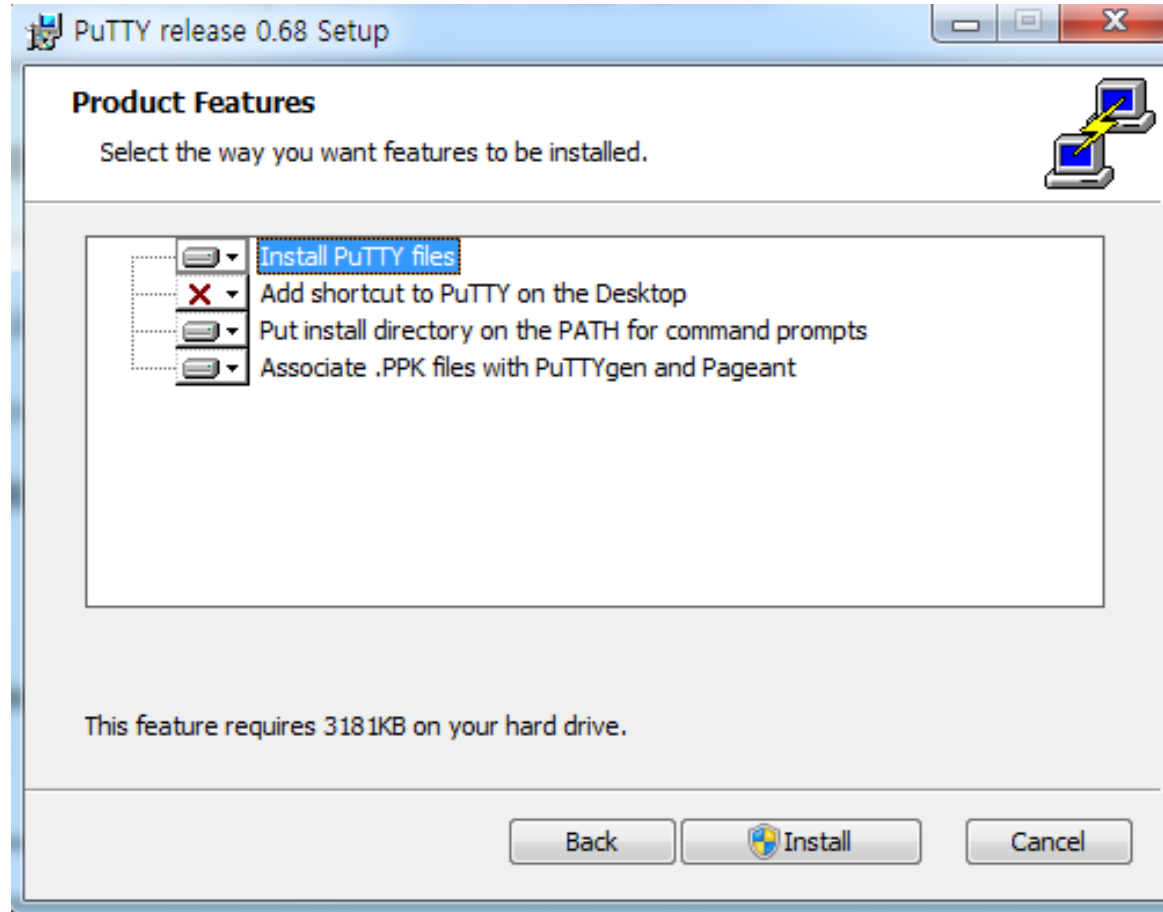
# Putty 설치



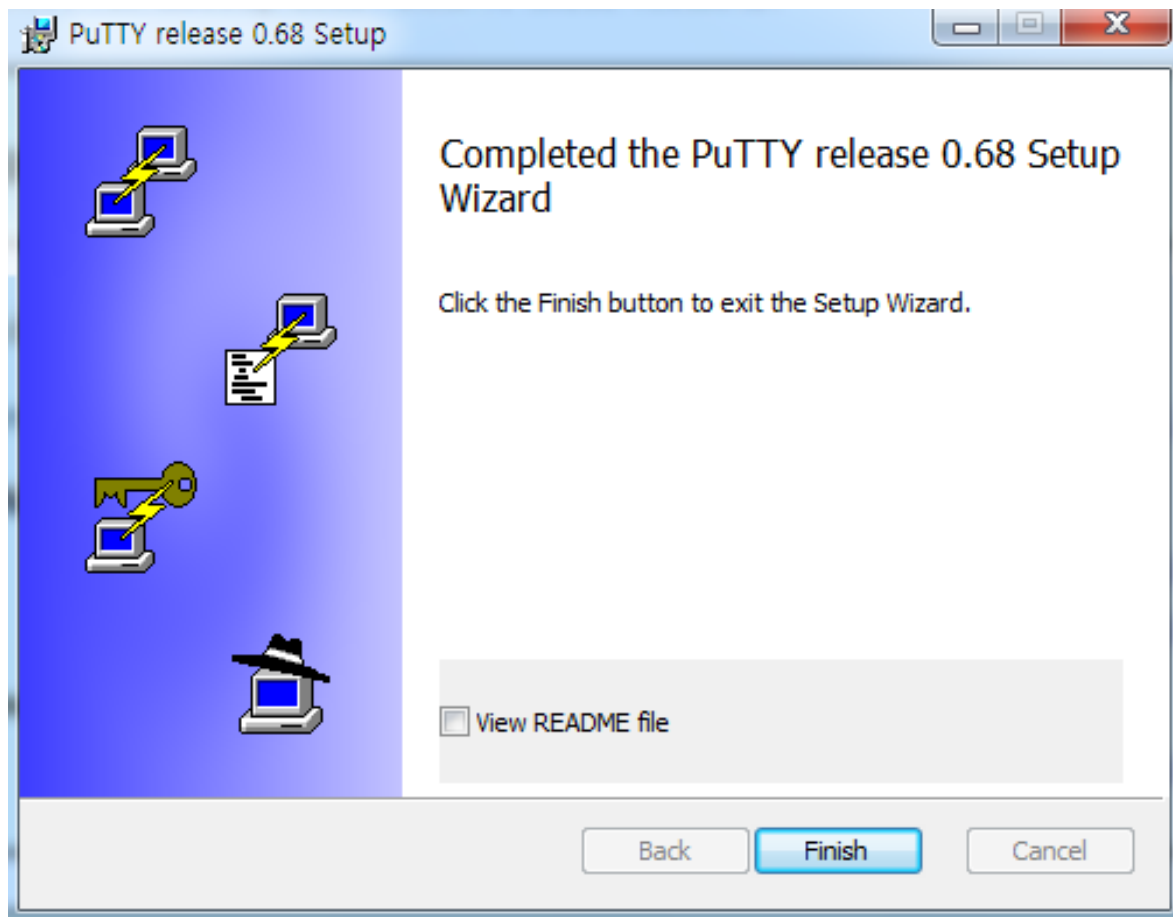
# Putty 설치



# Putty 설치



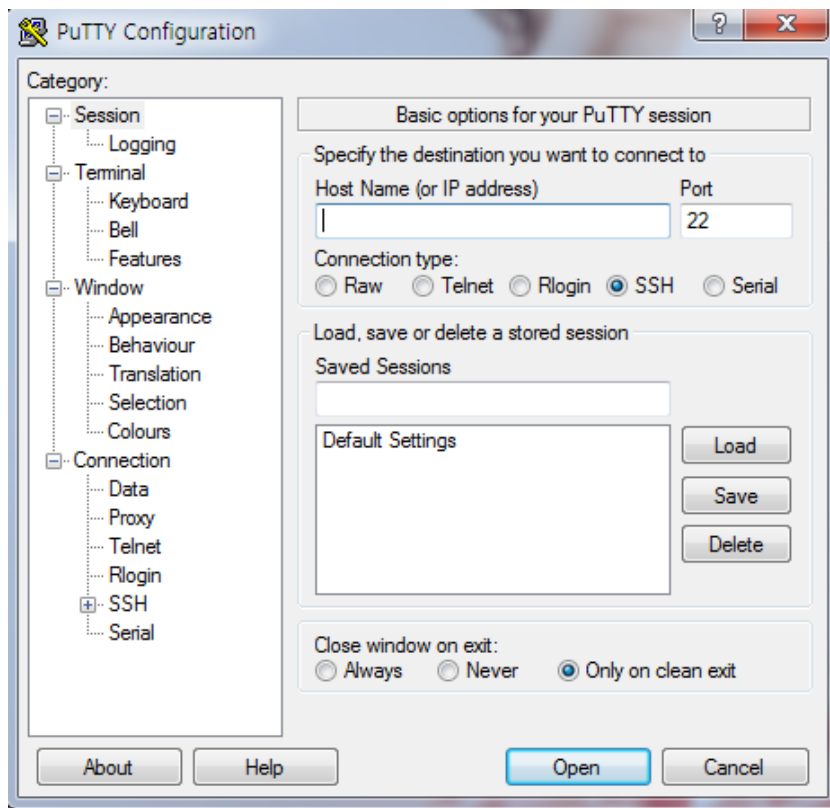
# Putty 설치





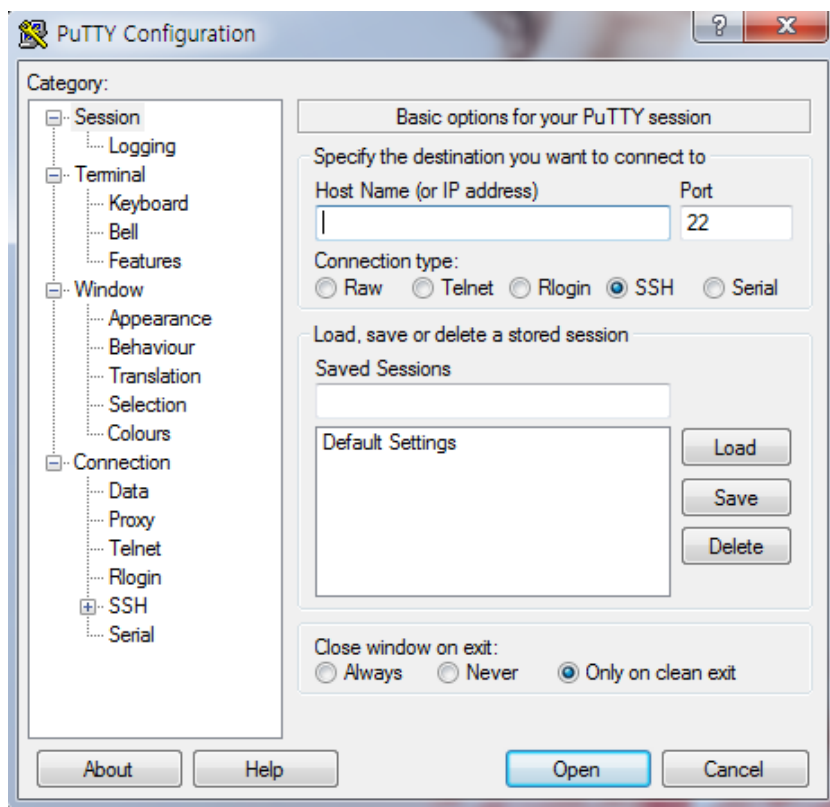
# Putty 실행

- 시작 → 모든 프로그램 → PuTTY (folder) → PuTTY



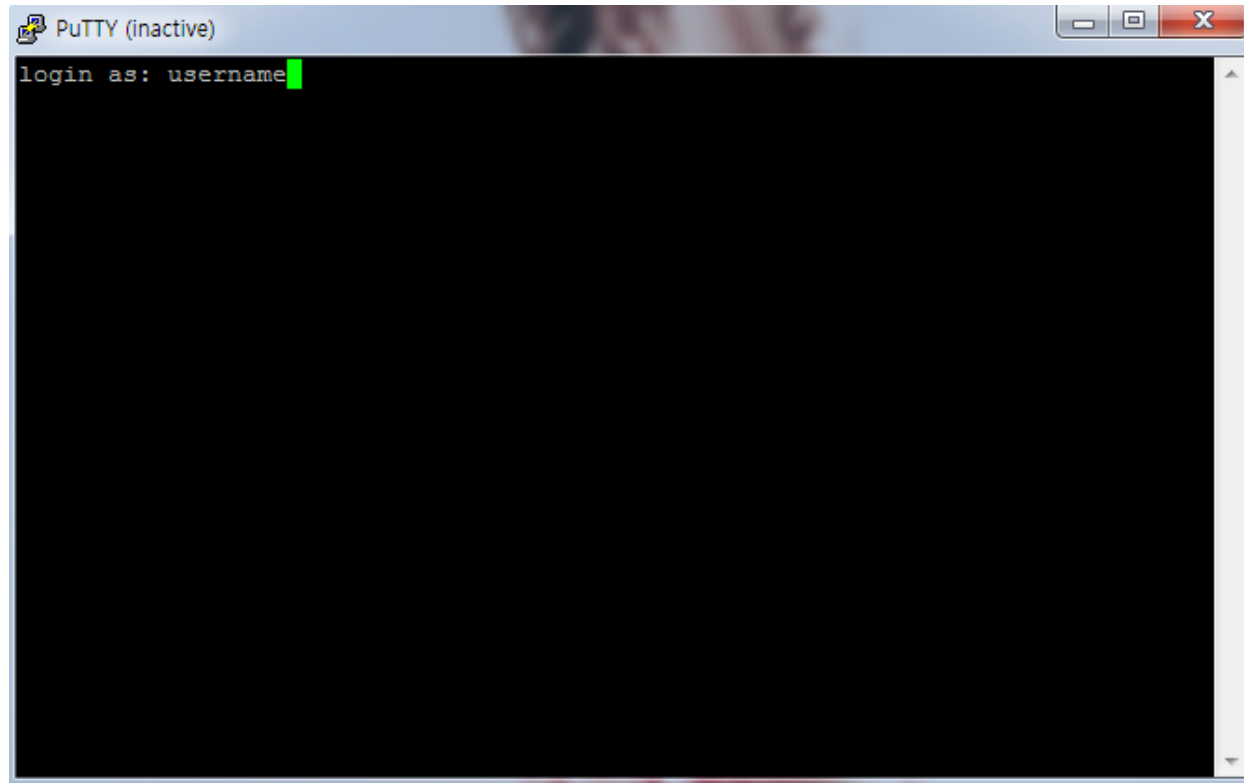
# Linux (ubuntu) 로 접속

- IP 입력 후 Open 버튼 클릭



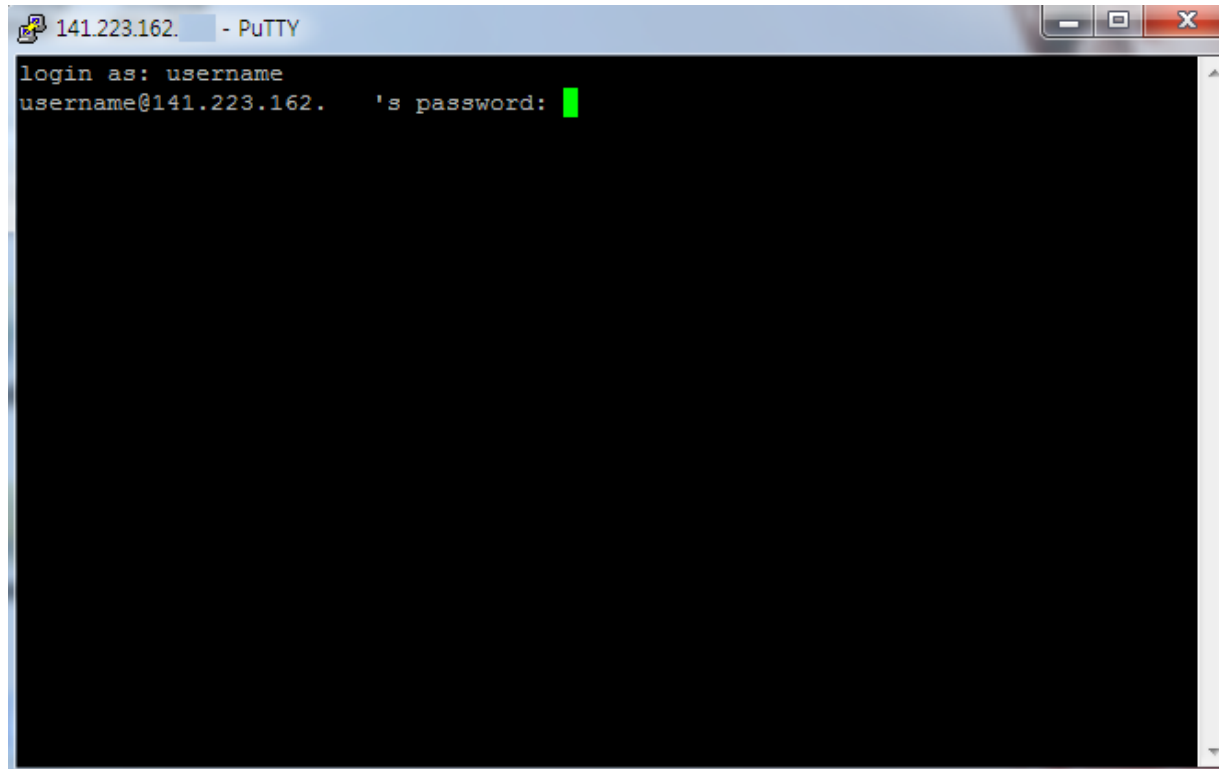
# Linux (ubuntu) 로 접속

- username 입력
- 각자의 username은?



