# BioInformatics Practicals

**Name:-Sanket Shivaji Thorat** **Roll.no.:-33**
**Class:-Msc CS part 1** **Subject:-BioInformatics**

# Practical No: 1

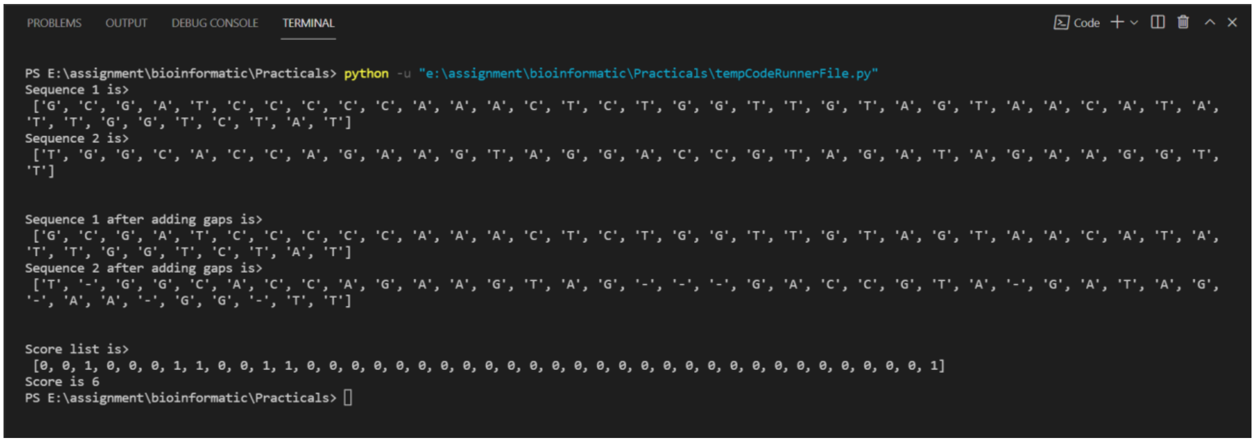
Aim: Write a Python/Java code to perform pairwise alignment. Take 2 sequences from user and calculate the score.

Code:

```
from random import choice, randint
from operator import eq

def get_sequences():
    char_sequence = 'ACTG'
    sequence_1 = [choice(char_sequence) for i in range(randint(10, 50))]
    sequence_2 = [choice(char_sequence) for i in range(randint(10, 50))]

    return sequence_1, sequence_2

def insert_gap(sequence):
    sequence.insert(randint(0, len(sequence) - 1), '-')

    return sequence

def insert_gaps(sequence_1, sequence_2):
    while len(sequence_1) != len(sequence_2):
        if len(sequence_1) < len(sequence_2):
            sequence_1 = insert_gap(sequence_1)
        else:
            sequence_2 = insert_gap(sequence_2)

    return sequence_1, sequence_2

def pairwise_alignment(sequence_1, sequence_2):
    return list(map(eq, sequence_1, sequence_2))

if __name__ == "__main__":
    sequence_1, sequence_2 = get_sequences()
    print("Sequence 1 is>\n", sequence_1)
    print("Sequence 2 is>\n", sequence_2)
    print("\n")
```

```
sequence_1, sequence_2 = insert_gaps(sequence_1, sequence_2)
print("Sequence 1 after adding gaps is>\n", sequence_1)
print("Sequence 2 after adding gaps is>\n", sequence_2)
print("\n")

score_list = pairwise_alignment(sequence_1, sequence_2)
print("Score list is>\n", [1 if i else 0 for i in score_list])
print(f"Score is {sum(score_list)}")
```

Output:



```
PS E:\assignment\bioinformatic\Practicals> python -u "e:\assignment\bioinformatic\Practicals\tempCodeRunnerFile.py"
Sequence 1 is>
 ['G', 'C', 'G', 'A', 'T', 'C', 'C', 'C', 'C', 'C', 'A', 'A', 'A', 'C', 'T', 'C', 'T', 'G', 'G', 'T', 'T', 'G', 'T', 'A', 'G', 'T', 'A', 'A', 'C', 'A', 'T', 'A',
'T', 'T', 'G', 'G', 'T', 'C', 'T', 'A', 'T']
Sequence 2 is>
 ['T', 'G', 'G', 'C', 'A', 'C', 'C', 'A', 'G', 'A', 'A', 'G', 'T', 'A', 'G', 'G', 'A', 'C', 'C', 'G', 'T', 'A', 'G', 'A', 'T', 'A', 'G', 'A', 'A', 'G', 'G', 'T',
'T']


Sequence 1 after adding gaps is>
 ['G', 'C', 'G', 'A', 'T', 'C', 'C', 'C', 'C', 'C', 'A', 'A', 'A', 'C', 'T', 'C', 'T', 'G', 'G', 'T', 'T', 'G', 'T', 'A', 'G', 'T', 'A', 'A', 'C', 'A', 'T', 'A',
'T', 'T', 'G', 'G', 'T', 'C', 'T', 'A', 'T']
Sequence 2 after adding gaps is>
 ['T', '-', 'G', 'G', 'C', 'A', 'C', 'C', 'A', 'G', 'A', 'A', 'G', 'T', 'A', 'G', '-', '-', '-', 'G', 'A', 'C', 'C', 'G', 'T', 'A', '-', 'G', 'A', 'T', 'A', 'G',
'-', 'A', 'A', '-', 'G', 'G', '-', 'T', 'T']


Score list is>
 [0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
Score is 6
PS E:\assignment\bioinformatic\Practicals> []
```

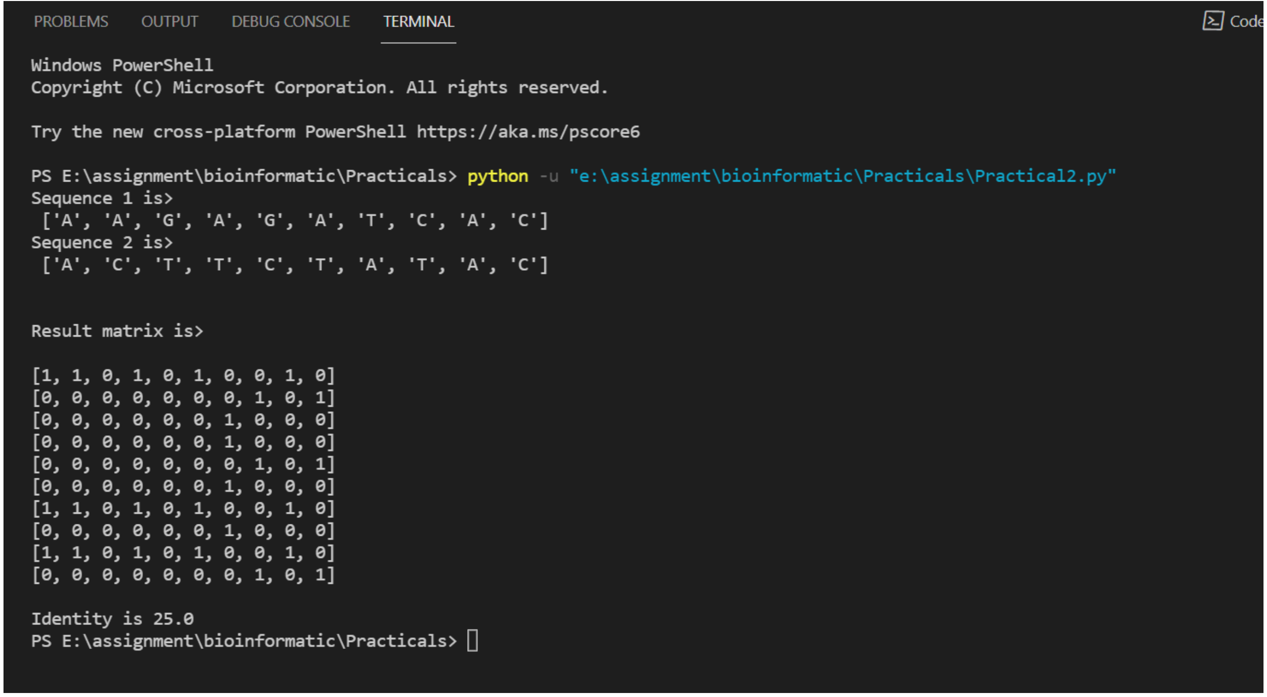
# Practical No:2

Aim: Write a Python/Java code to find the identity value of a given sequences. Take the sequence fromuser.

