01.프로그램 설치 및 설정

1. 아나콘다 설치
   - 구글 검색 (아나콘다 다운로드)
   - https://www.anaconda.com/products/distribution
   - 아나콘다 설치된 위치 확인하기
2. 작업공간 설정
   - 폴더 생성 및 주소 복사
   - 설치된 Anaconda prompt 열기

     > cd "생성한 폴더 주소"
     > jupyter notebook

3. 바이오파이썬 패키지 설치
   - 설치된 Anaconda prompt 을 **관리자권한** 으로 열기

     > C:\Users\82104\anaconda3\Scripts\pip.exe install biopython

   - 바이오파이썬 설치 확인

     > python
     > Bio._ *version* _

4. R 설치
   - 설치된 Anaconda prompt 을 **관리자권한** 으로 열기

     > conda install –c r r-essentials
     > Proceed([y]/n)? y

*주피터 노트북

In [ ]:
```
###단축키
## Alt + enter  : cell 추가
## Shift+ enter : cell 실행
## ESC -> M 마크다운
## ESC -> Y 코드
## ESC -> x 셀 삭제
```

* 구글 Colab

```
https://colab.research.google.com/?hl=ko
```

---

02. 파이썬 둘러보기

In [ ]:
```
# 화면에 출력하기
print("Hello, Bioinformatics")
```

In [ ]:
```
# 변수 와 사칙연산
# 자료형 : 숫자 (integer)
a = 3
b = 2
print(a)
print(a, "은 숫자입니다.")
print(a+b) # 더하기
print(a-b) # 빼기
print(a*b) # 곱하기
print(a/b) # 나누기
print(a%b) # 나눗셈 후 나머지
print(a**b) # 거듭제곱
```

In [ ]:
```
# 변수
# 자료형 : 문자열 (str)
a = '현수'
b = '정미'
print('누나 이름은', b,'이고, ', '내 친구는 이름은', a, '이다.')
```

In [ ]:
```
#1
a="Bio"
b="Informatics"
c=a+b
print(c)

#2
Met="ATG"
Trp="TGG" *10
His="CAT"

seq = Met+Trp+His
print(seq)
```

In [ ]:
```
print("seq=", seq)

# index: 0,1,2,3,4,,,,,,
print(seq[5])
print(seq[3:6])

print(len(seq))
print(seq.upper())
print(seq.lower())
print(seq.replace("A","T"))

# object.method
# 자동차.색상
# 자동차.앞으로가기
```

In [ ]:
```
#  a = b : a 변수에 b 값을 할당한다.
#  a += b : a 변수에 b 를 더하고 결과를 b 변수에 할당한다. c += a -> c = c + a
```

In [ ]:
```
# if - else 조건문 (단순조건)
a = 3
if a % 2 == 1:
    print(a, "은 홀수다")
else:
    print(b, "은 짝수다")
```

In [ ]:
```
# if - elif - else 조건문 (복수조건)
a = 21
if a % 3 == 0 and  a % 7 ==0:
    print(a , "은 3과 7의 배수이다.")
elif a % 3 == 0:
    print(a, "은 3의 배수이다.")
elif a % 7 == 0:
    print(a, "은 7의 배수이다.")
else:
    pritn(a, "은 3 또는 7의 배수가 아니다.")
```

In [ ]:
```
# for 문
# for 변수  in 순환 가능 객체:
# <명령문>

a = 0
for i in range(1, 11, 1):  # range(start, end, step)
    a += i
print(a)
```

In [ ]:
```
# 문자열
for s in "ACGT":
    print(s)
```

In [ ]:
```
# 리스트 : 숫자나 문자의 모음
for s in ["TTA","TAG","TGA"]:
    print(s)
```

In [ ]:
```
# 딕셔너리 : key 와 value 의 쌍
for s in {"TTA":2, "TAG":3, "TGA":1}:
    print(s)
```

In [ ]:
```
d = {"TTA":2, "TAG":3, "TGA":1}
d.items()
```

In [ ]:
```
d = {"TTA":2, "TAG":3, "TGA":1}
for k, v in d.items():
    print(k,v)
```

In [ ]:
```
# while 문
# while < 조건문 > :
#<명령문>

a = 5
result =1

while a > 0:
    result *= a
    print(result,a)
    a -=1
    print(a)
```

In [ ]:
```
# 함수 만들기
#y=f(x) = 2x+1

def greet():
    print("Hello, Bioinformatics.")

greet()
```

