CSE 3024

Web Mining

LAB ASSESSMENT - 4

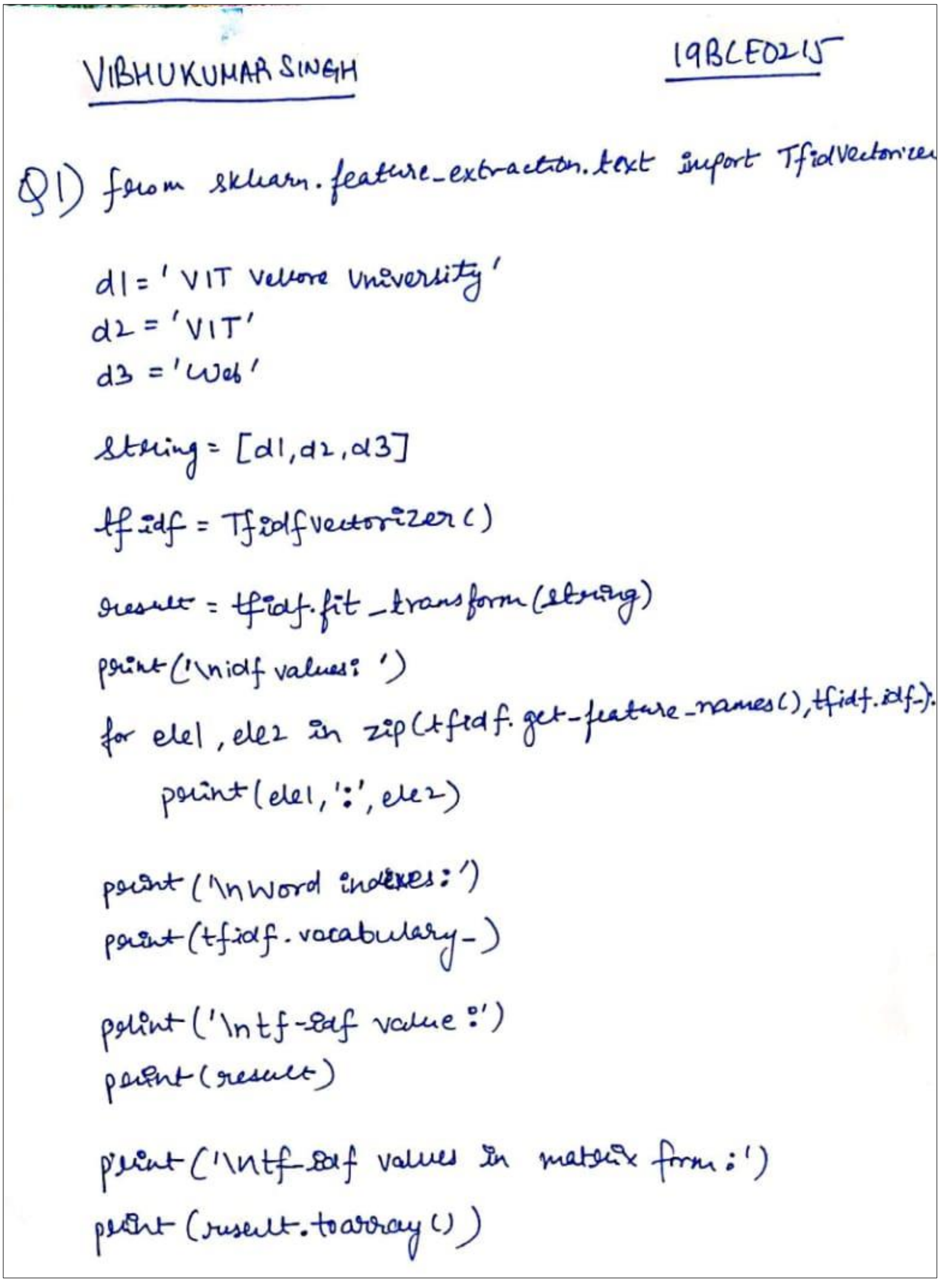
NAME: Vibhu Kumar Singh
REG. NO: 19BCE0215
TEACHER: Mr. Hiteshwar Kumar Azad

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**

Ans 1.

