

---

# Bioinformatics Python Programming

July 2019  
Han Joohyun

# Bioinfomatics Python Programming

01 강사 소개 및 강의 소개

02 초급자를 위한 Github

03 아나콘다 파이썬 설치 / 파이썬 라이브러리 설치 방법

04 기본 파일 종류 설명

05 GATK Best Practice Pipeline

06 Bioinformatics Python Programming

01

## 강사 소개 및 강의 소개

---

# Introduce

## 강사 소개

이름

한주현

소속

전) 마크로젠 데이터분석부  
현) 쓰리빌리언 Bioinformatics Engineer  
현) 서울대학교 의료정보학 전공 이규언 교수님 연구실

메일

[kenneth.jh.han@snu.ac.kr](mailto:kenneth.jh.han@snu.ac.kr)

주요 업무

Rare disease genetic algorithms  
Bioinformatics tool development  
Human genome analysis (WGS, WES)  
Analysis pipeline / platform Development  
Full stack development

주 언어

Python, JAVA, JavaScript, Bash shell

저서

니콜라스 볼커 이야기 (2016.10, 금창원 외 공역)  
바이오파이썬으로 만나는 생물정보학 (2019.03, 한주현)

강의

바이오협회 유전체 예비 전문가 과정 3-7기  
서울여자대학교 초청 강의

웹 페이지

<https://korbillgates.tistory.com> (블로그)



# 저의 관심사

---

- Bioinformatics for clinical genomics
- Machine learning in clinical genomics
- Algorithms in computational biology
- Application of NGS technology in precision medicine
- 도서 및 논문 출판
- Bioinformatics 강의

# Bioinformatics Python Programming

---



『생물정보학 파이썬 프로그래밍』은  
컴퓨터 프로그래밍 언어인  
[Python](#)으로 생물정보학 연구 및 업무에  
필요한 지식을 배우는 강좌입니다

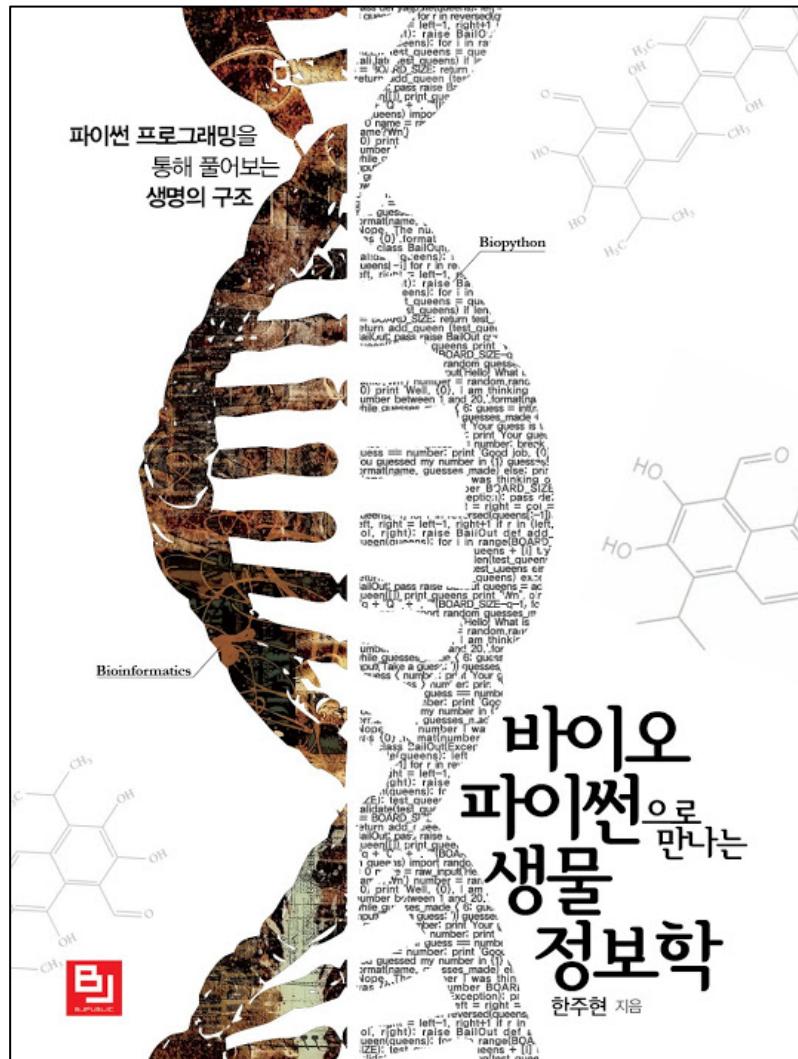
# Bioinformatics Python Programming

---

우리가 Bioinformatics Python Programming 과정에서 배울 내용들

- Server 환경에서의 Python Programming
- Github의 사용 방법
- 기본 텍스트 파일의 종류
- 기본적 Python Programming 실습
- Python으로 기본 텍스트 파일의 파싱
- DNA 분석 Pipeline의 개요 및 Input/Output 파일들의 소개와 파일 프로그래밍을 통한 파일의 파싱

# 바이오판으로 만나는 생물정보학



## 3.4 VCF

VCF(Variant Calling Format)는 변이(Variant)를 표기하기 위해 만든 포맷으로 크게 메타데이터와 내용 부분으로 나눌 수 있다. 내용 부분은 8개의 필수 열과 샘플에 따라 추가되는 열로 이루어진다. 각 열은 템으로 나누어 있다.

### ◆ 파일 형식:

#### - 메타데이터

메타데이터는 #(샵)으로 시작되는 부분이다. 2개의 샵으로 시작되는 부분은 VCF 파일에 대한 정보로 키=값(Key=Value) 관계로 표현한다. 어떠한 풀로 어떠한 분석을 진행하였는가에 따라 메타데이터에 쓰인 값은 다양하지만 일반적으로 쓰이는 값들에 대해 설명해보겠다.

| 값  | 설명  |   |
|--|---|---|
| ##fileformat=VCFv4.2   | VCF 파일 v4.2를 기준으로 생성한 파일을 의미한다.   |   |
| ##FORMAT= $D=AD, \dots$  | 구조로 구분되어 있으며 ref, alt를 순서에 맞게 표기한다.   |   |
| ,Description="Allelic depths for the ref and all alleles in the order listed") | 인코딩 특정 위치에서 REF에 G, ALT에 T, A가 있고 AD 열이 3, 10, 5로 있으면 REF 세대인 G 3개, ALT 세대인 T 10, A 5의 depth가 있음을 의미한다. |   |
| ##FORMAT= $D=DP, \dots$  | 위치의 depth를 나타낸다.  |   |
| ,Description="Approximate read depth")   | ##FILTER=<SNP_>,filter,Description="QD(2.0    MQ(40,0)"   | Description에 기재된 필터 조건에 해당하면 FILTER 열에 필터 이름에 해당하는 SNP_filter를 표기하고 조건에 해당하지 않는다면 PASS를 표기한다. |

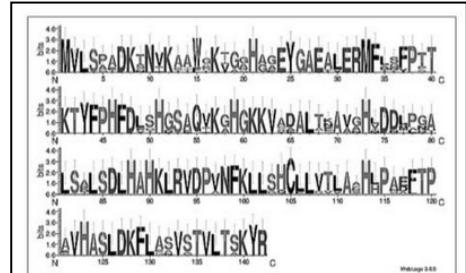
{ 표 3-8 VCF 메타데이터 }

1개의 샵으로 시작되는 부분은 헤더로 8개의 필수 열로 구성되어 있다.

| 값     | 설명                  | 예시       |
|-------|---------------------|----------|
| CHROM | chromosome, 염색체 번호. | chr19    |
| POS   | position, 위치.       | 45412079 |

43

섹션 1. 바이오판 이론



[그림 7-7] WebLogo 분석 결과

|       |  |
|-------|--|
| 문장 69 | 사전을 이용하여 아미노산 서열의 종류 개수 세기<br>다음 아미노산 서열을 아미노산 종류에 맞게 세어 출력하는 프로그램을 작성하십시오.<br><br>seq = "MLSSMPPGGLACHADDQII"                                 |
| 답안 69 | ##069.py<br><br>seq = "MLSSMPPGGLACHADDQII"<br><br>d = {}<br><br>for s in seq:<br>if s in d:<br>d[s] += 1<br>else:<br>d[s] = 1<br><br>print(d) |
| 답안 69 | {M: 2, L: 2, S: 3, P: 2, G: 2, A: 2, C: 1, H: 1, D: 3, Y: 2}   |

2019.03.22 출간 - 312p  
섹션1. 바이오판 라이브러리 활용  
섹션2. 생물정보학 파이썬 스크립팅 문제

참고도서 흥 흥  
구매자 분들께  
싸인해드림 (거부가능)



# 바이오파이썬으로 만나는 생물정보학 (리뷰)

★★★★★ 생물정보학을 위한 맞춤 파이썬 교재 [+ 구매]

현 시대는 정보의 시대이다. 이에 발 맞춰 생물정보학은 생물학의 가장자리에서 중심으로 그 자리 이동을 하였다. 우리나라에도 생물정보학이라는 학문이 더 이상 생소한 학문이 아니라 생물학을 하는데 반드시 필요한 학문이라는 인식들이 자리 잡았다. 애석하게도 이러한 생물정보학 지식을 전파하려는 책들은 많지 않다. 생물정보학이라는 학문을 하면서 필요한 지식들은 방대하다. 생물학적 지식부터 코딩능력까지 배양하는 것은 쉽지 않은 일이다. 코딩분야만 해도 JAVA, C, C++, php, R, python 등 수많은 언어들이 있어 기존 생물학자들이 접근하기 어려운것이 사실이다. 이 중 python은 접근 가능한 라이브러리들이 방대하고 다른 코딩언어를 아우르기 쉬운 코딩언어중 하나로, 다른 코딩언어들을 굳이 배우지 않아도 사용할 수 있고, 그 언어 또한 다른 언어들에 비해 비교적 간단하게 배울 수 있어, 생물정보학을 배우는 사람들이 많이 접하고 있는 코딩언어이다.

이 책은 파이썬이라는 언어의 설명과 동시에 생물정보학자이기 이전에 생물학적 지식을 갖고 있어야 하는 이 필드의 구조를 잘 이해하고 쓴 책이라 할만하다. 어떻게 파이썬으로 생물학적 지식을 잘 이용하고 스스로의 코딩을 잘 접목 시킬 수 있을지에 대한 길을 열어주는 길라잡이 역할을 잘 하고 있다. 생물정보학은 DNA를 기본으로 하는 text로 표현할 수 있는 모든 정보를 다루는 학문이다. 이 책으로 처음부터 끝까지 task를 다 수행해 본다면 어느새 생물정보학자라 불릴 만큼 파이썬으로 다양한 일을 해낼 수 있을 것이다.

다만, 생물학과 코딩언어를 동시에 설명하기란 쉽지 않은데 한계점에서 적절히 모든것을 설명하고 있어, 반대급부로 생물학, 코딩언어가 동시에 살짝 기초가 부족해 질 수 있기 때문에 다른 책들, 인터넷을 참고하면서 공부하면 좋을 것 같다.

- 접기

leeimchang 2019-04-12 공감(0) 댓글(0)

Thanks to 공감

★★★★★ 데이터 과학을 접하는 생물학도들의 로드맵 [+]

최근 과학 분야의 가장 핫한 트렌드는 역시 데이터 사이언스이다. 본 책은 프로그래밍/데이터와 거리가 멀었을 생물학도들을 위한 안내서이다. 생물학 분야에서 과학적으로 생각해라, 데이터를 분석해 보아라라고 하였을 때 그래서 어떻게 하라는거야?라는 질문에 딱 알맞는 답을 준다. 어떤 데이터부터 시작하면 좋은지, 어떤 데이터를 어떻게 가져와서 어떻게 분석해야 할지는 프로그래밍을 통해서 접근 할 수 있는 로드맵을 제공해 준다.

비 컴퓨터 전공자들은 프로그래밍 분야에 대해서 깊게 파고 들기가 쉬운 일이 아니다. 그래서 이 책은 크게 두 부분으로 나누어져 있다. 첫 파트는 파이썬을 이용한 생물 정보를 직접적으로 다루는 방법을 알려주는 파트이고 두 번째 파트는 파이썬 프로그래밍 그 자체에 대한 것이다. 독자는 첫 파트를 통해 무엇이 생물 정보를 프로그래밍을 통해 어떻게 다루어야 하는지 알게 된다. 두 번째 파트에서는 프로그래밍 스킬을 항상 시켜주어 추후에 필요할 좀 더 복잡한 데이터 분석 기술을 가질 수 있게 도와 준다.

개인적으로 이 책에서 가장 큰 가치를 주고 싶은 것은 연습 문제 부분이다. 타 전공자가 파이썬을 배우면서 알고리즘을 같이 배우는 것은 정말 쉽지 않은 일이다. 그리고 알고리즘을 배우는 것은 단순히 읽는 것을 넘어서 문제를 풀면서 많이 체득하게 된다. 본 책은 연습 문제를 통해서 파이썬 코딩 스킬과 동시에 프로그래밍 사고력을 기르게 도와주어 독자에게 단순한 코딩 학습을 넘어선 본질적인 사고의 함양을 가져다 준다. 이 책을 읽은 독자는 타 학도가 단순히 파이썬을 배운 것과 다른 생물학과 데이터 과학을 융합하는 복합적인 사고력에 있어서 질적인 향상을 경험하게 될 것이다.

rlwlsdnlwjerm 2019-05-14 공감(0) 댓글(0)

Thanks to 공감

리뷰 너무너무  
감사합니다

ㅜㅜ  
이 자리를 빌어  
감사드립니다



02

## 초급자를 위한 Github

---

# Github이란?



- Github (깃허브) 은 분산 버전 관리 툴인 Git (깃)의 웹 호스팅 서비스
- 사용자들의 코드를 버전관리를 해줌.
- 공개용 repository를 무료로 사용할 수 있으며, private repository는 유료였다.
- 2018년 6월 4일 Microsoft가 Github을 7,500,000,000 달러에 인수 (8조7562억5000만원..) MS의 인수 이후 private repository가 조건부 무료로 사용 가능
- 어.. 그런데 버전관리가 뭐지?

