```
12)(a) Program to Split sentences


print('Unit-Porgram-12 ...THIS IS  THE FIRST PROGRAM TO SPLIT 
SENTENCES')
lines=[
    'How to tokenize?\n\n\n\n\n\nLike $$\t a boss.',
    'Google is_ accessible via http://www.google.com',
    '1000 new followers! a #TwitterFamous'
    ]
print('Before\t\t\tsplitting \n ',lines)

print('This is first type of splitting\n\n\n')

for line in lines:
    print(line.split()) # It will split 'How to tokenize?\nLike $$\t a 
boss.' to 'How', 'to', 'tokenize?', 'Like', '$$', 'a', 'boss.'
                        # It removes $$ and #
```

Output

```
Unit-Porgram-12 ...THIS IS  THE FIRST PROGRAM TO SPLIT SENTENCES
Before                  splitting
  ['How to tokenize?\n\n\n\n\n\nLike $$\t a boss.', 'Google is_
accessible via http://www.google.com', '1000 new followers! a
#TwitterFamous']
This is first type of splitting



['How', 'to', 'tokenize?', 'Like', '$$', 'a', 'boss.']
['Google', 'is_', 'accessible', 'via', 'http://www.google.com']
['1000', 'new', 'followers!', 'a', '#TwitterFamous']
```

```
12)(b) Program to Split sentences
lines2=[
    'India has many historical\n\n\n\n\n\n monuments\t',
    'Many information\t\t\t\t\t #can be %obtained from 
http://www.google.com',
    '1000 new followers! a #TwitterFamous'
    ]

print('This is first type\n\n\n')
for line in lines2:
    print(line.split())
```

OUTPUT

```
This is first type



['India', 'has', 'many', 'historical', 'monuments']
['Many', 'information', '#can', 'be', '%obtained', 'from',
'http://www.google.com']
['1000', 'new', 'followers!', 'a', '#TwitterFamous']
```

```
12)(C) Program to Split sentences
```

```
lines2=[
    'India has many historical\n\n\n\n\n\n monuments\t',
    'Many information\t\t\t\t\t #can be %obtained from
http://www.google.com',
    '1000 new followers! a #TwitterFamous'
    ]

print('This is first type\n\n\n')
for line in lines2:
    print(line.split())
```

OUTPUT

```
This is first type



['India', 'has', 'many', 'historical', 'monuments']
['Many', 'information', '#can', 'be', '%obtained', 'from',
'http://www.google.com']
['1000', 'new', 'followers!', 'a', '#TwitterFamous']
```

13) (a) Second Program for Splitting sentences

```
print('13.  Second Program for splitting sentences')
print('THIS IS THE SECOND PROGRAM FOR SPLITTING SENTENCES')
lines=[
    'How to tokenize?\nLike a boss.',
    'Google is_ accessible via http://www.google.com',
    '1000 new %followers! a #TwitterFamous'
    ]

#A Regular Expressions (RegEx) is a special sequence of characters
#that uses a search pattern to find a string or set of strings.
#It can detect the presence or absence of a text by matching
```

```
#it with a particular pattern, and also can split a pattern into
#one or more sub-patterns. Python provides a re module that supports
#the use of regex in Python. Its primary function is to offer a search,
#where it takes a regular expression and a string.

# import re

import re
#_token_pattern=r"\w+"

_token_pattern=r"\w"


#Python’s re.compile() method is used to compile a regular expression
pattern provided
#as a string into a regex pattern object (re.Pattern).
#Later we can use this pattern object to search
#for a match inside different target strings using regex methods

token_pattern=re.compile(_token_pattern)


print('This is second type\n\n\n')
for line in lines:
    print(token_pattern.findall(line))
```

OUTPUT

```
13.   Second Program for splitting sentences
THIS IS THE SECOND PROGRAM FOR SPLITTING SENTENCES
This is second type



['H', 'o', 'w', 't', 'o', 't', 'o', 'k', 'e', 'n', 'i', 'z', 'e', 'L',
'i', 'k', 'e', 'a', 'b', 'o', 's', 's']
['G', 'o', 'o', 'g', 'l', 'e', 'i', 's', '_', 'a', 'c', 'c', 'e', 's',
's', 'i', 'b', 'l', 'e', 'v', 'i', 'a', 'h', 't', 't', 'p', 'w', 'w',
'w', 'g', 'o', 'o', 'g', 'l', 'e', 'c', 'o', 'm']
['1', '0', '0', '0', 'n', 'e', 'w', 'f', 'o', 'l', 'l', 'o', 'w', 'e',
'r', 's', 'a', 'T', 'w', 'i', 't', 't', 'e', 'r', 'F', 'a', 'm', 'o',
'u', 's']
```