HANDWRITTEN CODE:



VIBHUKUMAR SINGH

19BCE0215

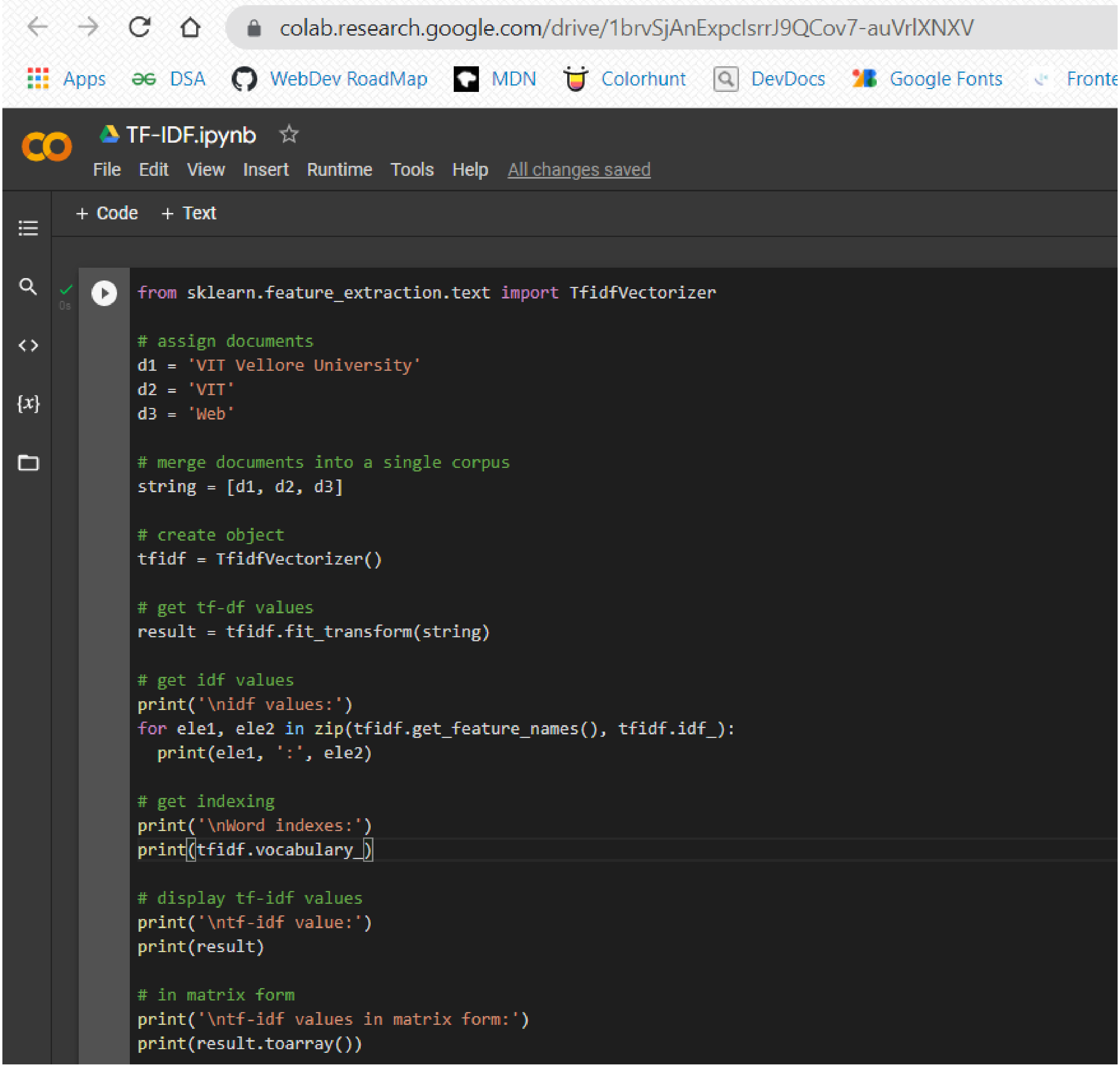
```
Q1) from sklearn.feature_extraction.text import TfidVectorizer

d1 = 'VIT Vellore University'
d2 = 'VIT'
d3 = 'Web'

string = [d1,d2,d3]

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())
```