미국 USD  
7,500,000,000  
75억 달러

대한민국 KRW  
8,756,250,000,000.00  
8조 7,562억 5,000만 원

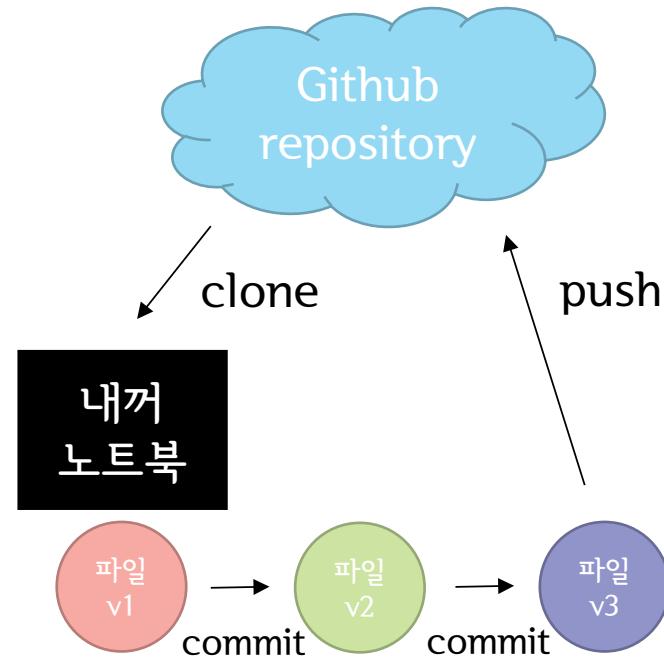
# Github을 사용해야하는 이유

---

- 코드의 버전 관리가 쉬워진다. (내가 어떤것을 고쳤는지 확인이 쉬움)  
예를 들어, 과제를 하는데 다음과 같이 관리 안되는 파일구조를 안가져도 된다는 얘기  
`assignment1.py -> assignment1.final.py -> assignment1.final.1.py  
-> assignment.final.2.py -> assignment.final.real.py ...`
- 협업이 편해진다.  
전체 메인코드(`main.py`)에 개발할 내부 모듈을 나눠서 개발하는 경우,  
개발자들이 서로 코드를 버전관리 시스템에 자유롭게 올리고 받으면 개발할 수 있음.  
만약 버전관리 시스템이 없다면, 메신저나 메일로 파일들을 주고 받아야함.
- 그러면 왜 Github? Git을 웹상에서 GUI로 다룰 수 있고,  
다른 사용자들과의 공유가 쉽기 때문
- 다른 사람들이 볼 수 있다는 점으로 **포트폴리오로 활용 가능!**
- 마크다운(Markdown) 언어 지원으로 웹페이지 생성 가능

# 이번 기회에 배울 Github 사용방법!

- Github의 repository 만들기
- Repository 받아오기
- 코드 만들어서 repository에 올리기
- Repository에 올라간 코드 확인하기
- 코드 수정하여 repository에 올리기
- Repository 확인하기
- Branch 만들기
- Branch에서 작업하기
- Branch에서 repository에 올리기
- Branch에서 master에 합치기



Github의 간단  
한 기능들 위주  
로 배워보자



# Github의 사용방법 - 회원가입, repository 만들기

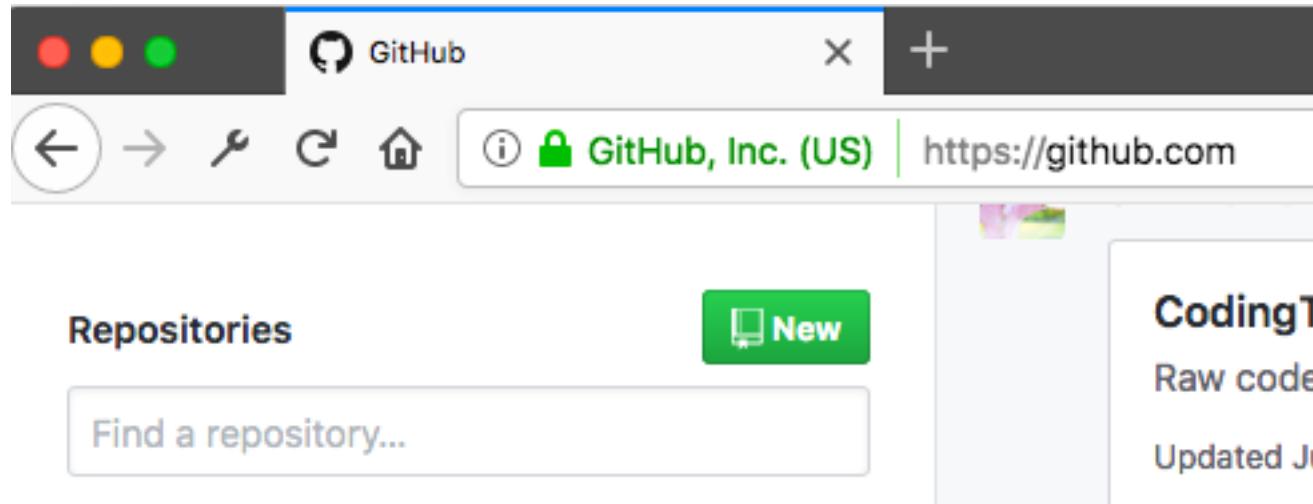
1) github 페이지에 접속합니다.

<https://github.com>

2) 회원가입을 합니다.

3) 로그인을 합니다.

4) New 버튼을 누릅니다.



# Github의 사용방법 - repository 만들기

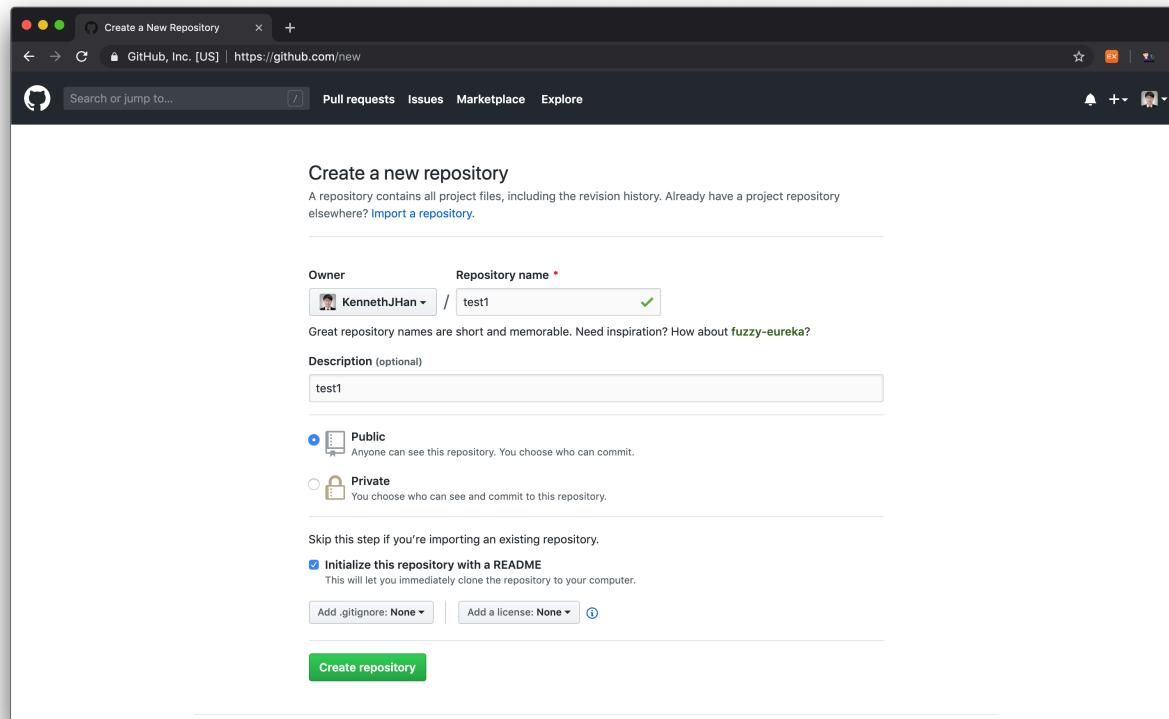
5) 새로운 repository를 만듭니다.

Repository name은 test1로 해봅시다.

Public, Private을 고를 수 있습니다.

Initialize this repository with a README를 체크해줍니다.

Create repository 버튼을 누릅니다.



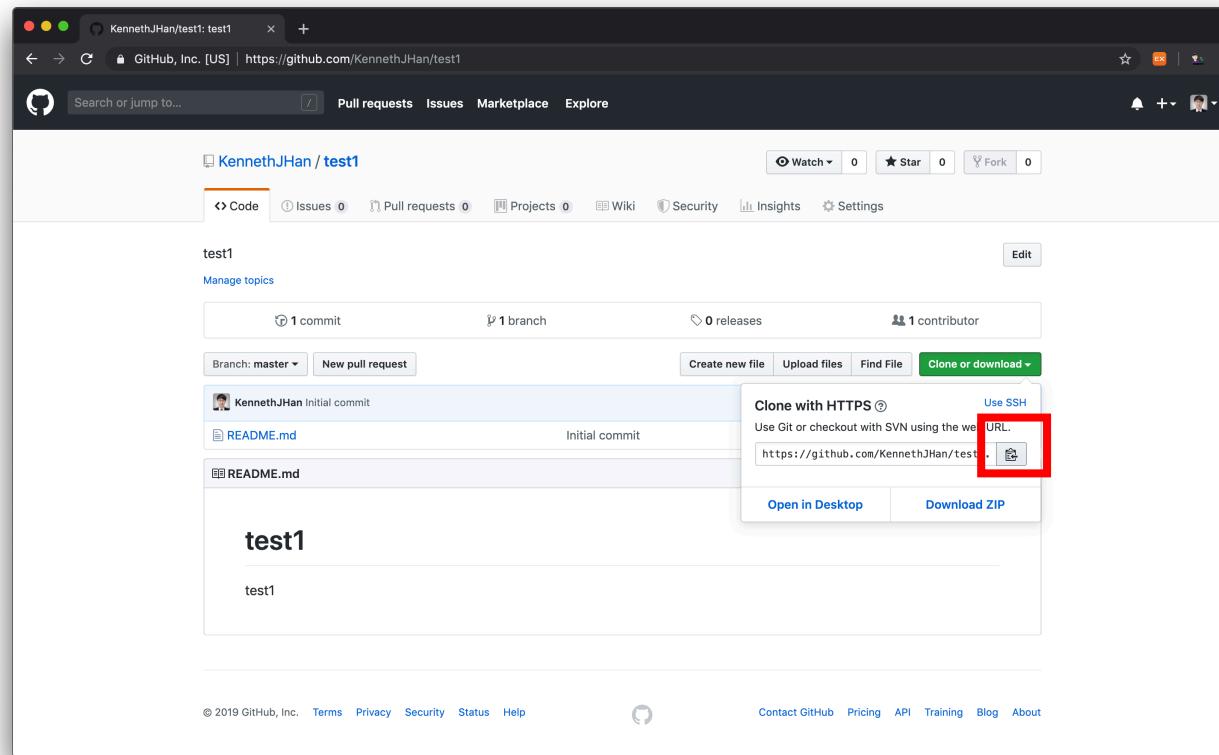
# Github의 사용방법 - repository 만들기

6) 다음 그림과 같이 새 repository가 생성되었습니다.

Clone or download 버튼을 누르고 다음 그림에서 빨간 네모로

표시한 버튼을 눌러 repository의 주소를 복사해둡시다.

지금 복사한 주소는 터미널에서 repository를 복사할 때 사용할 것 입니다.



# Github의 사용방법 - repository 다운로드

7) github을 터미널 환경에서 다루어봅시다.

새 터미널을 띄워봅시다.

터미널에서 적당한 디렉터리를 생성하여 들어갑니다. (mkdir)

git clone 을 타이핑 한 후 이전 슬라이드에서 복사한 내용을 붙여넣고 실행합니다.  
clone이 잘 되었는지 디렉터리 내부의 파일들을 살펴봅니다.

```
[11:38:58 jhan@Acer-A0532h github_test]$ git clone https://github.com/KennethJHa/test1.git
Cloning into 'test1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.
[11:39:32 jhan@Acer-A0532h github_test]$ ls -l
total 4
drwxrwxr-x 3 jhan jhan 4096 Jul 1 11:39 test1
[11:39:38 jhan@Acer-A0532h github_test]$ cd test1
[11:39:41 jhan@Acer-A0532h test1]$ ls -l
total 4
-rw-rw-r-- 1 jhan jhan 14 Jul 1 11:39 README.md
[11:39:42 jhan@Acer-A0532h test1]$ cat README.md
# test1
test1
[11:39:46 jhan@Acer-A0532h test1]$
```

# Github의 사용방법 - local에 파일 만들기

8) Hello world를 출력하는 파이썬 스크립트를 하나 만들고 실행하여 봅시다.

```
[11:39:46 jhan@Acer-A0532h test1]$ vi hello.py  
[11:40:13 jhan@Acer-A0532h test1]$ python hello.py  
Hello World  
[11:40:16 jhan@Acer-A0532h test1]$
```

```
1 #!/usr/bin/python  
2  
3 print("Hello World")  
4
```

9) github에 스크립트를 업로드 해봅시다.

다음과 같이 git status 를 입력하여 나오는 로그를 읽어봅시다.

처음 clone 한 repository와 비교하니 hello.py 가 추가됐다고 빨간 글씨로 보여줍니다.

```
[11:40:16 jhan@Acer-A0532h test1]$ git status  
On branch master  
Your branch is up-to-date with 'origin/master'.  
  
Untracked files:  
(use "git add <file>..." to include in what will be committed)  
  
    hello.py  
  
nothing added to commit but untracked files present (use "git add" to track)  
[11:40:22 jhan@Acer-A0532h test1]$
```

# Github의 사용방법 - add, status

---

- 10) git add [파일이름] 을 입력하여 업로드 할 파일을 추가해줍니다.  
git status 를 실행해보면 commit 할 파일을 보여줍니다.

```
[11:40:22 jhan@Acer-A0532h test1]$ git add hello.py
[11:40:35 jhan@Acer-A0532h test1]$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   hello.py

[11:40:43 jhan@Acer-A0532h test1]$ █
```

# Github의 사용방법 - git commit -m

---

11) git commit -m 으로 github에 commit 시 파일에 코멘트를 적을 수 있습니다.

```
[11:40:43 jhan@Acer-A0532h test1]$ git commit -m "Add hello.py"
[master 60c5102] Add hello.py
Committer: jhan <jhan@Acer-A0532h>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"
git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

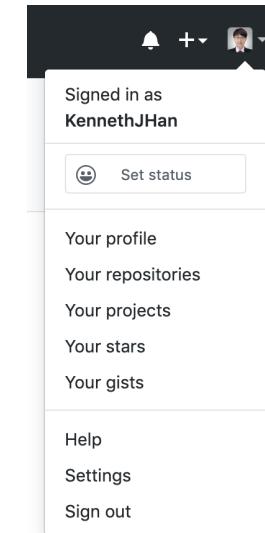
1 file changed, 4 insertions(+)
create mode 100644 hello.py
[11:41:03 jhan@Acer-A0532h test1]$ █
```

# Github의 사용방법 - git push origin

12) git push origin master 로 local에서 작업한 repository의 변경사항을 github에 반영합니다.

Username 은 github 페이지 우상단의 사진을 눌러서 나오는 창에서 확인할 수 있습니다. 비밀번호는 설정한 비밀번호를 넣습니다.

```
[11:41:03 jhan@Acer-A0532h test1]$ git push origin master
Username for 'https://github.com': KennethJHan
Password for 'https://KennethJHan@github.com':
Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/KennethJHan/test1.git
  e8c683c..60c5102  master -> master
[11:42:13 jhan@Acer-A0532h test1]$
```



# Github의 사용방법 - 업로드 확인

12) github 페이지에서 hello.py 가 생성된 것을 확인할 수 있습니다.  
hello.py를 눌러보면 우리가 작성한 코드를 확인할 수 있습니다.

The image shows two screenshots of a GitHub repository named 'test1' owned by 'KennethJHan'.  
The left screenshot shows the repository overview with 2 commits, 1 branch, 0 releases, and 1 contributor. It lists two commits: 'Initial commit' (README.md) and 'Add hello.py'. The 'hello.py' file content is shown as:

```
#!/usr/bin/python
print("Hello World")
```

The right screenshot shows the details of the 'hello.py' file. It indicates 5 lines (2 sloc) and 41 Bytes. The code is displayed as:

```
1  #!/usr/bin/python
2
3  print("Hello World")
4
```

# Github의 사용방법 - 파일의 수정

---

13) 코드를 수정하는 경우는 어떻게 될까요?

다음과 같이 hello.py에서 Hello World를 Hello Bioinformatics로 변경해봅시다.

```
1 #!/usr/bin/python
2
3 print("Hello Bioinformatics")
4 █
```

# Github의 사용방법 - 파일의 수정

14) git status를 실행해보니 hello.py가 modified 되었다고 나옵니다.  
git add hello.py 를 하여 commit에 추가해줍니다.

```
[11:42:13 jhan@Acer-A0532h test1]$ vi hello.py
[11:43:54 jhan@Acer-A0532h test1]$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   hello.py

no changes added to commit (use "git add" and/or "git commit -a")
[11:44:00 jhan@Acer-A0532h test1]$ git add hello.py
[11:44:23 jhan@Acer-A0532h test1]$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   hello.py

[11:44:26 jhan@Acer-A0532h test1]$ █
```

# Github의 사용방법 - 파일의 수정

---

15) 수정한 파일의 commit 도 마찬가지로 commit -m 으로 설명을 적어줄 수 있습니다.

```
[11:44:26 jhan@Acer-A0532h test1]$ git commit -m "Modified hello.py"
[master 7982127] Modified hello.py
Committer: jhan <jhan@Acer-A0532h>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"
git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 file changed, 1 insertion(+), 1 deletion(-)
[11:44:44 jhan@Acer-A0532h test1]$ █
```

# Github의 사용방법 - 파일의 수정

16) git push origin master 로 local의 내용을 repository로 업로드 합니다.

```
[11:44:44 jhan@Acer-A0532h test1]$ git push origin master
Username for 'https://github.com': KennethJHan
Password for 'https://KennethJHan@github.com':
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 326 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/KennethJHan/test1.git
  60c5102..7982127 master -> master
[11:45:12 jhan@Acer-A0532h test1]$
```

17) 짠.. 멋지게 바뀌었습니다. 여기 페이지에서 History 버튼을 눌러봅시다.