Code:

```
from random import choice, randint

def get_sequences():
    char_sequence = 'ACTG'
    sequence_length = randint(10, 20)
    sequence_1 = [choice(char_sequence) for i in range(sequence_length)]
    sequence_2 = [choice(char_sequence) for i in range(sequence_length)]

    return sequence_1, sequence_2

def identity(sequence_1, sequence_2):

    result_matrix = [[1 if i == j else 0 for j in sequence_1] for i in sequence_2]
    result = sum([sum(i) for i in result_matrix])

    return result, result_matrix

def print_matrix(matrix):
    for i in matrix:
        print(i)
    print()


if __name__ == "__main__":
    sequence_1, sequence_2 = get_sequences()
    print("Sequence 1 is>\n", sequence_1)
    print("Sequence 2 is>\n", sequence_2)
    print("\n")

    result, result_matrix = identity(sequence_1, sequence_2)
    print("Result matrix is>\n")
    print_matrix(result_matrix)
    print(f"Identity is {round((result / (len(sequence_1) * len(sequence_2))) * 100, 2)}")
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\assignment\bioinformatic\Practicals> python -u "e:\assignment\bioinformatic\Practicals\Practical2.py"
Sequence 1 is>
 ['A', 'A', 'G', 'A', 'G', 'A', 'T', 'C', 'A', 'C']
Sequence 2 is>
 ['A', 'C', 'T', 'T', 'C', 'T', 'A', 'T', 'A', 'C']


Result matrix is>

[1, 1, 0, 1, 0, 1, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
[1, 1, 0, 1, 0, 1, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
[1, 1, 0, 1, 0, 1, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 0, 0, 1, 0, 1]

Identity is 25.0
PS E:\assignment\bioinformatic\Practicals> 
```

# Practical No:3

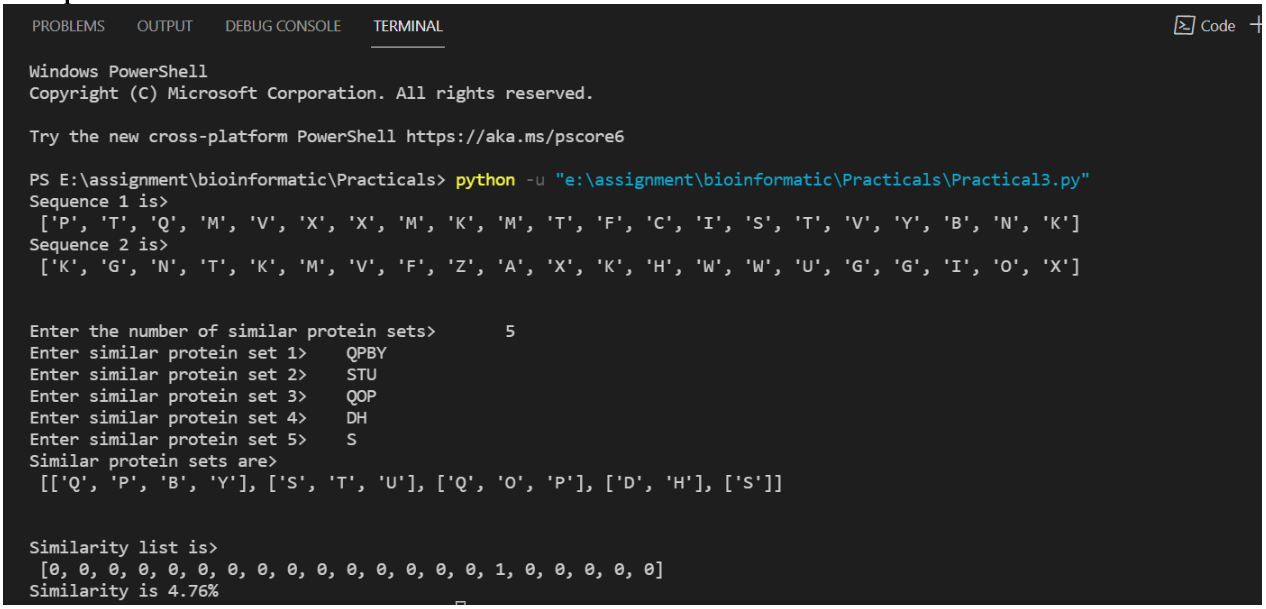
Aim: Write a Python/Java code to find the Similarity value of a given sequences. Take the sequence fromuser.

Code:

```
from random import choice, randint
from string import ascii_uppercase

sequence_list = []

def get_sequences():
        sequence_length = randint(8, 50)
        sequence_1 = [choice(ascii_uppercase) for i in range(sequence_length)]
        sequence_2 = [choice(ascii_uppercase) for i in range(sequence_length)]

        return sequence_1, sequence_2



def get_similar_protein_set():
        sequence_count = int(input("Enter the number of similar protein sets>\t"))

        global sequence_list
        for i in range(sequence_count):
                sequence_list.append(list(input(f"Enter similar protein set {i + 1}>\t")))



def check_similarity(char_1 : str, char_2 : str) -> bool:
        global sequence_list
        for i in sequence_list:
                if (char_1 != char_2):
                        if char_1 in i and char_2 in i:
                                return True

        return False



def similarity(sequence_1, sequence_2):
        similarity_list = [1 if i else 0 for i in list(map(check_similarity, sequence_1,
sequence_2))]
        similarity_value = sum(similarity_list)

        return similarity_value, similarity_list
```

```
if __name__ == "__main__":
    sequence_1, sequence_2 = get_sequences()
    print("Sequence 1 is>\n", sequence_1)
    print("Sequence 2 is>\n", sequence_2)
    print("\n")

    get_similar_protein_set()
    print("Similar protein sets are>\n", sequence_list)
    print("\n")

    similarity_value, similarity_list = similarity(sequence_1, sequence_2)
    print("Similarity list is>\n", similarity_list)
    print(f"Similarity is {round((similarity_value / len(sequence_1)) * 100, 2)}%")
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\assignment\bioinformatic\Practicals> python -u "e:\assignment\bioinformatic\Practicals\Practical3.py"
Sequence 1 is>
 ['P', 'T', 'Q', 'M', 'V', 'X', 'X', 'M', 'K', 'M', 'T', 'F', 'C', 'I', 'S', 'T', 'V', 'Y', 'B', 'N', 'K']
Sequence 2 is>
 ['K', 'G', 'N', 'T', 'K', 'M', 'V', 'F', 'Z', 'A', 'X', 'K', 'H', 'W', 'W', 'U', 'G', 'G', 'I', 'O', 'X']


Enter the number of similar protein sets>       5
Enter similar protein set 1>    QPBY
Enter similar protein set 2>    STU
Enter similar protein set 3>    QOP
Enter similar protein set 4>    DH
Enter similar protein set 5>    S
Similar protein sets are>
 [['Q', 'P', 'B', 'Y'], ['S', 'T', 'U'], ['Q', 'O', 'P'], ['D', 'H'], ['S']]


Similarity list is>
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
Similarity is 4.76%
```

