

# BioInformatics Practicals

Name:-Sanket Shivaji Thorat  
Class:-Msc CS part 1

Roll.no.:-33  
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:

## Practical No:2

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

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:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  Code

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, 0, 1, 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 from user.

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:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
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> QQP
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:")

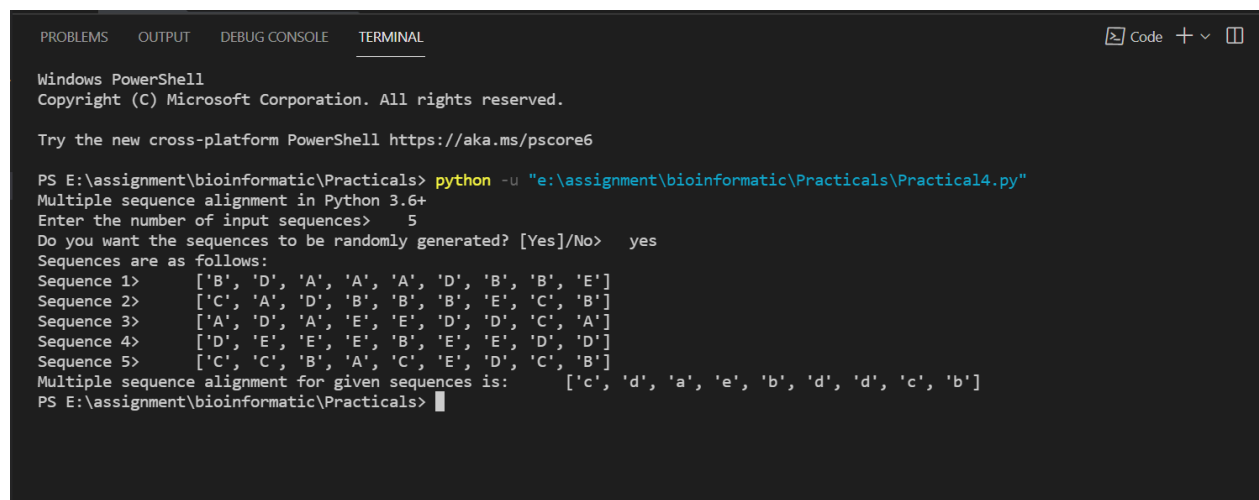
for i in range(len(sequence_list)):

    print(f"Sequence {i + 1}>\t", sequence_list[i])

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

```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
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>