13) (b) Second Program for Splitting sentences

```
print('2. Second Program for splitting sentences')
print('THIS IS THE SECOND PROGRAM FOR SPLITTING SENTENCES')
lines=[
    'India is great'
    ]

#A Regular Expressions (RegEx) is a special sequence of characters
```

```
#that uses a search pattern to find a string or set of strings.
#It can detect the presence or absence of a text by matching
#it with a particular pattern, and also can split a pattern into
#one or more sub-patterns. Python provides a re module that supports
#the use of regex in Python. Its primary function is to offer a search,
#where it takes a regular expression and a string.

# import re

import re
#_token_pattern=r"\w+"

_token_pattern=r"\w+"


#Python’s re.compile() method is used to compile a regular expression
pattern provided
#as a string into a regex pattern object (re.Pattern).
#Later we can use this pattern object to search
#for a match inside different target strings using regex methods

token_pattern=re.compile(_token_pattern)

print('The given sentence',lines)
print('This is second type\n\n\n')
for line in lines:
    print(token_pattern.findall(line))
```

OUTPUT

```
2.    Second Program for splitting sentences
THIS IS THE SECOND PROGRAM FOR SPLITTING SENTENCES
The given sentence ['India is great']
This is second type



['India', 'is', 'great']
```

13) (c) Second Program for Splitting sentences

```
import re
#_token_pattern=r"\w+"




lines=[
    'India is great'
    ]
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)

print('The given sentence',lines)
print('This is second type\n\n\n')
for line in lines:
    print(token_pattern.findall(line))
```

OUTPUT

```
The given sentence ['India is great']
This is second type



['India', 'is', 'great']
```

13) (d) Second Program for Splitting sentences

```
import re
#_token_pattern=r"\w+"



lines=[
    'India is great'
    ]
_token_pattern=r"\w"
token_pattern=re.compile(_token_pattern)

print('The given sentence',lines)
print('This is second type\n\n\n')
for line in lines:
    print(token_pattern.findall(line))
```

OUTPUT

```
The given sentence ['India is great']
This is second type



['I', 'n', 'd', 'i', 'a', 'i', 's', 'g', 'r', 'e', 'a', 't']
```

13) (e) Second Program for Splitting sentences

```
print('2. Second Program for splitting sentences')
print('THIS IS THE SECOND PROGRAM FOR SPLITTING SENTENCES')
lines=[
    'How to tokenize?\nLike a boss.',
    'Google is_ accessible via http://www.google.com',
    '1000 new %followers! a #TwitterFamous'
    ]

#A Regular Expressions (RegEx) is a special sequence of characters
#that uses a search pattern to find a string or set of strings.
#It can detect the presence or absence of a text by matching
#it with a particular pattern, and also can split a pattern into
#one or more sub-patterns. Python provides a re module that supports
#the use of regex in Python. Its primary function is to offer a search,
#where it takes a regular expression and a string.

# import re

import re
#_token_pattern=r"\w+"
#_token_pattern=r"\w"
_token_pattern=r"\w+"

token_pattern=re.compile(_token_pattern)
print('This is second type\n\n\n')
for line in lines:
    print(token_pattern.findall(line))
```

OUTPUT

```
2.    Second Program for splitting sentences
THIS IS THE SECOND PROGRAM FOR SPLITTING SENTENCES
This is second type



['How', 'to', 'tokenize', 'Like', 'a', 'boss']
['Google', 'is_', 'accessible', 'via', 'http', 'www', 'google', 'com']
['1000', 'new', 'followers', 'a', 'TwitterFamous']
```

14) (a)Program to split sentences

```
print(' Program to split sentences where one character words are not
considered')
print('THIS  TYPE-3')
lines=[
    'How to tokenize?\nLike a boss.',
    'Google is_ accessible\t\t via http://www.google.com',
    '1000 new followers! a #TwitterFamous'
    ]
```

```
import re


#_token_pattern=r"(?u)\b\w\w+\b"
#_token_pattern=r"\b\w\w+\b"
#_token_pattern=r"\w\w+\b"
_token_pattern=r"\w\w+"

token_pattern=re.compile(_token_pattern)
print('The given sentences ',lines)
print('This is third type\n\n')
for line in lines:
    print(token_pattern.findall(line))
```