In [ ]:
```

```

In [ ]:
```
arr = range(10)
print(list(arr))
arr[5]
#arr[A:B:C]의 의미는, index A 부터 index B 까지 C의 간격으로 배열을 만들어라는 말입니
arr[::2] # 처음부터 끝까지 두 칸 간격으로
arr[1::2] # index 1 부터 끝까지 두 칸 간격으로
arr[::-1] # 처음부터 끝까지 -1칸 간격으로 ( == 역순으로)
arr[::-2] # 처음부터 끝까지 -2칸 간격으로 ( == 역순, 두 칸 간격으로)
arr[3::-1] # index 3 부터 끝까지 -1칸 간격으로 ( == 역순으로)
arr[1:6:2] # index 1 부터 index 6 까지 두 칸 간격으로
```

In [ ]:
```
# 파일 쓰기 및 읽기

write_string = "HelloWnMy name is JinWn"

with open("write_sample.txt","w") as w:
    w.write(write_string)


with open("write_sample.txt", 'r') as r:
    for line in r:
        print(line.strip()) # .strip() : line 의 공백 제거

with open("write_sample.txt") as f:
    contents = f.read()

print(contents)
```

*파이썬 모듈

- 클래스(class) :똑같은 무엇인가를 계속해서 만들어 낼 수 있는 설계, 틀 (함수) .py 파일내 여러개의 클래스 존재
- 모듈(module) : 특정 기능을 .py 파일 단위로 작성한 것
- 패키지(package) : 특정 기능과 관련된 여러 모듈을 묶은 것 (ex. 패키지.모듈 / Bio.Seq)

import 모듈 *#모듈가져오기*
from 모듈 import 변수 (함수, 클래스) *# 모듈의 일부분만 가져오기*
import 모듈 as 이름 *#모듈이름 지정하기*

import Bio # 바이오 파이썬 내 모은 모듈을 가져온다.
from Bio import SeqIO # 바이오파이썬 중 SeqIO 모듈을 불러온다.
from Bio.Seq import Seq # Bio 패키지의 Seq 모듈에서 Seq라는 클래스를 불러온다
from Bio.SeqUtils import GC
from Bio.Data import CodonTable

In [ ]:
```
# 파일 읽고 저장하기
import pandas as pd
df = pd.read_csv('gwas_results', sep="₩t")
df
df[["POS","ID"]]
df.to_csv("test.csv", sep =',')
```

- 바이오파이썬으로 할 수 있는 일

- 유전체 서열정보를 문자열 수준에서 다루기
- 파싱하기
- 웹 정보 가져오기 (NCBI)
- 툴 활용

03. 유전체 서열을 문자열 수준에서 다루기

03-1. 유전체 서열정보

In [ ]:
```
from Bio.Seq import Seq
```

In [ ]:
```
help(Seq) ## 객체 Seq 에 대한 설명
```

In [ ]:
```
# Seq 라는 객체(object)의 . 속성과 기능들
Seq.count()
Seq.lower()
Seq.complement()
Seq.reverse_complement()
len(Seq)
Seq.transcribe()
Seq.translate()
```

In [ ]:
```
tatabox_seq = Seq("tataaaggcAATATGCAGTAG") # tatabox_seq 변수에 서열("tataaaggcAATAT
print(tatabox_seq) # tatabox_seq 의 내용(값)을 출력하라
```

In [ ]:
```
tatabox_seq.count("AT") # AT 의 갯수
```

In [ ]:
```
tatabox_seq.lower() # 소문자 반환
```

In [ ]:
```
tatabox_seq.split("a") # "a" 를 기준으로 나누어라
```

In [ ]:
```
tatabox_seq.complement() # 서열의 상보적 서열
```

In [ ]:
```
tatabox_seq.reverse_complement() # 역 상보적 서열을 만들어 반환
```

In [ ]:
```
len(tatabox_seq)
```

In [ ]:
```
tatabox_seq.transcribe()
```

In [ ]:
```
tatabox_seq.translate()
```

03-2. GC-contents(%) 계산하기

GC-contents(%) = (g 염기 + c 염기)/전체 염기수 * 100

03-2-1. 공식으로 직접

In [ ]:
```
exon_seq = Seq("ATGCAGTAG")
a_count = exon_seq.count("A")
g_count = exon_seq.count("G")
c_count = exon_seq.count("C")
t_count = exon_seq.count("T")
```