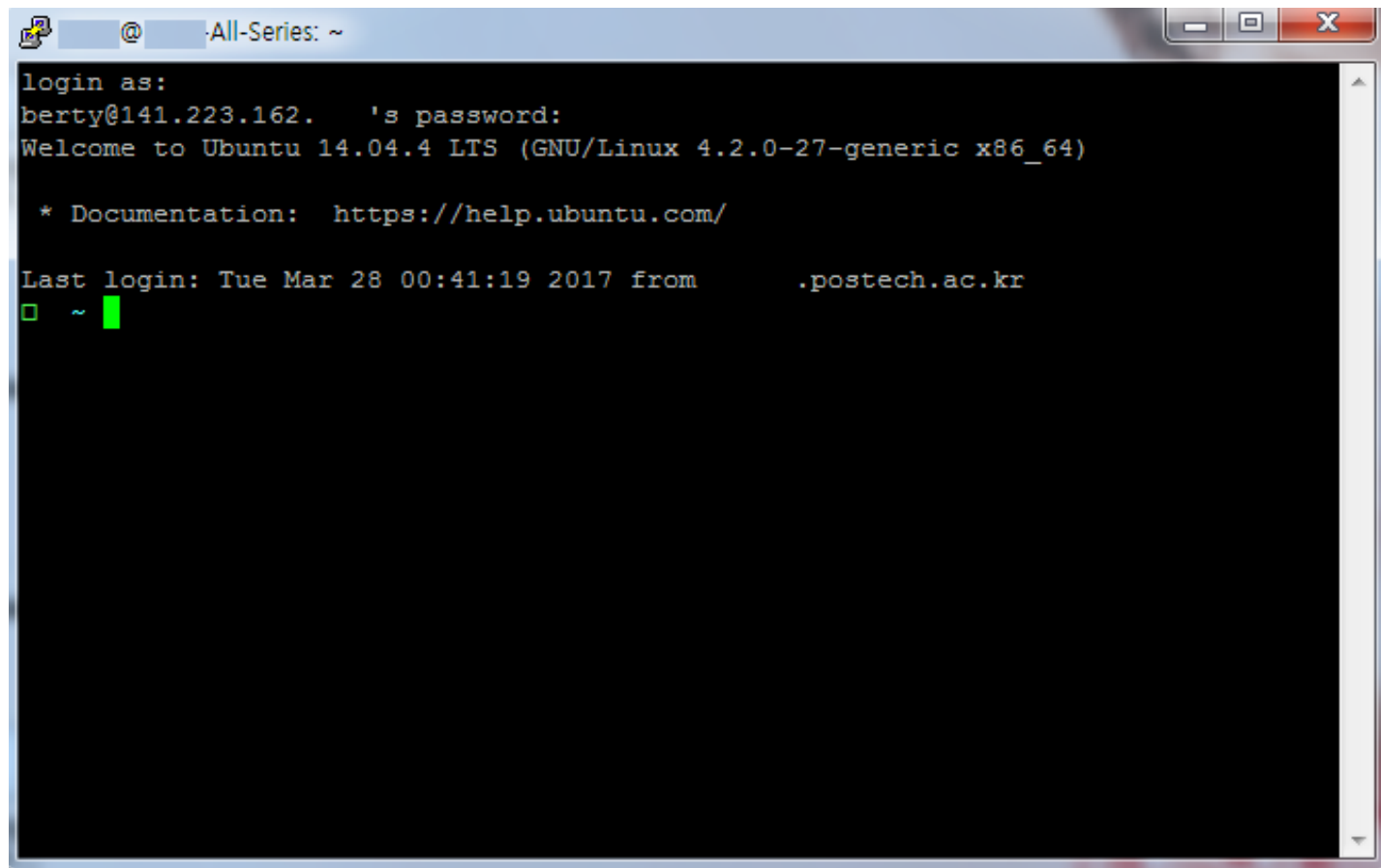
# Linux (ubuntu) 로 접속

- password 입력
- 기본 password는?



# Linux (ubuntu) 로 접속

- 접속 성공 시 화면

A screenshot of a terminal window titled ".All-Series: ~". The terminal displays the following text: "login as:", "berty@141.223.162. 's password:", "Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86\_64)", "\* Documentation: https://help.ubuntu.com/", and "Last login: Tue Mar 28 00:41:19 2017 from .postech.ac.kr". The prompt is "b ~" with a green cursor. The window has standard Ubuntu window controls (minimize, maximize, close) in the top right corner.

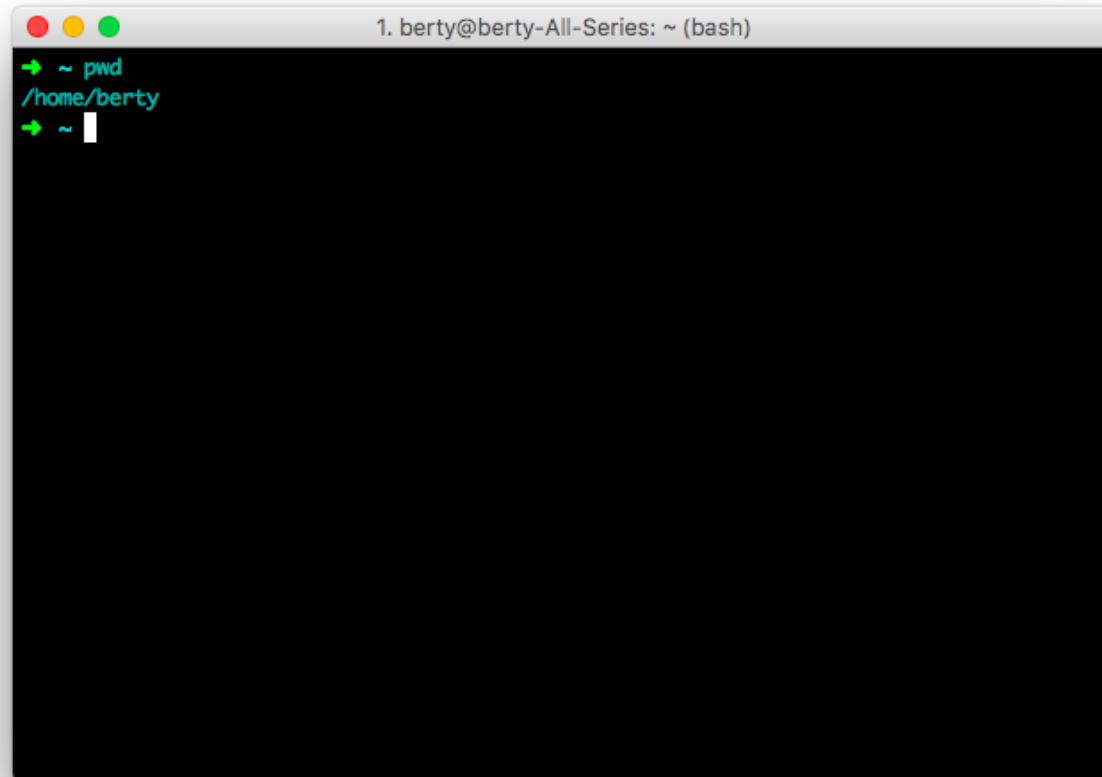
```
login as:
berty@141.223.162.  's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Tue Mar 28 00:41:19 2017 from      .postech.ac.kr
b ~
```

# 디렉토리 관련 명령어

- 윈도우에서는 폴더, 리눅스에서는 디렉토리
- 현재 위치하는 디렉토리는?
  - pwd

A terminal window with a dark background and light green text. The title bar at the top reads "1. berty@berty-All-Series: ~ (bash)". The terminal shows two lines of input and output: the first line is a prompt followed by "~ pwd" and the output "/home/berty"; the second line is a prompt followed by "~" and a cursor.

```
1. berty@berty-All-Series: ~ (bash)
→ ~ pwd
/home/berty
→ ~
```

# 디렉토리 관련 명령어

- 디렉토리에 존재하는 파일 조회
  - ls
- 더 자세하게 조회하려면 al 옵션을 붙임
  - ls -al

```
1. berty@berty-All-Series: ~ (bash)
→ ~ ls
101_1080_server.sh Documents hw1_linguistics.odt Pictures Videos
113_1080_server.sh Downloads ipython Public
Desktop ftp Music tensorflow_venv
→ ~
```

```
1. berty@berty-All-Series: ~ (bash)
→ ~ ls -al
total 736
drwxr-xr-x 37 berty berty 4096 Apr  1 02:18 .
drwxr-xr-x  4 root  root  4096 Oct 21 20:26 ..
-rwxrwxr-x  1 berty berty   24 Aug  5 2016 101_1080_server.sh
-rwxrwxr--  1 berty berty   24 Jul 11 2016 113_1080_server.sh
-rw-----  1 berty berty 9002 Jan  1 04:56 .bash_history
-rw-r--r--  1 berty berty 220 May 10 2016 .bash_logout
-rw-r--r--  1 root  root   932 Jan  1 04:51 .bash_profile
-rw-r--r--  1 berty berty 4353 Jan  1 04:55 .bashrc
drwxr-xr-x  4 berty berty 4096 May 10 2016 .bazel
-rw-rw-r--  1 berty berty   0 May 10 2016 .bazelrc
drwx----- 29 berty berty 4096 Jan  1 04:34 .cache
drwx-----  3 berty berty 4096 May 13 2016 .compiz
drwx----- 31 berty berty 4096 Nov  5 12:15 .config
drwx-----  3 root  root  4096 May 10 2016 .dbus
drwxrwxr-x 19 berty berty 4096 Mar 30 21:11 Desktop
drwxrwxr-x  3 berty berty 4096 May 10 2016 .distlib
-rw-r--r--  1 berty berty   25 Aug 25 2016 .dmrc
drwxr-xr-x  2 berty berty 4096 May 10 2016 Documents
drwxr-xr-x  2 berty berty 4096 Mar 30 21:11 Downloads
drwxrwxr-x  5 berty berty 4096 Jan  1 04:34 .edm
-rw-rw-r--  1 berty berty 152 Jan  1 04:34 .edm.yaml
drwxrwxr-x  2 berty berty 4096 Oct  5 21:33 ftp
```

# 디렉토리 관련 명령어

- 디렉토리 이동
  - ls 명령어 결과에서 디렉토리는 다른 색으로 표시 됨
  - cd <이동할 디렉토리>



```
1. berty@berty-All-Series: ~/Desktop (bash)
→ ~ pwd
/home/berty
→ ~ cd Desktop
→ Desktop pwd
/home/berty/Desktop
→ Desktop
```

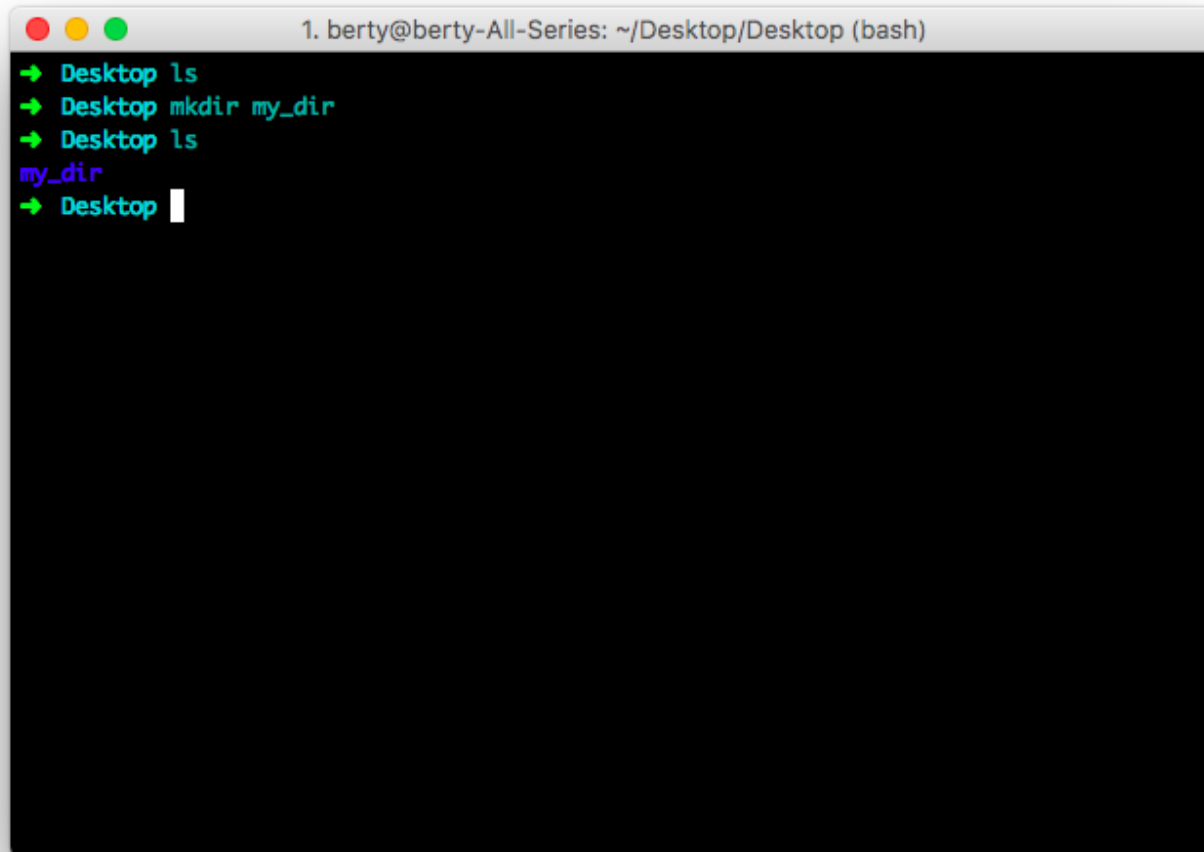


# 디렉토리 관련 명령어

- 기본 디렉토리
  - 현재 디렉토리: .
    - 윈도우로 비유하자면 현재 보고있는 폴더라 생각하시면 됩니다.
  - 부모 디렉토리: ..
    - 윈도우로 비유하자면 현재 폴더를 포함하고 있는 폴더를 나타냅니다.
  - 홈 디렉토리: ~
    - 윈도우로 비유하자면 바탕화면입니다 (처음 시작하는 폴더).
  - 루트 디렉토리: /
    - 윈도우로 비유하자면 '내 컴퓨터'에서의 C드라이브입니다.
- cd 명령어로 위 디렉토리들로 이동해 보세요