The screenshot shows a GitHub repository page for 'KennethJHan / test1'. The 'Code' tab is selected. Below it, the file 'test1 / hello.py' is shown with the content:  
1 #!/usr/bin/python  
2  
3 print("Hello Bioinformatics")  
4

At the bottom right of the code editor, there is a row of buttons: Raw, Blame, History, and a trash bin icon. The 'History' button is highlighted with a red box.

# Github의 사용방법 - 파일의 수정

18) 지금까지 hello.py의 수정된 기록을 볼 수 있습니다.  
Modified hello.py 를 클릭해봅시다.

The screenshot shows the GitHub repository page for 'test1'. The 'Code' tab is selected. Below it, the 'History for test1 / hello.py' section displays two commits:

- Modified hello.py**: jhan authored and jhan committed 2 minutes ago. Commit ID: 7982127
- Add hello.py**: jhan authored and jhan committed 5 minutes ago. Commit ID: 60c5102

19) 와우.. hello.py 에서 이전 파일과 비교하여 무엇이 바뀌었는지 확인할 수 있습니다.

The screenshot shows the GitHub diff view for the 'Modified hello.py' commit. It compares the previous version (commit 60c5102) with the current version (commit 7982127). The diff highlights one change:

```
diff --git a/hello.py b/hello.py
index 60c5102..7982127 100644
--- a/hello.py
+++ b/hello.py
@@ -1,4 +1,4 @@
 1     1     #!/usr/bin/python
 2     2
 3 -   print("Hello World")
 3 +   print("Hello Bioinformatics")
```

# Github의 사용방법 - Branch

20) 지금까지는 master라는 branch에서 작업을 하였습니다.  
그런데 테스트 코드를 만들어 테스트 해보고 나중에 버전을 합칠 수 있을까요?

The screenshot shows a GitHub repository page for 'KennethJHan / test1'. At the top, there are buttons for 'Watch' (0), 'Star' (0), and 'Fork' (0). Below the header, there are links for 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Security', 'Insights', and 'Settings'. The main area shows the repository name 'test1' and an 'Edit' button. A 'Manage topics' link is also present. Key statistics are displayed: '3 commits', '1 branch', '0 releases', and '1 contributor'. A dropdown menu shows the current branch is 'master'. There are buttons for 'New pull request' and 'Clone or download'. A sidebar allows switching between branches and tags, with 'master' selected. The commit history shows an 'Initial commit' made 42 minutes ago and a modification to 'hello.py' made 11 minutes ago. The repository contains a single file named 'test1'.

# Github의 사용방법 - Branch

21) git checkout -b [branch 이름] 으로 새 branch를 만들 수 있습니다.

```
[14:43:41 jhan@Acer-A0532h test1]$ git checkout -b ken_dev  
Switched to a new branch 'ken_dev'  
[14:48:36 jhan@Acer-A0532h test1]$ █
```

22) hello.py 에 새로운 줄을 추가해보겠습니다.

```
1 #!/usr/bin/python  
2  
3 print("Hello Bioinformatics")  
4 print("Hi")  
5 █
```

23) git status를 해보니 hello.py가 modified 되었다고 나옵니다.  
branch이름이 master가 아닌 우리가 정해준 이름으로 나온것을 확인해보세요.

```
[14:49:45 jhan@Acer-A0532h test1]$ git status  
On branch ken_dev  
Changes not staged for commit:  
  (use "git add <file>..." to update what will be committed)  
  (use "git checkout -- <file>..." to discard changes in working directory)  
  
        modified:   hello.py  
  
no changes added to commit (use "git add" and/or "git commit -a")  
[14:49:48 jhan@Acer-A0532h test1]$ █
```

# Github의 사용방법 - Branch

24) git add 로 hello.py를 추가하고 commit -m 으로 설명도 써줍니다.

```
[[14:49:48 jhan@Acer-A0532h test1]$ git add hello.py
[[14:50:47 jhan@Acer-A0532h test1]$ git status
On branch ken_dev
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   hello.py

[[14:50:50 jhan@Acer-A0532h test1]$ git commit -m "Add hi on dev branch"
[ken_dev eef7a08] Add hi on dev branch
Committer: jhan <jhan@Acer-A0532h>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

  git config --global user.name "Your Name"
  git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

  git commit --amend --reset-author

  1 file changed, 1 insertion(+)
[14:51:13 jhan@Acer-A0532h test1]$ ]
```

# Github의 사용방법 - Branch

25) git push로 local의 내용을 branch로 업로드 해줍니다.

```
[14:51:13 jhan@Acer-A0532h test1]$ git status
On branch ken_dev
nothing to commit, working directory clean
[14:51:33 jhan@Acer-A0532h test1]$ git push origin ken_dev
Username for 'https://github.com': KennethJHan
Password for 'https://KennethJHan@github.com':
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 327 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'ken_dev' on GitHub by visiting:
remote:     https://github.com/KennethJHan/test1/pull/new/ken_dev
remote:
To https://github.com/KennethJHan/test1.git
 * [new branch]      ken_dev -> ken_dev
[14:51:59 jhan@Acer-A0532h test1]$
```

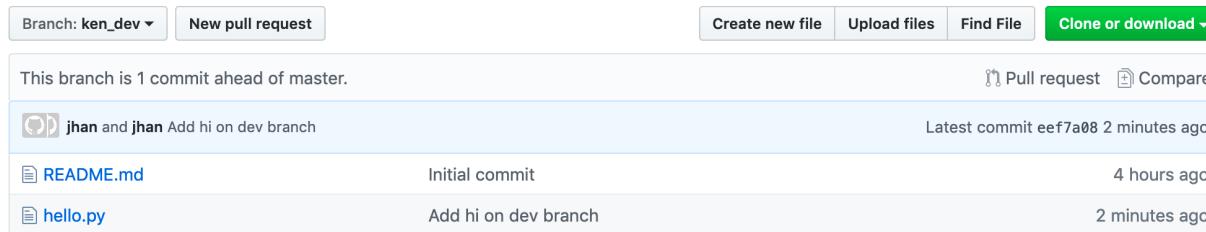
26) github 페이지에 보면 다음과 같이 branch 생성을 확인할 수 있습니다.

The screenshot shows a GitHub repository interface. On the left, there's a sidebar with a dropdown for 'Branch: master' and a button for 'New pull request'. Below that is a 'Switch branches/tags' section with a search bar and tabs for 'Branches' (which is selected) and 'Tags'. The 'ken\_dev' branch is listed here with a blue background. At the bottom of the sidebar, 'master' is also listed with a checkmark. The main area shows a table of commits:

|  |                   | Latest commit 7982127 3 hours ago |
|--|-------------------|-----------------------------------|
|  | Initial commit    | 4 hours ago                       |
|  | Modified hello.py | 3 hours ago                       |
|  |                   | (pencil icon)                     |

# Github의 사용방법 - Branch

27) 우리가 생성한 branch로 옮겨서 hello.py를 클릭해봅시다.



Branch: ken\_dev ▾ New pull request Create new file Upload files Find File Clone or download ▾

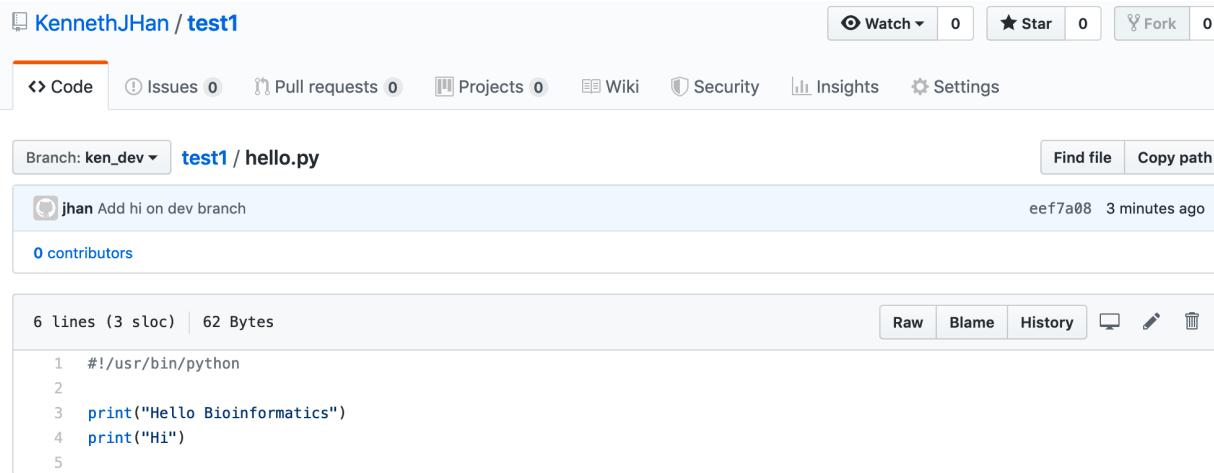
This branch is 1 commit ahead of master.

jhan and jhan Add hi on dev branch Latest commit eef7a08 2 minutes ago

README.md Initial commit 4 hours ago

hello.py Add hi on dev branch 2 minutes ago

28) 우리가 만든 코드가 잘 들어갔군요.



KennethJHan / test1 Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Branch: ken\_dev ▾ test1 / hello.py Find file Copy path

jhan Add hi on dev branch eef7a08 3 minutes ago

0 contributors

6 lines (3 sloc) | 62 Bytes Raw Blame History

```
1 #!/usr/bin/python
2
3 print("Hello Bioinformatics")
4 print("Hi")
5
```

# Github의 사용방법 - Branch

29) history도 확인해봅시다.

The screenshot shows the GitHub repository page for `KennethJHan/test1`. The navigation bar includes options like Watch (0), Star (0), Fork (0), Code, Issues (0), Pull requests (0), Projects (0), Wiki, Security, Insights, and Settings. The main content displays the commit history for the file `hello.py`. It shows three commits made on July 1, 2019:

- Add hi on dev branch**: jhan authored and jhan committed 3 minutes ago. Commit hash: `eef7a08`.
- Modified hello.py**: jhan authored and jhan committed 3 hours ago. Commit hash: `7982127`.
- Add hello.py**: jhan authored and jhan committed 3 hours ago. Commit hash: `60c5102`.

30) Add hi on dev branch를 누르면 변화된 부분을 볼 수 있습니다.

The screenshot shows the GitHub diff view for the file `hello.py`. The interface indicates "Showing 1 changed file with 1 addition and 0 deletions." There are two tabs at the top: Unified (selected) and Split. The diff view shows the following changes:

```
diff --git a/hello.py b/hello.py
index 111111..222222 100644
--- a/hello.py
+++ b/hello.py
@@ -1,4 +1,5 @@
 1     1     #!/usr/bin/python
 2     2
 3     3     print("Hello Bioinformatics")
+ 4     +     print("Hi")
```

# Github의 사용방법 - master에 branch merge

---

- 31) 이제 branch의 내용을 master에 merge해보겠습니다.  
git checkout master 로 branch에서 master로 전환합시다.

```
[14:51:59 jhan@Acer-A0532h test1]$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
```

- 32) git merge [branch 이름] 을 하여 지정한 branch를 master에 합칩니다.

```
[14:55:23 jhan@Acer-A0532h test1]$ git merge ken_dev
Updating 7982127..eef7a08
Fast-forward
 hello.py | 1 +
 1 file changed, 1 insertion(+)
[14:55:36 jhan@Acer-A0532h test1]$
```

- 33) git push origin master 를 하여 변경사항을 repository에 업로드합니다.

```
[14:58:53 jhan@Acer-A0532h test1]$ git push origin master
Username for 'https://github.com': KennethJHan
Password for 'https://KennethJHan@github.com':
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/KennethJHan/test1.git
 7982127..eef7a08  master -> master
[14:59:35 jhan@Acer-A0532h test1]$
```

# Github의 사용방법

이렇게 Github을 사용하면

code의 나의 코드 짠 내용들을 시간별로 볼 수 있고 (버전관리가 용이함)  
남들과 같이 일하기 쉽고 (협업의 편리)  
내가 짠 코드들을 남들에게 공유하기 쉽고 ..

여러 장점들이 많으므로 코드를 작성하고 github에 올리는 습관을 가져봅시다

이번에 배운 내용 이외에도  
Github을 사용하면서 필요한 내용들이 많습니다.  
여러분들이 하나씩 부딪히고 검색해 보면서  
사용법을 익혀봅시다 ^^



## 아나콘다 파이썬 설치 03 | 파이썬 라이브러리 설치 방법

---

# 파이썬의 주요 역사



Python logo (1990s - 2005)

\*\* Python is named after the British TV show Monty Python

1980s

1991

1994

1995

2000

Conceptualized by Guido van Rossum

Python 0.9.0 released

\* Exception handling, function, list, dict, str, module system

Python 1.0 released

\* lambda, map, filter, reduce

Python 1.2 released

Python 1.6, Python 2.0 released

\*\* new python license (GPL-compatible)

2000

2001

2001

2003

2004

Python 2.0 released

\* List comprehension, Full GC, unicode

Python 2.1 Released, Python 1.6.1 closed

Python 2.2 Released

\* Purely object oriented, Generator added

Python 2.3 Released

Python 2.4 Released

# 파이썬의 주요 역사



Python logo (2006 - current)

2006

2008

2010

Python 2.5  
released

Python 2.6  
released

Python 2.7  
released

\*\* Python 2.7 is the  
last major release

\* lambda, map, filt  
er, reduce, Order  
edDict

\* with statement

2008

2009

2011

2012

Python 3.0  
released

Python 3.1  
Released

Python 3.2  
Released

Python 3.3  
Released

\* Purely object ori  
ented,  
Generator added

\* Purely object ori  
ented,  
Generator added

\* Purely object ori  
ented,  
Generator added

# 파이썬의 주요 역사



Python logo (2006 - current)

| 2014                    | 2015   | 2016   | 2018  | 2019? 2020? | 20xx |
|-------------------------|--|--|---|-------------|------|
| Python 3.4 released     | Python 3.5 Released                          | Python 3.6 Released                          | Python 3.7 Released                           |             |      |
|                         | * Purely object oriented,<br>Generator added | * Purely object oriented,<br>Generator added | * Purely object oriented,<br>Generator added  |             |      |
|                         |  |  | ** Guido quit from<br>Python chief<br>2018.07 |             |      |
| Python 3.8              |  | Python 3.9                                   |   |             |      |
| * Walrus operator<br>:= |  | * Purely object oriented,<br>Generator added |   |             |      |

# 파이썬 창시자 Guido 선생님께 온 메일



Guido van Rossum ▶ Public

Sep 18, 2013

⋮

Do **not** send me email like this:

.....

Hi Guido,

I came across your resume in a Google web search. You seem to have an awesome expertise on Python. I would be glad if you can reply my email and let me know your interest and availability.

.....

Our client immediately needs a PYTHON Developers at its location in \*, NJ.  
Below are the job details. If interested and available, kindly fwd me your updated resume along with the expected rate and the availability.

[...]

.....

I might reply like this:

.....

I'm not interested and not available.

.....

+1

1883



588

Shared publicly • View activity

안녕 Guido,

구글 검색으로 너의 이력서를 보았어.  
너는 파이썬에서 멋진 이력이 있네.  
관심 있으면 답메일 부탁해.

-----  
우리의 고객이 PYTHON 개발자를 급히  
구하고 있습니다. 지역은 \*, NJ.



# 주의!

---

주의!!

다음부터 있는 내용은 초급자들을 위한  
내용이 절대 절대 아닙니다!