OUTPUT

```
      Program to split sentences where one character words are not
considered
THIS  TYPE-3
The given sentences  ['How to tokenize?\nLike a boss.', 'Google is_
accessible\t\t via http://www.google.com', '1000 new followers! a
#TwitterFamous']
This is third type


['How', 'to', 'tokenize', 'Like', 'boss']
['Google', 'is_', 'accessible', 'via', 'http', 'www', 'google', 'com']
['1000', 'new', 'followers', 'TwitterFamous']
```

14) (b) Program to split sentences

```
print(' Program to split sentences where one character words are not
considered')
print('THIS  TYPE-3')
lines=[
    'India is a #great country'
    ]



import re



_token_pattern=r"\w\w+"
#_token_pattern=r"\w+"

token_pattern=re.compile(_token_pattern)
print('The given sentences ',lines)
print('This is third type\n\n')
```

```
for line in lines:
    print(token_pattern.findall(line))
```

OUTPUT

```
     Program to split sentences where one character words are not
considered
THIS  TYPE-3
The given sentences  ['India is a #great country']
This is third type


['India', 'is', 'great', 'country']
```

15) (a) Program to split sentences

```
print(' Program to split sentences where hashtag and url are
represented')
print('THIS IS THE FOURTH TYPE')
lines=[
    'How to #tokenize?\nLike a %boss.',
    'Google @is_ $accessible @via http://www.google.com.com.abc',
    'https://abc.ybc.cef','$abc,$edfg'
    '1000 10new #followers! a #TwitterFamous $TwitterFamous'
    ]

print('The given data\n\n\n',lines)
#for line in lines:
#    print(line.split())


import re


_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)

def tokenizer(line):
    line=line.lower()

    line=re.sub(r'http[s]?://[\w\.\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)

    line=re.sub(r'%\w+','_percent_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)
```

```
print('This is fourth type\n\n\n')

for line in lines:
    print(tokenizer(line))
```

OUTPUT

```
      Program to split sentences where hashtag and url are represented
THIS IS THE FOURTH TYPE
The given data


 ['How to #tokenize?\nLike a %boss.', 'Google @is_ $accessible @via
http://www.google.com.com.abc', 'https://abc.ybc.cef', '$abc,$edfg1000
10new #followers! a #TwitterFamous $TwitterFamous']
This is fourth type



['how', 'to', '_hashtag_', 'like', 'a', '_percent_']
['google', 'is_', 'accessible', 'via', '_url_']
['_url_']
['abc', 'edfg_num_', '_num_new', '_hashtag_', 'a', '_hashtag_',
'twitterfamous']
```

15) (b) **Program to split sentences**

```
print('b. Program to split the words')
lines=[
    'How to tokenize?\nLike a boss.',
     'Google is_ accessible via http://www.google.com',
    'Google is_ accessible via http://www.google.com.com.com.com',
    'Google is_ accessible via https://www.google.com.com.com.com',
    '1000 new followers! a #TwitterFamous'
    ]

#print('This is first type\n\n\n')
#for line in lines:
#    print(line.split())


import re


_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)
```

```
def tokenizer(line):
    line=line.lower()

    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    return token_pattern.findall(line)

print('This is fourth type\n\n\n')
print('\n It converts to lower case and removes Special characters and
https[s]')

for line in lines:
    print(tokenizer(line))
```

OUTPUT

```
b. Program to split the words
This is fourth type




 It converts to lower case and removes Special characters and https[s]
['how', 'to', 'tokenize', 'like', 'a', 'boss']
['google', 'is_', 'accessible', 'via', '_url_', 'google', 'com']
['google', 'is_', 'accessible', 'via', '_url_', 'google', 'com', 'com',
'com', 'com']
['google', 'is_', 'accessible', 'via', '_url_', 'google', 'com', 'com',
'com', 'com']
['1000', 'new', 'followers', 'a', 'twitterfamous']
```

15) (c) **Program to split sentences**

```
print('c.Program to split the words')
lines=[
    'How to tokenize?\nLike a boss.',
    'Google is_ accessible via http://www.google.com',
    '1000 new followers! a #TwitterFamous and speed'
    ]

#print('This is first type\n\n\n')
#for line in lines:
#    print(line.split())


import re


_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)

def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\.\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)

    return token_pattern.findall(line)

print('This is fourth type\n\n\n')

for line in lines:
    print(tokenizer(line))
```