In [ ]:
```
print(t_count)
```

In [ ]:
```
gc_contents = (g_count + c_count)/ len(exon_seq)*100
```

In [ ]:
```
print(gc_contents)
```

03-2-2. Bio.SeqUtils 이용하기

In [ ]:
```
from Bio.SeqUtils import GC
```

In [ ]:
```
gc_contents = GC(exon_seq)
print(gc_contents)
```

03-3. 상보적, 역상보적 서열 만들기

In [ ]:
```
## 바이오 파이썬으로 DNA Sequence 의 상보적, 역상보적 서열 만들기
seq = Seq("TATAAAGGCAATATGCAGTAG")
comp_seq = seq.complement()
rev_comp_seq = seq.reverse_complement()
print(comp_seq)
print(rev_comp_seq)
```

03-4. 코돈테이블

In [ ]:
```
from Bio.Data import CodonTable
```

In [ ]:
```
condon_table = CodonTable.unambiguous_dna_by_name["Standard"]
print(condon_table)
```

03-5. DNA, RNA, 아미노산 서열

- DNA 는 mRNA 로 전사되고, 번역되어 단백질이 생성된다.
- 코딩 가닥, 주형가닥에서 전사와 번역이 일어나는 과정

- 코드 가닥 : 5'- ATGCAGTAG-3'
- 주형 가닥 : 3'-TACGTCATC-5'

- 전사 : mRNA : 5'-AUGCAGUAG-3'
- 번역 : 아미노산 : Met-Gln-STOP

In [ ]:
```
dna = Seq("ATGCAGTAG")
mrna = dna.transcribe()
ptn = dna.translate()
```

In [ ]:
```
print("DNA 서열:", dna)
print("mRNA 서열: ",mrna)
print("아미노산 서열: ", ptn)
```

In [ ]:
```
mRNA = Seq("AUGAACUAAGUUUAA")
ptn = mRNA.translate()
print(ptn)
```

In [ ]:
```
# 첫 생성되는 종결코돈이전의 서열까지 결과를 보여주기
mRNA = Seq("AUGAACUAAGUUUAAU")
ptn = mRNA.translate(to_stop = True)
print(ptn)
```

03-6. Open Reading Frame 찾기

In [ ]:
```
# 오픈 리딩 프레임은 mRNA로 전사되어 단백질이 될 가능성이 있는 염기서열들을 의미
# 시작 코돈 ATG 시작으로 3개의 염기서열씩 읽다가 종결코돈 (TAA, TAG, TGA) 중 하나를
print(tatabox_seq)
start_idx = tatabox_seq.find("ATG")
end_idx = tatabox_seq.find("TAG", start_idx)
orf = tatabox_seq[start_idx : end_idx+3]
print(orf)
```

---

04. 파싱하기

- 파싱이란: 데이터에서 원하는 정보를 가져오는 것을 말한다.
- FASTA, FASTQ, GeneBank 등의 데이터로부터 필요한 부분을 가져오는 작업을 말한다.

04-1. fasta
04-2. fastq
04-3. vcf

04-1. fasta 파일 읽기

- fasta 포맷은 텍스트 기반 포맷으로 염기서열 또는 단백질 서열을 나타내기위해 만든 파일 포맷

In [ ]:
```
from Bio import SeqIO
```

In [ ]:
```
fa = SeqIO.parse("sample.fasta","fasta")

for s in fa:
    print(s)
    print(" ")
    print(s.annotations)
```

In [ ]:
```
fa = SeqIO.parse("sample.fasta","fasta")
for seq_record in fa:
    print(seq_record.id)
    print(seq_record.seq)
```

```
    print(len(seq_record))
    print(" ")
```

In [ ]:
```
records = list(SeqIO.parse("sample.fasta","fasta"))
last_record = records[-1]
first_record = records[0]

first_record.id
```

04-2. fastq 파일 읽기

- fasta 파일에 품질정보가 추가된 파일

In [ ]:
```
fq = SeqIO.parse("sample.fastq","fastq")

for s in fq:
    print(type(s))
    print(s)
    print("")
```

In [ ]:
```
# 서열만 출력하기
fq = SeqIO.parse("sample.fastq","fastq")
for s in fq:
    print(s.seq)
```