결론은 Python3 가 기능이 많고  
앞으로도 계속 발전하고 있는  
살아있는 언어라는 사실을  
여러분들께 말씀드리기 위해 넣은  
내용이니 그냥 쭉~ 보시고 넘어가세요~



# New Features in Python3

- f-String

PEP498 – Literal String Interpolation  
python3.6

```
1 #!/usr/bin/python3
2
3 name = "ken"
4 age = 34
5
6 result = f"""My name is {name}.
7 I'm {age} years old.
8 My age next year is {age+1}."""
9
10 print(result)
11
12 # My name is ken.
13 # I'm 34 years old.
14 # My age next year is 35.
15
```

```
1 #!/usr/bin/python3
2
3 import datetime
4
5 date = datetime.date(1986, 10, 15)
6
7 print(f"{date} was on a {date:%A}.")
8
9 # 1986-10-15 was on a Wednesday.
10
```

- Merge Dictionary

## PEP448 – Additional Unpacking Generalizations Python3.5

```
1 #!/usr/bin/python3
2
3 d1 = {'a':1, 'b':2}
4 d2 = {'c':3, 'd':4}
5 d3 = {**d1, **d2}
6 print(d3)
7 ## {'a': 1, 'b': 2, 'c': 3, 'd': 4}
8
9
10 d1 = {'a':1, 'b':2}
11 d2 = {'b':3, 'c':4}
12 d3 = {**d1, **d2}
13 print(d3)
14 ## {'a': 1, 'b': 3, 'c': 4}
15
16
17 d1 = {'a':1, 'b':2}
18 d2 = {'c':3, 'd':4}
19 d3 = {'e':5, 'f':6}
20 d4 = {**d1, **d2, **d3}
21 print(d4)
22 ## {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6}
23
```

- Async

PEP492 – Coroutines with `async` and `await` syntax  
python3.5

```
1 import asyncio
2 import time
3
4 async def sleep():
5     await asyncio.sleep(5)
6
7 start = time.time()
8
9 # Define Event loop
10 loop = asyncio.get_event_loop()
11
12 # Run Event loop
13 loop.run_until_complete(sleep())
14
15 end = time.time()
16
17 print(f"{end-start}s")
18 ## 5.007434606552124s
19
```

```
1 import asyncio
2 import time
3
4
5 async def coroutine_1():
6     print('## START: coroutine_1')
7     print('await 5 seconds')
8     await asyncio.sleep(5)
9     print('coroutine1 resume')
10
11
12 async def coroutine_2():
13     print('## START: coroutine_2')
14     print('await 4 seconds')
15     await asyncio.sleep(4)
16     print('coroutine2 resume')
17
18 if __name__ == "__main__":
19     # Define Event loop
20     loop = asyncio.get_event_loop()
21
22     start = time.time()
23
24     # Run Two Coroutines in Event loop
25     # by using asyncio.gather (coroutine scheduled in order)
26     loop.run_until_complete(asyncio.gather(coroutine_1(), coroutine_2()))
27     end = time.time()
28
29     print(f'time taken: {end-start}')
30
31 """
32 ## START: coroutine_1
33 await 5 seconds
34 ## START: coroutine_2
35 await 4 seconds
36 coroutine2 resume
37 coroutine1 resume
38 time taken: 5.004393815994263
39 """
40
```

```
1 import asyncio
2 import time
3
4
5 async def coroutine_1():
6     print('## START: coroutine_1')
7     print('await time.sleep 5')
8     loop = asyncio.get_event_loop()
9
10    await loop.run_in_executor(None, time.sleep, 5)
11    # run_in_executor:
12    # Method which can run function not in coroutine
13
14    # Params of run_in_executor:
15    # executor(None: default loop executor), func, *args
16
17    print('coroutine_1 resume')
18
19
20 async def coroutine_2():
21     print('## START: coroutine_2')
22
23     print('await time.sleep 4')
24     await loop.run_in_executor(None, time.sleep, 4)
25     print('coroutine_2 resume')
26
27 if __name__ == "__main__":
28     # Define Event loop
29     loop = asyncio.get_event_loop()
30
31     # Schedule coroutine
32     start = time.time()
33     loop.run_until_complete(asyncio.gather(coroutine_1(), coroutine_2()))
34     end = time.time()
35
36     print(f'time taken: {end-start}')
37
38 """
39 ## START: coroutine_2
40 await time.sleep 4
41 ## START: coroutine_1
42 await time.sleep 5
43 coroutine_2 resume
44 coroutine_1 resume
45 time taken: 5.0130133628845215
46 """
47
```

- Type hints

PEP484 – Type Hints  
python3.5

```
1 #!/usr/bin/python3
2
3 def helloWorld(name: str) -> str:
4     result = f"Hello {name}"
5     return result
6
7 if __name__ == "__main__":
8     print(helloWorld("ken"))
9
```

- Guido made *mypy* which can check type without IDE.

- Getter, Setter

PEP318 – Decorators for Functions and Methods  
python2.4

```
1 #!/usr/bin/python3
2
3 class Obj:
4     def __init__(self):
5         self._prop = ""
6
7     @property
8     def prop(self):
9         return self._prop
10
11    @prop.setter
12    def prop(self, val):
13        self._prop = val
14
15
16 if __name__ == "__main__":
17     obj = Obj()
18     obj.prop = "123"
19     print(obj.prop) ## 123
20
```

- Decorator

## PEP318 – Decorators for Functions and Methods python2.4

```
1 #!/usr/bin/python3
2
3 from datetime import datetime
4 import time
5
6 def stamp_decorator(func):
7     def deco():
8         print(f"# START - {datetime.now()}")
9         func()
10        print(f"# END - {datetime.now()}")
11    return deco
12
13 @stamp_decorator
14 def func1():
15     print("job in func1")
16     time.sleep(3)
17
18 @stamp_decorator
19 def func2():
20     print("some other job in func2")
21     time.sleep(2)
22
23
24 if __name__ == "__main__":
25     func1()
26     func2()
```

```
28 """
29 # START - 2019-06-28 09:27:42.936900
30 job in func1
31 # END - 2019-06-28 09:27:45.940389
32
33 # START - 2019-06-28 09:27:45.940749
34 some other job in func2
35 # END - 2019-06-28 09:27:47.943317
36 """
37
```

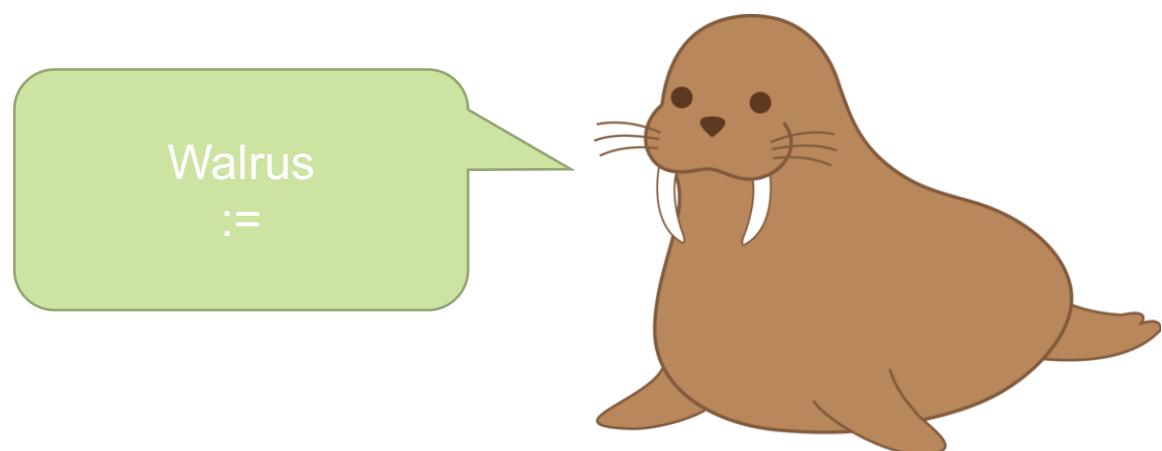
- Decorator

```
1 #!/usr/bin/python3
2
3 from datetime import datetime
4 import time
5
6 class stamp_decorator:
7     def __init__(self, func):
8         self.func = func
9
10    def __call__(self, *args, **kwargs):
11        print(f"# START - {datetime.now()}")
12        self.func(*args, **kwargs)
13        print(f"# END - {datetime.now()}")
14
15
16 @stamp_decorator
17 def func1():
18     print("job in func1")
19     time.sleep(3)
20
21 @stamp_decorator
22 def func2():
23     print("some other job in func2")
24     time.sleep(2)
25
26
27 if __name__ == "__main__":
28     func1()
29     func2()
30
31
32
33 # START - 2019-06-28 09:27:42.936900
34 job in func1
35 # END - 2019-06-28 09:27:45.940389
36
37
38 """
39 # START - 2019-06-28 09:27:45.940749
40 some other job in func2
41 # END - 2019-06-28 09:27:47.943317
42 """
43
```

- Walrus operator

PEP572 – Assignment Expression  
python3.8

```
1 #!/usr/bin/python3
2
3 a = 11
4
5 if (b := a) > 10:
6     print(f"The value of b is {b} and is greater than 10.")
7     ## The value of b is 11 and is greater than 10.
8
```



- Walrus operator (PEP572, python3.8)

```
1 #!/usr/bin/python3
2
3 sample_data = [
4     {"userId": 1, "id": 1, "title": "delectus aut autem", "completed": False},
5     {"userId": 1, "id": 2, "title": "quis ut nam facilis", "completed": False},
6     {"userId": 1, "id": 3, "title": "fugiat veniam minus", "completed": False},
7     {"userId": 1, "id": 4, "title": "et porro tempora", "completed": True},
8     {"userId": 1, "id": 4, "title": None, "completed": True},
9 ]
10
11 print("With Python 3.8 Walrus Operator:")
12 for entry in sample_data:
13     if title := entry.get("title"):
14         print(f'Found title: {title}')
15
16 print("Without Walrus operator:")
17 for entry in sample_data:
18     title = entry.get("title")
19     if title:
20         print(f'Found title: {title}')
21
```

# 주의!

---

쓱~ 지나가셨죠?  
같이 아나콘다 파이썬3을 설치하고  
이제 정말로 몸풀기 해봐요



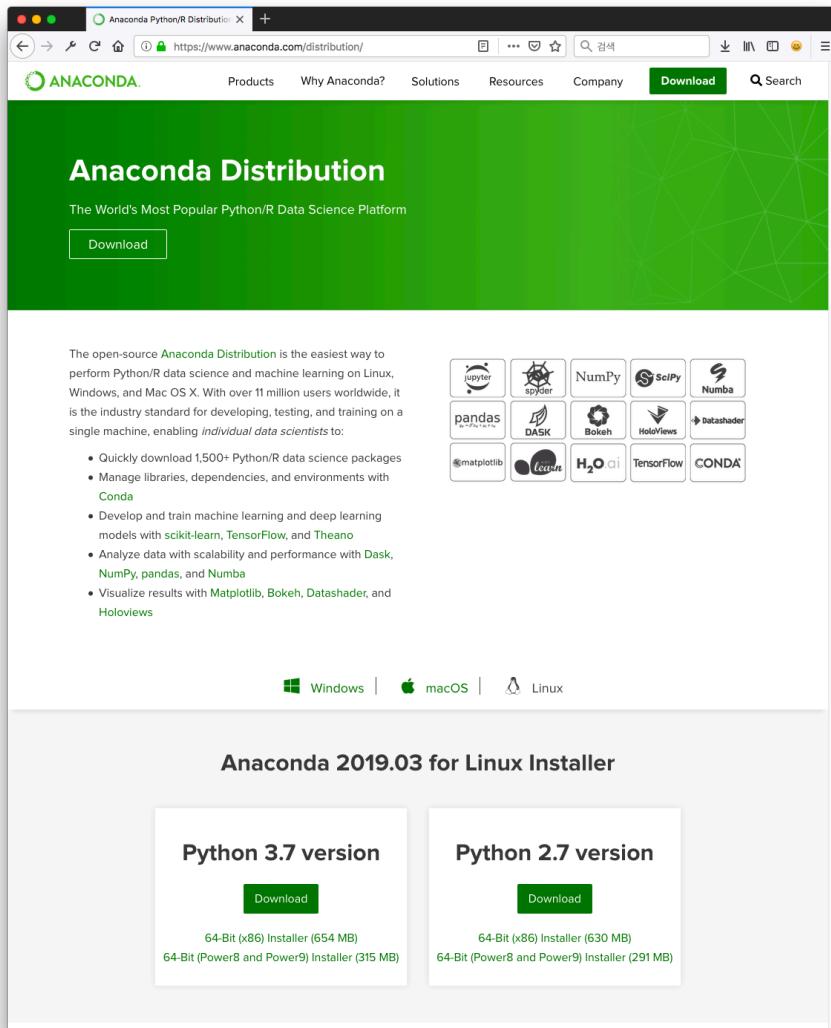
# Anaconda python3 설치 - 전체 과정 요약

---

- 1) google에 anaconda python 검색하여 사이트 들어가기
- 2) Download 버튼을 눌러 download 페이지에 들어가기
- 3) 파일을 다운 받는다.
- 4) 받은 파일을 실행한다. (파일 이름 다를 수도 있음)  
\$ sh Anaconda3-5.2.0-Linux-x86\_64.sh
- 5) 설치를 진행한다.  
PATH를 잡아주는것에 주의한다.
- 6) python -V로 파이썬 버전을 확인한다. (대문자 V)

```
$ python -V  
Python 3.6.0 :: Anaconda 4.3.1 (x86_64)
```

# Anaconda python download (LINUX)



<https://www.anaconda.com/distribution>

사이트에 접속하여 LINUX 용 Python3.7을  
다운로드 합니다.

링크 주소를 복사하여 wget 명령어로 받으면  
쉽겠죠?

# Anaconda python download (LINUX)

---

URL:

[https://repo.anaconda.com/archive/Anaconda3-2019.03-Linux-x86\\_64.sh](https://repo.anaconda.com/archive/Anaconda3-2019.03-Linux-x86_64.sh)

```
$ wget https://repo.anaconda.com/archive/Anaconda3-2019.03-Linux-x86_64.sh
--2019-07-03 20:09:55--  https://repo.anaconda.com/archive/Anaconda3-2019.03-Linux-x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3, 104.16.131.3, 2606:4700::6810:8203, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443
... connected.
HTTP request sent, awaiting response... 200 OK
Length: 685906562 (654M) [application/x-sh]
Saving to: 'Anaconda3-2019.03-Linux-x86_64.sh'

9% [=>                                              ] 62,995,164  11.1MB/s  eta 54s
```

# Anaconda python install

```
$ sh Anaconda3-2019.03-Linux-x86_64.sh
```

Welcome to Anaconda3 2019.03

In order to continue the installation process, please review the license  
agreement.

Please, press ENTER to continue  
>>> █

엔터를 누릅니다

```
=====
Anaconda End User License Agreement
=====
```

Copyright 2015, Anaconda, Inc.

라이센스 얘기

Do you accept the license terms? [yes|no]  
[no] >>> yes

동의 하시면 yes를 타이핑

Anaconda3 will now be installed into this location:  
/home/jhan/anaconda3

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

[/home/jhan/anaconda3] >>> /home/jhan/etc/anaconda3 █

설치 경로를 바꾸고 싶으면 저처럼  
경로를 입력해줍니다

# Anaconda python install

---

```
installing: scikit-learn-0.20.3-py3/hd81dba3_0 ...
installing: astropy-3.1.2-py37h7b6447c_0 ...
installing: statsmodels-0.9.0-py37h035aef0_0 ...
installing: seaborn-0.9.0-py37_0 ...
installing: anaconda-2019.03-py37_0 ...
installation finished.
Do you wish the installer to initialize Anaconda3
by running conda init? [yes|no]
[no] >>> █
```

yes 타이핑 후 엔터

```
$ source ~/.bashrc
$ which python
여러분이 넣었던 파일경로가 나옵니다
```

```
$ python -V
Python 3.7.3
```

이렇게 나오면 성공!

# 파이썬 라이브러리 설치 방법

pip (Python Package Index) 를 사용하여 파이썬 라이브러리를 설치해보겠습니다.

사용방법:

```
$ pip install [Package name]
```

우리는 이번 실습에서 바이오파이썬을 설치해보겠습니다!

```
$ python
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
[>>> import Bio
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'Bio'
>>>
```

TIP

Biopython이 설치되었는지 확인

Module: python 문법으로 쓰인 .py 파일  
Package: python module이 있는 디렉토리  
Library: python module이 하나로 묶인 것

# 파이썬 라이브러리 설치 방법

---

파이썬이 설치된 경로와 pip의 경로를 확인..

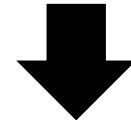
```
$ which python  
/home/jhan/etc/anaconda3/bin/python  
$  
$ which pip  
/home/jhan/etc/anaconda3/bin/pip  
$  
$ █
```

python, pip 가 설치된 경로에서 실행한 python에  
라이브러리가 설치되는 것입니다!

\$ pip install biopython 을 타이핑 하고 엔터!

# 파이썬 라이브러리 설치 방법

```
$ python
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import Bio
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'Bio'
>>> █
```



성공!

```
$ python
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import Bio
>>> Bio.__version__
'1.73'
>>> █
```

04

## 파이썬 기본문법 문제풀이로 정리하기

---

# 파이썬 기본문법 문제풀이로 정리하기



프로그래밍을 어떻게 하면 잘할 수 있을까?