OUTPUT

```
c.Program to split the words
This is fourth type



['how', 'to', 'tokenize', 'like', 'a', 'boss']
['google', 'is_', 'accessible', 'via', '_url_']
['1000', 'new', 'followers', 'a', '_hashtag_', 'and', 'speed']
```

15) (d) **Program to split sentences**

```
print('d.Program to split the word')
lines=[
    'How to tokenize?\nLike a boss.',
    'Google is_ accessible via http://www.google.com',
    '1000 new followers! a #TwitterFamous',
    '1000 new 10followers! 5a #TwitterFamous'
    ]

#print('This is first type\n\n\n')
#for line in lines:
#    print(line.split())


import re


_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)

def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\.\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)

print('This is fourth type\n\n\n')

for line in lines:
    print(tokenizer(line))
```

OUTPUT

```
d.Program to split the word
This is fourth type



['how', 'to', 'tokenize', 'like', 'a', 'boss']
['google', 'is_', 'accessible', 'via', '_url_']
['_num_', 'new', 'followers', 'a', '_hashtag_']
['_num_', 'new', '_num_followers', '_num_a', '_hashtag_']
```

15) (e) **Program to split sentences**

```
print('e.Program to split the word')
lines=[
    'How to tokenize?\nLike a boss.',
    'Google is_ accessible via http://www.google.com',
    'Google is_ accessible via http://www.google.com.com',
    '1000 new followers! a #TwitterFamous',
    '1000 new 10followers! 5a #TwitterFamous'
    ]

#print('This is first type\n\n\n')
#for line in lines:
#    print(line.split())


import re


_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)

def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)
```

```
print('This is fourth type\n\n\n')

for line in lines:
    print(tokenizer(line))
```

OUTPUT

```
e.Program to split the word
This is fourth type



['how', 'to', 'tokenize', 'like', 'a', 'boss']
['google', 'is_', 'accessible', 'via', '_url_', 'google', 'com']
['google', 'is_', 'accessible', 'via', '_url_', 'google', 'com', 'com']
['_num_', 'new', 'followers', 'a', '_hashtag_']
['_num_', 'new', '_num_followers', '_num_a', '_hashtag_']
```

(15) (f) Program to split sentences

```
print('f. Program to split the words')
lines=[
    'How to tokenize?\nLike a boss.',
     'Google is_ accessible via http://www.google.com',
    'Google is_ accessible via http://www.google.com.com.com.com',
    'Google is_ accessible via https://www.google.com.com.com.com',
    '1000 new followers! a #TwitterFamous'
    ]
print('The given ',lines)
#print('This is first type\n\n\n')
#for line in lines:
#    print(line.split())


import re


_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)

def tokenizer(line):
    line=line.lower()

    line=re.sub(r'http[s]?://[\w\.\?]+','url',line)
    return token_pattern.findall(line)

print('This is fourth type\n\n\n')
print('\n It converts to lower case and removes Special characters and
http[s]')
```

```
for line in lines:
    print(tokenizer(line))
```

OUTPUT

```
f. Program to split the words
The given  ['How to tokenize?\nLike a boss.', 'Google is_ accessible
via http://www.google.com', 'Google is_ accessible via
http://www.google.com.com.com.com', 'Google is_ accessible via
https://www.google.com.com.com.com', '1000 new followers! a
#TwitterFamous']
This is fourth type



 It converts to lower case and removes Special characters and http[s]
['how', 'to', 'tokenize', 'like', 'a', 'boss']
['google', 'is_', 'accessible', 'via', 'url']
['google', 'is_', 'accessible', 'via', 'url']
['google', 'is_', 'accessible', 'via', 'url']
['1000', 'new', 'followers', 'a', 'twitterfamous']
```

16) (a) Program on Count Vectorizer

```
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)
```

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print('x.todense = ',x.todense)
pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)

print('It is the result of xyz=pd.DataFrame')
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
x.todense =  <bound method spmatrix.todense of <3x16 sparse matrix of
type '<class 'numpy.int64'>'
      with 17 stored elements in Compressed Sparse Row format>>
It is the result of xyz=pd.DataFrame
   _hashtag_  _num_  _url_  a  accessible  boss  com  followers  google
how  \
0          0      0      0  1           0     1    0          0       0
1
1          0      0      1  0           1     0    1          0       3
0
2          1      1      0  1           0     0    0          1       0
0

   is_  like  new  to  tokenize  via
0    0     1    0   1         1    0
1    1     0    0   0         0    1
2    0     0    1   0         0    0
```

16) (b) Program on Count Vectorizer