# 디렉토리 관련 명령어

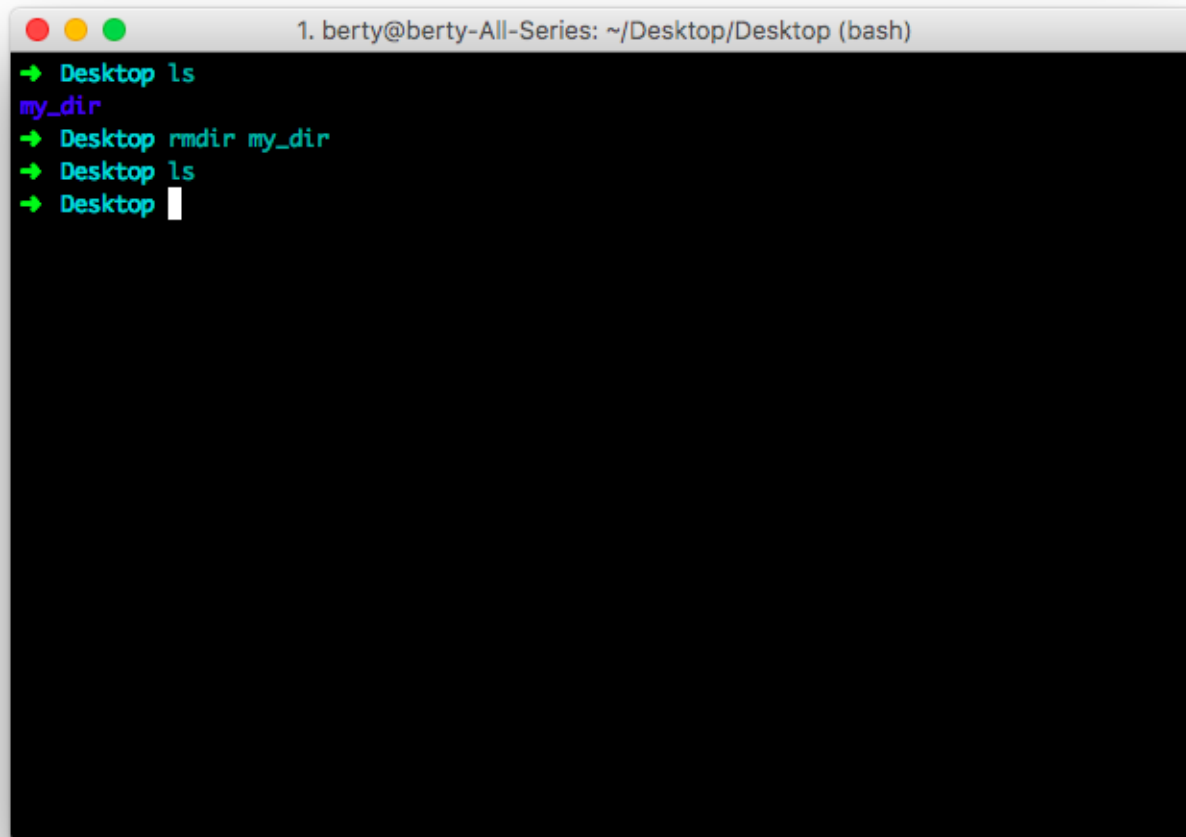
- 디렉토리 만들기
  - mkdir <디렉토리 이름>

A terminal window with a dark background and light green text. The window title bar shows '1. berty@berty-All-Series: ~/Desktop/Desktop (bash)'. The terminal content shows a sequence of commands: 'ls' (output: Desktop), 'mkdir my\_dir', 'ls' (output: my\_dir), and a prompt 'Desktop' with a cursor.

```
1. berty@berty-All-Series: ~/Desktop/Desktop (bash)
→ Desktop ls
→ Desktop mkdir my_dir
→ Desktop ls
my_dir
→ Desktop
```

# 디렉토리 관련 명령어

- 디렉토리 지우기
  - rmdir <디렉토리 이름>

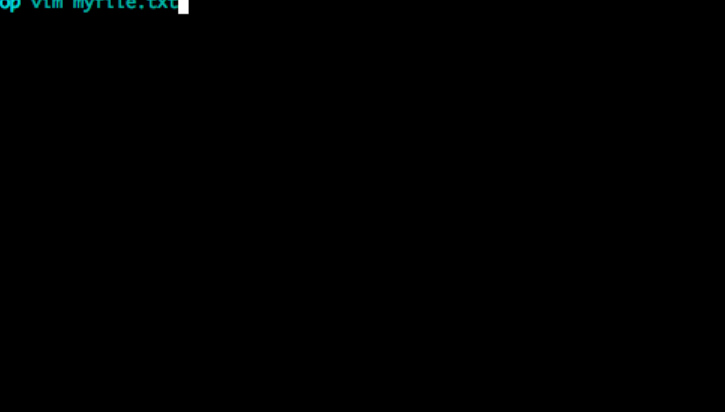
A terminal window with a dark background and light green text. The title bar shows '1. berty@berty-All-Series: ~/Desktop/Desktop (bash)'. The terminal content shows a sequence of commands: 'ls' followed by 'my\_dir' on a new line, then 'rmdir my\_dir', then 'ls', and finally a prompt 'Desktop' with a cursor. The output of the first 'ls' command is not visible.

```
1. berty@berty-All-Series: ~/Desktop/Desktop (bash)
→ Desktop ls
my_dir
→ Desktop rmdir my_dir
→ Desktop ls
→ Desktop
```

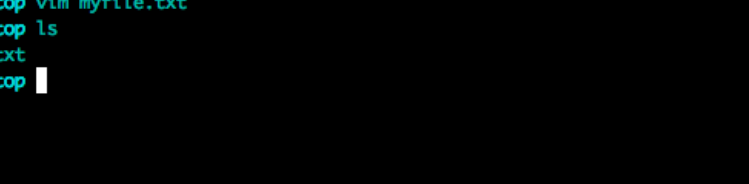
# 파일 관련 명령어

- 실습을 위한 임의의 파일 만들기
  - vim <파일 이름>
  - Shift 키 + ; (:)
  - wq 입력 후 enter 키

# 파일 관련 명령어



A terminal window with a title bar containing three colored circles (red, yellow, green) on the left and the text "1. berty@berty-All-Series: ~/Desktop/Desktop (bash)" on the right. The terminal area has a black background. A green prompt character "➔" is followed by the text "Desktop vim myfile.txt" in a light blue font. A white cursor is positioned at the end of the text.

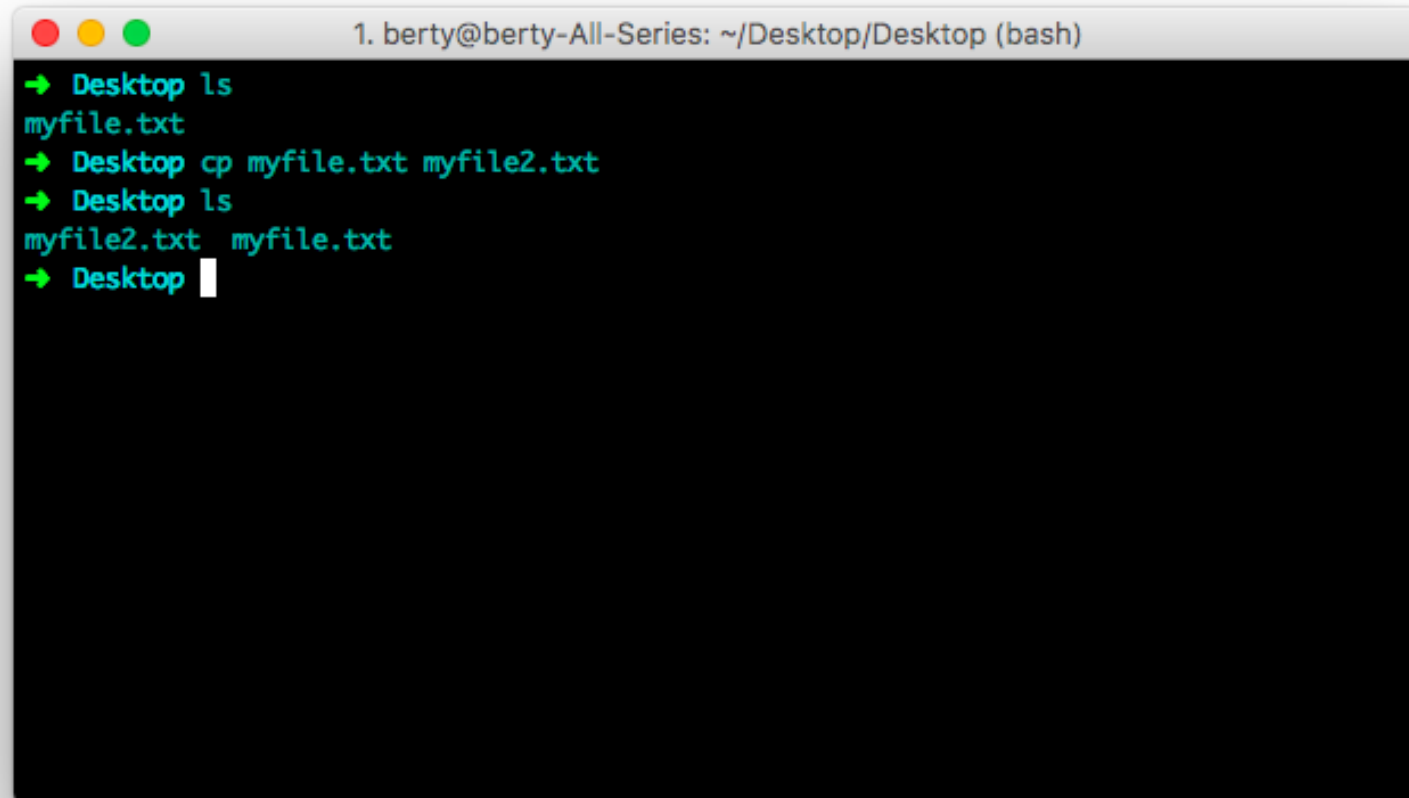
[illegible]A screenshot of a terminal window titled "1. vim myfile.txt (bash)". The window has three colored window control buttons (red, yellow, green) at the top left. The main area is black, representing the Vim editor. On the far left, there are several tilde (~) symbols indicating line numbers. A yellow cursor is positioned at the first line, which is labeled with a yellow "1". At the bottom, a red status bar displays "0:1 [All]" on the left and "~/Desktop/Desktop/myfile.txt\" on the right. In the bottom-left corner, below the status bar, the text ":wq" is visible, indicating the user is entering the write-and-quit command.

A terminal window titled "1. berty@berty-All-Series: ~/Desktop/Desktop (bash)". The terminal shows the following commands and output:

```
→ Desktop vim myfile.txt
→ Desktop ls
myfile.txt
→ Desktop
```

# 파일 관련 명령어

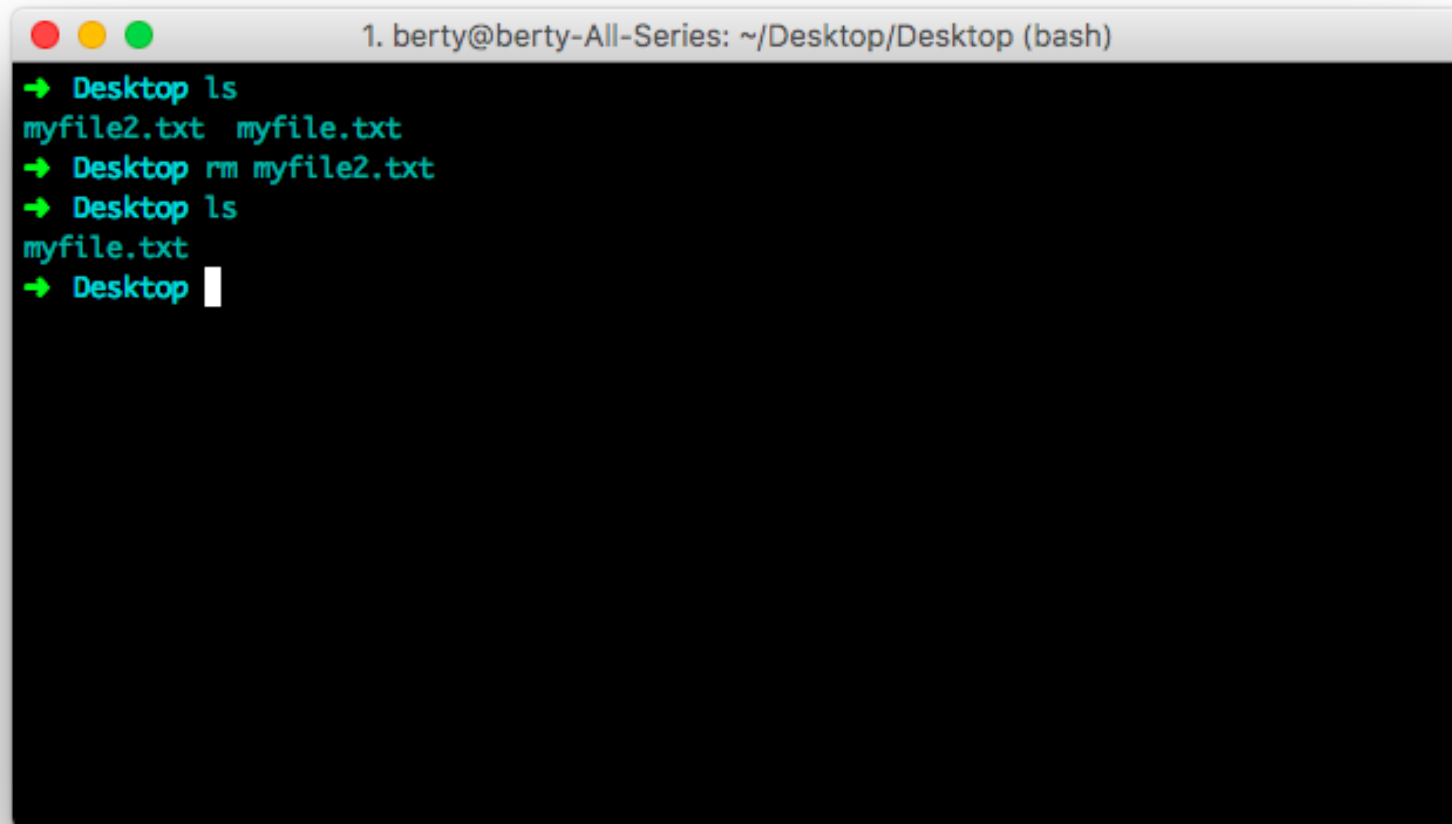
- 파일 복사하기
  - cp <복사될 파일 이름> <새 파일 이름>

A terminal window with a dark background and light green text. The window title bar shows '1. berty@berty-All-Series: ~/Desktop/Desktop (bash)'. The terminal content shows a sequence of commands and their outputs: 'ls' followed by 'myfile.txt', 'cp myfile.txt myfile2.txt', and 'ls' followed by 'myfile2.txt' and 'myfile.txt'. The prompt 'Desktop' is followed by a cursor.

```
1. berty@berty-All-Series: ~/Desktop/Desktop (bash)
→ Desktop ls
myfile.txt
→ Desktop cp myfile.txt myfile2.txt
→ Desktop ls
myfile2.txt  myfile.txt
→ Desktop
```

# 파일 관련 명령어

- 파일 삭제하기
  - `rm <삭제할 파일 이름>`

A terminal window with a dark background and light green text. The window title bar shows '1. berty@berty-All-Series: ~/Desktop/Desktop (bash)'. The terminal content shows a sequence of commands and their outputs: 'ls' lists 'myfile2.txt' and 'myfile.txt'; 'rm myfile2.txt' removes the file; a second 'ls' shows only 'myfile.txt' remains. The prompt is currently 'Desktop ' with a cursor.

```
1. berty@berty-All-Series: ~/Desktop/Desktop (bash)
→ Desktop ls
myfile2.txt  myfile.txt
→ Desktop rm myfile2.txt
→ Desktop ls
myfile.txt
→ Desktop
```

# 파일 관련 명령어

- 파일 옮기기 (이름 변경하기)
  - mv <옮길 파일 이름> <옮겨질 경로/파일이름>
  - mv a.txt b.txt
    - a.txt가 b.txt로 이름이 변경됨

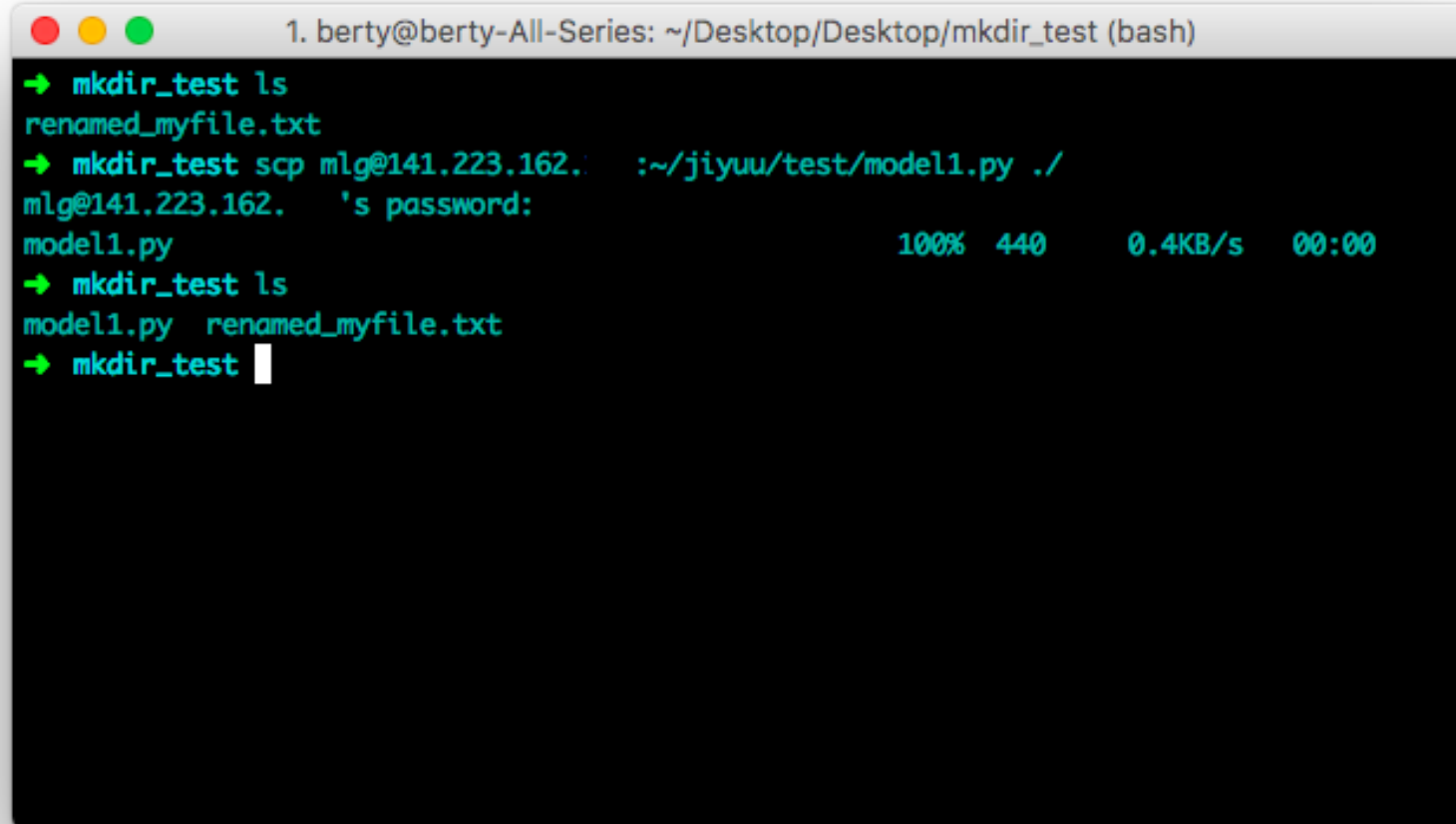
```
1. berty@berty-All-Series: ~/Desktop/Desktop/mkdir_test (bash)
→ Desktop mkdir mkdir_test
→ Desktop ls
mkdir_test myfile.txt
→ Desktop mv myfile.txt mkdir_test
→ Desktop ls
mkdir_test
→ Desktop cd mkdir_test
→ mkdir_test ls
myfile.txt
→ mkdir_test
```

```
1. berty@berty-All-Series: ~/Desktop/Desktop/mkdir_test (bash)
→ mkdir_test ls
myfile.txt
→ mkdir_test mv myfile.txt renamed_myfile.txt
→ mkdir_test ls
renamed_myfile.txt
→ mkdir_test
```