1. **프로그래밍의 목적을 세운다.** -> Bioinformatics
2. **프로그래밍 언어를 선택한다.**  
-> 쉬운 언어로 시작한다. Python은 배우기 쉬우면서도 성능이 강력하다. 그리고 다른 많이 사용되는 언어인 C++, JAVA 와 같이 객체지향언어이다.
3. 간단한 것 부터 차근 차근 프로그래밍 해본다.
4. 눈이 아닌 직접 손으로 코딩 해 본다
5. 오류를 잡는 디버깅을 통해 실력을 키운다.
6. 다른 사람들과 코드를 비교해 본다.



# 파이썬 기본문법 문제풀이로 정리하기

여러분들께서 생물정보학 분야에서 원하는 연구 및 업무를 하시기 위해서는 탄탄한 프로그래밍 지식과 실력이 필수입니다.

프로그래밍 실력을 키우기 위해서는 스스로 문제를 해결하고 머리속 생각을 코드로 옮기는 연습을 해보아야 합니다.



# 파이썬 기본문법 문제풀이로 정리하기

---

파이썬 열심히 배우신 여러분들!  
몸풀기 해볼까요?

지금 부터 기본 문법에 대한 복습을 문제풀이를 통해 해볼게요.

그리고 작성한 파일들은 여러분들께서 github에 올려주세요.

# 001. Hello Bioinformatics

---

파이썬의 print 함수를 사용하여

Hello Bioinformatics

를 출력해봅시다.

# 001. Hello Bioinformatics

---

파이썬의 print 함수를 사용하여

Hello Bioinformatics

를 출력해봅시다.

```
1 print("Hello Bioinformatics")
```

## 002. Variable

---

반지름을 나타내는 변수  $r$ 을 사용하여

원의 넓이를 구해봅시다.

## 002. Variable

---

반지름을 나타내는 변수 r을 사용하여

원의 넓이를 구해봅시다.

```
1 r = 3
2 PI = 3.14
3
4 surface_area = r * r * PI
5
6 print(surface_area) ## 28.26
```

# 003. Operator

---

다음 두 변수의 사칙연산, 나머지 연산, 제곱을 구해보세요.

num1 = 3

num2 = 5

# 003. Operator

---

다음 두 변수의 사칙연산, 나머지 연산, 제곱을 구해보세요.

num1 = 3

num2 = 5

```
1 num1 = 3
2 num2 = 5
3
4 print(num1 + num2) ## 8
5 print(num1 - num2) ## -2
6 print(num1 * num2) ## 15
7 print(num1 / num2) ## 0.6
8 print(num1 // num2) ## 3
9 print(num1 ** num2) ## 243
```

## 004. if - else

---

변수 num1에 들어있는 수가 짝수인지 홀수인지 판별해보세요.  
예시로 num1에 3을 넣어보겠습니다.

## 004. if - else

---

변수 num1에 들어있는 수가 짝수인지 홀수인지 판별해보세요.  
예시로 num1에 3을 넣어보겠습니다.

```
1 num1 = 3
2 if num1 % 2 == 1:
3     print(num1, "은 홀수다.")
4 else:
5     print(num1, "은 짝수다.")
```

## 005. if-elif-else

---

변수 num1에 들어있는 수가 3의 배수인지 7의 배수인지  
판별해보세요.

예시로 num1에 21을 넣어보겠습니다.

## 005. if-elif-else

---

변수 num1에 들어있는 수가 3의 배수인지 7의 배수인지 판별해보세요.  
예시로 num1에 21을 넣어보겠습니다.

```
1 num1 = 21
2 if num1 % 3 == 0 and num1 % 7 == 0:
3     print(num1, "은 3과 7의 배수다.")
4 elif num1 % 3 == 0:
5     print(num1, "은 3의 배수다.")
6 elif num1 % 7 == 0:
7     print(num1, "은 7의 배수다.")
8 else:
9     print(num1, "은 3 또는 7의 배수가 아니다.")
```

## 006. for

---

1 부터 10까지 숫자 합을 구하는 프로그램을 작성해보세요.

## 006. for

---

1 부터 10까지 숫자 합을 구하는 프로그램을 작성해보세요.

```
1 s = 0
2 for i in range(1,11,1):
3     s += i
4 print(s)
```

## 007. 중첩 for 문

---

구구단의 짹수단만 출력해보세요.

## 007. 중첩 for 문

---

구구단의 짹수단만 출력해보세요.

```
1 for i in range(2,9,2):
2     for j in range(1,10,1):
3         print(i, "*" , j, "=" , i*j)
```

## 008. while

---

while 문을 사용하여  $5!$  (factorial) 값을 구해보세요.

## 008. while

---

while 문을 사용하여 5! (factorial) 값을 구해보세요.

```
1 num = 5
2 result = 1
3
4 while num > 0:
5     result *= num
6     num -= 1
7
8 print(result)
```

## 009. 함수

---

greet라는 이름의 함수를 만든 후,  
이 함수를 호출할 때마다 “Hello, Bioinformatics”  
를 출력하는 프로그램을 작성해보세요.

## 009. 함수

---

greet라는 이름의 함수를 만든 후,  
이 함수를 호출할 때마다 “Hello, Bioinformatics”  
를 출력하는 프로그램을 작성해보세요.

```
1 def greet():
2     print("Hello, Bioinformatics.")
3
4 greet()
5 greet()
```

## 010. 함수 - 함수에 값 전달

---

mySum이라는 이름의 함수를 만든 후,  
이 함수에 (2, 3), (5, 7), (10, 15)를 함수의 값으로 전달하여  
전달한 두 수를 더하여 출력하는 프로그램을 작성해보세요.

# 010. 함수 - 함수에 값 전달

mySum이라는 이름의 함수를 만든 후,  
이 함수에 (2, 3), (5, 7), (10, 15)를 함수의 값으로 전달하여  
전달한 두 수를 더하여 출력하는 프로그램을 작성해보세요.

```
1 def mySum(num1, num2):  
2     print("%s + %s = %s" %(num1, num2, num1+num2))  
3  
4 mySum(2, 3)  
5 mySum(5, 7)  
6 mySum(10, 15)
```

## 011. 함수 - 함수에서 값의 반환

---

5! 을 계산하여 반환하는 프로그램을 작성해보세요.

# 011. 함수 - 함수에서 값의 반환

5! 을 계산하여 반환하는 프로그램을 작성해보세요.

```
1 def Factorial():
2     result = 1
3     num = 5
4
5     while num > 0:
6         result *= num
7         num -= 1
8
9     return result
10
11 result = Factorial()
12 print(result)
```

## 012. 함수 - 함수에 값 전달과 반환값 받기

---

앞의  $5!$  을 계산하여 반환하는 프로그램을 수정하여  
함수에 값을 전달하여 반환하는 프로그램을 작성해보세요.  
예를 들어  $3$ 을 넣으면  $3!$  인  $6$ 이 반환되게 작성합니다.

## 012. 함수 - 함수에 값 전달과 반환값 받기

앞의 5! 을 계산하여 반환하는 프로그램을 수정하여  
함수에 값을 전달하여 반환하는 프로그램을 작성해보세요.  
예를 들어 3을 넣으면 3! 인 6이 반환되게 작성합니다.

```
1 def Factorial(num):  
2     result = 1  
3     while num > 0:  
4         result *= num  
5         num -= 1  
6     return result  
7  
8 num = 3  
9 result = Factorial(num)  
10 print(result)
```

## 013. 사용자로부터 값 받기 - 하드코딩 피하기

---

사용자로부터 값(이름)을 입력받아

Hello [입력받은 값] 을 출력하는 프로그램을 작성해보세요.

# 013. 사용자로부터 값 받기 - 하드코딩 피하기

사용자로부터 값(이름)을 입력받아

Hello [입력받은 값] 을 출력하는 프로그램을 작성해보세요.

```
1 name = input("이름 입력: ")  
2 print("Hello %s." % name)
```

## 014. 사용자로부터 값 받기 활용 문항

---

사용자로부터 한 글자를 입력받아, 입력받은 문자가 숫자인지 문자인지 출력하는 프로그램을 작성해보세요.

# 014. 사용자로부터 값 받기 활용 문항

사용자로부터 한 글자를 입력받아, 입력받은 문자가 숫자인지 문자인지 출력하는 프로그램을 작성해보세요.

```
1 s = input("입력: ")
2
3 if s.isalpha():
4     print("%s는 문자." % s)
5 else:
6     print("%s는 숫자." % s)
```

## 015. 커맨드라인에서 인수 입력받기

---

작성한 파이썬 스크립트를 콘솔 환경에서 실행할 때,  
외부에서 인수를 입력받아 이를 출력하도록 해봅시다.

# 015. 커맨드라인에서 인수 입력받기

---

작성한 파이썬 스크립트를 콘솔 환경에서 실행할 때,  
외부에서 인수를 입력받아 이를 출력하도록 해봅시다.

```
1 import sys  
2  
3 s = sys.argv[1]  
4  
5 print("Hello %s." % s)
```

# 016. 파일 읽기

---

read\_sample.txt 파일을 읽어 내용을 출력해보세요.

# 016. 파일 읽기

---

read\_sample.txt 파일을 읽어 내용을 출력해보세요.

```
1 f = open("read_sample.txt", 'r')
2 r = f.readlines()
3 f.close()
4
5 for s in r:
6     print(s.strip())
```

# 016. 파일 읽기

---

read\_sample.txt 파일을 읽어 내용을 출력해보세요.

```
1 with open("read_sample.txt",'r') as handle:  
2     for line in handle:  
3         print(line.strip())
```

## 017. 파일 쓰기

---

다음 내용을 write\_sample.txt 파일에 써보세요.

# 017. 파일 쓰기

---

다음 내용을 write\_sample.txt 파일에 써보세요.

```
1 f = open("write_sample.txt", 'w')
2
3 f.write("Hello\n")
4 f.write("write_sample text file\n")
5
6 f.close()
```

# 017. 파일 쓰기

---

다음 내용을 write\_sample.txt 파일에 써보세요.

```
1 write_string = "Hello\nwrite_sample text file\n"
2
3 with open("write_sample.txt", 'w') as handle:
4     handle.write(write_string)
```

# 018. 주석 달기

---

파이썬에서 주석은 다음과 같이 작성합니다.

```
1 # Single line comment  
2  
3 """  
4 Multiple line  
5 comment  
6 """  
7
```

# 019. 예외 처리하기 (Debugging)

다음 스크립트는 파일이 없다는 오류가 발생합니다.  
오류가 발생하지 않도록 try-except를 사용해보세요.

```
1 with open("noname.txt",'r') as fr:  
2     read = fr.readlines()  
3  
4 print(read)
```

# 019. 예외 처리하기 (Debugging)

다음 스크립트는 파일이 없다는 오류가 발생합니다.  
오류가 발생하지 않도록 try-except를 사용해보세요.

```
1 with open("noname.txt",'r') as fr:  
2     read = fr.readlines()  
3  
4 print(read)
```

```
1 try:  
2     with open("noname.txt",'r') as fr:  
3         read = fr.readlines()  
4     print(read)  
5 except FileNotFoundError:  
6     print("파일이 없습니다.")
```

# 019. 예외 처리하기 (Debugging)

다음 스크립트는 여러 오류가 있습니다.

값을 0을 넣는 경우

값을 넣지 않는 경우

두 가지 오류가 발생하는 상황을 처리해 봅시다.

```
1 num = int(input("Enter: "))
2 print(10 / num)
```

# 019. 예외 처리하기 (Debugging)

다음 스크립트는 여러 오류가 있습니다.

값을 0을 넣는 경우

값을 넣지 않는 경우

두 가지 오류가 발생하는 상황을 처리해 봅시다.

```
1 num = int(input("Enter: "))
2 print(10 / num)
```

```
1 try:
2     num = int(input("Enter: "))
3     print(10 / num)
4 except ZeroDivisionError:
5     print("0으로는 나눌 수 없습니다.")
6 except ValueError:
7     print("값을 입력해주세요.")
```

## 020. 문자열 더하기

---

두 개의 변수를 선언 한 후 각각 Bio와 Informatics를 넣은 후 다른 변수에는 두 변수를 합친 값을 집어넣고 출력해보세요.

## 021. 문자열 n 번째 출력하기

---

문자열 변수 Seq1에 “AGTTTATAG” 가 담겨있습니다.  
Seq1의 6번째 문자를 출력해보세요.

## 022. 문자열 나누기

---

문자열 변수 Seq1에 “AGTTTATAG” 가 담겨있습니다.  
Seq1의 4번째 문자부터 6번째 문자까지 출력해보세요.

## 023. 문자열 길이 구하기

---

문자열 변수 Seq1에 “AGTTTATAG” 가 담겨있습니다.  
Seq1의 길이를 구해보세요.

## 024. 문자열 대소문자 변환하기

---

문자열 변수 Seq1에 “ATGttATaG” 가 담겨있습니다.  
Seq1을 모두 대문자, 모두 소문자로 바꾸어보세요.

## 025. 문자열 n씩 건너뛰며 출력하기

---

문자열 변수 Seq1에 “ATGTTATAG” 가 담겨있습니다.  
Seq1에서 3칸씩 건너뛰며 출력, 즉 첫 번째, 네 번째, 일곱 번째 만  
출력하는 프로그램을 작성해보세요.

## 026. 문자열 n씩 건너뛰며 출력하기

---

문자열 변수 Seq1에 “ATGTTATAG” 가 담겨있습니다.  
Seq1에서 3칸씩 건너뛰며 다음과 같이 출력해보세요.

ATG

TTA

TAG

## 027. 문자열 순서 뒤집기

---

문자열 변수 Seq1에 “ATGTTATAG” 가 담겨있습니다.  
Seq1을 거꾸로 뒤집어 출력해보세요.

## 028. 문자열 바꾸기

---

문자열 변수 Seq1에 “ATGTTATAG” 가 담겨있습니다.  
Seq1에서 A는 T로 T는 A로 C는 G로 G는 C로 바꾸어 출력해보세요.

## 029. 특정 문자 확인하기

---

문자열 변수 Seq1에 “ATGTTATAG” 가 담겨있습니다.  
Seq1에서 C가 들어있는지 확인해보세요.

## 030. 문자열 index 번호 출력하기

---

문자열 변수 Seq1에 “AGTTTATAG” 가 담겨있습니다.  
Seq1에서 TT가 출현하는 index를 출력해보세요.

TTT 가 붙어있음을 주의합시다.

## 031. 문자열 리스트로 바꾸기

---

문자열 AA,AC,AG,AT 를 쉼표를 기준으로 리스트로 바꿔보세요.

## 032. 리스트에 요소 추가하기

---

[“AA”, “AC”, “AG”, “AT”] 에 문자열 “CA” 를 추가해보세요.

## 033. 리스트 순서 뒤집기

---

[“AA”, “AC”, “AG”, “AT”] 의 순서를 뒤집어보세요.

## 034. 리스트에서 가장 큰 값, 작은 값을 구하기

---

리스트 [3, 1, 1, 2, 0, 0, 2, 3, 3]에서 가장 큰 값과 가장 작은 값을 구해보세요

## 034. 리스트의 요소값을 사전으로 세기

---

리스트 [3, 1, 1, 2, 0, 0, 2, 3, 3]에서 각 요소의 출현 빈도를 사전으로 세어보세요.