# Practical No:4

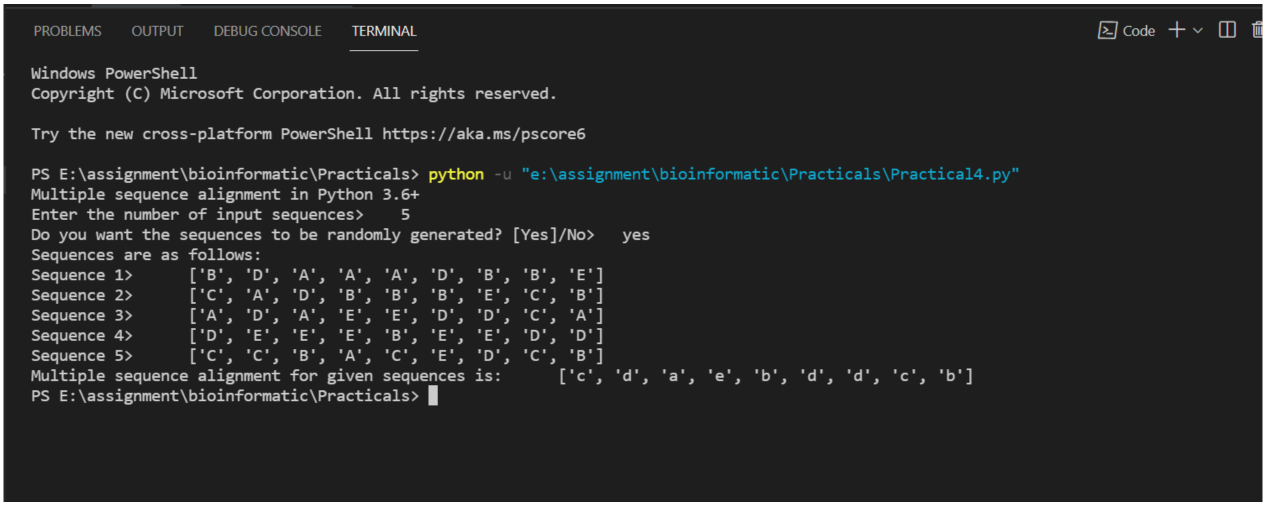
Aim: Enter genome of five different organism and write a python/java program to find consensus sequence using Multiple Sequence Alignment (MSA) technique.

Code:

```
from random import choice, randint


def get_sequences(no_of_sequences : int):
    sequence_list = []
    print("Enter the sequences (All sequences should have equal length)>\t")

    for i in range(no_of_sequences):
        sequence_list.append(list(input(f"Enter sequence {i + 1}>\t")))

    return sequence_list



def get_random_sequences(no_of_sequences : int):
    sequence_length = randint(8, 20)
    sequence_list = [[choice('ABCDE') for j in range(sequence_length)] for i in
range(no_of_sequences)]
    return sequence_list



def multiple_sequence_alignment(sequence_list):
    output_sequence = []

    for i in range(len(sequence_list[0])):
        char_list = list()

        for j in range(len(sequence_list)):
            char_list.append(sequence_list[j][i])

        char_set = set(char_list)
        char_at_pos = ""

        if len(char_set) == 1:
            char_at_pos = char_set[0]

        elif len(sequence_list) % len(char_set) == 0:
            for i in char_set:
                char_at_pos += f"{i}/"
            char_at_pos = char_at_pos[: -1]

        else:
            largest_count = 0
            largest_char = None

            for i in char_set:
                if char_list.count(i) >= largest_count:
                    largest_char = i
                    largest_count = char_list.count(i)

            char_at_pos = largest_char.lower()

        output_sequence.append(char_at_pos)

    return output_sequence


if __name__ == "__main__":
```

```
print("Multiple sequence alignment in Python 3.6+")
no_of_sequences = int(input("Enter the number of input sequences>\t"))

random_flag = False if input("Do you want the sequences to be randomly generated? [Yes]/No>\t").lower() == "no" else True
sequence_list = get_random_sequences(no_of_sequences) if random_flag else get_sequences(no_of_sequences)

print("Sequences are as follows:")
for i in range(len(sequence_list)):
    print(f"Sequence {i + 1}>\t", sequence_list[i])

print("Multiple sequence alignment for given sequences is:\t", multiple_sequence_alignment(sequence_list))
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\assignment\bioinformatic\Practicals> python -u "e:\assignment\bioinformatic\Practicals\Practical4.py"
Multiple sequence alignment in Python 3.6+
Enter the number of input sequences>    5
Do you want the sequences to be randomly generated? [Yes]/No>   yes
Sequences are as follows:
Sequence 1>      ['B', 'D', 'A', 'A', 'A', 'D', 'B', 'B', 'E']
Sequence 2>      ['C', 'A', 'D', 'B', 'B', 'B', 'E', 'C', 'B']
Sequence 3>      ['A', 'D', 'A', 'E', 'E', 'D', 'D', 'C', 'A']
Sequence 4>      ['D', 'E', 'E', 'E', 'B', 'E', 'E', 'D', 'D']
Sequence 5>      ['C', 'C', 'B', 'A', 'C', 'E', 'D', 'C', 'B']
Multiple sequence alignment for given sequences is:      ['c', 'd', 'a', 'e', 'b', 'd', 'd', 'c', 'b']
PS E:\assignment\bioinformatic\Practicals> 
```

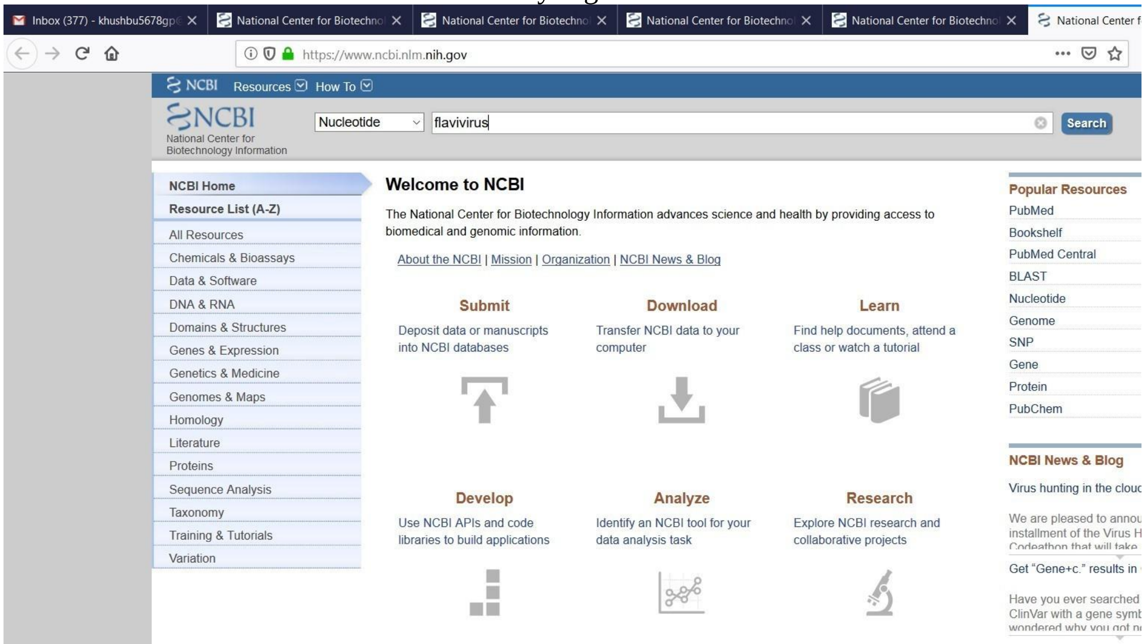
# Practical No:5

Aim: Perform a BLAST search on a specific gene sequence of a specify organism.

Steps:

Go to the National Center for Biotechnology Information Site https://www.ncbi.nlm.nih.gov/

Select Nucleotide from All Databases and find any organism in a search bar.



Welcome to NCBI

The National Center for Biotechnology Information advances science and health by providing access to biomedical and genomic information.