# 파일 관련 명령어

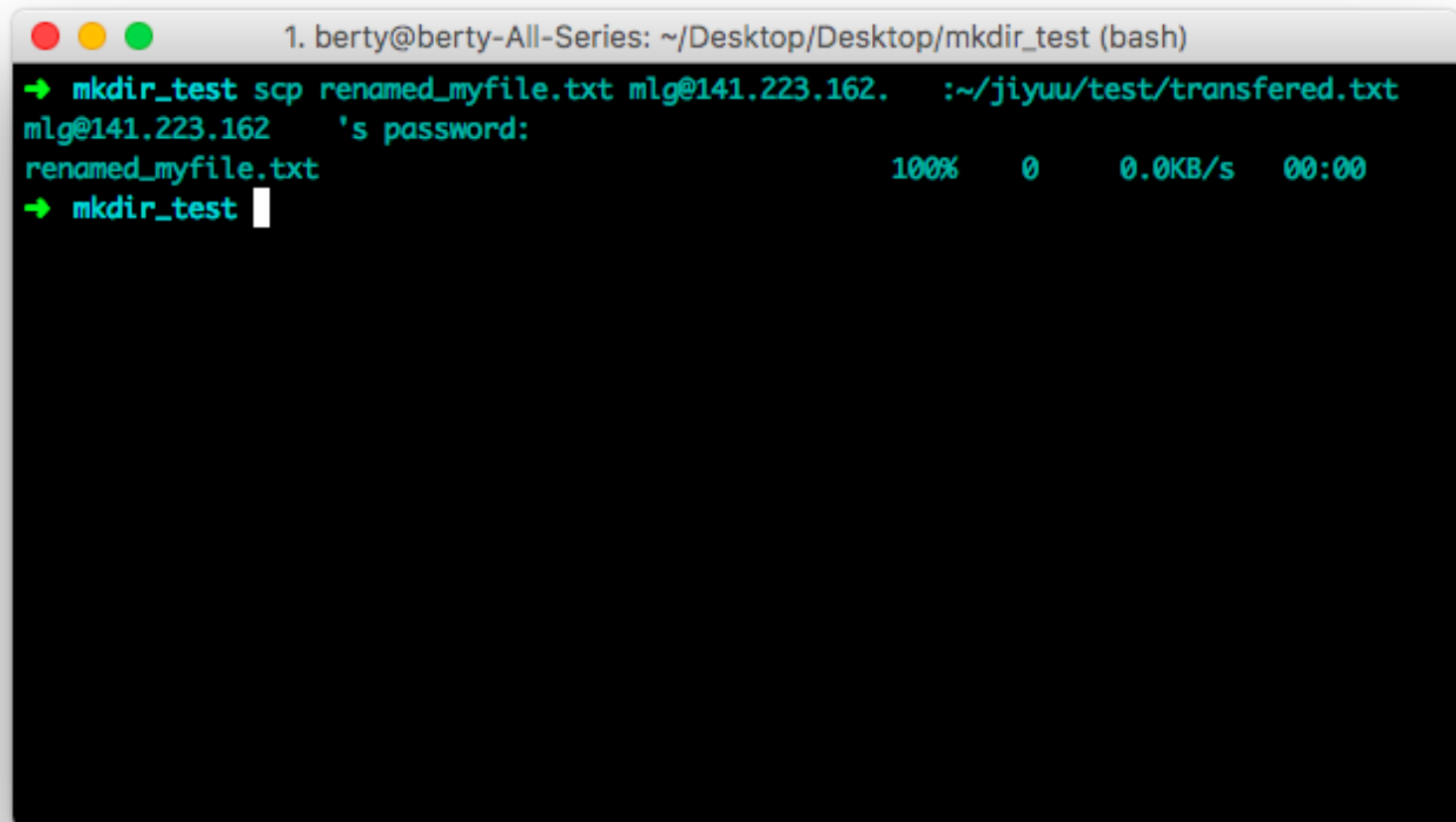
- 원격 서버에서 파일 가져오기
  - `scp <서버 ip> @<username>:<파일 경로> <옮길 디렉토리>`

A terminal window with a dark background and light green text. The window title is "1. berty@berty-All-Series: ~/Desktop/Desktop/mkdir\_test (bash)". The terminal shows a sequence of commands: "mkdir\_test ls" followed by "renamed\_myfile.txt"; "mkdir\_test scp mlg@141.223.162.: ~/jiyuu/test/model1.py ./" followed by "mlg@141.223.162. 's password:" and then "model1.py" with progress "100% 440 0.4KB/s 00:00"; "mkdir\_test ls" followed by "model1.py renamed\_myfile.txt"; and finally "mkdir\_test" with a cursor.

```
1. berty@berty-All-Series: ~/Desktop/Desktop/mkdir_test (bash)
→ mkdir_test ls
renamed_myfile.txt
→ mkdir_test scp mlg@141.223.162.: ~/jiyuu/test/model1.py ./
mlg@141.223.162. 's password:
model1.py                                100% 440    0.4KB/s   00:00
→ mkdir_test ls
model1.py renamed_myfile.txt
→ mkdir_test
```

# 파일 관련 명령어

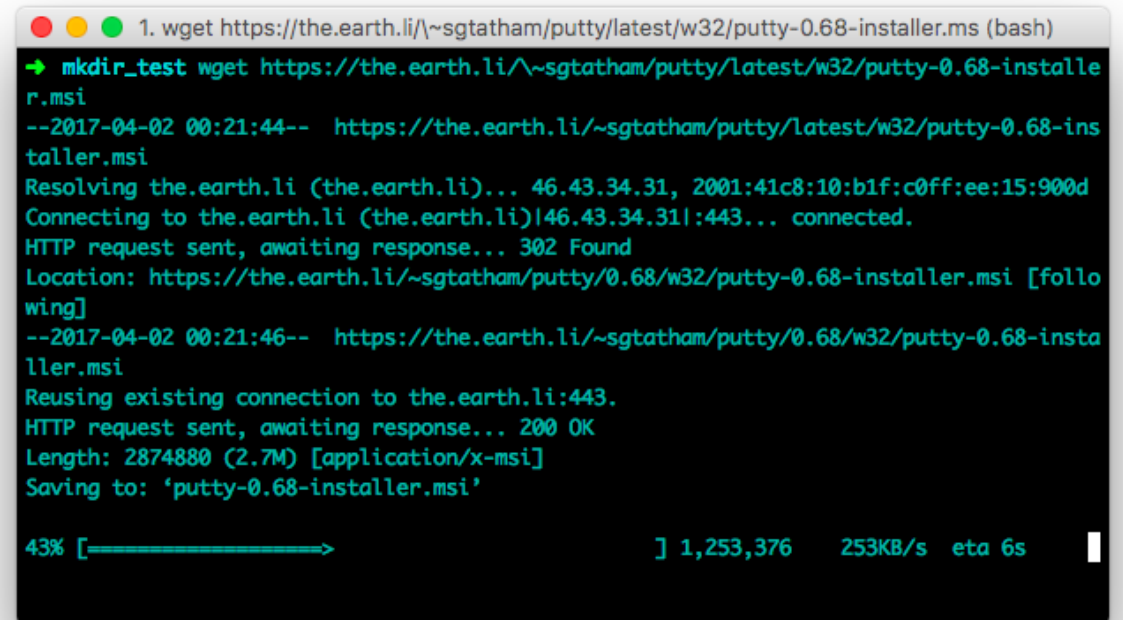
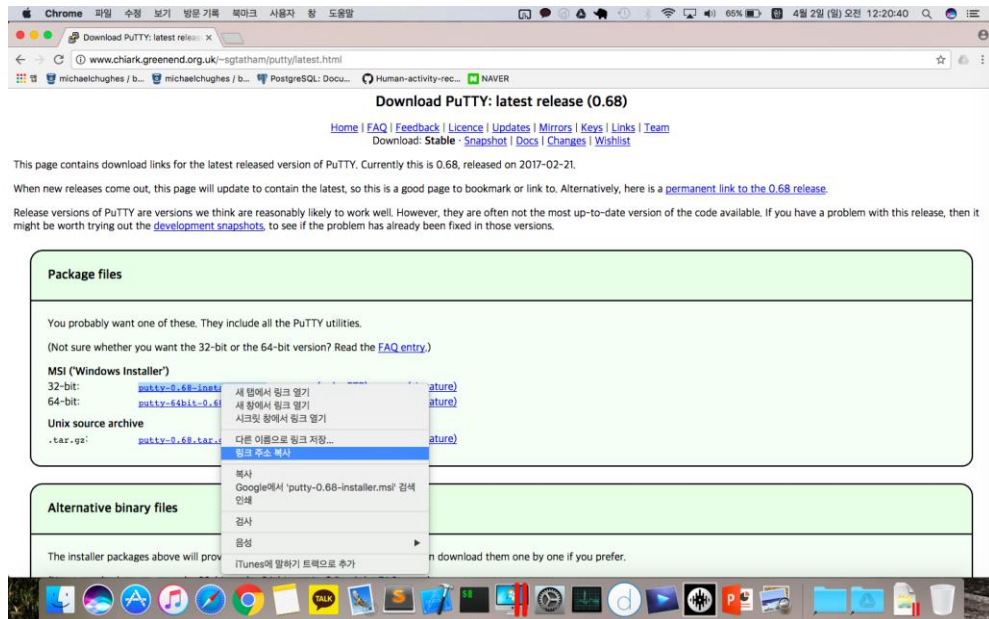
- 원격 서버로 파일 옮기기
  - `scp <옮길 파일이름> <서버 ip>@<username>:<옮길경로/파일이름>`



```
1. berty@berty-All-Series: ~/Desktop/Desktop/mkdir_test (bash)
→ mkdir_test scp renamed_myfile.txt mlg@141.223.162. :~/jiyuu/test/transferred.txt
mlg@141.223.162 's password:
renamed_myfile.txt          100%   0   0.0KB/s   00:00
→ mkdir_test
```

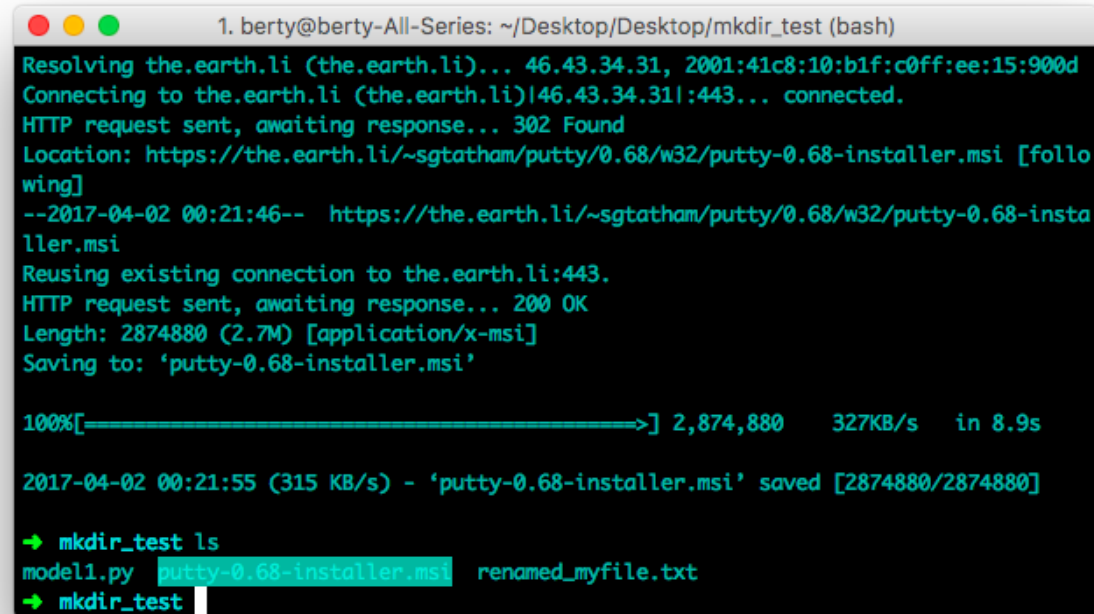
# 파일 관련 명령어

- 웹에서 파일 다운로드 하기
- 윈도우는 UI기반이라 단순히 클릭만 하면 가능
- 터미널로 접속하여 사용할 땐 어떻게 다운로드?
  - `wget <파일 URL>`



# 파일 관련 명령어

- 웹에서 파일 다운로드 하기
- 윈도우는 UI기반이라 단순히 클릭만 하면 가능
- 터미널로 접속하여 사용할 땐 어떻게 다운로드?
  - `wget <파일 URL>`



```
1. berty@berty-All-Series: ~/Desktop/Desktop/mkdir_test (bash)
Resolving the.earth.li (the.earth.li)... 46.43.34.31, 2001:41c8:10:b1f:c0ff:ee:15:900d
Connecting to the.earth.li (the.earth.li)|46.43.34.31|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://the.earth.li/~sgtatham/putty/0.68/w32/putty-0.68-installer.msi [following]
--2017-04-02 00:21:46-- https://the.earth.li/~sgtatham/putty/0.68/w32/putty-0.68-installer.msi
Reusing existing connection to the.earth.li:443.
HTTP request sent, awaiting response... 200 OK
Length: 2874880 (2.7M) [application/x-msi]
Saving to: 'putty-0.68-installer.msi'

100%[=====>] 2,874,880 327KB/s in 8.9s

2017-04-02 00:21:55 (315 KB/s) - 'putty-0.68-installer.msi' saved [2874880/2874880]

→ mkdir_test ls
model1.py putty-0.68-installer.msi renamed_myfile.txt
→ mkdir_test
```

# 파일 관련 명령어

- 파일 압축하기

- `tar -cvzf <압축파일 이름> <압축할 디렉토리>`
  - tar: 단순히 여러 파일을 하나로 합침
  - gunzip (gz): 파일 하나를 압축
  - tar를 이용하여 여러 파일을 하나로 묶고, 묶인 파일을 gz로 압축
  - -c: 새로운 묶음파일 생성
  - -v: 압축시 진행률 보여줌
  - -z: gzip 압축 및 압축해제
  - -f: 파일 이름 지정

