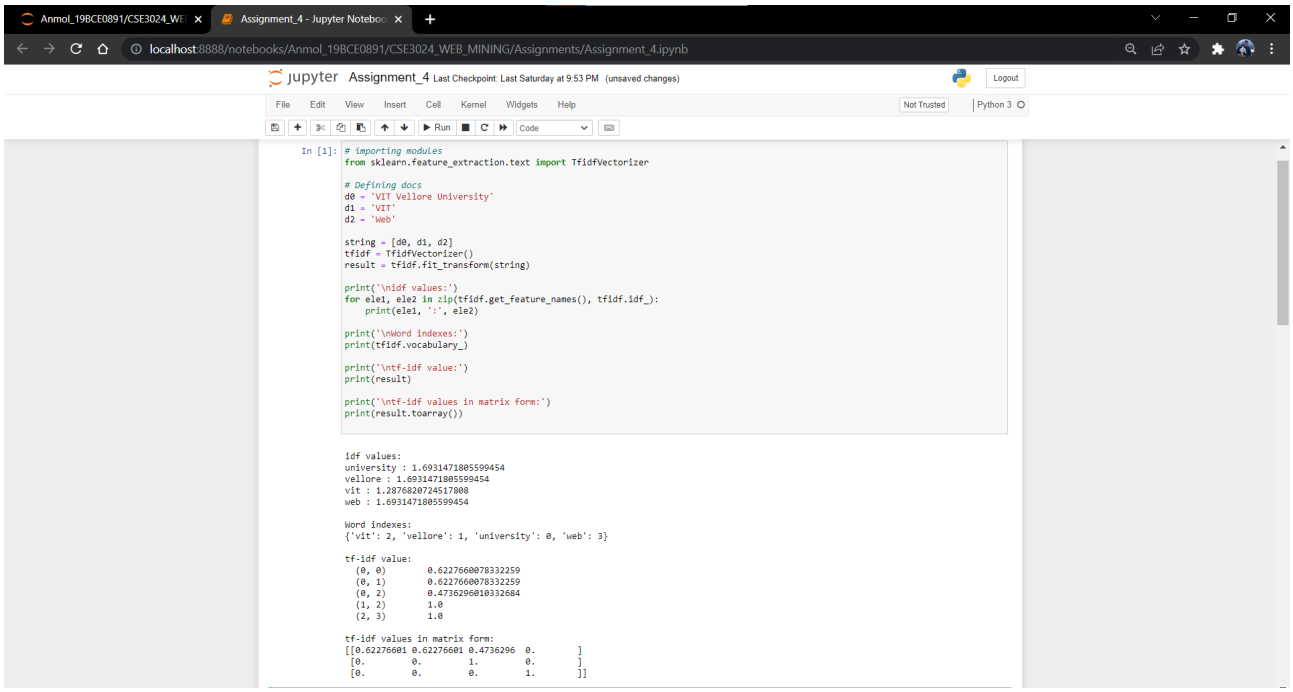
CSE3024 - WEB MINING (L41 + L42)

NAME – ANMOL
REG. NO. - 19BCE0891

DIGITAL ASSIGNMENT – 4

1. Create a Python programme that uses TF-IDF to find the important words in the given corpus. Note: Collect strings from the following documents and create a corpus containing strings from documents d1, d2, and d3.