```

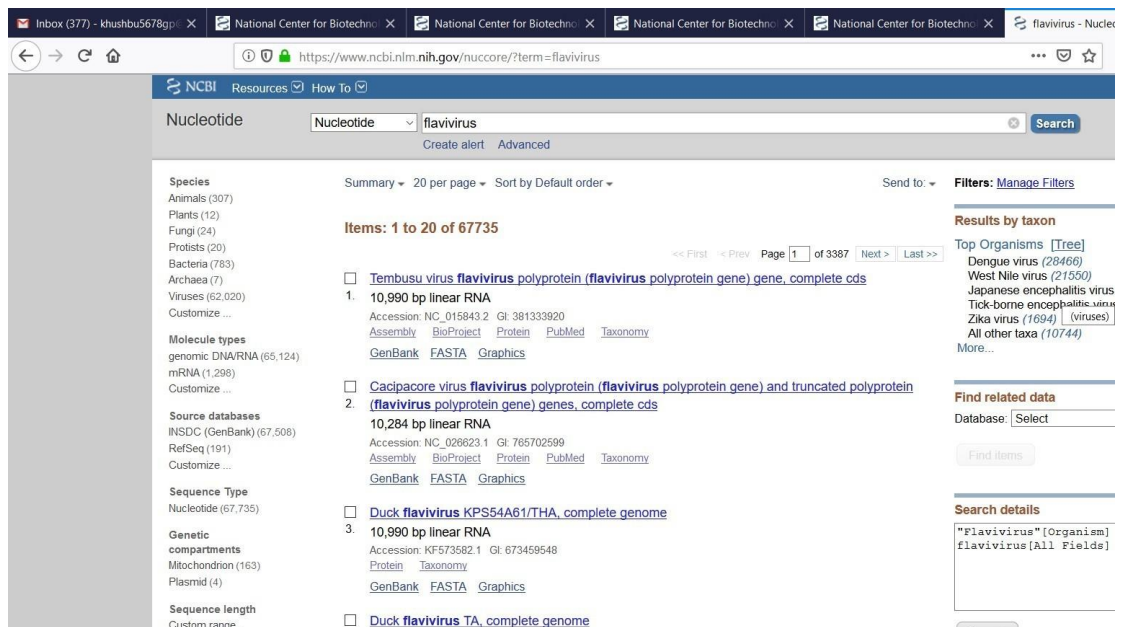
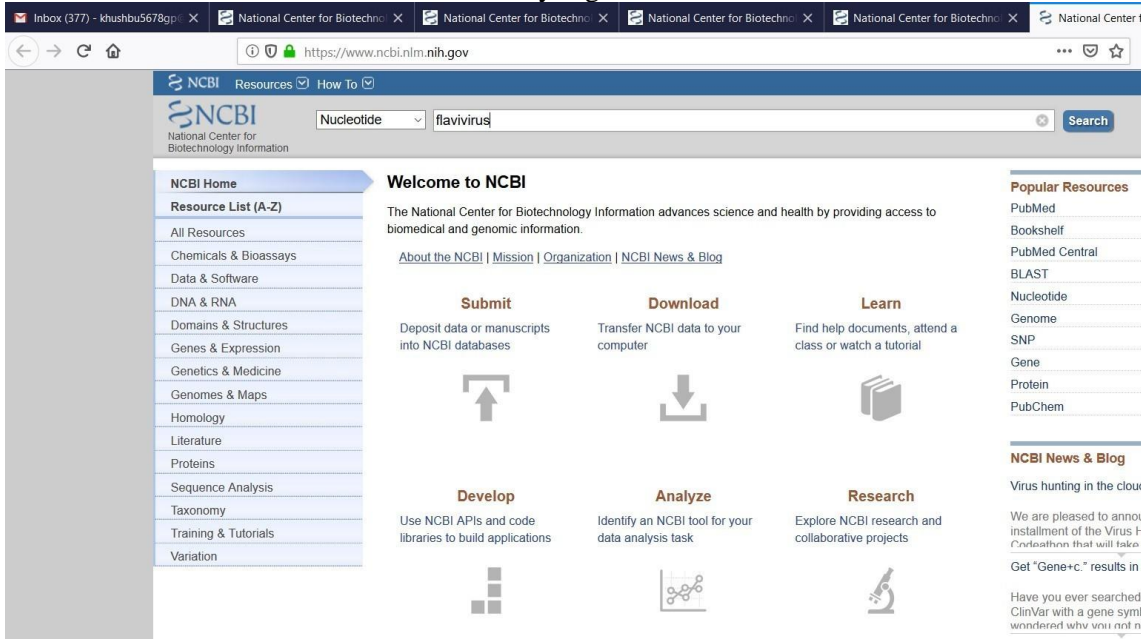
# Practical No:5

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



NCBI Resources How To

Nucleotide

Advanced

GenBank

**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

**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

2 (bases 1 to 10990)

Run BLAST option we have to select

Align two or more sequences

Choose Search Set

Database ☐ Human genomic + transcript ☐ Mouse genomic + transcript ☒ Others (nr etc.):

Nucleotide collection (nr/nt)

Organism

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

☐ Models (XM/XP) ☐ Uncultured/environmental sample sequences

☐ Sequences from type material

Limit to   [Create custom database](#)

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

BLAST

Sequences producing significant alignments					Download	Manage Columns	Show
<input checked="" type="checkbox"/> select all 99 sequences selected					GenBank	Graphics	Dis
	Description	Max Score	Total Score	Query Cover	E value		
<input checked="" type="checkbox"/>	<a href="#">Tembusu virus strain JS804, complete genome</a>	20064	20064	99%	0.0	11	
<input checked="" type="checkbox"/>	<a href="#">Tembusu virus strain JS/2010, complete genome</a>	20064	20064	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Duck egg-drop syndrome virus strain byd1, complete genome</a>	20048	20048	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Tembusu virus isolate Tembusu virus strain, complete genome</a>	20026	20026	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Duck Tembusu virus isolate df-2, complete genome</a>	20020	20020	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Duck egg-drop syndrome virus strain JXSP, complete genome</a>	20020	20020	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Tembusu virus isolate YY5, complete genome</a>	20015	20015	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Tembusu virus isolate SDMS, complete genome</a>	20009	20009	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Tembusu virus isolate ZJ-6, complete genome</a>	20009	20009	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Tembusu virus strain AH-F10 from China, complete genome</a>	20004	20004	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Duck egg-drop syndrome virus strain pigeon, complete genome</a>	20004	20004	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Tembusu virus genomic RNA, complete genome, strain: TMUV-YY1Du</a>	19998	19998	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Duck Tembusu virus strain BZ, 2010, complete genome</a>	19998	19998	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Duck egg-drop syndrome virus strain duan, complete genome</a>	19998	19998	99%	0.0	9	
<input checked="" type="checkbox"/>	<a href="#">Duck Tembusu virus strain GDLH01, complete genome</a>	19989	19989	99%	0.0	9	