{0: 2, 1: 2, 2: 2, 3: 3}

# 프로그래밍 과제1

---

1. 지금까지 프로그래밍 한 스크립트를 github repository에 올려보세요. 디렉토리 를 만들어 업로드 해도 좋고 그냥 올려도 좋고 자유입니다.
2. 자신의 이름을 출력하는 프로그램을 작성해보세요.
3. 염기 한 글자를 받는 함수를 만들어  
A를 넣으면 Adenine,  
C를 넣으면 Cytosine,  
G를 넣으면 Guanine,  
T를 넣으면 Thymine,  
U를 넣으면 Uracil  
을 출력하는 프로그램을 작성해보세요.
4. 어떠한 수  $n$ 을 입력받아  $10 / n$ 의 결과를 출력하는 프로그램을 작성해 보세요.  
만약  $n$ 이 0인 경우는 어떻게 되나요? 발생하는 오류를 try-except로 잡아보세요.
5. 재귀 알고리즘을 사용하여  $5!$ 을 구하는 프로그램을 만들어보세요

04

## 기본 파일 종류 설명

---

# 파일의 종류 - 텍스트 파일과 바이너리 파일

파일의 종류는 크게 두 가지로 나눌 수 있음

## 1) 텍스트 파일

간단히 설명하자면 사람이 눈으로 읽을 수 있는 파일 예를 들어 메모장에서 작성한 txt 파일

```
$ cat test.txt  
This is a test file.  
$ file test.txt  
test.txt: ASCII text  
$
```

## 2) 바이너리 파일

## 간단히 설명하자면 사람이 눈으로 읽을 수 없는 파일

예를 들어 그림파일..을 less로 열어보면  
오른쪽처럼 보인다. 😣

```
$ file info_1.png
info_1.png: PNG image data, 1080 x 1920, 8-bit/color RGBA, non-interlaced
$ less info_1.png
"info_1.png" may be a binary file. See it anyway? █
```

# 파일의 종류 - 텍스트 파일

텍스트 파일의 종류에 대해 좀 더 알아보자.

## 1) txt 파일

plain한 text 문자로 이루어진 파일

```
$ cat test.txt  
This is a test file.  
$ file test.txt  
test.txt: ASCII text  
$ █
```

## 2) csv 파일 (comma separated value)

각 항목들이 쉼표(comma)로 구분된  
텍스트 파일

```
$ cat test.csv  
id,sequence,species  
1,ACAGGGTTA,Influenza  
2,TTAACCAAG,Herpes  
3,GCGAATGAC,Epstein-barr  
$
```

## 3) tsv 파일 (tab separated value)

각 항목들이 탭(tab)으로 구분된  
텍스트 파일

```
$ cat test.tsv  
id sequence species  
1 ACAGGGTTA Influenza  
2 TTAACCAAG Herpes  
3 GCGAATGAC Epstein-barr  
$
```

# 파일의 종류 - 텍스트 파일

---

## 4) xml 파일

eXtensible Markup Language 의 약자로 구조화된 데이터를 표현하고 데이터를 주고 받을 때 사용하는 범용 포맷

우리가 흔히 아는 웹페이지의 HTML(HyperText Markup Language)도 XML과 같이 마크업 언어로 형태가 비슷함

```
$ cat test.xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <row>
    <id>1</id>
    <sequence>ACAGGGTTA</sequence>
    <species>Influenza</species>
  </row>
  <row>
    <id>2</id>
    <sequence>TTAACCAAG</sequence>
    <species>Herpes</species>
  </row>
  <row>
    <id>3</id>
    <sequence>GCGAATGAC</sequence>
    <species>Epstein-barr</species>
  </row>
</root>
$
```

---

# 파일의 종류 - 텍스트 파일

## 5) json 파일

JavaScript Object Notation의 약자로 XML과 마찬가지로 구조화된 데이터를 표현하고 정보를 주고받기위해 만든 포맷이다.

key : value 의 구조로 이루어 져있으며 파일의 dictionary 자료형과 형태가 같다.

```
$ cat test.json
[
  {
    "id": 1,
    "sequence": "ACAGGGTTA",
    "species": "Influenza"
  },
  {
    "id": 2,
    "sequence": "TTAACCAAG",
    "species": "Herpes"
  },
  {
    "id": 3,
    "sequence": "GCGAATGAC",
    "species": "Epstein-barr"
  }
]
```

# 파이썬으로 파일 읽고 파싱하는 방법

---

## 파이썬으로 파일을 읽고 파싱하는 방법

- 1) 파일을 less로 열어서 어떠한 구조로 생겼는지 확인, 원하는 데이터가 어디에 어떠한 형태로 있는지 확인한다.
- 2) 파이썬으로 파일을 연다.
- 3) 파일을 읽어서 문자열에서 원하는 부분들을 가져온다.

# txt 파일 파이썬으로 읽기

---

python으로 txt 파일을 읽어서 출력해봅시다.

# txt 파일 파이썬으로 읽기

---

python으로 txt 파일을 읽어서 출력해봅시다.

```
1 f = "test.txt"  
2  
3 with open(f, 'r') as fr:  
4     for line in fr:  
5         print(line.strip())  
6
```

---

# csv, tsv 파일 파이썬으로 읽기

---

python으로 csv 파일과 tsv 파일을 읽어서 출력해봅시다.

각 라인을 csv는 comma 구분자로

tsv는 tab 구분자로 split 하여 나온 결과를 출력해봅시다.

# csv, tsv 파일 파일 쪼개기

python으로 csv 파일과 tsv 파일을 읽어서 출력해봅시다.

각 라인을 csv는 comma 구분자로

tsv는 tab 구분자로 split 하여 나온 결과를 출력해봅시다.

```
1 f = "test.csv"  
2  
3 with open(f, 'r') as fr:  
4     for line in fr:  
5         print(line.strip().split(", "))  
6
```

```
1 f = "test.tsv"  
2  
3 with open(f, 'r') as fr:  
4     for line in fr:  
5         print(line.strip().split("\t"))  
6
```

```
$ python readCsv.py  
['id', 'sequence', 'species']  
['1', 'ACAGGGTTA', 'Influenza']  
['2', 'TTAACCAAG', 'Herpes']  
['3', 'GCGAATGAC', 'Epstein-barr']  
$  
$ python readTsv.py  
['id', 'sequence', 'species']  
['1', 'ACAGGGTTA', 'Influenza']  
['2', 'TTAACCAAG', 'Herpes']  
['3', 'GCGAATGAC', 'Epstein-barr']  
$
```

# json 파일 파이썬으로 읽기

---

python으로 csv 파일과 tsv 파일을 읽어서 출력해봅시다.

각 라인을 csv는 comma 구분자로

tsv는 tab 구분자로 split 하여 나온 결과를 출력해봅시다.

# json 파일 파일쓰으로 읽기

python으로 csv 파일과 tsv 파일을 읽어서 출력해봅시다.

각 라인을 csv는 comma 구분자로

tsv는 tab 구분자로 split 하여 나온 결과를 출력해봅시다.

```
1 import json
2
3 f = "test.json"
4
5 with open(f, 'r') as fr:
6     data = json.load(fr)
7
8 print(data)
9

$ python readJson.py
[{"id": 1, "sequence": "ACAGGGTTA", "species": "Influenza"}, {"id": 2
, "sequence": "TTAACCAAG", "species": "Herpes"}, {"id": 3, "sequence"
: "GCGAATGAC", "species": "Epstein-barr"}]
$
```

# json 파일 파일쓰기로 읽기

The screenshot shows a web browser with two tabs open. The left tab is a Google search results page for "python document json". The right tab is the official Python documentation for the `json` module.

**Google Search Results:**

- 검색 결과 약 28,400,000개 (0.40초)
- 도움말: 한국어 검색결과만 검색합니다. 환경설정에
- json — JSON encoder and decoder**  
https://docs.python.org/3/library/json.html  
json exposes an API familiar to users of the standard library `marshal` module to encode and decode a JSON document from a string that may be a Python object or a Unicode instance containing a JSON document.
- 18.2. json — JSON encoder and decoder**  
https://docs.python.org/2/library/json.html  
json exposes an API familiar to users of the standard library `marshal` module to encode and decode a JSON document from a string that may be a Python object or a Unicode instance containing a JSON document.
- json --- JSON エンコーダおよびデコード**  
https://docs.python.org/ja/3/library/json.html  
Deserialize fp (a.read() -supporting text file or file-like object) to a Python object using this conversion table. object\_hookはオブジェクトをJSONオブジェクトに変換するための関数。
- 19.2. json — JSON encoder and decoder**  
https://docs.python.org/3.4/library/json.html  
json exposes an API familiar to users of the standard library `marshal` module to encode and decode a JSON document from a string that may be a Python object or a Unicode instance containing a JSON document.
- Working With JSON Data in Python**  
https://realpython.com/python-json/ 0 | 100%  
You'll see hands-on examples of working with JSON data using the Python standard library's `json` module, as well as how to use it to interact with an API or storing your data in a document database.

**Python Documentation for json — JSON encoder and decoder:**

- Table of Contents**
  - json — JSON encoder and decoder
    - Basic Usage
    - Encoders and Decoders
    - Exceptions
    - Standard Compliance and Interoperability
      - Character Encodings
      - Infinite and NaN Number Values
      - Repeated Names Within an Object
      - Top-level Non-Object, Non-Array Values
      - Implementation Limitations
    - Command Line Interface
      - Command line options
- Source code: Lib/json/\_init\_.py**
- JSON (JavaScript Object Notation)**, specified by [RFC 7159](#) (which obsoletes [RFC 4627](#)) and by [ECMA-404](#), is a lightweight data interchange format inspired by [JavaScript](#) object literal syntax (although it is not a strict subset of [JavaScript](#) [1]).
- json** exposes an API familiar to users of the standard library `marshal` and `pickle` modules.
- Encoding basic Python object hierarchies:**

```
>>> import json
>>> json.dumps([{'foo': {'bar': ('baz', None, 1.0, 2)}}])
'[{"foo": {"bar": ["baz", null, 1.0, 2]}}]'
>>> print(json.dumps("\\"foo\\bar"))
"\\"foo\\bar"
>>> print(json.dumps('\u1234'))
"\u1234"
>>> print(json.dumps('\\\\'))
"\\\\"
>>> print(json.dumps({'c': 0, 'b': 0, 'a': 0}, sort_keys=True))
{'a': 0, 'b': 0, 'c': 0}
>>> from io import StringIO
>>> io = StringIO()
>>> json.dump(['streaming API'], io)
>>> io.getvalue()
['"streaming API"]'
```

- Compact encoding:**

# 프로그래밍 과제2

---

1. 파일을 파싱할 때 고려해야 할 사항은 무엇인가요?
2. csv, tsv 파일의 차이점은 무엇인가요?
3. json 파일은 무엇인가요?
4. 다음 csv 파일을 json으로 바꾸어 보세요.

CSV

sample1,sample2,sample3,sample4,sample5  
20.1,30.4,28.7,22.8,31.8

JSON

{“sample1”: 20.1, “sample2”: 30.4, “sample3”: 28.7, “sample4”: 22.8  
“sample5”: 31.8}

# 프로그래밍 과제2

---

5. k-mer를 만드는 프로그램을 작성해보세요

사용자가 3을 입력하면

AAA

AAC

AAG

AAT

...

TTT

까지 출력합니다.

# 프로그래밍 과제2

---

6. 앞서 만든 kmer를 활용하여 7mer 중 문자열이 회문 구조 (palindrome) 를 만족하는 문자열들의 개수를 출력해보세요.

답: 256

palindrome 한 문자열의 예시는 다음과 같습니다.

AAAAA  
ACACA  
CCGCC  
TTATT

## 05 | GATK Best Practice Pipeline

---

# file type

---

| Menu  | Short | Tall  |
|-------|-------|-------|
| 아메리카노 | 3,500 | 4,000 |
| 라떼    | 3,800 | 4,300 |
| 그린티   | 3,000 | 3,500 |

아메리카노 Tall 사이즈  
가 먹고싶다



생물 정보학에서 다루는 파일의 형식들을 알아봅시다.

FASTA, FASTQ

SAM, BAM

BED

VCF

# FASTA

---

KT225476.2.fasta

```
>KT225476.2 Middle East respiratory syndrome coronavirus isol  
ate MERS-CoV/THA/CU/17_06_2015, complete genome  
AGTGAATAGCTTGGCTATCTCACTTCCCCTCGTTCTTGCAGAACTTGATTAAACGAA  
CTTAAATAA  
AAGCCCTGTTGTTAGCGTATTGTTGCACTTGTCTGGTGGGATTGTGGCACTAATTGCCT  
GCTCATCTA  
... 중간 생략 ...  
ATCAACAGACCAAATGTCTGAACCTCCAAAGGAGCAGCGTGTGCAAGGTAGCATCACTCAG  
CGCACTCGC  
ACCCGTCCAAGTGTTCAGCCTGGTCCAATGATTGATGTTAACACTGATTAGTGTCACTC
```

> 로 시작하는 헤더와  
두 번째 줄 부터는 서열이 있다.



# FASTQ

```
@EAS123:456:FC789VJ:3:1103:26362:2088 1:N:0:ACGTACGT  
GCATGGAGGTGGCGCTGCAGACTGAGCCCCACTTGCTGGCTGGCACCGTCAACCCACC GT  
GGGCAAGAGGAATGTCACGCTGCCATCGACAACTGCCTC  
+  
AAFFFJFJ|||||||||||||||||||||||||||||||||||||||||||||||||||||  
|||||||||||||||||||||||||||||||||||||||||||||||||||||  
... 하락
```

FASTQ 는 네 줄로 구성된다.  
헤더, 서열, 구분자(+), 퀼리티  
시퀀서에서 나온 데이터가 FASTQ다.



# SAM/BAM

BAM은 SAM의 binary 버전이다.