About the NCBI | Mission | Organization | NCBI News & Blog

Submit — Deposit data or manuscripts into NCBI databases

Download — Transfer NCBI data to your computer

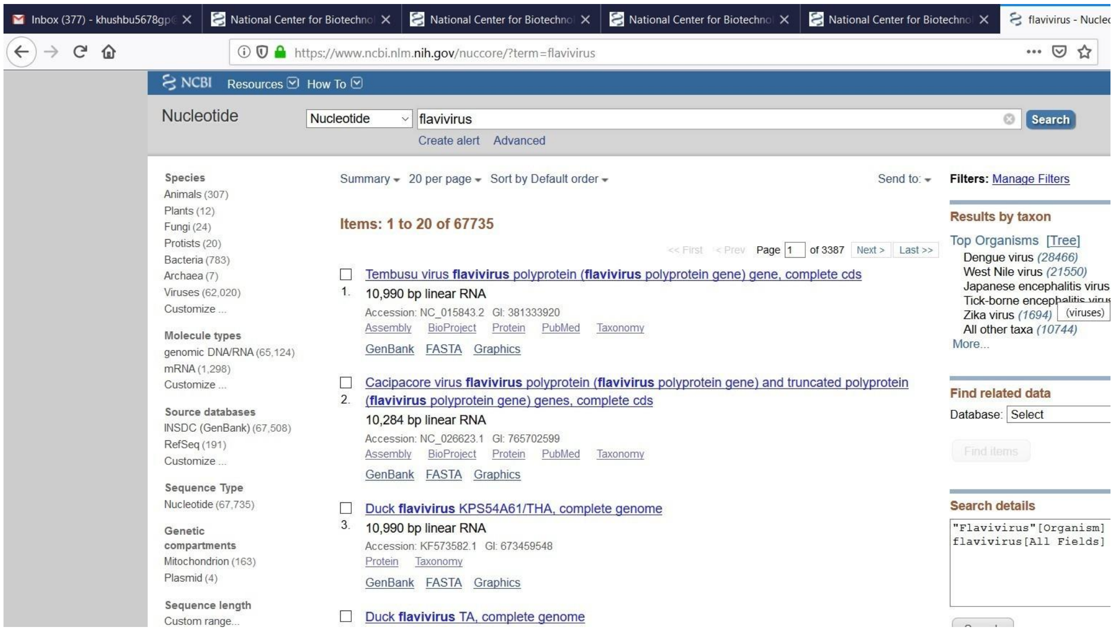
Learn — Find help documents, attend a class or watch a tutorial

Develop — Use NCBI APIs and code libraries to build applications

Analyze — Identify an NCBI tool for your data analysis task

Research — Explore NCBI research and collaborative projects

Popular Resources
PubMed
Bookshelf
PubMed Central
BLAST
Nucleotide
Genome
SNP
Gene
Protein
PubChem
NCBI News & Blog
Virus hunting in the clou
We are pleased to annou installment of the Virus H Codeathon that will take
Get "Gene+c." results in
Have you ever searched ClinVar with a gene symb wondered why you got n



Items: 1 to 20 of 67735

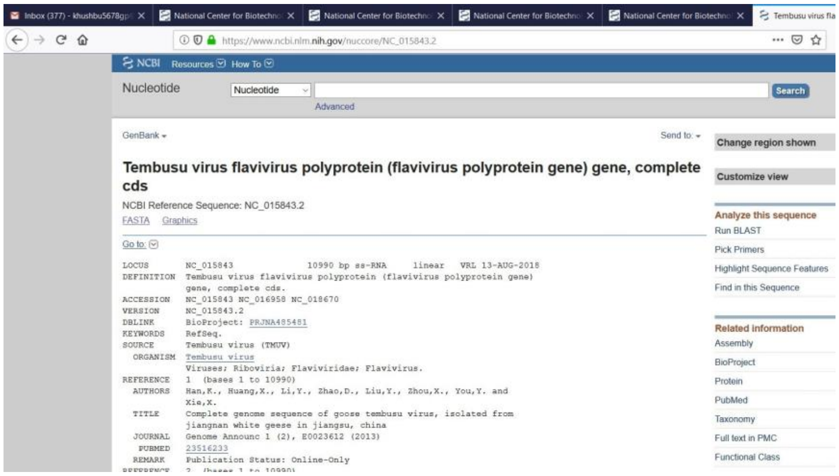
☐ 1. Tembusu virus **flavivirus** polyprotein (**flavivirus** polyprotein gene) gene, complete cds
10,990 bp linear RNA
Accession: NC_015843.2 GI: 381333920
Assembly BioProject Protein PubMed Taxonomy
GenBank FASTA Graphics

☐ 2. Cacipacore virus **flavivirus** polyprotein (**flavivirus** polyprotein gene) and truncated polyprotein (**flavivirus** polyprotein gene) genes, complete cds
10,284 bp linear RNA
Accession: NC_026623.1 GI: 765702599
Assembly BioProject Protein PubMed Taxonomy
GenBank FASTA Graphics

☐ 3. Duck **flavivirus** KPS54A61/THA, complete genome
10,990 bp linear RNA
Accession: KF573582.1 GI: 673459548
Protein Taxonomy
GenBank FASTA Graphics

☐ Duck **flavivirus** TA, complete genome

```
"Flavivirus"[Organism]
flavivirus[All Fields]
```



https://www.ncbi.nlm.nih.gov/nuccore/NC_015843.2

Nucleotide | Nucleotide | Search | Advanced

GenBank | Send to:

**Tembusu virus flavivirus polyprotein (flavivirus polyprotein gene) gene, complete cds**

NCBI Reference Sequence: NC_015843.2

FASTA Graphics

Go to:

```
LOCUS       NC_015843             10990 bp ss-RNA     linear   VRL 13-AUG-2018
DEFINITION  Tembusu virus flavivirus polyprotein (flavivirus polyprotein gene)
            gene, complete cds.
ACCESSION   NC_015843 NC_016958 NC_018670
VERSION     NC_015843.2
DBLINK      BioProject: PRJNA485481
KEYWORDS    RefSeq.
SOURCE      Tembusu virus (TMUV)
  ORGANISM  Tembusu virus
            Viruses; Riboviria; Flaviviridae; Flavivirus.
REFERENCE   1  (bases 1 to 10990)
  AUTHORS   Han,K., Huang,X., Li,Y., Zhao,D., Liu,Y., Zhou,X., You,Y. and
            Xie,X.
  TITLE     Complete genome sequence of goose tembusu virus, isolated from
            jiangnan white geese in jiangsu, china
  JOURNAL   Genome Announc 1 (2), E0023612 (2013)
   PUBMED   23516233
  REMARK    Publication Status: Online-Only
```

Change region shown

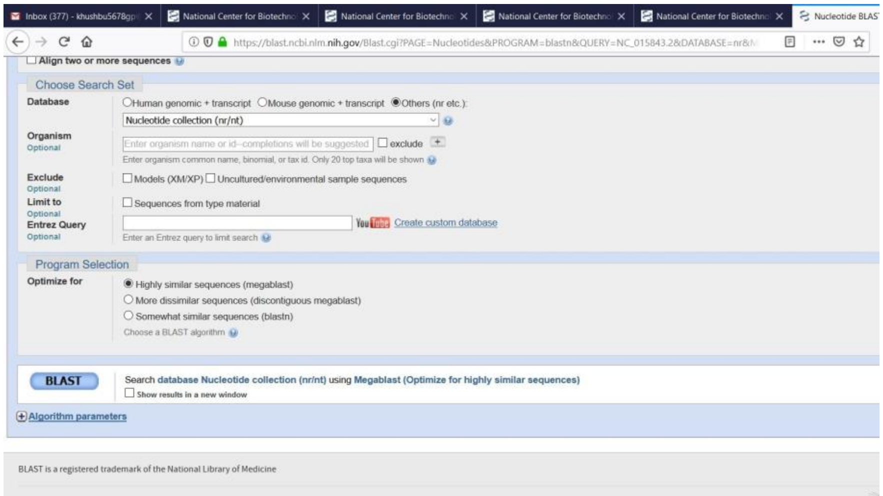
Customize view

Analyze this sequence
- Run BLAST
- Pick Primers
- Highlight Sequence Features
- Find in this Sequence

Related information
- Assembly
- BioProject
- Protein
- PubMed
- Taxonomy
- Full text in PMC
- Functional Class

Inbox (377) - khushbu5678gp
National Center for Biotechno
National Center for Biotechno
National Center for Biotechno
National Center for Biotechno
Tembusu virus fla
NCBI
Resources
How To

Run BLAST option we have to select



https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE=Nucleotides&PROGRAM=blastn&QUERY=NC_015843.2&DATABASE=nr&M