CODE:

```
from sklearn.feature_extraction.text import TfidfVectorizer

# assign documents
d1 = 'VIT Vellore University'
d2 = 'VIT'
d3 = 'Web'

# merge documents into a single corpus
string = [d1, d2, d3]

# create object
tfidf = TfidfVectorizer()

# get tf-df values
result = tfidf.fit_transform(string)

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

# get indexing
print('\nWord indexes:')
print(tfidf.vocabulary_)

# display tf-idf values
print('\ntf-idf value:')
print(result)

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

CODE SCREENSHOT:

(P.T.O)



```python
from sklearn.feature_extraction.text import TfidfVectorizer

# assign documents
d1 = 'VIT Vellore University'
d2 = 'VIT'
d3 = 'Web'

# merge documents into a single corpus
string = [d1, d2, d3]

# create object
tfidf = TfidfVectorizer()

# get tf-df values
result = tfidf.fit_transform(string)

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

# get indexing
print('\nWord indexes:')
print(tfidf.vocabulary_)

# display tf-idf values
print('\ntf-idf value:')
print(result)

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

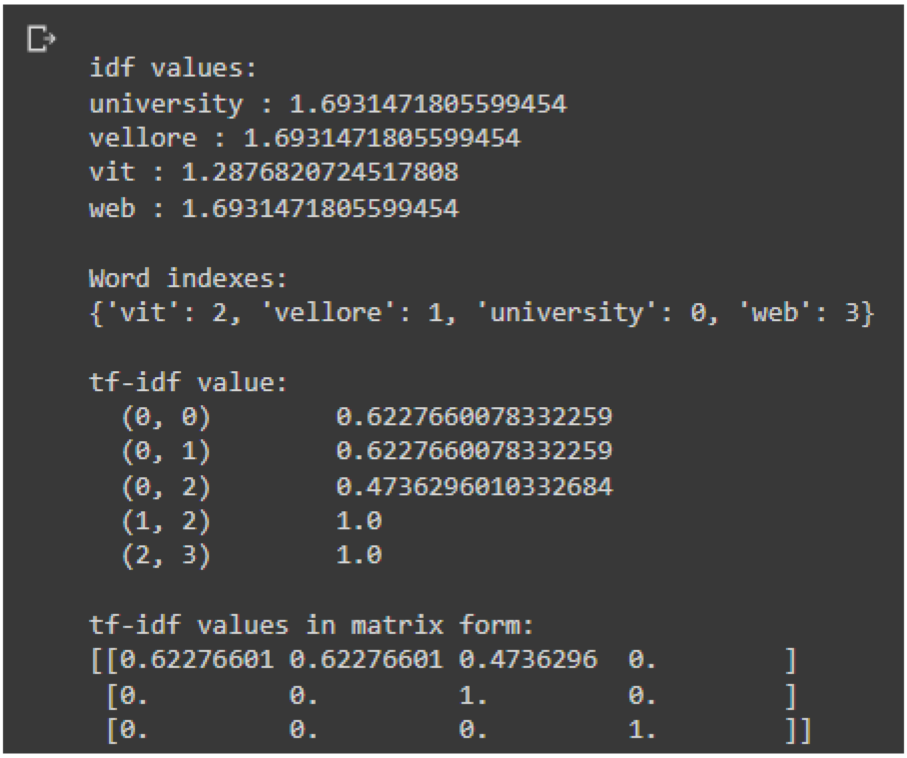
OUTPUT:

```
idf values:
university : 1.6931471805599454
vellore : 1.6931471805599454
vit : 1.2876820724517808
web : 1.6931471805599454

Word indexes:
{'vit': 2, 'vellore': 1, 'university': 0, 'web': 3}

tf-idf value:
  (0, 0)0.6227660078332259
  (0, 1)0.6227660078332259
  (0, 2)0.4736296010332684
  (1, 2)1.0
  (2, 3)1.0
```

tf-idf values in matrix form:

```
[[0.62276601 0.62276601 0.4736296  0.        ]
 [0.         0.         1.         0.        ]
 [0.         0.         0.         1.        ]]
```