# 파일 관련 명령어

- 파일 압축하기
  - `tar -cvzf <압축파일 이름> <압축할 디렉토리>`

A terminal window with a dark background and light green text. The window title is "1. berty@berty-All-Series: ~/Desktop/Desktop (bash)". The terminal shows a series of commands and their outputs. The user creates a directory named 'mkdir\_test' and moves into it. They then run 'ls' to see the contents: 'model1.py', 'putty-0.68-installer.msi', and 'renamed\_myfile.txt'. Next, they run 'cd ..' to move back to the parent directory. They run 'ls' again to see the new directory listed. Then, they run 'tar -cvzf mkdir\_test.tar.gz mkdir\_test' to create a compressed archive. The output shows the files being archived: 'mkdir\_test/', 'mkdir\_test/renamed\_myfile.txt', 'mkdir\_test/putty-0.68-installer.msi', and 'mkdir\_test/model1.py'. Finally, they run 'ls' to confirm the archive 'mkdir\_test.tar.gz' has been created.

```
1. berty@berty-All-Series: ~/Desktop/Desktop (bash)
→ mkdir_test ls
model1.py  putty-0.68-installer.msi  renamed_myfile.txt
→ mkdir_test cd ..
→ Desktop ls
mkdir_test
→ Desktop tar -cvzf mkdir_test.tar.gz mkdir_test
mkdir_test/
mkdir_test/renamed_myfile.txt
mkdir_test/putty-0.68-installer.msi
mkdir_test/model1.py
→ Desktop ls
mkdir_test  mkdir_test.tar.gz
→ Desktop
```

# 파일 관련 명령어

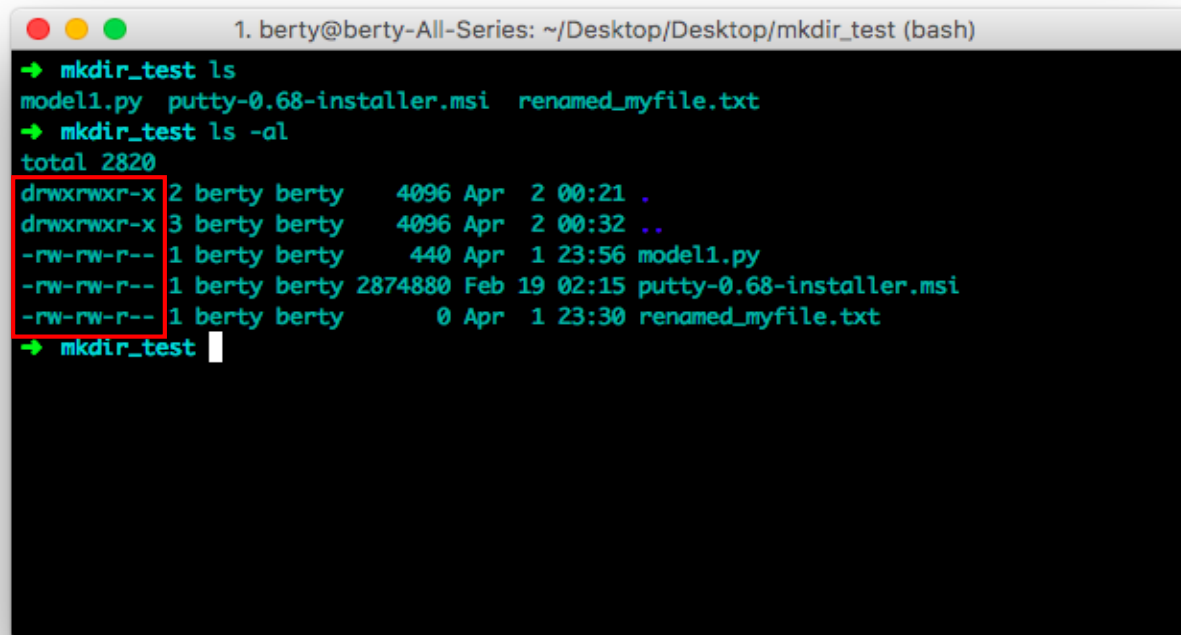
- 압축파일 풀기
  - `tar -xvzf <> <>`

A terminal window with a dark background and light-colored text. The window title is "1. berty@berty-All-Series: ~/Desktop/Desktop (bash)". The terminal shows a series of commands and their outputs. The commands are: "ls", "rm -rf mkdir\_test", "ls", "tar -xvzf mkdir\_test.tar.gz", "ls", and "ls". The outputs are: "mkdir\_test", "mkdir\_test/renamed\_myfile.txt", "mkdir\_test/putty-0.68-installer.msi", "mkdir\_test/model1.py", and "mkdir\_test". The terminal also shows the creation of the "mkdir\_test" directory and the extraction of files from "mkdir\_test.tar.gz".

```
1. berty@berty-All-Series: ~/Desktop/Desktop (bash)
→ Desktop ls
mkdir_test mkdir_test.tar.gz
→ Desktop rm -rf mkdir_test
→ Desktop ls
mkdir_test.tar.gz
→ Desktop tar -xvzf mkdir_test.tar.gz
mkdir_test/
mkdir_test/renamed_myfile.txt
mkdir_test/putty-0.68-installer.msi
mkdir_test/model1.py
→ Desktop ls
mkdir_test mkdir_test.tar.gz
→ Desktop
```

# 파일 관련 명령어

- 파일 권한 조회
  - `ls -al` 결과를 자세히 살펴봅시다
  - d: 디렉토리, r: 읽기, w: 쓰기, x: 실행
  - [디렉토리 1자리][소유자권한 3자리][그룹권한 3자리][전체권한 3자리]



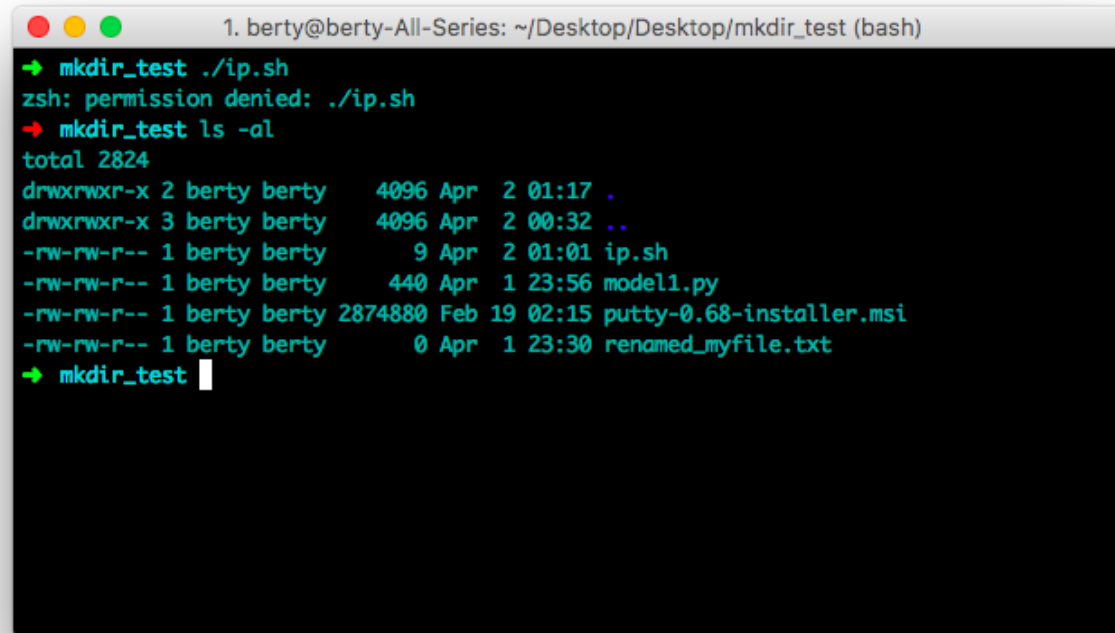
```
1. berty@berty-All-Series: ~/Desktop/Desktop/mkdir_test (bash)
→ mkdir_test ls
model1.py  putty-0.68-installer.msi  renamed_myfile.txt
→ mkdir_test ls -al
total 2820
drwxrwxr-x 2 berty berty 4096 Apr  2 00:21 .
drwxrwxr-x 3 berty berty 4096 Apr  2 00:32 ..
-rw-rw-r-- 1 berty berty 440 Apr  1 23:56 model1.py
-rw-rw-r-- 1 berty berty 2874880 Feb 19 02:15 putty-0.68-installer.msi
-rw-rw-r-- 1 berty berty 0 Apr  1 23:30 renamed_myfile.txt
→ mkdir_test
```

The terminal window shows the output of the `ls -al` command. The permissions for the directory `.` and `..` are `drwxrwxr-x`, which are highlighted with a red box. The permissions for the files `model1.py`, `putty-0.68-installer.msi`, and `renamed_myfile.txt` are `-rw-rw-r--`.



# 파일 관련 명령어

- 파일 접근 권한 변경
  - Wget을 이용하여 쉘파일 다운로드
    - `wget http://jungsoo.postech.ac.kr/ip.sh`
  - `./ip.sh` 명령어로 실행해 보세요.
  - 실행이 되지 않는다면 `ls -al` 명령어로 접근권한을 확인해보세요



```
1. berty@berty-All-Series: ~/Desktop/Desktop/mkdir_test (bash)
→ mkdir_test ./ip.sh
zsh: permission denied: ./ip.sh
→ mkdir_test ls -al
total 2824
drwxrwxr-x 2 berty berty 4096 Apr  2 01:17 .
drwxrwxr-x 3 berty berty 4096 Apr  2 00:32 ..
-rw-rw-r-- 1 berty berty   9 Apr  2 01:01 ip.sh
-rw-rw-r-- 1 berty berty  440 Apr  1 23:56 model1.py
-rw-rw-r-- 1 berty berty 2874880 Feb 19 02:15 putty-0.68-installer.msi
-rw-rw-r-- 1 berty berty   0 Apr  1 23:30 renamed_myfile.txt
→ mkdir_test
```

# 파일 관련 명령어

- 파일 접근 권한 변경
  - `chmod +x <변경할 파일 이름>`
  - 위 명령어는 대상 파일에 실행권한을 추가하는 명령어입니다

```
1. berty@berty-All-Series: ~/Desktop/Desktop/mkdir_test (bash)
→ mkdir_test chmod +x ip.sh
→ mkdir_test ls -al
total 2824
drwxrwxr-x 2 berty berty 4096 Apr  2 01:17 .
drwxrwxr-x 3 berty berty 4096 Apr  2 00:32 ..
-rwxrwxr-x 1 berty berty   9 Apr  2 01:01 ip.sh
-rw-rw-r-- 1 berty berty  440 Apr  1 23:56 model1.py
-rw-rw-r-- 1 berty berty 2874880 Feb 19 02:15 putty-0.68-installer.msi
-rw-rw-r-- 1 berty berty    0 Apr  1 23:30 renamed_myfile.txt
→ mkdir_test
```

```
1. berty@berty-All-Series: ~/Desktop/Desktop/mkdir_test (bash)
→ mkdir_test ./ip.sh
eth0      Link encap:Ethernet  HWaddr d8:50:e6:4e:67:8d
          inet addr:141.223.162.105  Bcast:141.223.162.255  Mask:255.255.255.0
          inet6 addr: fe80::da50:e6ff:fe4e:678d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8140637 errors:0 dropped:285384 overruns:0 frame:0
          TX packets:1620426 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3994231099 (3.9 GB)  TX bytes:181549834 (181.5 MB)

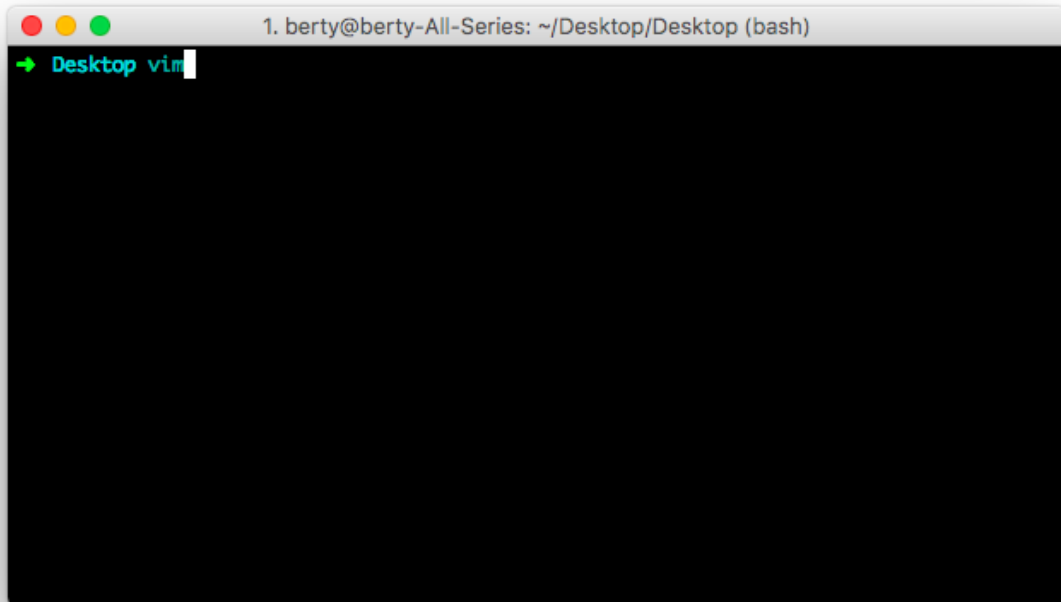
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:3356 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3356 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1606528 (1.6 MB)  TX bytes:1606528 (1.6 MB)
```

# sudo

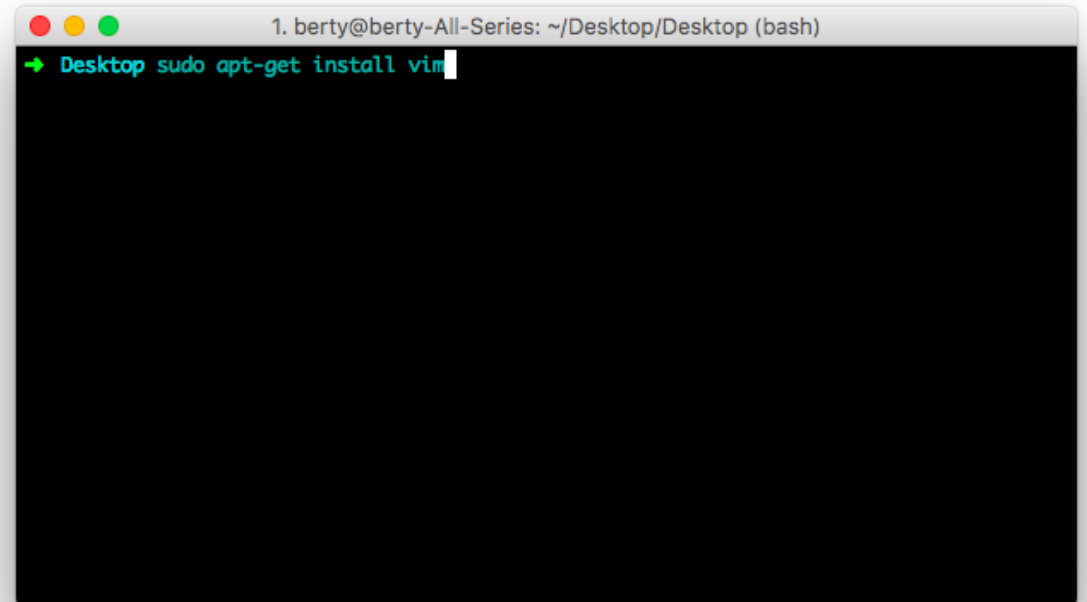
- sudo <실행할 명령어>
  - 앞으로 여러 패키지를 설치하는 과정에서 root 권한이 필요할 수 있음
  - sudo 명령어는 root권한 대행으로 원하는 명령어를 실행할 때 사용
  - 바로 다음 슬라이드에서 사용해 봅시다

# apt-get

- apt-get: 리눅스 소프트웨어를 설치하고 제거하는 일을 함.
  - 소스코드 컴파일을 통해 소프트웨어 패키지 확인, 구성, 설치 자동화
  - vim을 설치해봅시다
  - sudo를 붙이는 이유는 시스템에 소프트웨어를 설치하기 위해 root 권한이 필요하기 때문
  - `sudo apt-get install <패키지 이름>`