VCF

```

.../Tools/reference — ssh -i SWU_class1.pem centos@ec2-54-180-101-117.ap-northeast-2.compute.amazonaws.com ...
~/Genome-Analysis-Tutorial — less -S SRR000982.Filtered.Variants.vcf — 194x63

##contig=
##contig=
##contig=
##contig=
##contig=
##contig=
##contig=
##reference=file:///Users/jhan/etc/reference/hg19/etc/hg19.fasta
##source=GenotypeGVCFs

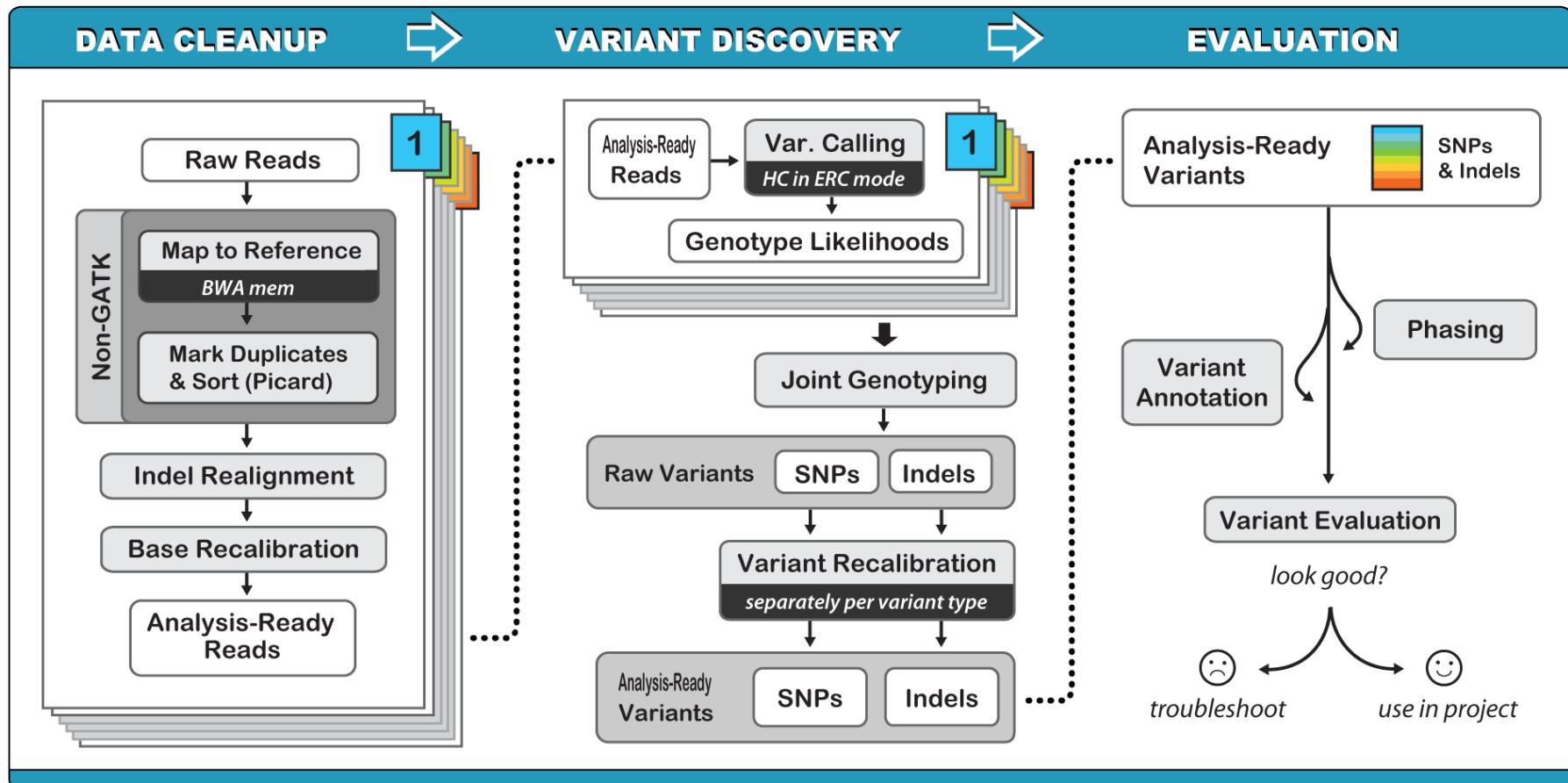
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT SRR000982
chrM 73 . G A 117.98 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=60.00;QD=23.58;SOR=1.022 GT:AD:DP:Q:PL 1/1:0.5:5:15:146,15,0
chrM 73 . G A 117.98 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=60.00;QD=23.58;SOR=1.022 GT:AD:DP:Q:PL 1/1:0.5:5:15:146,15,0
chrM 150 . T C 362.78 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=59.98;QD=25.36;SOR=0.892 GT:AD:DP:Q:PGT:PID:PL 1/1:0.8:8:27:1:1:1
chrM 150 . T C 362.78 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=59.98;QD=25.36;SOR=0.892 GT:AD:DP:Q:PGT:PID:PL 1/1:0.8:8:27:1:1:1
chrM 152 . T C 362.78 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=59.98;QD=25.36;SOR=0.892 GT:AD:DP:Q:PGT:PID:PL 1/1:0.8:8:27:1:1:1
chrM 152 . T C 362.78 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=59.98;QD=25.36;SOR=0.892 GT:AD:DP:Q:PGT:PID:PL 1/1:0.8:8:27:1:1:1
chrM 195 . C T 138.98 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=58.45;QD=26.18;SOR=1.981 GT:AD:DP:Q:PL 1/1:0.5:5:15:159,15,0
chrM 195 . C T 138.98 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=58.45;QD=26.18;SOR=1.981 GT:AD:DP:Q:PL 1/1:0.5:5:15:159,15,0
chrM 410 . A T 36.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=18.37;SOR=0.693 GT:AD:DP:Q:PL 1/1:0.2:2:6:64,6,0
chrM 410 . A T 36.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=18.37;SOR=0.693 GT:AD:DP:Q:PL 1/1:0.2:2:6:64,6,0
chrM 2261 . C T 156.84 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=57.64;QD=24.14;SOR=1.329 GT:AD:DP:Q:PL 1/1:0.6:16:18:185,18,0
chrM 2261 . C T 156.84 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=57.64;QD=24.14;SOR=1.329 GT:AD:DP:Q:PL 1/1:0.6:16:18:185,18,0
chrM 2354 . C T 63.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=21.99;SOR=1.179 GT:AD:DP:Q:PL 1/1:0.3:3:9:91,9,0
chrM 2354 . C T 63.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=21.99;SOR=1.179 GT:AD:DP:Q:PL 1/1:0.3:3:9:91,9,0
chrM 2708 . G A 35.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=17.87;SOR=0.693 GT:AD:DP:Q:PL 1/1:0.2:2:6:63,6,0
chrM 2708 . G A 35.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=17.87;SOR=0.693 GT:AD:DP:Q:PL 1/1:0.2:2:6:63,6,0
chrM 4746 . G T 87.83 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=51.74;QD=21.76;SOR=1.689 GT:AD:DP:Q:PL 1/1:0.4:12:115,12,0
chrM 4746 . G A 87.83 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=51.74;QD=21.76;SOR=1.689 GT:AD:DP:Q:PL 1/1:0.4:12:115,12,0
chrM 5501 . C T 124.98 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=45.00;QD=21.98;SOR=0.611 GT:AD:DP:Q:PL 1/1:0.5:15:15:153,15,0
chrM 5501 . C T 124.98 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=45.00;QD=21.98;SOR=0.611 GT:AD:DP:Q:PL 1/1:0.5:15:15:153,15,0
chrM 7338 . G A 37.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=58.00;QD=24.44;SOR=0.393 GT:AD:DP:Q:PL 1/1:0.2:2:6:65,6,0
chrM 7338 . G A 37.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=58.00;QD=24.44;SOR=0.393 GT:AD:DP:Q:PL 1/1:0.2:2:6:65,6,0
chrM 6702 . G A 64.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=60.00;QD=21.43;SOR=0.633 GT:AD:DP:Q:PL 1/1:0.3:3:9:92,9,0
chrM 6702 . G A 64.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=60.00;QD=21.43;SOR=0.633 GT:AD:DP:Q:PL 1/1:0.3:3:9:92,9,0
chrM 10399 . G A 157.84 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=26.31;SOR=0.329 GT:AD:DP:Q:PL 1/1:0.6:16:18:186,18,0
chrM 10399 . G A 157.84 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=26.31;SOR=0.329 GT:AD:DP:Q:PL 1/1:0.6:16:18:186,18,0
chrM 10920 . G A 63.26 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=58.69;QD=20.99;SOR=1.179 GT:AD:DP:Q:PL 1/1:0.3:3:9:92,9,0
chrM 10920 . G A 63.26 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=58.69;QD=20.99;SOR=1.179 GT:AD:DP:Q:PL 1/1:0.3:3:9:92,9,0
chrM 10874 . C T 159.84 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=55.00;QD=26.44;SOR=0.393 GT:AD:DP:Q:PL 1/1:0.3:3:9:180,18,0
chrM 10874 . C T 159.84 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=55.00;QD=26.44;SOR=0.393 GT:AD:DP:Q:PL 1/1:0.3:3:9:180,18,0
chrM 11018 . C T 88.03 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=54.58;QD=22.91;SOR=0.3256 GT:AD:DP:Q:PL 1/1:0.9:4:12:116,12,0
chrM 11018 . C T 88.03 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=54.58;QD=22.91;SOR=0.3256 GT:AD:DP:Q:PL 1/1:0.9:4:12:116,12,0
chrM 11720 . A G 99.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=76.00;QD=39.89;SOR=0.179 GT:AD:DP:Q:PL 1/1:0.3:3:9:11:11:11
chrM 11720 . A G 99.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=76.00;QD=39.89;SOR=0.179 GT:AD:DP:Q:PL 1/1:0.3:3:9:11:11:11
chrM 11723 . C T 99.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=69.00;QD=37.07;SOR=0.293 GT:AD:DP:Q:PL 1/1:0.2:1:2:9:11:11:11
chrM 11723 . C T 99.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=69.00;QD=37.07;SOR=0.293 GT:AD:DP:Q:PL 1/1:0.2:1:2:9:11:11:11
chrM 11723 . C T 99.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=69.00;QD=37.07;SOR=0.293 GT:AD:DP:Q:PL 1/1:0.2:1:2:9:11:11:11
chrM 11723 . C T 99.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=69.00;QD=37.07;SOR=0.293 GT:AD:DP:Q:PL 1/1:0.2:1:2:9:11:11:11
chrM 12706 . C T 152.84 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=69.00;QD=25.47;SOR=0.493 GT:AD:DP:Q:PL 1/1:0.1:16:18:181,18,0
chrM 12706 . C T 152.84 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=69.00;QD=25.47;SOR=0.493 GT:AD:DP:Q:PL 1/1:0.1:16:18:181,18,0
chrM 12851 . G A 65.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=69.00;QD=21.76;SOR=0.179 GT:AD:DP:Q:PL 1/1:0.3:3:9:93,9,0
chrM 12851 . G A 65.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=69.00;QD=21.76;SOR=0.179 GT:AD:DP:Q:PL 1/1:0.3:3:9:93,9,0
chrM 13327 . C T 33.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=51.79;QD=16.87;SOR=0.693 GT:AD:DP:Q:PL 1/1:0.8:2:2:6:61,6,0
chrM 13327 . C T 33.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=51.79;QD=16.87;SOR=0.693 GT:AD:DP:Q:PL 1/1:0.8:2:2:6:61,6,0
chrM 13681 . C T 34.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=37.37;SOR=0.293 GT:AD:DP:Q:PL 1/1:0.2:2:6:62,6,0
chrM 13681 . C T 34.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=37.37;SOR=0.293 GT:AD:DP:Q:PL 1/1:0.2:2:6:62,6,0
chrM 14213 . C T 35.74 PASS SNP_FILTER_AC2=AC2;AF=1.00;AN=2;DP=2;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=38.33;QD=17.87;SOR=0.693 GT:AD:DP:Q:PL 1/1:0.1:16:2:6:63,6,0
chrM 14213 . C T 35.74 PASS AC=2;AF=1.00;AN=2;DP=2;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=38.33;QD=17.87;SOR=0.693 GT:AD:DP:Q:PL 1/1:0.1:16:2:6:63,6,0
chrM 14581 . G A 66.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=59.34;QD=22.89;SOR=0.179 GT:AD:DP:Q:PL 1/1:0.3:3:9:94,9,0
chrM 14581 . G A 66.28 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=59.34;QD=22.89;SOR=0.179 GT:AD:DP:Q:PL 1/1:0.3:3:9:94,9,0
chrM 14767 . C T 32.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=37.37;SOR=0.693 GT:AD:DP:Q:PL 1/1:0.2:2:6:68,6,0
chrM 14767 . C T 32.74 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=68.00;QD=37.37;SOR=0.693 GT:AD:DP:Q:PL 1/1:0.2:2:6:68,6,0
chrM 14832 . G A 129.99 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=58.74;QD=25.98;SOR=1.022 GT:AD:DP:Q:PL 1/1:0.5:5:15:158,15,0
chrM 14832 . G A 129.99 PASS AC=2;AF=1.00;AN=2;DP=6;ExcessHet=3.0103;FS=0.000;MLEAC=2;MLEAF=1.00;MQ=58.74;QD=25.98;SOR=1.022 GT:AD:DP:Q:PL 1/1:0.5:5:15:158,15,0

```

VCF는 Variant Calling Format의 약자로  
윈도우의 vCard 파일과 혼동하지 말자.

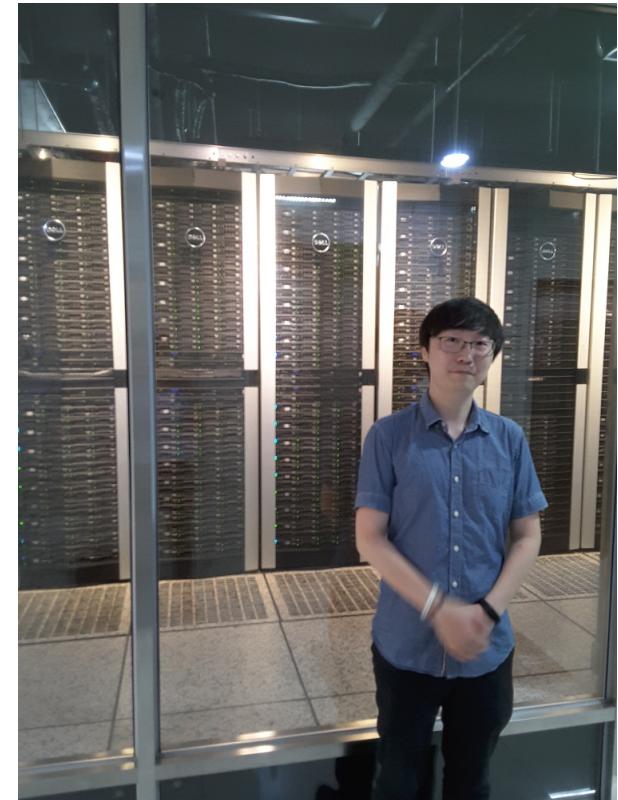


# GATK Best Practice Pipeline



<https://github.com/KennethJHan/Genome-Analysis-Tutorial>

# Sequencer



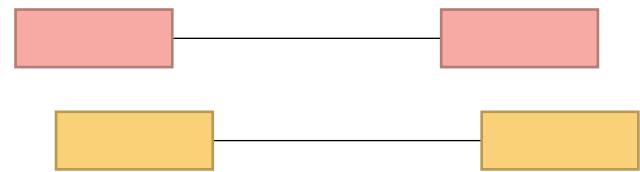
HiSeqX, NovaSeq Sequencer와  
데이터를 처리하는 Server

# Sequencing

sample  
e.g. blood, saliva,  
tissue ..



library  
construction



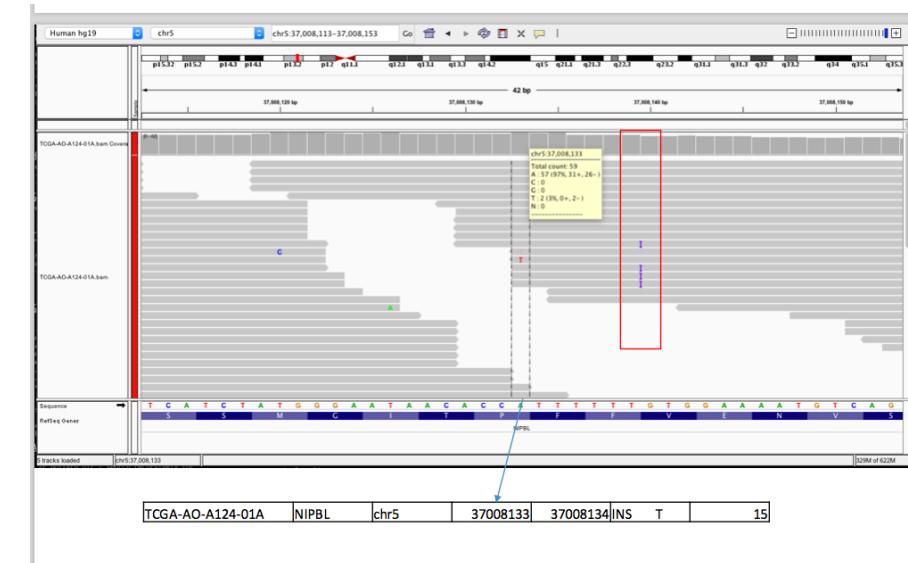
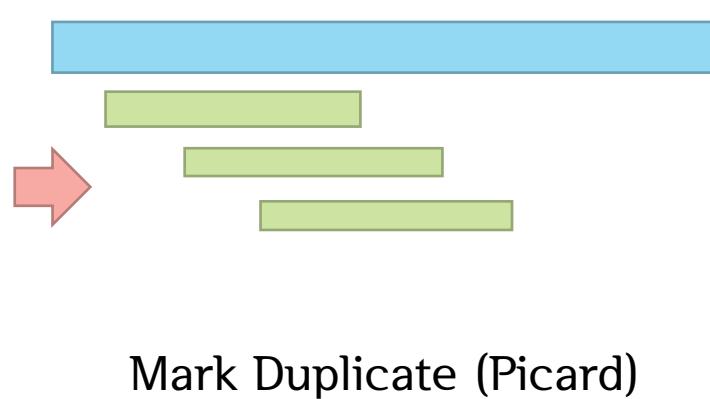
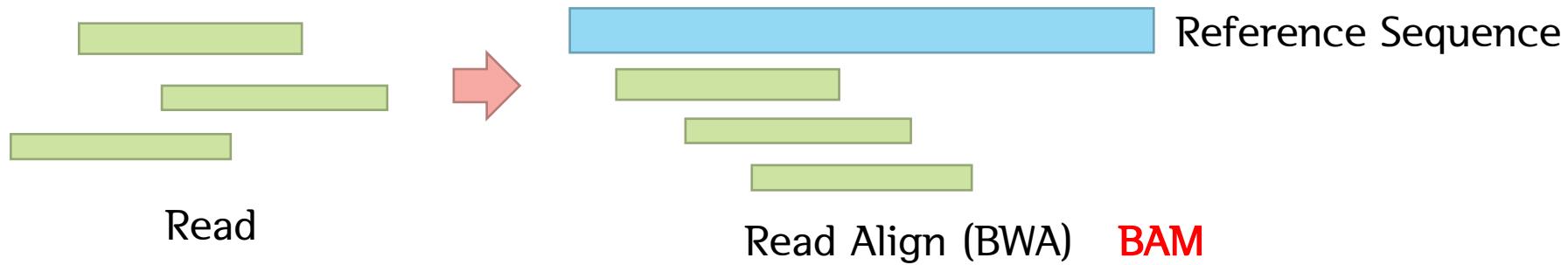
library