OUTPUT SCREENSHOT:



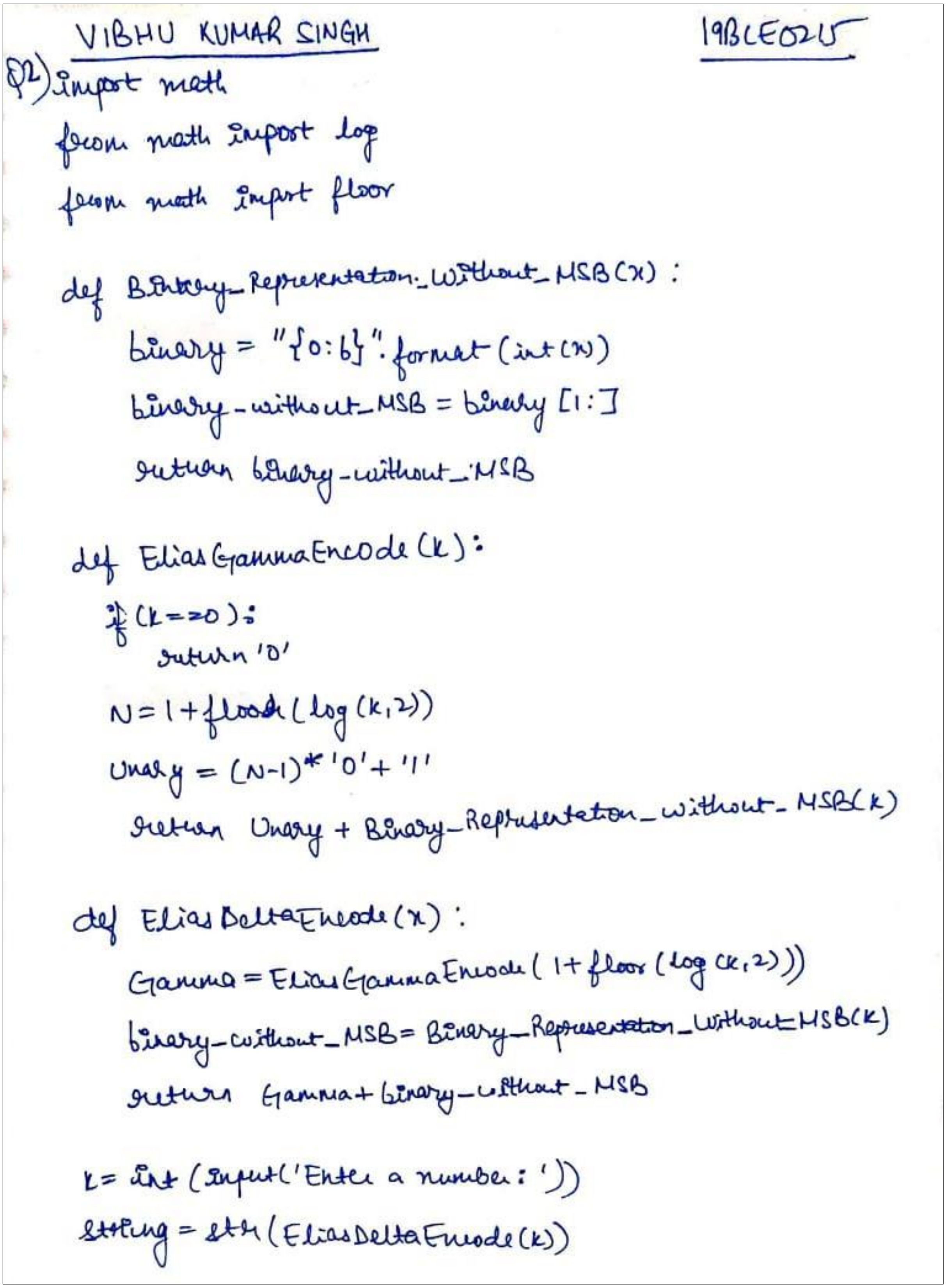
```
idf values:
university : 1.6931471805599454
vellore : 1.6931471805599454
vit : 1.2876820724517808
web : 1.6931471805599454

Word indexes:
{'vit': 2, 'vellore': 1, 'university': 0, 'web': 3}

tf-idf value:
  (0, 0)        0.6227660078332259
  (0, 1)        0.6227660078332259
  (0, 2)        0.4736296010332684
  (1, 2)        1.0
  (2, 3)        1.0

tf-idf values in matrix form:
[[0.62276601 0.62276601 0.4736296  0.        ]
 [0.         0.         1.         0.        ]
 [0.         0.         0.         1.        ]]
```