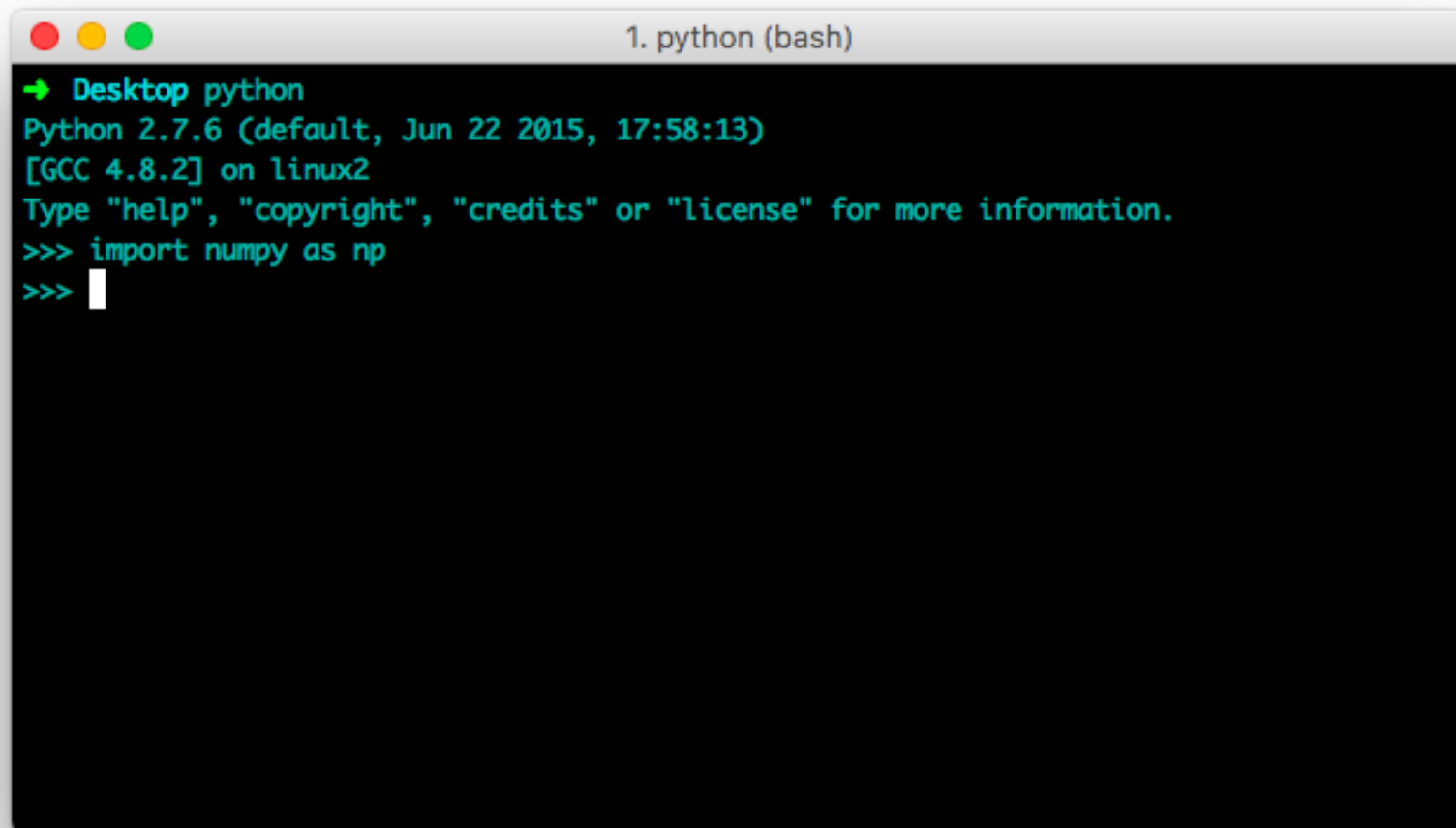
A terminal window with a title bar showing '1. berty@berty-All-Series: ~/Desktop/Desktop (bash)'. The prompt is '→ Desktop vim' with a cursor at the end.



A terminal window with a title bar showing '1. berty@berty-All-Series: ~/Desktop/Desktop (bash)'. The prompt is '→ Desktop sudo apt-get install vim' with a cursor at the end.

# pip

- Python에서 numpy 모듈 import 해보기



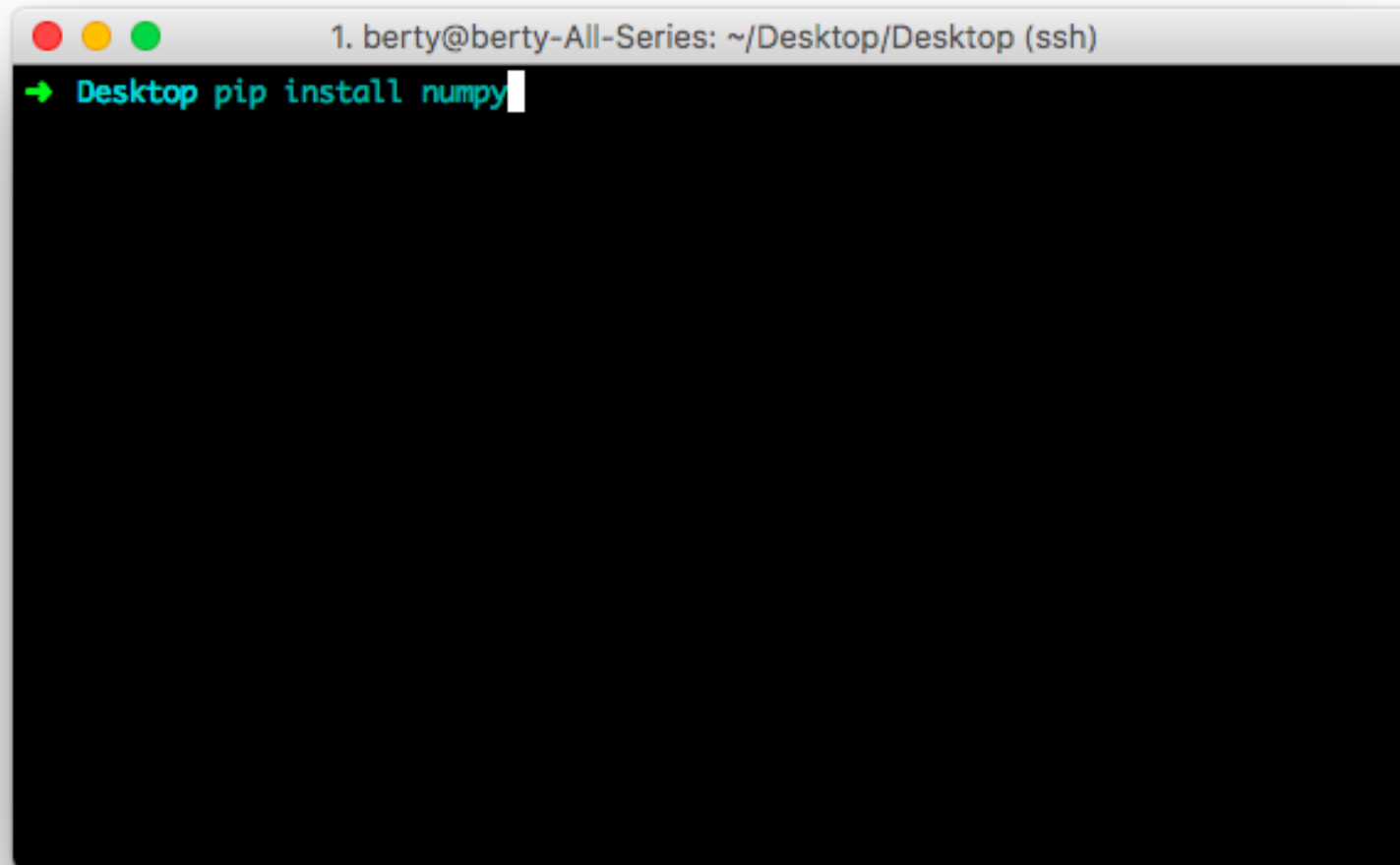
```
1. python (bash)
→ Desktop python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> 
```

# pip

- pip : python 패키지 관리자. Python에서 사용되는 모듈을 쉽게 설치할 수 있도록 함.
  - 앞으로 python에서 필요한 모듈들을 이 명령어로 많이 설치 할 겁니다
- Pip 설치: `sudo apt-get install python-pip`
  - Pip: python의 패키지 관리자. Python에서 사용되는 모듈을 쉽게 설치할 수 있도록 합니다.

# pip

- Numpy 설치하고 import 해보기

A terminal window with a light gray title bar containing three colored window control buttons (red, yellow, green) on the left and the text '1. berty@berty-All-Series: ~/Desktop/Desktop (ssh)' on the right. The main area of the terminal is black. A green prompt character '➔' is followed by the text 'Desktop pip install numpy' in a light blue/cyan font. A white cursor is positioned at the end of the command.

```
➔ Desktop pip install numpy
```

# man

- `man <명령어>`
  - 명령어 사용 매뉴얼을 볼 수 있습니다.

```
1. berty@berty-All-Series: ~/Desktop/Desktop/mkdir_test (bash)
→ mkdir_test man scp
```

```
1. man scp (bash)
SCP(1)                                BSD General Commands Manual                                SCP(1)

NAME
    scp - secure copy (remote file copy program)

SYNOPSIS
    scp [-123468Cpqrvt] [-c cipher] [-F ssh_config] [-i identity_file] [-l limit]
        [-o ssh_option] [-P port] [-S program] [[user@]host1:]file1 ...
        [[user@]host2:]file2

DESCRIPTION
    scp copies files between hosts on a network. It uses ssh(1) for data transfer,
    and uses the same authentication and provides the same security as ssh(1).
    Unlike rcp(1), scp will ask for passwords or passphrases if they are needed for
    authentication.

    File names may contain a user and host specification to indicate that the file
    is to be copied to/from that host. Local file names can be made explicit using
    Manual page scp(1) line 1 (press h for help or q to quit)
```



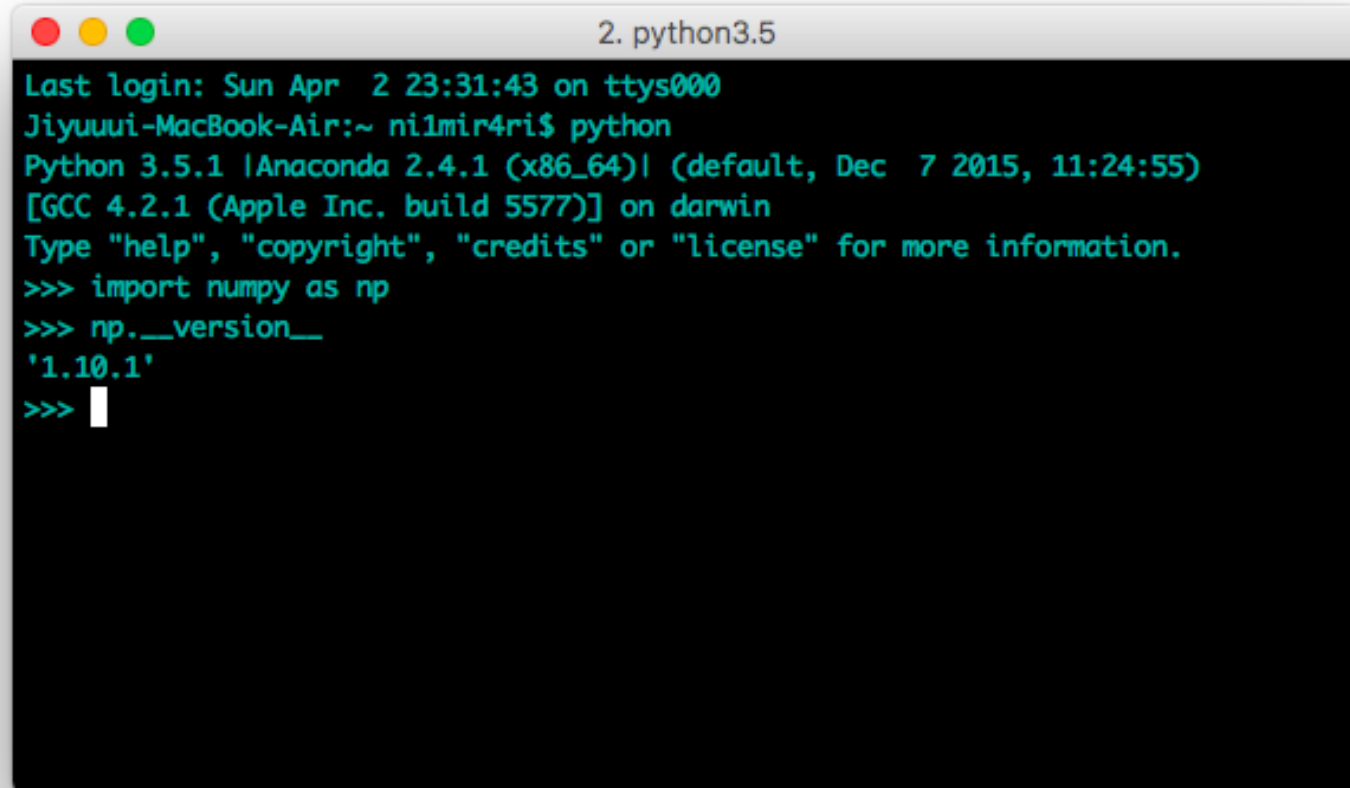
Numpy

# Numpy?

- 데이터/수치분석을 위한 python 모듈
- 행렬연산에 특화
- Matlab의 python 버전이라 생각할 수 있습니다
- Python으로 만들어지는 많은 머신러닝 모듈의 베이스가 됨

# Numpy 설치 확인

- import numpy as np
- np.\_\_version\_\_

A terminal window titled '2. python3.5' with a dark background and light green text. It shows the output of running 'python' in a shell, including login information, Python version (3.5.1), and the successful import of numpy with version 1.10.1.

```
2. python3.5
Last login: Sun Apr  2 23:31:43 on ttys000
Jiyuuui-MacBook-Air:~ nilmir4ri$ python
Python 3.5.1 |Anaconda 2.4.1 (x86_64)| (default, Dec  7 2015, 11:24:55)
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> np.__version__
'1.10.1'
>>> 
```

# Numpy 성능확인

```
performance.py  UNREGISTERED

1  import numpy as np
2  import timeit
3
4  # 0~999를 담고있는 python list 만들기
5  num_list = range(1000)
6  # 0~999를 담고있는 numpy array (vector) 만들기
7  num_np_array = np.arange(1000)
8
9  # python list에 있는 모든 수의 제곱을 구함
10 start = timeit.default_timer()
11 squared_list = [i**2 for i in num_list]
12 end = timeit.default_timer()
13 # 실행시간 출력
14 print('python list: %lf 초' % (end - start))
15
16 # numpy array에 있는 모든 수의 제곱을 구함
17 start = timeit.default_timer()
18 squared_np_array = num_np_array**2
19 end = timeit.default_timer()
20 # 실행시간 출력
21 print('numpy array: %lf 초' % (end - start))
22
```

Line 12, Column 29   Tab Size: 4   Python

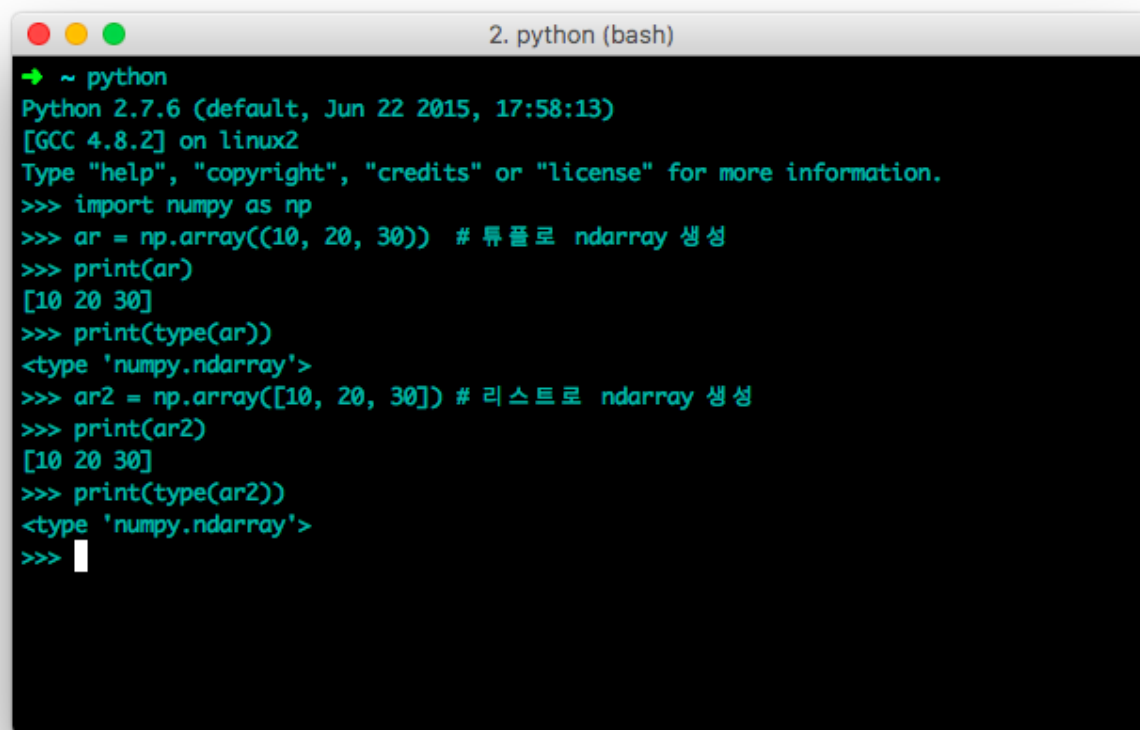
```
2. bash

Jiyuuui-MacBook-Air:numpy_practice  $ python performance.py
python list: 0.000473 초
numpy array: 0.000025 초
Jiyuuui-MacBook-Air:numpy_practice  $
```