In [ ]:
```
from Bio import SeqIO
count = 0
for rec in SeqIO.parse("sample.fastq", "fastq"):
    count += 1
print("%i reads" % count)
```

In [ ]:
```
from Bio import SeqIO
# Q20은 99% 즉 1%의 에러만 허용하겠다는 뜻이다
good_reads = (rec for rec in SeqIO.parse("sample.fastq", "fastq")
              if min(rec.letter_annotations["phred_quality"]) >= 20)
SeqIO.write(good_reads, "good_quality.fastq", "fastq")
```

In [ ]:
```
good= SeqIO.parse("good_quality.fastq", "fastq")
for s in good:
    print(type(s))
    print(s)
    print("")
```

04-3. vcf 파일 읽기

- 변이를 표기하기 위해 만든 포맷
- 메타데이터 부분(header) + 내용 부분(data)

- 메타데이터 : 어떠한 툴로 어떠한 분석을 진행하였는가에 따라 메타데이터에 쓰인값은 다양
- ##filteformat=VCFv4.2
- ##FORMAT=<ID=AD,Description="Allelic depths for the ref and alt alleres in the order listed">

- ##FORMAT=<ID=DP,Description="Approximate read depth">
- ##FILTER=<SNP_filter,Description="QD<2.0 || MQ<40.0">

- 내용부분
- CHROM: chromosome
- POS: 위치
- ID: rsID
- FILTER: 메타데이터 filter 조건에 해당하면 pass
- INOF:ananotation 툴을 사용하여 정보를 붙힘
- FORMAT:GT:Genotype, AD:Allteric depths, DP:Approximate read depth
- Sample: sample 당 Format 해당 값

CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	Sample
chr19	45412079	rs7412	C	T	136.03	PASS	IAC=2;AF=0.54	GT:AD:DP	0/1:30, 35:65

In [ ]:
```
with open("sample.vcf", "r") as vcf:
    print(vcf.read())
```

In [ ]:
```
header = ""
data= ""

with open("sample.vcf", "r") as fr:
    for line in fr:
        if line.startswith("#"):
            header += line
        else:
            data += line
```

In [ ]:
```
print(header)
print("")
print(data)
```

In [ ]:
```
import io
import os
import pandas as pd

def read_vcf(path):
    with open(path, 'r') as f:
        lines = [l for l in f if not l.startswith('##')]
    return pd.read_csv(
        io.StringIO(''.join(lines)),
        dtype={'#CHROM': str, 'POS': int, 'ID': str, 'REF': str, 'ALT': str,
               'QUAL': str, 'FILTER': str, 'INFO': str},
        sep='Wt'
    ).rename(columns={'#CHROM': 'CHROM'})



vcf_table = read_vcf("sample.vcf")
```

In [ ]:
```
vcf_table
```

In [ ]:
```
# VCF 에는 몇개의 샘플에 대한 정보가 있을까요?
with open("sample.vcf", "r") as fr:
    for line in fr:
```

```
        if line.startswith("#CHROM"):
            print(len(line.split())-9)
```

In [ ]:
```
# VCF 파일 중 데이터 부분 Filter 열에서 PASS 로만 찍힌 변이 개수 세기
cnt = 0
with open("sample.vcf","r") as fr:
    for line in fr:
        if line.startswith("#"):
            pass
        else:
            l= line.split()
            if l[6] == "PASS":
                print(l)
                cnt += 1

print("PASS 통과한 변이 갯수는 총 ",cnt, "개 입니다. ")
```

In [ ]:
```
## VCF 파일 중 SNP, InDel 개수 세기

SNP=0
Indersion = 0
Deletion = 0

with open("sample.vcf", "r") as fr:
    for line in fr:
        if line.startswith("#"):
            pass
        else:
            l = line.split()
            ref = l[3]
            al = l[4]

            if len(ref) == len(alt):
                SNP += 1
            elif len(ref) > len(alt):
                Deletion += 1
            elif len(ref) < len(alt):
                Insertion += 1
print("SNP :" , SNP)
print("Insersion:", Insertion)
print("Deletion",Deletion)
```

In [ ]:
```
## VCF 파일 중 dbSNP 에서 발견된 변이 개수 구하기

rs = 0
with open("sample.vcf", "r") as fr:
    for line in fr:
        if line.startswith("#"):
            pass
        else:
            l=line.split()
            rsID = l[2]
            if rsID != ".":
                rs += 1
print(rs)
```