(P.T.O)

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

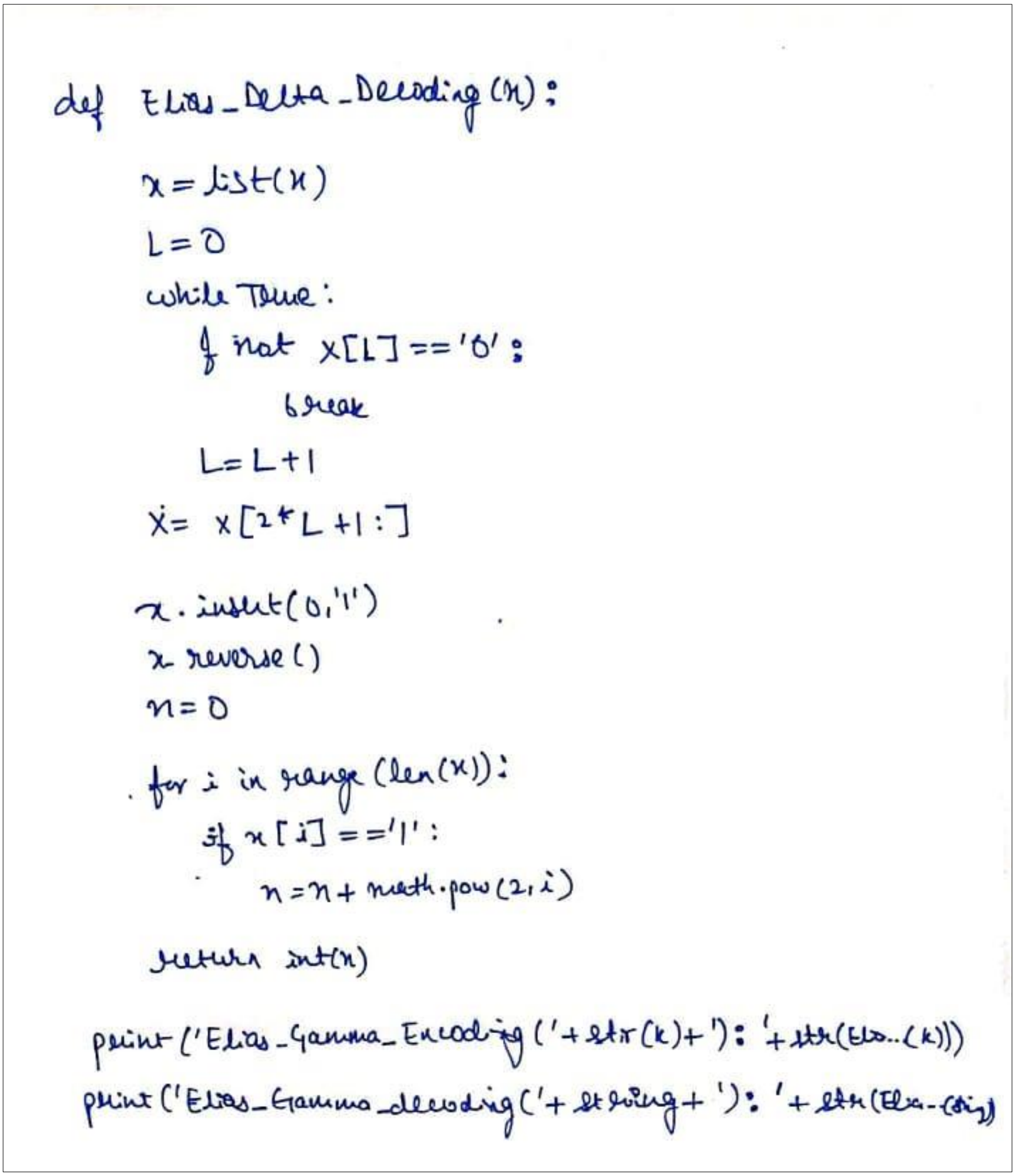
Ans 2.

