**ЛАБОРАТОРНАЯ РАБОТА №2**

**РАБОТА С ФАЙЛАМИ В PYTHON**

**Вариант 1. Вычисление GC-состава**

**Условие задачи:** Биологический аналог задачи определения языка, на котором написан заданный текст, возникает при исследовании молекул ДНК, полученных от особей неизвестного вида. Так как две ***спирали*** ДНК связаны спаренными основаниями, ***цитозин*** и ***гуанин***, будучи ***комплементарными*** ***основаниями***, всегда появляются в одинаковом количестве в любой молекуле ДНК. Поэтому для анализа частоты нуклеиновых ***оснований*** заданной ДНК можно вычислить ***GC-состав*** этой молекулы, т.е. долю оснований, являющихся либо цитозином, либо гуанином.

GC-состав геномов большинства ***эукариот*** располагается в районе 50%. Тем не менее, благодаря большой длине геномов мы можем отличить один вид от другого на основе очень малой разницы между их GC-составами; более того, GC-состав большинства ***прокариот*** значительно превышает 50%, значит, GCсостав можно использовать для быстрого разделения прокариот от эукариот, используя относительно малые образцы ДНК.

GC-состав ***последовательности ДНК*** определяется как доля символов C и G среди всех символов строки. Например, GC-состав последовательности AGCTATAG равен 37.5%. Заметим, что ***обратная комплементарная последовательность*** любой ДНК всегда имеет такой же GC-состав.

Последовательности ДНК должны так или иначе помечаться, например, при хранении в некоторой базе данных. Одним из широко используемых стандартов является ***формат FASTA***. В этом формате последовательности ДНК предшествует дополнительная строка, начинающаяся символом >, после которого следует некоторая основная информация для данной последовательности. Сама последовательность размещается на одной либо нескольких последующих строках. После одной последовательности ДНК может идти другая, при этом новая строка опять начнётся символом >.

**Входные данные**: не более 10 последовательностей ДНК в формате FASTA (*длины не более 1 килобазы каждая*).

**Выходные данные**: идентификатор последовательности с наибольшим GC-составом и значение GC-состава (в процентах) этой последовательности.

**Анализ задачи:**

Задача заключается в определении GC-состава последовательностей ДНК, представленных в формате FASTA. GC-состав определяется как процентное содержание нуклеотидов гуанина (G) и цитозина (C) в последовательности ДНК.

Для того что бы посчитать GC-состав нам потребуется узнать количество символов G и С а также общее количество символов. Ну ничего сложного, особенно если учесть что за нас уже всё сделали создатели библиотеки Biopython. В Biopython -е имеется модуль SeqIO , который предназначен для работы с биологическими последовательностями, в том числе и с форматом FASTA. В библиотеке Biopython есть специальная функция для подсчёта GC.

**Написание кода**

1. Для начала импортируем нашу библиотеку Biopython. Для этого в терминал впишем ‘pip install biopython’, после чего импортируем оттуда модуль SeqIO.

from Bio import SeqIO

1. Далее создадим переменные. Здесь их будет много. **max\_gc\_cont -** в этой переменной будет храниться максимальное значение GC-состава, найденное на данный момент.

Далее нам придётся создать переменную **seq**, которая Извлекает саму последовательность ДНК из объекта **seq\_rec(**в цикле**)** и преобразует её в строку.  **seq\_id**

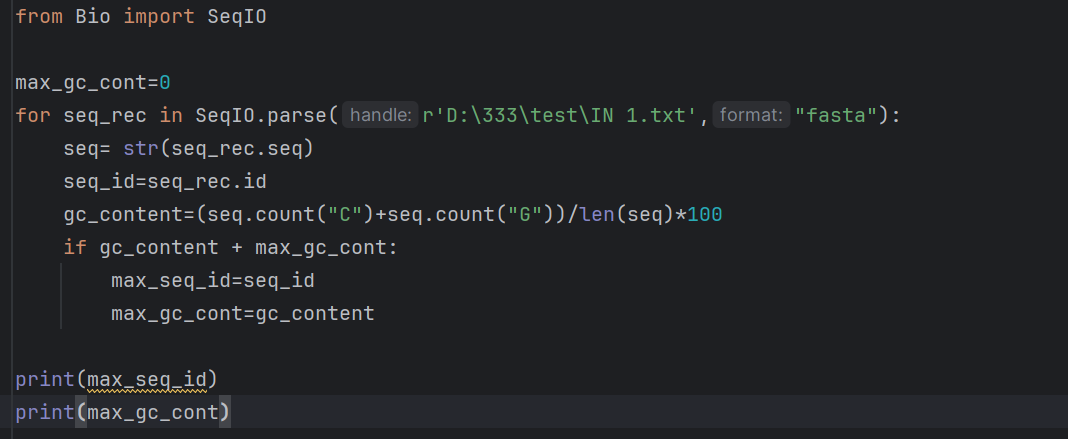
извлекает идентификатор последовательности с помощью метод **.id** из объекта. Это обычно строка, следующая за символом ">" в файле FASTA.

**max\_seq\_id** будет смотреть, е сли текущий GC-состав больше максимального, то обновляет max\_seq\_id идентификатором текущей последовательности.

**gc\_content** вычисляет GC-состав, подсчитывая количество символов "C" и "G" в последовательности, соответственно. Их сумма делится на общую длину последовательности **(len(seq))** и умножается на 100 для получения процентного значения.

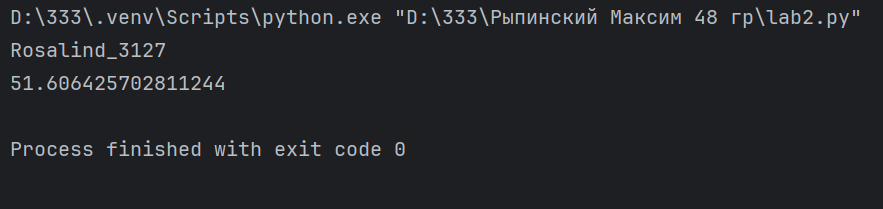
1. С переменными вроде бы как всё. Теперь осталось только импортировать файл в формате fasta с помощью SeqIO и скомпоновать всё в кучу.

Итоговый код выглядит следующим образом:



from Bio import SeqIO  
  
max\_gc\_cont=0  
for seq\_rec in SeqIO.parse(r'D:\333\test\IN 1.txt',"fasta"):  
 seq= str(seq\_rec.seq)  
 seq\_id=seq\_rec.id  
 gc\_content=(seq.count("C")+seq.count("G"))/len(seq)\*100  
 if gc\_content + max\_gc\_cont:  
 max\_seq\_id=seq\_id  
 max\_gc\_cont=gc\_content  
  
print(max\_seq\_id)  
print(max\_gc\_cont)

**Проверка кода**



**Всё готово!**