- 같은 작업인데도 numpy array를 이용한 연산이 약 19배 빠르다

# ndarray

- N-dimensional array
  - Matlab에서 사용하는 vector나 matrix로 생각할 수 있다
- 같은 종류 (type)의 데이터만 담을 수 있다
- 리스트, 튜플로 부터 ndarray 생성 가능



```
2. python (bash)
→ ~ python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> ar = np.array((10, 20, 30)) # 튜플로 ndarray 생성
>>> print(ar)
[10 20 30]
>>> print(type(ar))
<type 'numpy.ndarray'>
>>> ar2 = np.array([10, 20, 30]) # 리스트로 ndarray 생성
>>> print(ar2)
[10 20 30]
>>> print(type(ar2))
<type 'numpy.ndarray'>
>>> 
```

# ndarray

- 인덱스는 0부터 시작

```
2. python (bash)
>>>
>>>
>>> print(ar[0]) # 인덱스는 0부터 시작
10
>>> 
```

- Python for문을 이용하여 ndarray의 각 요소들을 가져올 수 있음

```
2. python (bash)
>>> for num in ar:
...     print(num)
...
10
20
30
>>> 
```

# Ndarray의 shape

- 이중 리스트를 이용하여 2차원 행렬을 만들 수도 있음
- np.shape() 함수는 ndarray의 모양을 얻을 수 있는 함수

```
2. python (bash)
>>>
>>>
>>> list_2d = [[1,2,3],[4,5,6]]
>>> mat = np.array(list_2d)
>>> print(np.shape(mat))
(2, 3)
>>> 
```

- 다차원 ndarray의 인덱싱도 python list와 똑같이 할 수 있음

```
2. python (bash)
>>> print( mat[0][0] )
1
>>> print( mat[0][1] )
2
>>> print( mat[1][0] )
4
>>> 
```

# Ndarray의 shape

- Ndarray의 shape: ndarray의 모양. 벡터 혹은 행렬의 차원
- np.reshape() 는 ndarray의 shape를 바꾸는 함수
  - 첫 번째 인자: shape를 바꿀 ndarray 변수
  - 두 번째 인자: 새로운 shape. Shape 또한 python list나 tuple로 표현

```
2. python (bash)
>>> print( np.shape(mat) )
(2, 3)
>>> flatten = np.reshape( mat, (6) )
>>> print(flatten)
[1 2 3 4 5 6]
>>> print(np.shape(flatten))
(6,)
>>> 
```



# Ndarray의 transpose

- 머신러닝 알고리즘을 구현하다보면 복잡한 행렬 연산들을 해야함
  - 그 과정에서 차원을 바꾸는 일이 많을 수 있습니다.
- `np.transpose()`
  - 전치행렬 구하기

```
2. python
>>> transposed_mat = np.transpose(mat)
>>> print( transposed_mat )
[[1 4]
 [2 5]
 [3 6]]
>>> print( np.shape(transposed_mat) )
(3, 2)
>>>
```

```
2. python (bash)
>>> mat
array([[1, 2, 3],
       [4, 5, 6]])
>>> mat.T
array([[1, 4],
       [2, 5],
       [3, 6]])
>>> print( np.shape(mat) )
(2, 3)
>>> print( np.shape(mat.T) )
(3, 2)
>>>
```

# 0~26의 값을 갖는 3x3x3 ndarray 만들어보기

- Hint

- range() python 내장함수 이용
- ndarray를 만드는 np.array() 함수
- Nddarray의 shape를 바꾸는 np.reshape() 이용

- 직접 해봅시다

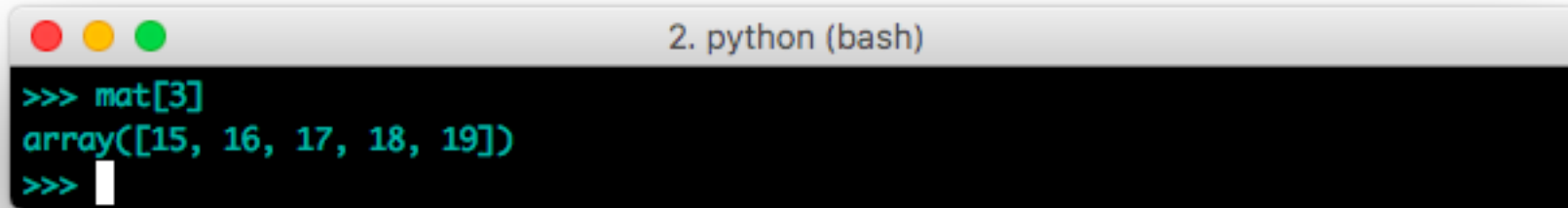
# Ndarray의 slicing

- Python slicing과 비슷하다
  - 하지만 numpy는 추가적인 기능 제공: 다차원, 값 할당
- 실습을 위한 행렬 만들기

```
2. python (bash)
>>> mat = np.reshape( np.array(range(20)), (4, 5) )
>>> mat
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
>>> 
```

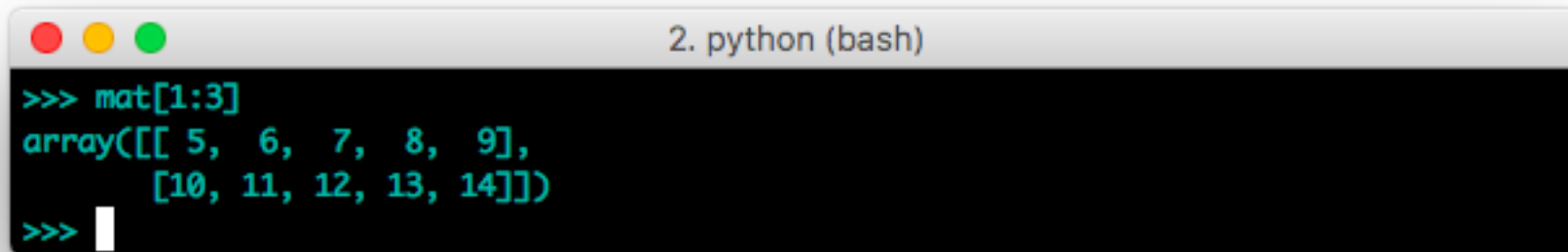
# Ndarray의 slicing

- 특정 행만 추출하기



```
2. python (bash)
>>> mat[3]
array([15, 16, 17, 18, 19])
>>>
```

- 몇 개의 행만 슬라이싱 하기



```
2. python (bash)
>>> mat[1:3]
array([[ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>>
```

# Ndarray의 slicing

- 처음 행부터 몇 개만 슬라이싱

```
2. python (bash)
>>> mat[ : 3]
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> 
```

- 특정 행부터 끝까지 슬라이싱

```
2. python (bash)
>>>
>>> mat[ 2 : ]
array([[10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
>>> 
```

# Ndarray의 slicing

- 다차원 ndarray 인덱싱

```
2. python (bash)
>>> mat[2, 1]
11
>>> 
```

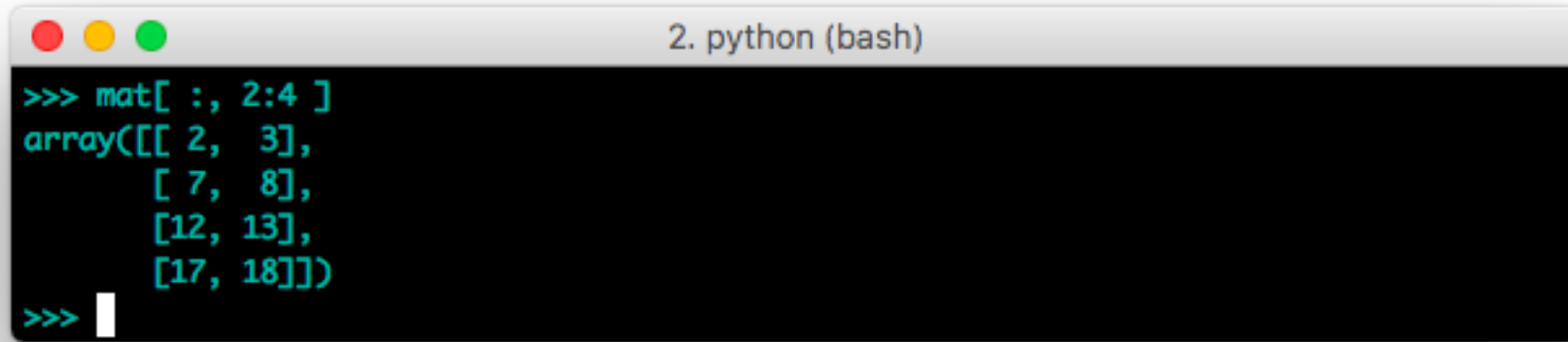
- 특정 열만 슬라이싱

- 행, 열 동시에 인덱싱 하는 것이 가능하다는 점을 이용

```
2. python (bash)
>>> mat[ : , 1]
array([ 1,  6, 11, 16])
>>> 
```

# Ndarray의 slicing

- 몇 개의 열만 슬라이싱

A terminal window titled "2. python (bash)" with a black background and green text. It shows a Python command to slice a matrix 'mat' to get columns 2 through 4. The output is a 4x2 array.

```
>>> mat[ :, 2:4 ]  
array([[ 2,  3],  
       [ 7,  8],  
       [12, 13],  
       [17, 18]])  
>>> 
```

- 행과 마찬가지로
  - 처음부터 특정 열까지, 특정 열부터 끝까지 슬라이싱도 가능

# Python list와 ndarray를 이용한 인덱싱

- Ndarrray를 이용한 인덱싱

```
2. python (bash)
>>> indices = np.array([1, 3, 4])
>>> mat[ : , indices ]
array([[ 1,  3,  4],
       [ 6,  8,  9],
       [11, 13, 14],
       [16, 18, 19]])
>>>
```

- Python list를 이용한 인덱싱

```
2. python (bash)
>>> python_list = [1, 2]
>>> mat[ python_list, : ]
array([[ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>>
```



# 조건부 인덱싱

- 10보다 크거나 같은 요소만 추출하기

```
2. python (bash)
>>> cond_indices = mat >= 10
>>> cond_indices
array([[False, False, False, False, False],
       [False, False, False, False, False],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True]], dtype=bool)
>>> mat[cond_indices]
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
>>>
```

- 짝수인 요소만 추출해보세요.

# narray 연산자

- + : 전체 요소에 덧셈

```
2. python (bash)
>>> mat + 3
array([[ 3,  4,  5,  6,  7],
       [ 8,  9, 10, 11, 12],
       [13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22]])
>>> 
```

- - : 전체 요소에 뺄셈

```
2. python (bash)
>>> mat - 5
array([[ -5,  -4,  -3,  -2,  -1],
       [  0,   1,   2,   3,   4],
       [  5,   6,   7,   8,   9],
       [10, 11, 12, 13, 14]])
>>> 
```

# narray 연산자

- \* : 전체 요소에 곱셈

```
2. python (bash)
>>> mat * 2
array([[ 0,  2,  4,  6,  8],
       [10, 12, 14, 16, 18],
       [20, 22, 24, 26, 28],
       [30, 32, 34, 36, 38]])
>>>
```

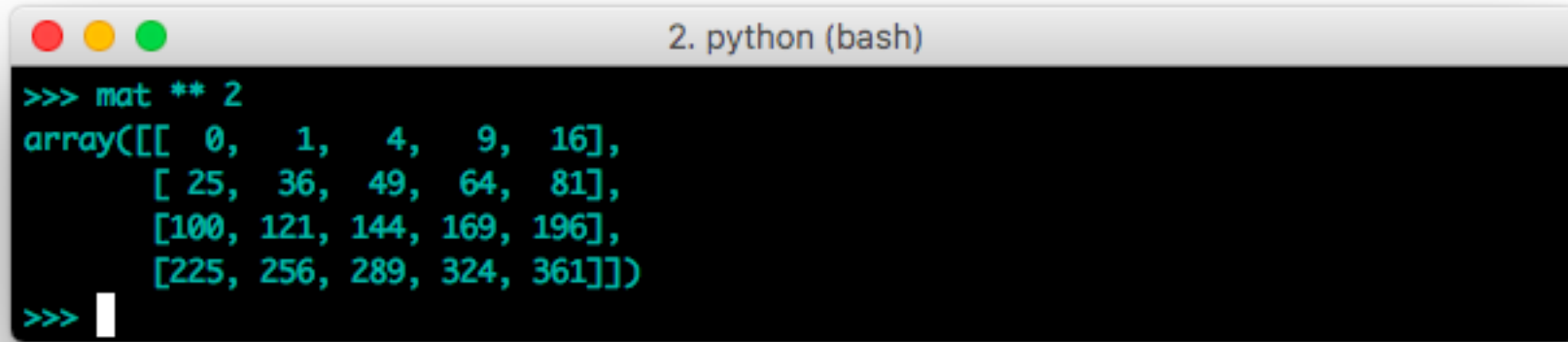
- / : 전체 요소에 나눗셈

```
2. python (bash)
>>> mat / 2
array([[0, 0, 1, 1, 2],
       [2, 3, 3, 4, 4],
       [5, 5, 6, 6, 7],
       [7, 8, 8, 9, 9]])
>>>
```

- 자료형 문제로 결과가 정확하지 않을겁니다.

# numpy 연산자

- \*\* : 전체 요소에 n승



```
2. python (bash)
>>> mat ** 2
array([[ 0,  1,  4,  9, 16],
       [25, 36, 49, 64, 81],
       [100, 121, 144, 169, 196],
       [225, 256, 289, 324, 361]])
>>>
```

- Python에서 사용하는 연산자는 거의 쓸 수 있습니다

# 행렬간의 요소별 연산

- 실습을 위한 두 번째 행렬 생성

```
2. python (bash)

>>> mat2 = - ( mat + 1 )
>>> mat2
array([[ -1,  -2,  -3,  -4,  -5],
       [ -6,  -7,  -8,  -9, -10],
       [-11, -12, -13, -14, -15],
       [-16, -17, -18, -19, -20]])
>>> 
```

- 두 행렬의 요소별 덧셈

```
2. python (bash)

>>> elementwise_add = mat + mat2
>>> elementwise_add
array([[ -1,  -1,  -1,  -1,  -1],
       [ -1,  -1,  -1,  -1,  -1],
       [ -1,  -1,  -1,  -1,  -1],
       [ -1,  -1,  -1,  -1,  -1]])
>>> 
```

# 행렬과 벡터의 요소별 연산

- 실습을 위한 벡터 생성

```
2. python (bash)
>>> vector = np.array([1, 2, 3, 4, 5])
>>> 
```

- 행렬의 각 행과 벡터의 요소별 덧셈

```
2. python (bash)
>>> mat + vector
array([[ 1,  3,  5,  7,  9],
       [ 6,  8, 10, 12, 14],
       [11, 13, 15, 17, 19],
       [16, 18, 20, 22, 24]])
>>> 
```

# 행렬과 벡터의 요소별 연산

- 실습을 위한 벡터 생성

```
2. python (bash)
>>> vector = np.array([1, 2, 3, 4, 5])
>>> 
```

- 행렬의 각 행과 벡터의 요소별 덧셈

```
2. python (bash)
>>> mat + vector
array([[ 1,  3,  5,  7,  9],
       [ 6,  8, 10, 12, 14],
       [11, 13, 15, 17, 19],
       [16, 18, 20, 22, 24]])
>>> 
```

# 행렬과 벡터의 요소별 연산

- 실습을 위한 벡터 만들기

```
2. python (bash)
>>> vector = np.array([[2], [3], [4], [5]])
>>> vector
array([[2],
       [3],
       [4],
       [5]])
>>> 
```

- 행렬의 각 열과 벡터의 요소별 덧셈

```
2. python (bash)
>>> mat + vector
array([[ 2,  3,  4,  5,  6],
       [ 8,  9, 10, 11, 12],
       [14, 15, 16, 17, 18],
       [20, 21, 22, 23, 24]])
>>> 
```

- 아까 모든 요소에대한 연산과 마찬가지로 다른 연산자에도 똑같이 적용됨
- ndarray의 차원이 더 높아져도 적용됨



# 행렬간의 곱셈

- 선형대수학에서 배웠던 행렬간의 곱셈
- `np.dot(행렬1, 행렬2)`
- 실습을 위한 두 행렬 만들기

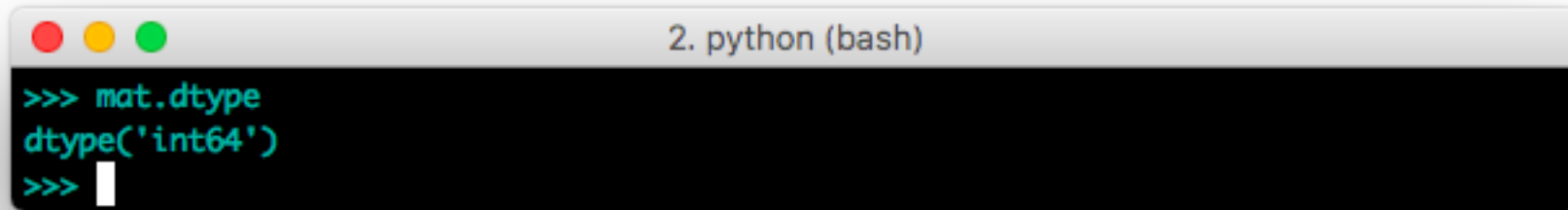
```
2. python (bash)
>>> mat1 = np.array([[1,1,1], [2,2,2]])
>>> mat2 = np.array([[1, 4], [2, 5], [3, 6]])
>>>
```

- 두 행렬의 곱

```
2. python (bash)
>>> np.dot(mat1, mat2)
array([[ 6, 15],
       [12, 30]])
>>>
```

# Numpy 데이터 타입

- ndarray.dtype

A terminal window titled "2. python (bash)" with a black background and green text. It shows the command >>> mat.dtype and the output dtype('int64'). A cursor is visible on the line >>>.

```
>>> mat.dtype
dtype('int64')
>>>
```

- 다양한 데이터 타입

- bool
- int
- float
- str
- ...