<u>HANDWRITTEN CODE:</u>



VIBHU KUMAR SINGH 19BCE0215

Q2)
```
import math
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 EliasDeltaEncode(x) :
    Gamma = EliasGammaEncode( 1+floor (log (k,2)))
    binary_without_MSB= Binary_Representation_Without_MSB(k)
    return Gamma+binary_without_MSB

k= int (input('Enter a number: '))
string = str(EliasDeltaEncode(k))
```



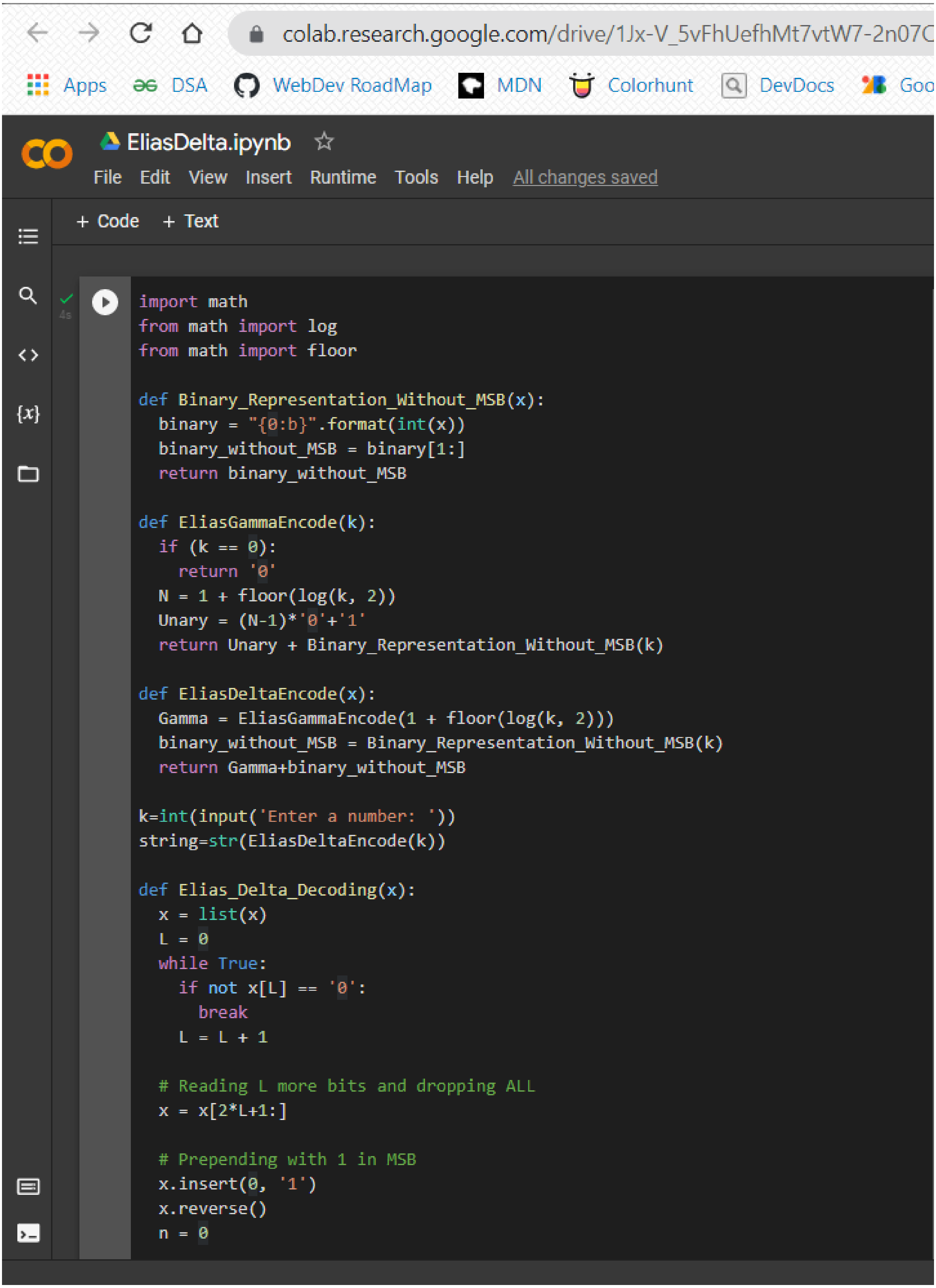
```
def Elias_Delta_Decoding(x):
    x = list(x)
    L = 0
    while True:
        if not x[L] == '0':
            break
        L = L+1
    x = x[2*L+1:]

    x.insert(0,'1')
    x.reverse()
    n = 0
    for i in range(len(x)):
        if x[i] == '1':
            n = n+math.pow(2,i)
    return int(n)

print('Elias_Gamma_Encoding('+str(k)+'): '+str(Elo..(k)))
print('Elias_Gamma_decoding('+string+'): '+str(Ela-(stin))
```