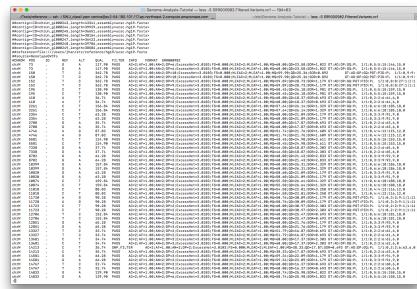
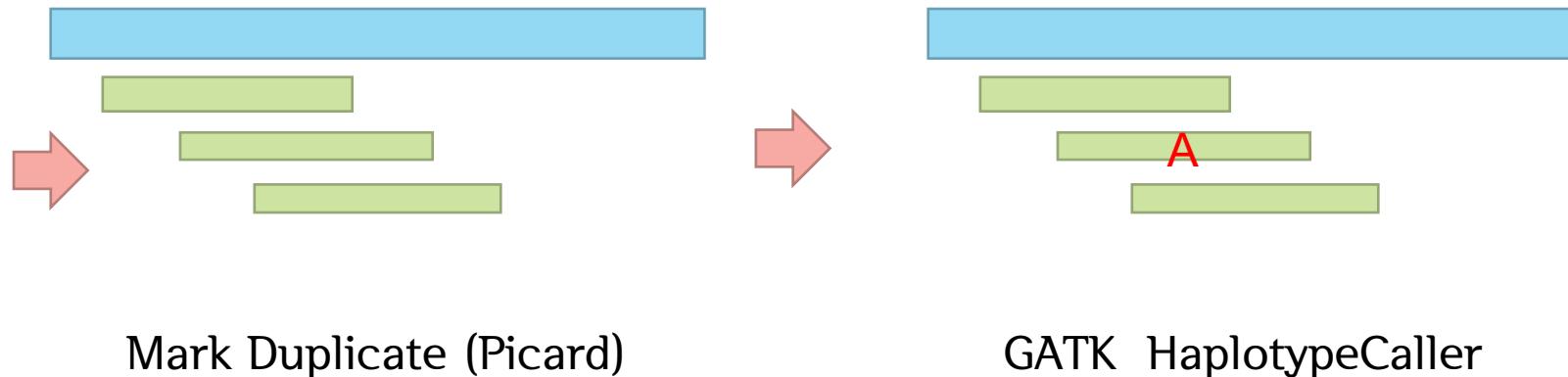
sequencer

Read **FASTQ**

# Sequencing



# Sequencing



## 06 Bioinformatics Python Programming

---

# Resource

---

[https://github.com/KennethJHan/Bioinformatics\\_Programming\\_101](https://github.com/KennethJHan/Bioinformatics_Programming_101)

이번 수업에서 진행할  
FASTQ, BAM, VCF 파일들을  
받을 수 있습니다.

<https://kennethjhan.github.io/Genome-Analysis-Tutorial/>

GATK4 DNA Analysis Pipeline  
을 Step을 따라 보고 만들 수 있는  
material입니다.  
(참고자료)

<https://kennethjhan.github.io/Genome-Analysis-Tutorial/resource>

# FASTA

---

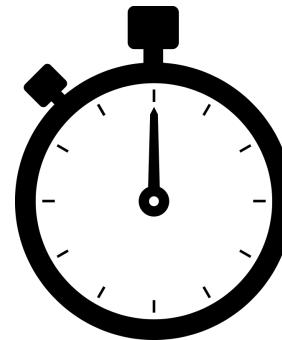
[https://raw.githubusercontent.com/KennethJHan/Bioinformatics\\_Programming\\_101/master/059.fasta](https://raw.githubusercontent.com/KennethJHan/Bioinformatics_Programming_101/master/059.fasta)

wget 으로 받으셔도 됩니다.

FASTA 포맷은 텍스트 기반 포맷으로 염기서열 또는 단백질서열을 나타내기 위해 만든 포맷입니다. > 로 시작하는 헤더와 서열로 이뤄진 레코드입니다.

```
>NC_009333.1:c91512-89473 Human herpesvirus 8, complete genome
ATTCTGACAGGTCAACATGGCGGGACGCAGGCTTACCTGGATTCTGAGTTATTGTAGGTGCTTGAC
TCTGATAAAATATCCTTGGTCAAGTGGCTAGATAGATCTACTGGAACATTCTTGCTCCGGCTGCCCGTA
ATGACGTAATT CCTCTGGATAGCCTACAGTTTCATTGATTAAAGAGGGAATGCCTATCGAAGGGCCT
GCATCCCAGAGATTTACTGGGCTGCCGATTACGGCTTTGGGAAAATATGTACCACGTCGGCGCCTT
```

059.fasta 파일의 염기 A, C, G, T의 개수를 세어보세요



7min

힌트: 파일을 읽어 문자열 메서드를 이용해보자



# FASTA

059.fasta 파일의 염기 A, C, G, T의 개수를 세어보세요

```
1 f = "059.fasta"
2
3 A, C, G, T = 0, 0, 0, 0
4
5 with open(f, 'r') as fr:
6     for line in fr:
7         if line.startswith(">"):
8             header = line.strip()
9         else:
10            seq = line.strip()
11            A += seq.count("A")
12            C += seq.count("C")
13            G += seq.count("G")
14            T += seq.count("T")
15
16 print(f"A: {A}") # A: 497
17 print(f"C: {C}") # C: 444
18 print(f"G: {G}") # G: 585
19 print(f"T: {T}") # T: 514
20
```

with open 으로 파일을 읽어 문자열의 count 메서드를 사용하였다.

print 쪽 16-19라인에  
f-string 을 사용한것에 주목해보자!



# FASTA

059.fasta 파일의 염기 A, C, G, T의 개수를 세어보세요

```
1 from Bio import SeqIO  
2  
3 record = SeqIO.read("059.fasta", "fasta")  
4  
5 A = record.seq.count("A")  
6 C = record.seq.count("C")  
7 G = record.seq.count("G")  
8 T = record.seq.count("T")  
9  
10 print(f"A: {A}") # A: 497  
11 print(f"C: {C}") # C: 444  
12 print(f"G: {G}") # G: 585  
13 print(f"T: {T}") # T: 514  
14
```

여러분께 드리는 보너스!  
Biopython의 SeqIO를 사용하여 간단하게 FASTA 파일  
을 파싱할 수 있다!



# FASTQ

[https://raw.githubusercontent.com/KennethJHan/Bioinformatics\\_Programming\\_101/master/061.fasta](https://raw.githubusercontent.com/KennethJHan/Bioinformatics_Programming_101/master/061.fasta)

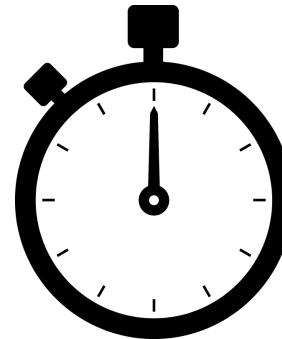
wget 으로 받으셔도 됩니다.

FASTQ 포맷은 텍스트 기반 포맷으로 염기서열과 염기서열에 해당하는 퀄리티 점수를 포함한 파일 포맷입니다. FASTQ는 네 줄이 하나의 리드를 구성합니다.  
첫 번째 줄은 헤더, 두 번째 줄은 서열,  
세 번째 줄은 구분자 +, 네 번째 줄은 퀄리티 입니다.

# FASTQ

---

061.fastq 파일의 염기의 개수를 세어보세요



10min

힌트: FASTQ 파일의 하나의 리드가 네 줄인것을 이용하자



# FASTQ

---

061.fastq 파일의 염기의 개수를 세어보세요

```
1 n = 0
2 length = 0
3 with open("061.fastq", 'r') as fr:
4     for line in fr:
5         if n % 4 == 1:
6             length += len(line.strip())
7         n += 1
8
9 print(length) # 10100
10
```

# FASTQ

061.fastq 파일의 염기의 개수를 세어보세요

```
1 from Bio import SeqIO  
2  
3 length = 0  
4 seq = SeqIO.parse("061.fastq", "fastq")  
5 for s in seq:  
6     length += len(s)  
7  
8 print(length) # 10100  
9
```

여러분께 드리는 보너스!  
Biopython의 SeqIO를 사용하여 간단하게 FASTQ 파일  
을 파싱할 수 있다!



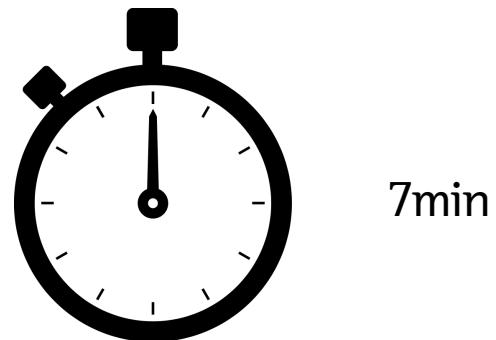
[https://raw.githubusercontent.com/KennethJHan/Bioinformatics\\_Programming\\_101/master/077.bed](https://raw.githubusercontent.com/KennethJHan/Bioinformatics_Programming_101/master/077.bed)

wget 으로 받으셔도 됩니다.

BED(Browser Extensible Data) 포맷은 유전체를 구간별로 나누어 각 구간의 특징을 표기할 수 있는 파일 포맷입니다.

```
chr21  9825763  9825854
chr21  9825960  9826081
chr21  9826185  9826312
chr21  9907173  9907501
chr21  9908283  9908478
chr21  9909046  9909136
chr21  9909146  9909301
chr21  10793870      10793961
chr21  10862590      10862681
chr21  10862819      10862989
```

077.bed 파일의 bed의 구간 길이 합을 구해보세요.



힌트:

첫 번째 컬럼은 chromosome

두 번째 컬럼은 start

세 번째 컬럼은 end

이며 각 컬럼은 탭으로 구분한다.

로 end - start를 빼면 한 행의 길이가 나온다



# BED

---

077.bed 파일의 bed의 구간 길이 합을 구해보세요.

```
1 length = 0
2
3 with open("077.bed", 'r') as fr:
4     for line in fr:
5         chrom_start_end = line.split("\t")
6         #chrom = chrom_start_end[0]
7         start = int(chrom_start_end[1])
8         end = int(chrom_start_end[2])
9         length += end - start
10
11 print(length) # 29681
12
```



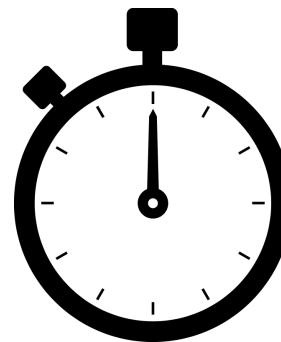
[https://raw.githubusercontent.com/KennethJHan/Bioinformatics\\_Programming\\_101/master/070.vcf](https://raw.githubusercontent.com/KennethJHan/Bioinformatics_Programming_101/master/070.vcf)

wget 으로 받으셔도 됩니다.

VCF (Variant Calling Format)는 변이(variant) 정보를 표기하기 위해 만든 파일 포맷입니다. VCF 포맷은 ##으로 시작하는 메타데이터와 #으로 시작하는 헤더 그리고 변이들의 위치가 적힌 행으로 구성되어 있습니다.

```
##INFO=<ID=SOR,Number=1>Type=Float>Description="Symmetric Odds Ratio of 2x
##INFO=<ID=set,Number=1>Type=String>Description="Source VCF for the merged
##contig=<ID=chr21,length=48129895,assembly=hg19>
##reference=file:///reference/ucsc.hg19.fasta
##source=SelectVariants
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT Sa
chr21 18269600 rs12345 T TGCG 1378.77 PASS AC=1;AF=0.
chr21 18269704 . A C 20434.77 SNP_Filter
chr21 25887701 . T C 91026.77 SNP_Filter
chr21 26698376 . CG C 1052.77 PASS AC=1;AF=0.
chr21 28328640 . A G 205.84 SNP_Filter AC
chr21 31713201 . C T 116.77 SNP_Filter AC
chr21 31713271 rs12346 C T 1052.77 SNP_Filter AC
```

070.vcf 파일에서 #을 제외한 행의 개수를 구해보세요.



7min

힌트:  
파일을 한 줄씩 읽어서 #을 제외하며 읽습니다.

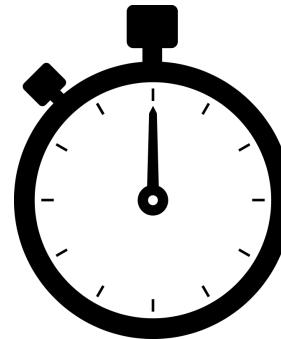


070.vcf 파일에서 #을 제외한 행의 개수를 구해보세요.

```
1 num = 0
2
3 with open("070.vcf", 'r') as fr:
4     for line in fr:
5         if line.startswith("#"):
6             continue
7         num += 1
8
9 print(num) # 12
10
```



070.vcf 파일에서 FILTER 열이 PASS인 행의 개수를 세어보세요.



7min

힌트:

파일을 한 줄씩 읽어서 문자열을 split 하여 if 문에 넣습니다.



070.vcf 파일에서 FILTER 열이 PASS인 행의 개수를 세어보세요.

```
1 cnt = 0
2
3 with open("070.vcf", 'r') as fr:
4     for line in fr:
5         if line.startswith("#"):
6             continue
7         rowList = line.strip().split("\t")
8         if rowList[6] == "PASS":
9             cnt += 1
10
11 print(cnt) # 5
12
```

힌트:

파일을 한 줄씩 읽어서 문자열을 split 하여 if 문에 넣습니다.



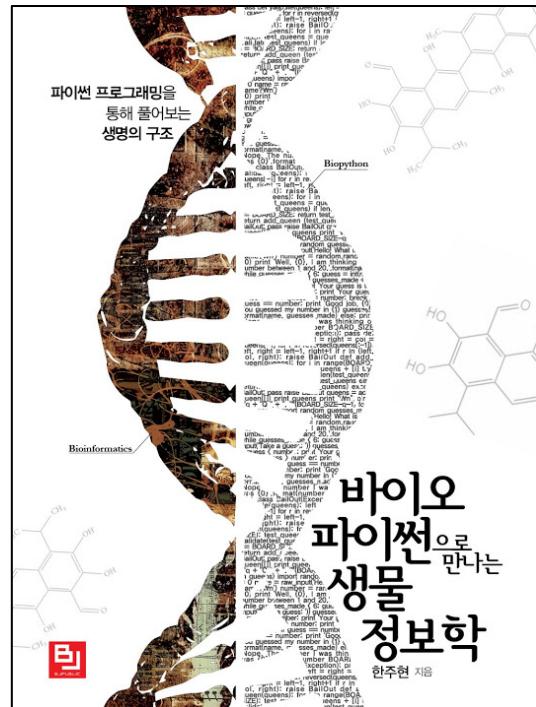
# 프로그래밍 과제3

---

1. 059.fasta 파일의 각 엔지를 파일로 저장한 dictionary를 사용하여 세어보세요.
2. 059.fasta 파일의 각 엔지를 파일로 저장한 dictionary를 사용하여 센 다음 결과를 json으로 저장해보세요
3. 059.fasta 파일의 역상보서열을 출력하는 프로그램을 작성해보세요.
4. 레코드가 여러개 있는 FASTA 파일을 임의로 생성하여 각 레코드의 엔기 개수를 세는 프로그램을 작성해보세요.

# 프로그래밍 과제3

5. 070.vcf 파일에서 INFO 컬럼의 rs값이 있는 행의 chromosome, position, rs값, REF, ALT를 출력해보세요.
6. 070.vcf 파일에서 ALT의 개수를 세어보세요. (multi alt allele에 주의)



더 많은 문항들은 교재에서 만나요~



## 강사 소개

|      |   |
|------|---|
| 이름   | 한주현   |
| 소속   | 전) 마크로젠 데이터분석부<br>현) 쓰리빌리언 Bioinformatics Engineer<br>현) 서울대학교 의료정보학 전공 이규언 교수님 연구실 |
| 메일   | <a href="mailto:kenneth.jh.han@snu.ac.kr">kenneth.jh.han@snu.ac.kr</a>              |
| 웹페이지 | <a href="https://korbillgates.tistory.com">https://korbillgates.tistory.com</a>     |



짧은 시간이기에 못다한 얘기들이 많은데  
여러분들과 헤어지는게 아쉽네요  
필요한 일 있으시면 언제든지 연락주세요 😊

그럼 나중에 또 만나요 !!