□ Align two or more sequences

Choose Search Set

Database: ○Human genomic + transcript ○Mouse genomic + transcript ●Others (nr etc.):

Nucleotide collection (nr/nt)

Organism (Optional): Enter organism name or id--completions will be suggested □ exclude

Enter organism common name, binomial, or tax id. Only 20 top taxa will be shown

Exclude (Optional): □ Models (XM/XP) □ Uncultured/environmental sample sequences

Limit to (Optional): □ Sequences from type material

Entrez Query (Optional): Create custom database

Enter an Entrez query to limit search

Program Selection

Optimize for:
● Highly similar sequences (megablast)
○ More dissimilar sequences (discontiguous megablast)
○ Somewhat similar sequences (blastn)

Choose a BLAST algorithm

BLAST — Search database Nucleotide collection (nr/nt) using Megablast (Optimize for highly similar sequences)

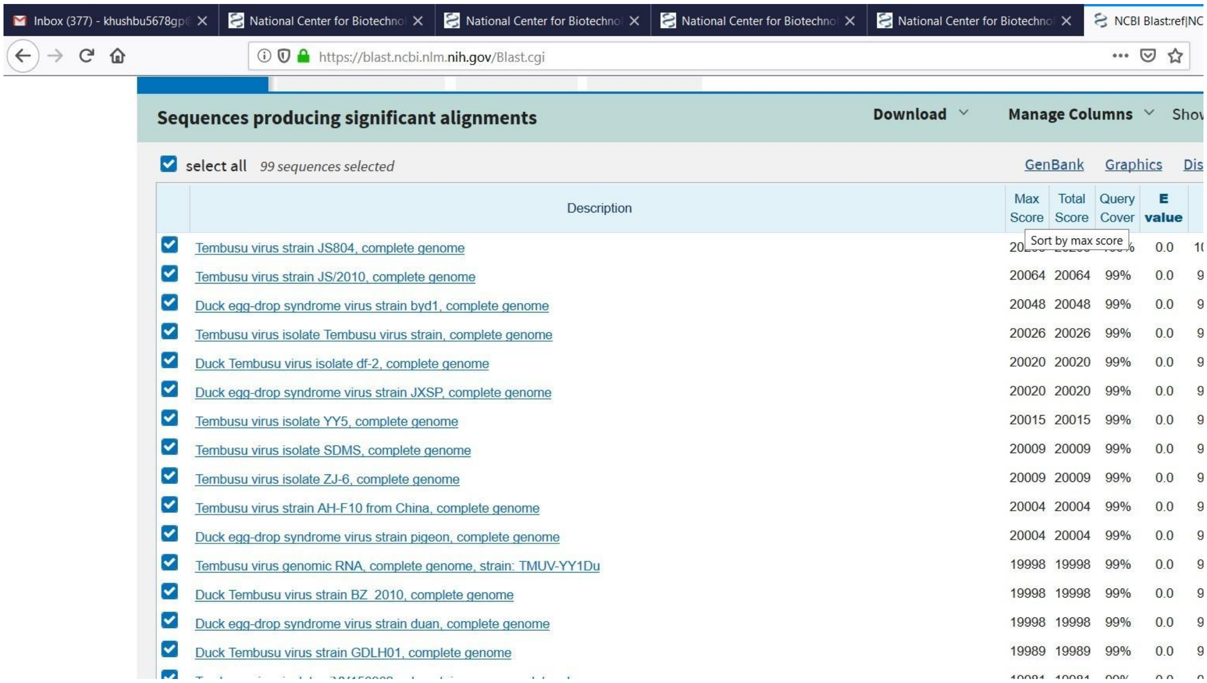
□ Show results in a new window

Algorithm parameters

BLAST is a registered trademark of the National Library of Medicine

Inbox (377) - khushbu5678gp
National Center for Biotechno
National Center for Biotechno
National Center for Biotechno
National Center for Biotechno
Nucleotide BLAS
BLAST

BLAST



https://blast.ncbi.nlm.nih.gov/Blast.cgi

**Sequences producing significant alignments**

Download | Manage Columns | Sho

☑ select all *99 sequences selected*

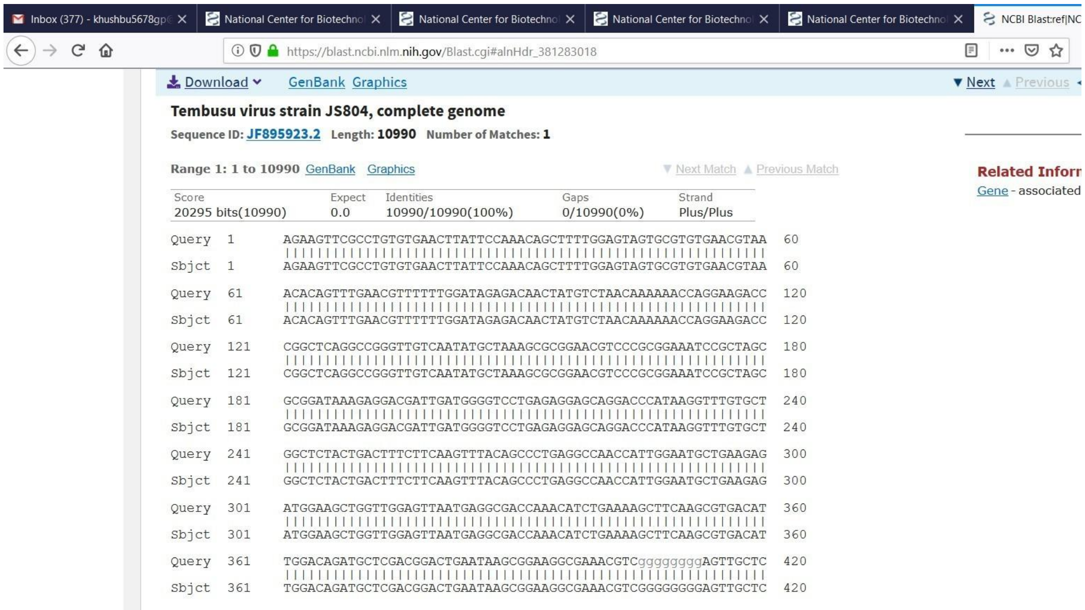
GenBank Graphics Dis

| | Description | Max Score | Total Score | Query Cover | E value | |
|---|---|---|---|---|---|---|
| ☑ | Tembusu virus strain JS804, complete genome | 20[illegible] | [illegible] | [illegible]% | 0.0 | 1( |
| ☑ | Tembusu virus strain JS/2010, complete genome | 20064 | 20064 | 99% | 0.0 | 9 |
| ☑ | Duck egg-drop syndrome virus strain byd1, complete genome | 20048 | 20048 | 99% | 0.0 | 9 |
| ☑ | Tembusu virus isolate Tembusu virus strain, complete genome | 20026 | 20026 | 99% | 0.0 | 9 |
| ☑ | Duck Tembusu virus isolate df-2, complete genome | 20020 | 20020 | 99% | 0.0 | 9 |
| ☑ | Duck egg-drop syndrome virus strain JXSP, complete genome | 20020 | 20020 | 99% | 0.0 | 9 |
| ☑ | Tembusu virus isolate YY5, complete genome | 20015 | 20015 | 99% | 0.0 | 9 |
| ☑ | Tembusu virus isolate SDMS, complete genome | 20009 | 20009 | 99% | 0.0 | 9 |
| ☑ | Tembusu virus isolate ZJ-6, complete genome | 20009 | 20009 | 99% | 0.0 | 9 |
| ☑ | Tembusu virus strain AH-F10 from China, complete genome | 20004 | 20004 | 99% | 0.0 | 9 |
| ☑ | Duck egg-drop syndrome virus strain pigeon, complete genome | 20004 | 20004 | 99% | 0.0 | 9 |
| ☑ | Tembusu virus genomic RNA, complete genome, strain: TMUV-YY1Du | 19998 | 19998 | 99% | 0.0 | 9 |
| ☑ | Duck Tembusu virus strain BZ_2010, complete genome | 19998 | 19998 | 99% | 0.0 | 9 |
| ☑ | Duck egg-drop syndrome virus strain duan, complete genome | 19998 | 19998 | 99% | 0.0 | 9 |
| ☑ | Duck Tembusu virus strain GDLH01, complete genome | 19989 | 19989 | 99% | 0.0 | 9 |