CODE:

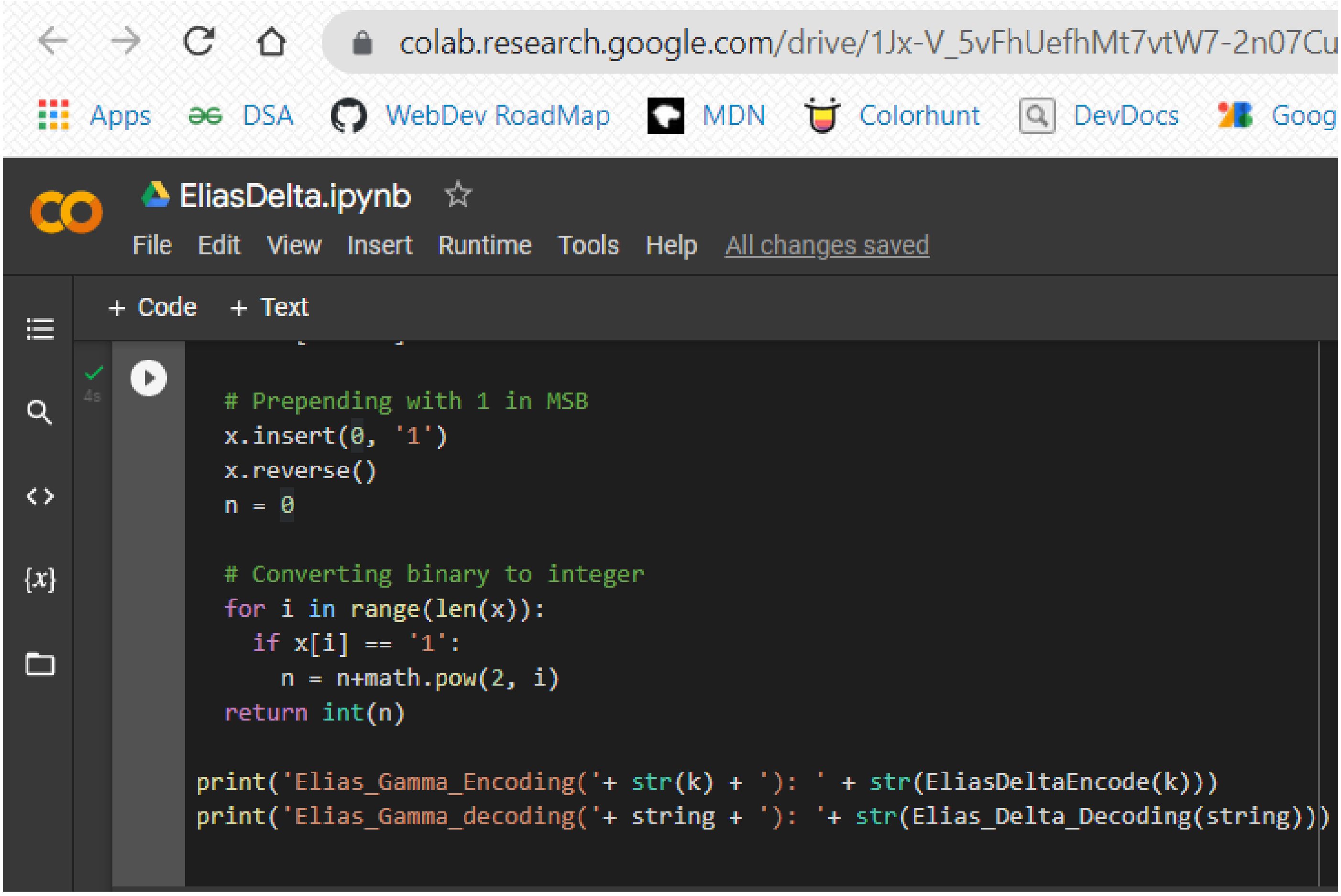
```
from sklearn.feature_extraction.text import TfidfVectorizer

# assign documents
d1 = 'VIT Vellore University'
d2 = 'VIT'
d3 = 'Web'

# merge documents into a single corpus
string = [d1, d2, d3]

# create object
tfidf = TfidfVectorizer()

# get tf-df values
result = tfidf.fit_transform(string)

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

# get indexing
print('\nWord indexes:')
print(tfidf.vocabulary_)

# display tf-idf values
print('\ntf-idf value:')
print(result)

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

CODE SCREENSHOT:

(P.T.O)



```python
import math
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 EliasDeltaEncode(x):
  Gamma = EliasGammaEncode(1 + floor(log(k, 2)))
  binary_without_MSB = Binary_Representation_Without_MSB(k)
  return Gamma+binary_without_MSB

k=int(input('Enter a number: '))
string=str(EliasDeltaEncode(k))

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
```

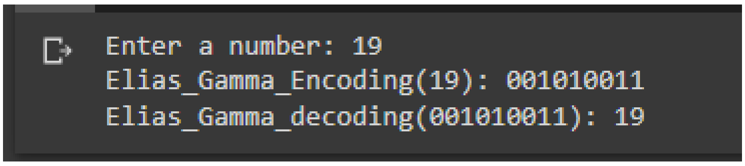


```
  # Prepending with 1 in MSB
  x.insert(0, '1')
  x.reverse()
  n = 0

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

print('Elias_Gamma_Encoding('+ str(k) + '): ' + str(EliasDeltaEncode(k)))
print('Elias_Gamma_decoding('+ string + '): '+ str(Elias_Delta_Decoding(string)))
```

OUTPUT:

Enter a number: 19
Elias_Gamma_Encoding(19): 001010011
Elias_Gamma_decoding(001010011): 19

OUTPUT SCREENSHOT:



```
Enter a number: 19
Elias_Gamma_Encoding(19): 001010011
Elias_Gamma_decoding(001010011): 19
```