- **d1: VIT Vellore University**
- **d2: VIT**
- **d3: Web**



CODE -

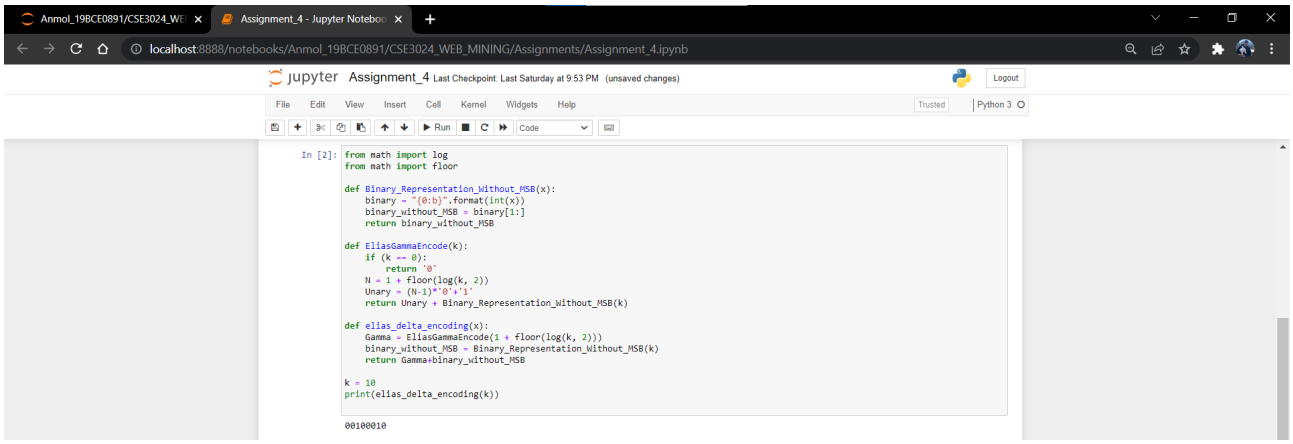
```
# importing modules
from sklearn.feature_extraction.text import TfidfVectorizer

# Defining docs
d0 = 'VIT Vellore University'
d1 = 'VIT'
d2 = 'Web'

string = [d0, d1, d2]
tfidf = TfidfVectorizer()
result = tfidf.fit_transform(string)

print('\nidf values:')
for ele1, ele2 in zip(tfidf.get_feature_names(), tfidf.idf_):
    print(ele1, ':', ele2)
```

```
print('\nWord indexes:')
print(tfidf.vocabulary_)

print('\ntf-idf value:')
print(result)

print('\ntf-idf values in matrix form:')
print(result.toarray())
```

2. Create a Python programme that performs Elias Delta Encoding and Decoding for a given number.

ENCODING :

METHOD - 1

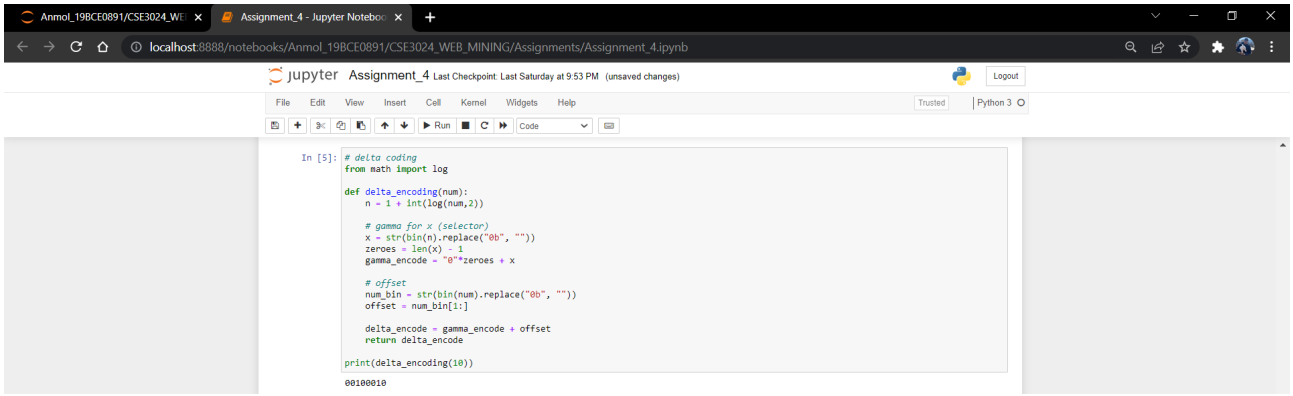


CODE (method-1)

```
from math import log
from math import floor

def Binary_Representation_Without_MSB(x):
    binary = "{0:b}".format(int(x))
    binary_without_MSB = binary[1:]
    return binary_without_MSB

def EliasGammaEncode(k):
    if (k == 0):
        return '0'
    N = 1 + floor(log(k, 2))
    Unary = (N-1)*'0'+'1'
    return Unary + Binary_Representation_Without_MSB(k)

def elias_delta_encoding(x):
```

```
    Gamma = EliasGammaEncode(1 + floor(log(k, 2)))
    binary_without_MSB = Binary_Representation_Without_MSB(k)
    return Gamma+binary_without_MSB

k = 10
print(elias_delta_encoding(k))
```