Here the result will be display

Inbox (377) - khusbu5678g

2025/10/20 - National Center for Biotechnology Information

2025/10/20 - National Center for Biotechnology Information

2025/10/20 - National Center for Biotechnology Information

2025/10/20 - National Center for Biotechnology Information

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 Information

[Gene](#) - associated

Score	Expect	Identities	Gaps	Strand			
20295 bits(10990)	0.0	10990/10990(100%)	0/10990(0%)	Plus/Plus			
Query 1	AGAA	AGTT	CGCCT	GTGT	GAAC	GTAA	60
Sbjct 1	AGAA	AGTT	CGCCT	GTGT	GAAC	GTAA	60
Query 61	ACAC	AGTTT	GAACGTTTTTTT	GGATAGAGACA	CACTATGTCTA	ACAAAAAAC	120
Sbjct 61	ACAC	AGTTT	GAACGTTTTTTT	GGATAGAGACA	CACTATGTCTA	ACAAAAAAC	120
Query 121	CGGCT	CAGGCCGGGTTGT	CAATATGCTAAAGCGCGGAACGTC	CCGCGGAAATCCGCTAGC			180
Sbjct 121	CGGCT	CAGGCCGGGTTGT	CAATATGCTAAAGCGCGGAACGTC	CCGCGGAAATCCGCTAGC			180
Query 181	GCGGATAA	AGGACGATTGATGGGGTCCT	GAGAGGAGCAGGACCCATAAGGTTTGTGCT				240
Sbjct 181	GCGGATAA	AGGACGATTGATGGGGTCCT	GAGAGGAGCAGGACCCATAAGGTTTGTGCT				240
Query 241	GGCTCTACTGACTTTCTT	CAAGTTTACAGCCCTGAGGCCAACCAATTGGAATGCTGAAGAG					300
Sbjct 241	GGCTCTACTGACTTTCTT	CAAGTTTACAGCCCTGAGGCCAACCAATTGGAATGCTGAAGAG					300
Query 301	ATGGAAGCTGGTTGGAGTTAAT	GAGCGACCAAAACATCTGAAAAGCTTCAAGCGTGACAT					360
Sbjct 301	ATGGAAGCTGGTTGGAGTTAAT	GAGCGACCAAAACATCTGAAAAGCTTCAAGCGTGACAT					360
Query 361	TGGACAGATGCTCGACGGACTGAATAAGCGGAAGCGGAAACGTC	ggggggggAGTTTGCTC					420
Sbjct 361	TGGACAGATGCTCGACGGACTGAATAAGCGGAAGCGGAAACGTC	ggggggggAGTTTGCTC					420

Related Inform  
[Gene](#) - associated

## 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:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

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:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
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\Practical7.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"))

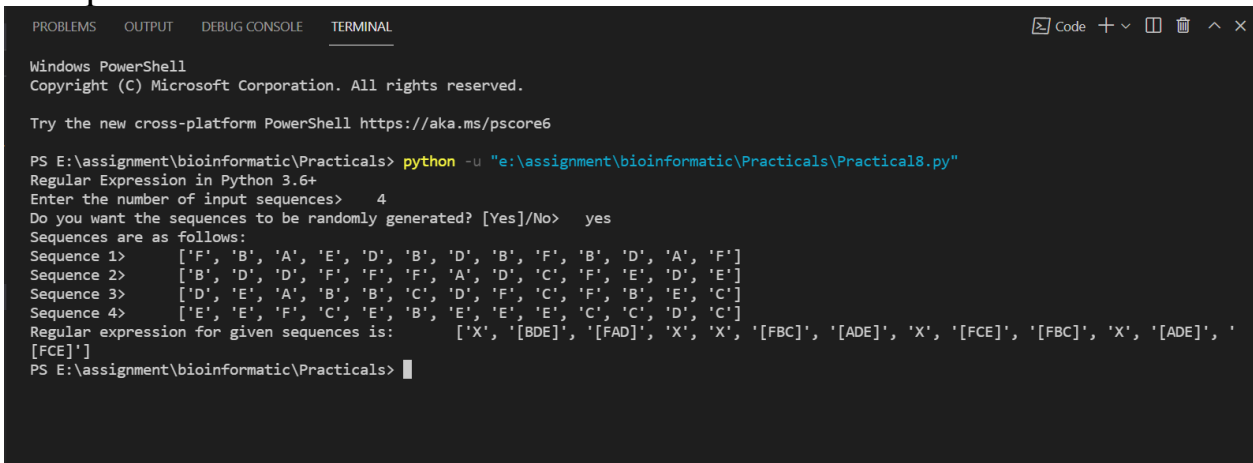
    random_flag = False if input("Do you want the sequences to be randomly generated?
[Yes]/No>\t").lower() == "no" else True
    sequence_list = get_random_sequences(no_of_sequences) if random_flag else
get_sequences(no_of_sequences)

    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:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
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:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
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|
+-----+-----+-----+-----+

```