Sort by max score

Inbox (377) - khushbu5678gp
National Center for Biotechno
National Center for Biotechno
National Center for Biotechno
National Center for Biotechno
NCBI Blast:ref|NC

Here the result will be display


Inbox (377) - khushbu5678gp
National Center for Biotechno
National Center for Biotechno
National Center for Biotechno
National Center for Biotechno
NCBI Blast:ref|NC
https://blast.ncbi.nlm.nih.gov/Blast.cgi#alnHdr_381283018
Download
GenBank
Graphics
▼ Next
▲ Previous

Tembusu virus strain JS804, complete genome

Sequence ID: JF895923.2 Length: 10990 Number of Matches: 1

Range 1: 1 to 10990 GenBank Graphics ▼ Next Match ▲ Previous Match

Related Inforr

Gene - associated

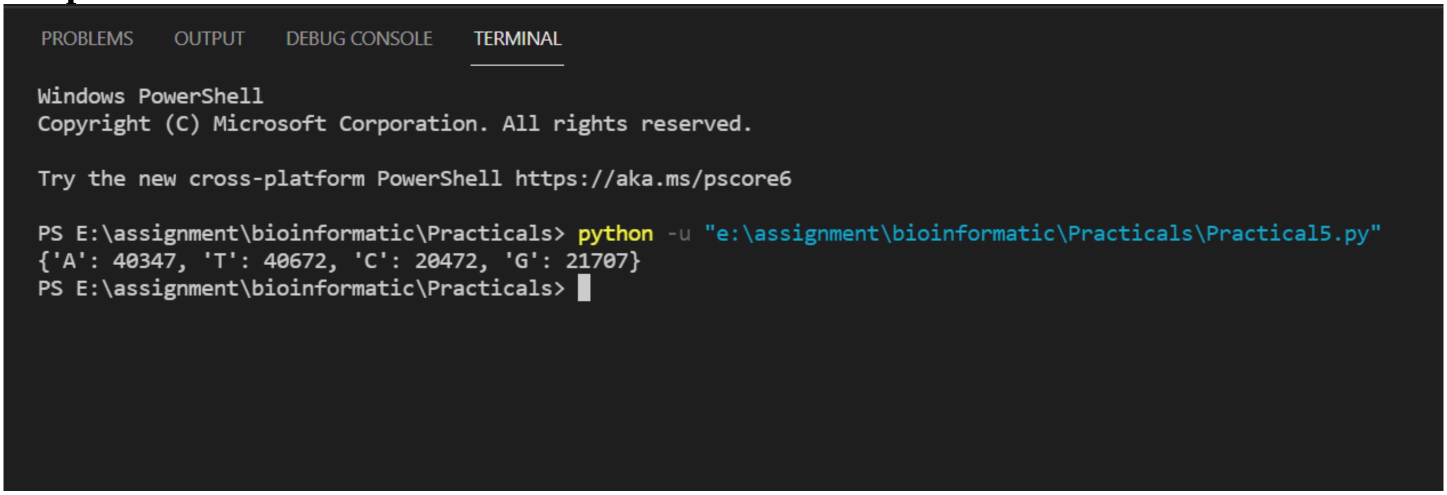
| Score | Expect | Identities | Gaps | Strand |
|---|---|---|---|---|
| 20295 bits(10990) | 0.0 | 10990/10990(100%) | 0/10990(0%) | Plus/Plus |

```
Query  1      AGAAGTTCGCCTGTGTGAACTTATTCCAAACAGCTTTTGGAGTAGTGCGTGTGAACGTAA  60
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  1      AGAAGTTCGCCTGTGTGAACTTATTCCAAACAGCTTTTGGAGTAGTGCGTGTGAACGTAA  60

Query  61     ACACAGTTTGAACGTTTTTTGGATAGAGACAACTATGTCTAACAAAAAACCAGGAAGACC  120
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  61     ACACAGTTTGAACGTTTTTTGGATAGAGACAACTATGTCTAACAAAAAACCAGGAAGACC  120

Query  121    CGGCTCAGGCCGGGTTGTCAATATGCTAAAGCGCGGAACGTCCCGCGGAAATCCGCTAGC  180
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  121    CGGCTCAGGCCGGGTTGTCAATATGCTAAAGCGCGGAACGTCCCGCGGAAATCCGCTAGC  180

Query  181    GCGGATAAAGAGGACGATTGATGGGGTCCTGAGAGGAGCAGGACCCATAAGGTTTGTGCT  240
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  181    GCGGATAAAGAGGACGATTGATGGGGTCCTGAGAGGAGCAGGACCCATAAGGTTTGTGCT  240

Query  241    GGCTCTACTGACTTTCTTCAAGTTTACAGCCCTGAGGCCAACCATTGGAATGCTGAAGAG  300
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  241    GGCTCTACTGACTTTCTTCAAGTTTACAGCCCTGAGGCCAACCATTGGAATGCTGAAGAG  300

Query  301    ATGGAAGCTGGTTGGAGTTAATGAGGCGACCAAACATCTGAAAAGCTTCAAGCGTGACAT  360
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  301    ATGGAAGCTGGTTGGAGTTAATGAGGCGACCAAACATCTGAAAAGCTTCAAGCGTGACAT  360

Query  361    TGGACAGATGCTCGACGGACTGAATAAGCGGAAGGCGAAACGTCggggggggAGTTGCTC  420
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  361    TGGACAGATGCTCGACGGACTGAATAAGCGGAAGGCGAAACGTCGGGGGGGGAGTTGCTC  420
```

## Code:-

```
fasta=open('seqdump.txt','r')
seq=fasta.read()
data={'A':seq.count('A'),'T':seq.count('T'),'C':seq.count('C'),'G':seq.count('G')}
print(data)
```

## Output:-



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\assignment\bioinformatic\Practicals> python -u "e:\assignment\bioinformatic\Practicals\Practical5.py"
{'A': 40347, 'T': 40672, 'C': 20472, 'G': 21707}
PS E:\assignment\bioinformatic\Practicals> 
```

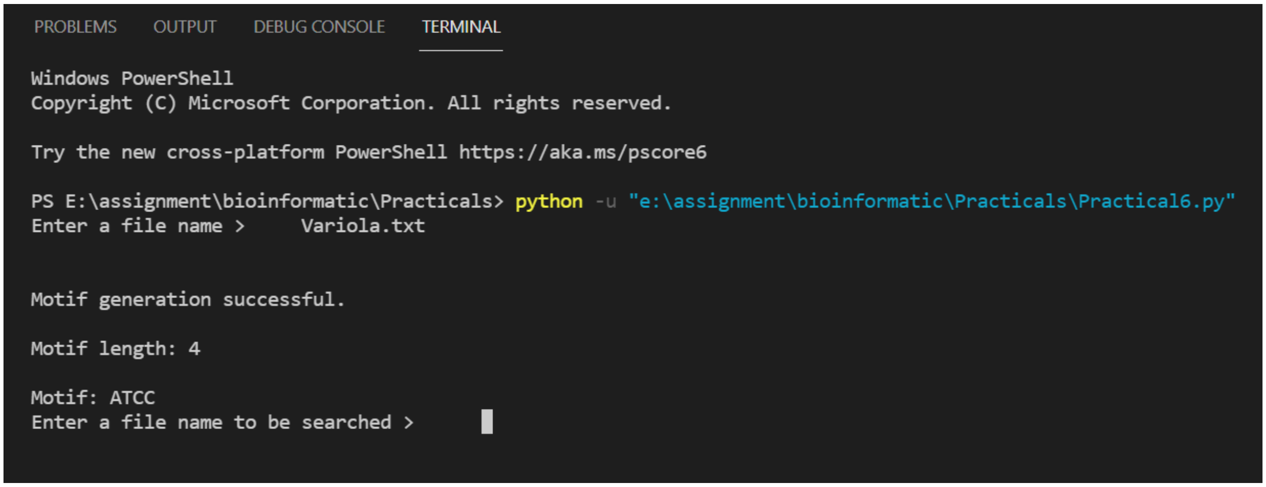
# Practical No:6

Aim: Write a Python/Java code to find motif in a given sequence. Code:

```
from random import randint

def motif(input_file_name : str) -> str:

    with open(input_file_name) as input_file_handle:

        input_file_data = input_file_handle.read()

        input_file_handle.close()



    input_file_data = input_file_data.replace("\n", "")



    #motif_length = randint(2, len(input_file_data) - 1)

    motif_length = randint(2, 10)

    start_index = randint(0, len(input_file_data) - motif_length)



    return input_file_data[start_index : start_index + motif_length]
```

```
def search_for_motif(motif : str, search_file_name : str) -> int:
    with open(search_file_name) as search_file_handle:
        search_file_data = search_file_handle.read()
        search_file_handle.close()

    search_file_data = search_file_data.replace("\n", "")

    index = search_file_data.find(motif)

    return index

if __name__ == "__main__":
    input_file_name = input("Enter a file name >\t")
    generated_motif = motif(input_file_name)
    print("\n\nMotif generation successful.")
    print(f"\nMotif length: {len(generated_motif)}")
    print(f"\nMotif: {generated_motif}")

    search_file_name = input("Enter a file name to be searched >\t")
    index = search_for_motif(generated_motif, search_file_name)



    if index > 0:

        print("Given motif is found at index:\t", index)

    else:

        print("Motif not found in document")
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\assignment\bioinformatic\Practicals> python -u "e:\assignment\bioinformatic\Practicals\Practical6.py"
Enter a file name >     Variola.txt


Motif generation successful.

Motif length: 4

Motif: ATCC
Enter a file name to be searched >
```

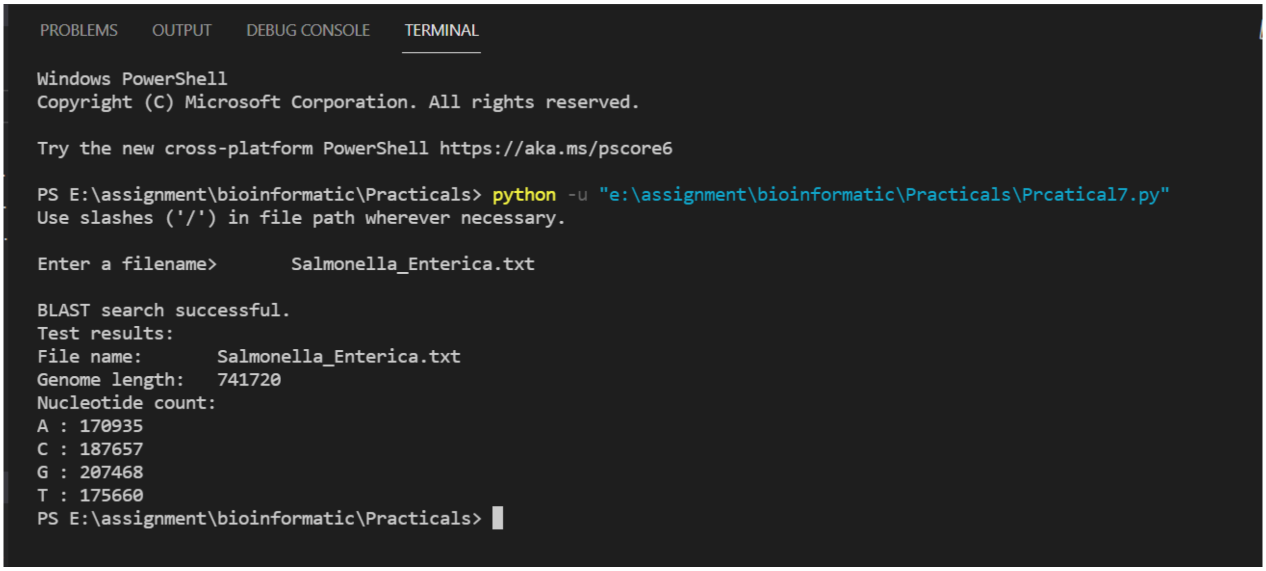
# Practical No:7

Aim: Perform a BLAST search on any genes sequence and writer a java/python code to count the no of repetition of each nucleotide in the sequence.

Code:

```
if __name__ == "__main__":

    print("Use slashes ('/') in file path wherever
necessary.\n")

    file_handle = open(input("Enter a
filename>\t"))

    file_data = file_handle.read()

    file_handle.close()



    base_dict = {}



    for i in "ACGT":

        base_dict[i] = file_data.count(i)



    print("\nBLAST search successful.\nTest
results:")

    print(f"File name:\t
{file_handle.name.split('/')[- 1]}")

    print(f"Genome length:\t

{sum(base_dict.values())}")

    print(f"Nucleotide count:")



    for i in "ACGT":

        print(f"{i} : {base_dict[i]}")
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\assignment\bioinformatic\Practicals> python -u "e:\assignment\bioinformatic\Practicals\Prcatical7.py"
Use slashes ('/') in file path wherever necessary.

Enter a filename>       Salmonella_Enterica.txt

BLAST search successful.
Test results:
File name:       Salmonella_Enterica.txt
Genome length:   741720
Nucleotide count:
A : 170935
C : 187657
G : 207468
T : 175660
PS E:\assignment\bioinformatic\Practicals> 
```

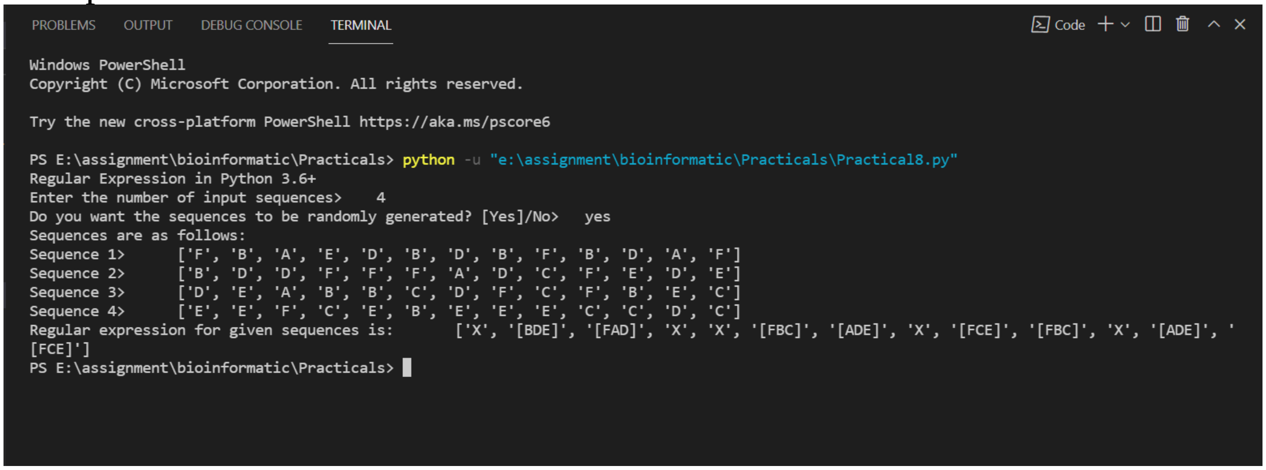
# Practical No:8

Aim: Generate a regular expression enter three protein sequence of three different organism. Write Python/Java code to find regular expression for this sequences.

Code:

```
from random import choice, randint
from string import ascii_uppercase

def get_sequences(no_of_sequences : int):
    sequence_list = []
    print("Enter the sequences (All sequences should have equal length)>\t")

    for i in range(no_of_sequences):
        sequence_list.append(list(input(f"Enter sequence {i + 1}>\t")))

    return sequence_list


def get_random_sequences(no_of_sequences : int):
    sequence_length = randint(8, 20)
    sequence_list = [[choice(ascii_uppercase[: 6]) for j in range(sequence_length)] for i in
range(no_of_sequences)]
    return sequence_list


def get_regular_expression(sequence_list):
    output_sequence = []

    for i in range(len(sequence_list[0])):
        char_column = set()

        for j in range(len(sequence_list)):
            if sequence_list[j][i] != '-':
                char_column.add(sequence_list[j][i])

        char_at_i = ""
        if len(char_column) == 1:
            char_at_i = char_column[0]

        else:
            if len(char_column) == len(sequence_list):
                char_at_i = 'X'
            else:
                char_at_i += "["
                for i in char_column:
                    char_at_i += i
                char_at_i += "]"

        output_sequence.append(char_at_i)

    return output_sequence

if __name__ == "__main__":
    print("Regular Expression in Python 3.6+")
    no_of_sequences = int(input("Enter the number of input sequences>\t"))

    random_flag = False if input("Do you want the sequences to be randomly generated?
[Yes]/No>\t").lower() == "no" else True
    sequence_list = get_random_sequences(no_of_sequences) if random_flag else
get_sequences(no_of_sequences)

    print("Sequences are as follows:")
    for i in range(len(sequence_list)):
        print(f"Sequence {i + 1}>\t", sequence_list[i])

    print("Regular expression for given sequences is:\t", get_regular_expression(sequence_list))
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\assignment\bioinformatic\Practicals> python -u "e:\assignment\bioinformatic\Practicals\Practical8.py"
Regular Expression in Python 3.6+
Enter the number of input sequences>    4
Do you want the sequences to be randomly generated? [Yes]/No>   yes
Sequences are as follows:
Sequence 1>      ['F', 'B', 'A', 'E', 'D', 'B', 'D', 'B', 'F', 'B', 'D', 'A', 'F']
Sequence 2>      ['B', 'D', 'D', 'F', 'F', 'F', 'A', 'D', 'C', 'F', 'E', 'D', 'E']
Sequence 3>      ['D', 'E', 'A', 'B', 'B', 'C', 'D', 'F', 'C', 'F', 'B', 'E', 'C']
Sequence 4>      ['E', 'E', 'F', 'C', 'E', 'B', 'E', 'E', 'E', 'C', 'C', 'D', 'C']
Regular expression for given sequences is:       ['X', '[BDE]', '[FAD]', 'X', 'X', '[FBC]', '[ADE]', 'X', '[FCE]', '[FBC]', 'X', '[ADE]', '
[FCE]']
PS E:\assignment\bioinformatic\Practicals> 
```

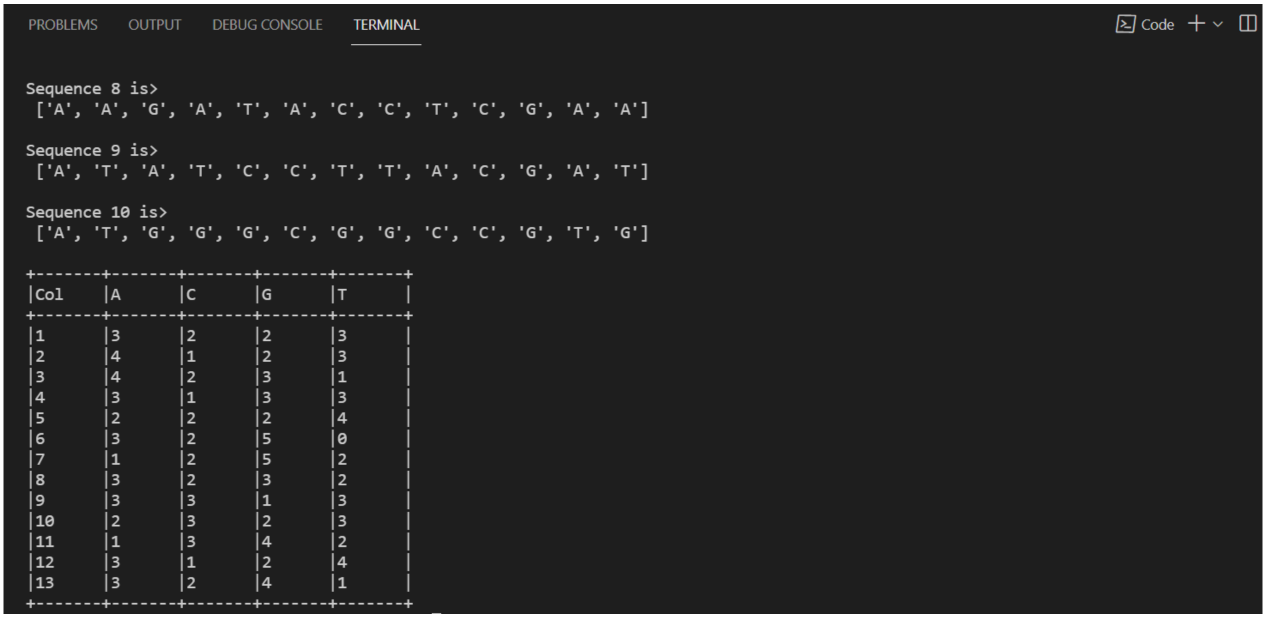
# Practical No:9

Aim: Enter six protein sequence of different organism and write a program to find a fingerprint of sequence.

Code:

```
from random import choice, randint
import sys

def generate_sequences():
    sequence_length = randint(8, 20)
    sequence_count = randint(8, 20)

    return [[choice("ACTG") for i in range(sequence_length)] for j in
range(sequence_count)]


def calculate_fingerprint(sequence_list):
    fingerprint_list = []

    for j in range(len(sequence_list[0])):
        finger_print_dict = dict()

        for i in range(len(sequence_list)):
            finger_print_dict[sequence_list[i][j]] =
finger_print_dict.get(sequence_list[i][j], 0) + 1

        for i in 'ACTG':
            if i not in finger_print_dict:
                finger_print_dict[i] = 0
            fingerprint_list.append(finger_print_dict)

    return fingerprint_list


def print_result(fingerprint_list):

    print("+-------+-------+-------+-------+-------+")
    print("|Col\t|A\t|C\t|G\t|T\t|")
    print("+-------+-------+-------+-------+-------+")

    for i in range(len(fingerprint_list)):
        print(f"|{i + 1}\t|{fingerprint_list[i]['A']}\t|{fingerprint_list[i]['C']}\t|{fingerprint_list[i]['G']}\t|{fingerprint_list[i]['T']}\t|")

    print("+-------+-------+-------+-------+-------+")


if __name__ == "__main__":
    sequence_list = generate_sequences()

    for i in range(len(sequence_list)):
        print(f"Sequence {i + 1} is>\n", sequence_list[i], "\n")

    fingerprint_list = calculate_fingerprint(sequence_list)
    print_result(fingerprint_list)
```

Output:



```
Sequence 8 is>
 ['A', 'A', 'G', 'A', 'T', 'A', 'C', 'C', 'T', 'C', 'G', 'A', 'A']

Sequence 9 is>
 ['A', 'T', 'A', 'T', 'C', 'C', 'T', 'T', 'A', 'C', 'G', 'A', 'T']

Sequence 10 is>
 ['A', 'T', 'G', 'G', 'G', 'C', 'G', 'G', 'C', 'C', 'G', 'T', 'G']

+-------+-------+-------+-------+-------+
|Col    |A      |C      |G      |T      |
+-------+-------+-------+-------+-------+
|1      |3      |2      |2      |3      |
|2      |4      |1      |2      |3      |
|3      |4      |2      |3      |1      |
|4      |3      |1      |3      |3      |
|5      |2      |2      |2      |4      |
|6      |3      |2      |5      |0      |
|7      |1      |2      |5      |2      |
|8      |3      |2      |3      |2      |
|9      |3      |3      |1      |3      |
|10     |2      |3      |2      |3      |
|11     |1      |3      |4      |2      |
|12     |3      |1      |2      |4      |
|13     |3      |2      |4      |1      |
+-------+-------+-------+-------+-------+
```