- 더 많은 정보는

[https://www.tutorialspoint.com/numpy/numpy\\_data\\_types.htm](https://www.tutorialspoint.com/numpy/numpy_data_types.htm)

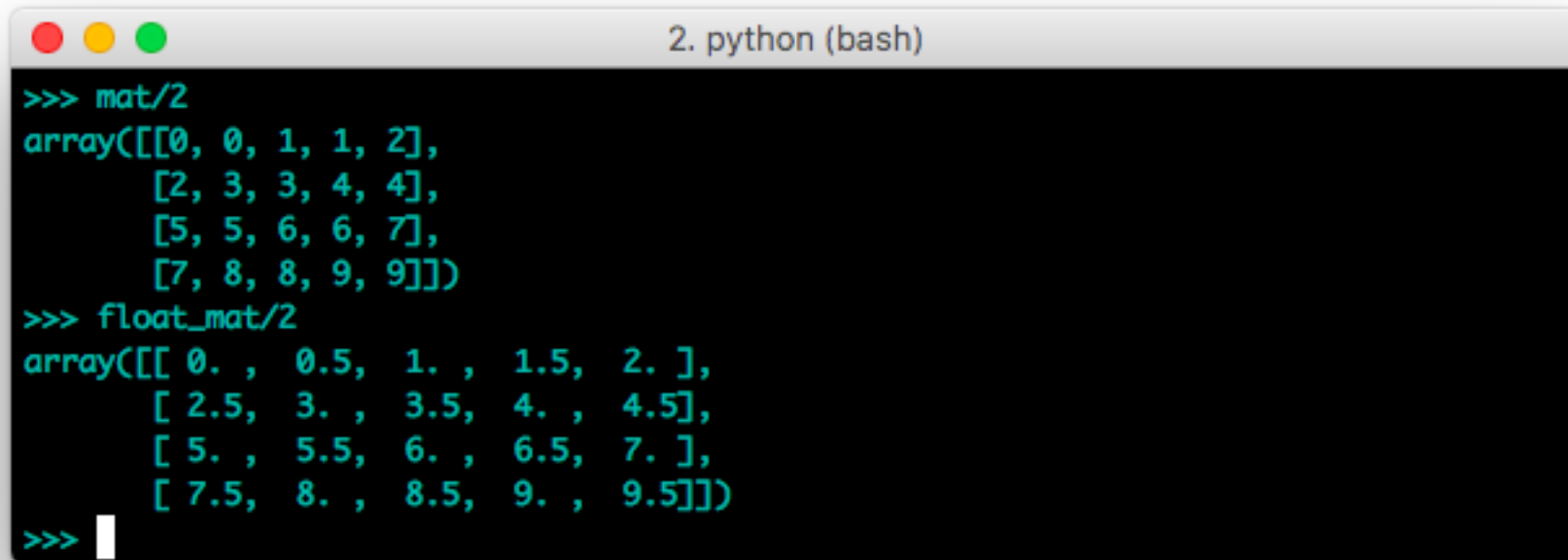
# Numpy 데이터 타입

- `ndarray.astype(타입명)`

```
2. python (bash)
>>> float_mat = mat.astype(np.float)
>>> float_mat
array([[ 0.,  1.,  2.,  3.,  4.],
       [ 5.,  6.,  7.,  8.,  9.],
       [10., 11., 12., 13., 14.],
       [15., 16., 17., 18., 19.]])
>>> mat
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19.]])
>>> 
```

# Numpy 데이터 타입

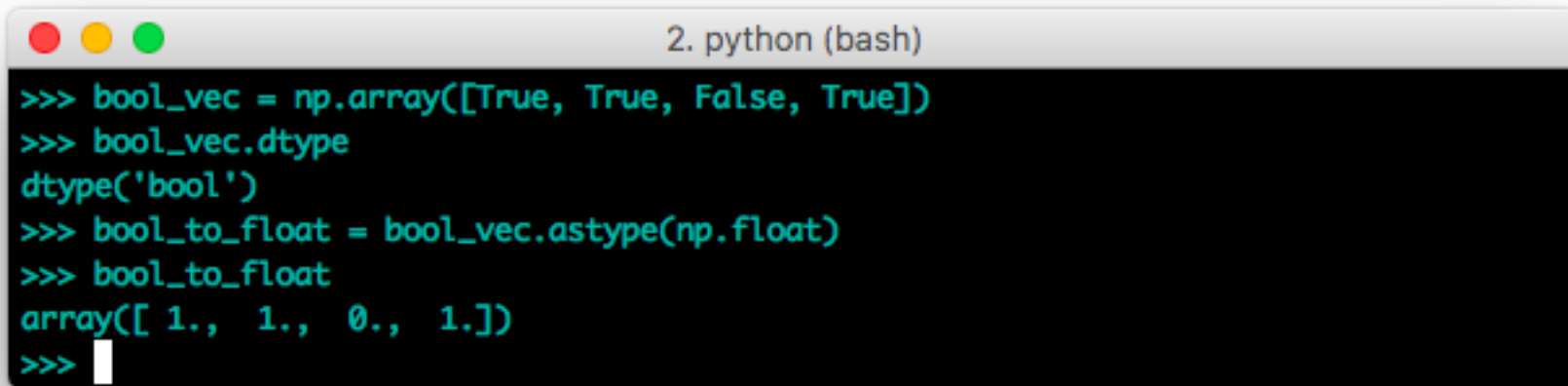
- `ndarray.astype(타입명)`
  - `np.float` 형으로 변경하면 나눗셈이 제대로 됩니다.



```
2. python (bash)
>>> mat/2
array([[0, 0, 1, 1, 2],
       [2, 3, 3, 4, 4],
       [5, 5, 6, 6, 7],
       [7, 8, 8, 9, 9]])
>>> float_mat/2
array([[ 0. ,  0.5,  1. ,  1.5,  2. ],
       [ 2.5,  3. ,  3.5,  4. ,  4.5],
       [ 5. ,  5.5,  6. ,  6.5,  7. ],
       [ 7.5,  8. ,  8.5,  9. ,  9.5]])
>>>
```

# Numpy 데이터 타입

- `ndarray.astype(타입명)`
  - bool 형도 float로 변환할 수 있습니다

A terminal window titled "2. python (bash)" with a dark background and light green text. It shows a sequence of Python commands and their outputs. The commands create a boolean array, check its dtype, and then convert it to a float array.

```
>>> bool_vec = np.array([True, True, False, True])
>>> bool_vec.dtype
dtype('bool')
>>> bool_to_float = bool_vec.astype(np.float)
>>> bool_to_float
array([ 1.,  1.,  0.,  1.])
>>> 
```

# 자주 사용할 numpy 함수

- `np.sum(합을 구할 행렬, axis=합을 구할 차원)`
  - 모든 요소의 합, 또는 차원별 합을 구할 수 있음

```
2. python (bash)
>>> np.sum( mat )
190
>>>
```

- 각 행의 열에대한 합

```
2. python (bash)
>>> np.sum( mat, axis=1 )
array([10, 35, 60, 85])
>>>
```

- 각 열의 행에대한 합

```
2. python (bash)
>>> np.sum( mat, axis=0 )
array([30, 34, 38, 42, 46])
>>>
```

# 자주 사용할 numpy 함수

- `np.sum(합을 구할 행렬, axis=합을 구할 차원)`
  - 모든 요소의 합, 또는 차원별 합을 구할 수 있음

```
2. python (bash)
>>> np.sum( mat )
190
>>>
```

- 각 행의 열에대한 합

```
2. python (bash)
>>> np.sum( mat, axis=1 )
array([10, 35, 60, 85])
>>>
```

- 각 열의 행에대한 합

```
2. python (bash)
>>> np.sum( mat, axis=0 )
array([30, 34, 38, 42, 46])
>>>
```

# 자주 사용할 numpy 함수

- `np.mean(평균을 구할 행렬, axis=평균을 구할 차원)`
  - 모든 요소의 평균, 또는 차원별 평균을 구할 수 있음

```
2. python (bash) 🔔
>>> np.mean( mat )
9.5
>>> 
```

- 각 행의 열에대한 평균

```
2. python (bash)
>>> np.mean( mat, axis=1 )
array([ 2.,  7., 12., 17.])
>>> 
```

- 각 열의 행에대한 평균

```
2. python (bash)
>>> np.mean( mat, axis=0 )
array([ 7.5,  8.5,  9.5, 10.5, 11.5])
>>> 
```



# 자주 사용할 numpy 함수

- np.max(최댓값을 구할 행렬, axis=최댓값을 구할 차원)
  - 모든 요소 중 최댓값, 또는 차원별 최댓값을 구할 수 있음

```
2. python (bash)
>>> np.max( mat, axis=1 )
array([ 4,  9, 14, 19])
>>>
```

- np.min(최솟값을 구할 행렬, axis=최솟값을 구할 차원)
  - 모든 요소 중 최댓값, 또는 차원별 최댓값을 구할 수 있음

```
2. python (bash)
>>> np.min( mat, axis=0 )
array([0, 1, 2, 3, 4])
>>>
```

# 자주 사용할 numpy 함수

- np.argmax(행렬, axis=최댓값의 인덱스를 구할)
  - 모든 요소 중 최댓값의 인덱스

```
2. python (bash)
>>> np.argmax( mat )
19
>>>
```

- 행에서 최댓값의 인덱스

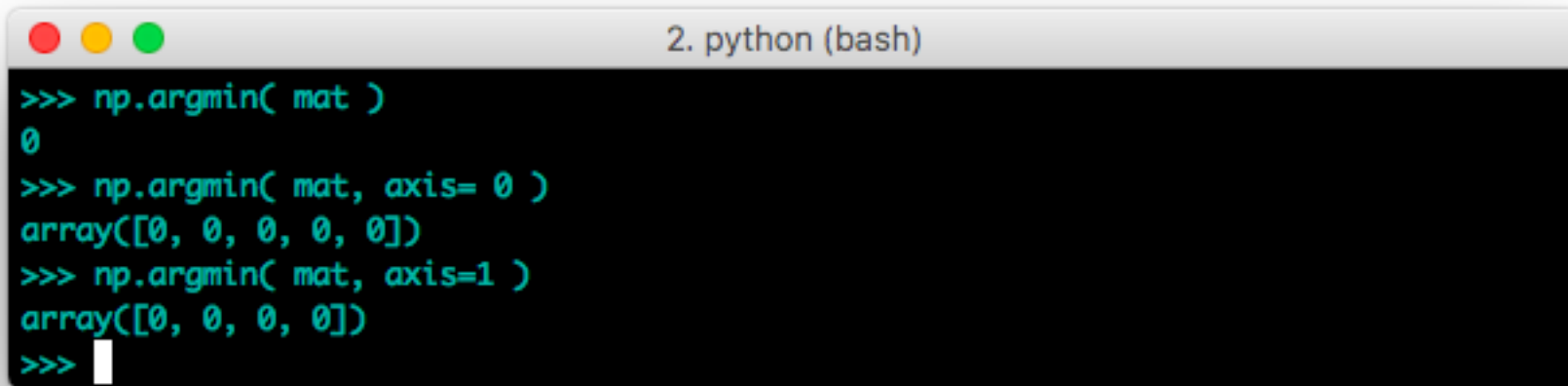
```
2. python (bash)
>>> np.argmax( mat, axis=1 )
array([4, 4, 4, 4])
>>>
```

- 열에서 최댓값의 인덱스

```
2. python (bash)
>>> np.argmax( mat, axis=0 )
array([3, 3, 3, 3, 3])
>>>
```

# 자주 사용할 numpy 함수

- np.argmin(행렬, axis=최솟값의 인덱스를 구할)
  - np.argmax()와 유사

A terminal window titled "2. python (bash)" with a dark background and light green text. It shows four lines of Python code and their outputs. The first line is a comment. The second line calls np.argmin on a variable 'mat' and returns 0. The third line calls np.argmin on 'mat' with axis=0 and returns an array of five 0s. The fourth line calls np.argmin on 'mat' with axis=1 and returns an array of four 0s. The prompt '>>>' is followed by a white cursor on the last line.

```
>>> np.argmin( mat )  
0  
>>> np.argmin( mat, axis= 0 )  
array([0, 0, 0, 0, 0])  
>>> np.argmin( mat, axis=1 )  
array([0, 0, 0, 0])  
>>> 
```

# 자주 사용할 numpy 함수

- `np.random.normal(평균, 표준편차, ndarray shape)`
  - 인자로 넘겨진 평균, 표준편차의 정규분포를 따르는 랜덤 값 생성
  - 머신러닝에서는 overfitting을 예방하기 위해 학습데이터에 정규분포를 따르는 잡음을 더하는 경우에 사용

```
1. python3.5
>>> np.random.normal( 0, 0.1, [2, 3] )
array([[ -0.05774687, -0.04996592, -0.12387002],
       [ -0.13395694,  0.10747827,  0.19811271]])
>>> np.random.normal( 10, 0.01, [3, 3, 2] )
array([[[ 10.00555815,  9.98903243],
        [ 9.98828184, 10.01734258],
        [ 9.99437717, 10.02196912]],

       [[ 10.00021965, 10.00248266],
        [ 9.99804544, 10.00014633],
        [ 9.99919242,  9.98977444]],

       [[ 10.0019738 ,  9.99197586],
        [ 10.00126182, 10.02242233],
        [ 9.97905301, 10.012992  ]]])
>>>
```

# 나머지 numpy 함수

- 모든 함수를 이 수업에서 다뤄볼 순 없습니다.
- 필요할 때 마다 구글링을 하세요
- Reference를 봐도 좋습니다
  - <https://docs.scipy.org/doc/numpy/reference/>

# Numpy 응용해보기

- 여러 점들과 특정 점의 유클리드 거리 구하기

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

data1	3.0	1.3	7.6
data2	8.4	6.5	-32.6
data3	7.4	37.3	12.1
data4	6.9	-25.3	0.0
data5	3.2	-4.1	22.8

mean	4.3	4.2	-6.1
------	-----	-----	------

- `np.sqrt(ndarray)`: 모든 요소의 루트값을 구하는 함수

# Numpy 응용해보기

- 데이터를 3개의 클래스로 분류하는 문제
  - Ont-hot encoded 예측값과 정답값을 이용하여 정확도 구해보기

예측값	class1	class2	class3
data1	0.8	0.05	0.05
data2	0.1	0.2	0.7
data3	0.6	0.4	0.0
data4	0.3	0.4	0.3
data5	0.1	0.1	0.8

정답값	class1	class2	class3
data1	1.0	0.0	0.0
data2	0.0	0.0	1.0
data3	0.0	0.0	1.0
data4	0.0	1.0	0.0
data5	1.0	0.0	0.0

# Numpy 응용해보기

- 데이터를 3개의 클래스로 분류하는 문제
  - Ont-hot encoded 예측값과 정답값을 이용하여 cross entropy 구해보기

$$H_{y'}(y) = - \sum_i y'_i \log(y_i)$$

예측값	class1	class2	class3
data1	0.8	0.05	0.05
data2	0.1	0.2	0.7
data3	0.6	0.4	0.0
data4	0.3	0.4	0.3
data5	0.1	0.1	0.8

정답값	class1	class2	class3
data1	1.0	0.0	0.0
data2	0.0	0.0	1.0
data3	0.0	0.0	1.0
data4	0.0	1.0	0.0
data5	1.0	0.0	0.0



Thank you