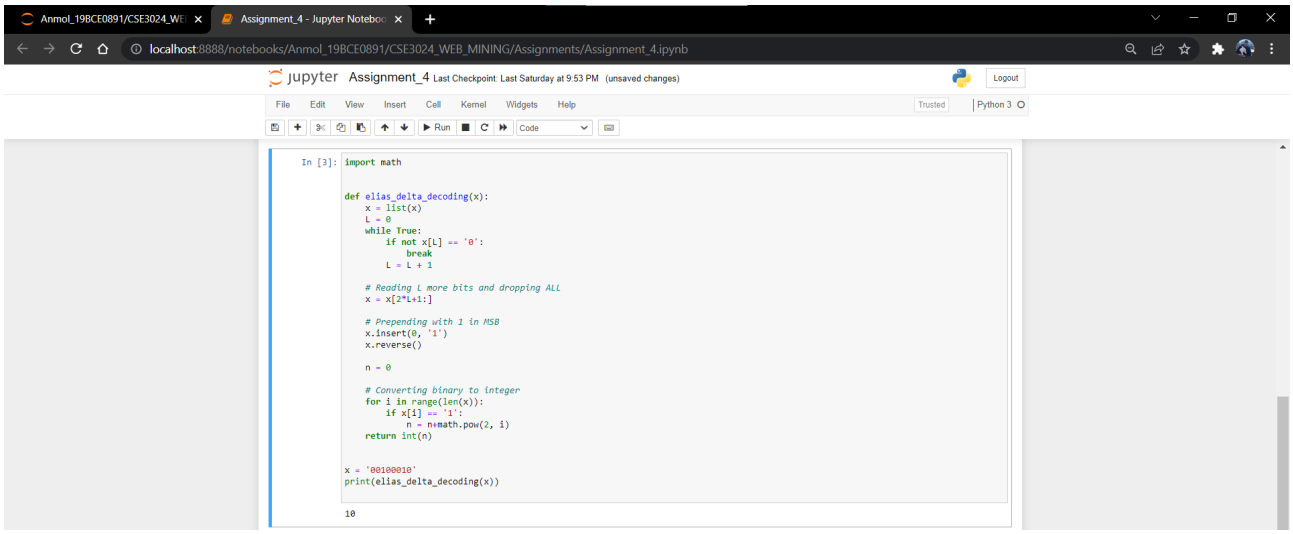
METHOD – 2



CODE (method -2)

```
# delta coding
from math import log

def delta_encoding(num):
    n = 1 + int(log(num,2))

    # gamma for x (selector)
    x = str(bin(n).replace("0b", ""))
    zeroes = len(x) - 1
    gamma_encode = "0"*zeroes + x

    # offset
    num_bin = str(bin(num).replace("0b", ""))
    offset = num_bin[1:]

    delta_encode = gamma_encode + offset
    return delta_encode

print(delta_encoding(10))
```

DECODING



CODE -

```
import math

def elias_delta_decoding(x):
    x = list(x)
    L = 0
    while True:
        if not x[L] == '0':
            break
        L = L + 1

    # Reading L more bits and dropping ALL
    x = x[2*L+1:]

    # Prepending with 1 in MSB
    x.insert(0, '1')
    x.reverse()

    n = 0

    # Converting binary to integer
    for i in range(len(x)):
        if x[i] == '1':
            n = n+math.pow(2, i)
    return int(n)


x = '00100010'
print(elias_delta_decoding(x))
```