```
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')
```

```python
lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)



flight_delayed_lines=[
    'Flight was delayed , I am not happy much',
    'Flight was not delayed,I am happy'
    ]


x=vec.fit_transform(flight_delayed_lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
   am  delayed  flight  happy  i  much  not  was
0   1        1       1      1  1     1    1    1
1   1        1       1      1  1     0    1    1
```

16) (c) Program on CountVectorizer

```python
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)

flight_delayed_lines=[
    'Flight was delayed , I am not happy'

    ]
```

```


x=vec.fit_transform(flight_delayed_lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print('x.todense() and its zize ',x.todense(),x.todense().shape)
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
x.todense() and its zize  [[1 1 1 1 1 1 1]] (1, 7)
   am  delayed  flight  happy  i  not  was
0   1        1       1      1  1    1    1
```

16) (d) Program on Count Vectorizer

```
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
```

```
vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print('x.todense = ',x.todense)
pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)

print('It is the result of xyz=pd.DataFrame')
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
x.todense =  <bound method spmatrix.todense of <3x16 sparse matrix of
type '<class 'numpy.int64'>'
      with 17 stored elements in Compressed Sparse Row format>>
It is the result of xyz=pd.DataFrame
   _hashtag_  _num_  _url_  a  accessible  boss  com  followers  google
how  \
0          0      0      0  1           0     1    0          0       0
1
1          0      0      1  0           1     0    1          0       3
0
2          1      1      0  1           0     0    0          1       0
0

   is_  like  new  to  tokenize  via
0    0     1    0   1         1    0
1    1     0    0   0         0    1
2    0     0    1   0         0    0
```

17) (a) Program on TfidfTransformer

```
#TfidfTransformer. Transform a count matrix to a normalized tf or tf-
idf representation.

#Tf means term-frequency while tf-idf means term-frequency times
inverse document-frequency.
#This is a common term weighting scheme in information retrieval,
#that has also found good use in document classification.

from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tvec = TfidfVectorizer()
documents = [
    "apple apple",
    "apple orange"
]
x = tvec.fit_transform(documents) # vectorizing documents
print(tvec.vocabulary_) # printing learned vocab of vectorizer
print(x.toarray()) # converting sparse to dense array
```

OUTPUT

```
{'apple': 0, 'orange': 1}
[[1.         0.        ]
 [0.57973867 0.81480247]]
```

17) (b) Program on CountVetorizer

```
print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

import pandas as pd
lines=[
    'India is a great country',
    'Bharatha has rich divine culture'
   ]


def tokenizer(line):
    #line=line.lower()          #Changed
    #line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    #line=re.sub(r'#\w+','_hashtag_',line)
    #line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=False,tokenizer=tokenizer)  #Changed
x=vec.fit_transform(lines)
```

```
xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )

#pd.set_option("display.max_rows", None)
#pd.set_option("display.max_columns", None)

print('It is the result of xyz=pd.DataFrame')
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
It is the result of xyz=pd.DataFrame
   Bharatha  India  a  country  culture  divine  great  has  is  rich
0         0      1  1        1        0       0      1    0   1     0
1         1      0  0        0        1       1      0    1   0     1
```

17) (c) Program on CountVectorizer

```
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
```

```
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)



flight_delayed_lines=[
    'Flight was delayed , I am not happy much',
    'Flight was not delayed,I am happy'
    ]


x=vec.fit_transform(flight_delayed_lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
   am  delayed  flight  happy  i  much  not  was
0   1        1       1      1  1     1    1    1
1   1        1       1      1  1     0    1    1
```

17) (d) Program on CountVetorizer

```
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)

flight_delayed_lines=[
    'Flight was delayed , I am not happy'

    ]


x=vec.fit_transform(flight_delayed_lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print('x.todense() and its zize ',x.todense(),x.todense().shape)
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
x.todense() and its zize  [[1 1 1 1 1 1 1]] (1, 7)
   am  delayed  flight  happy  i  not  was
0   1        1       1      1  1    1    1
```

17) (e) Program on CountVectorizer

```
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print('x.todense = ',x.todense)
pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)

print('It is the result of xyz=pd.DataFrame')
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
x.todense =  <bound method spmatrix.todense of <3x16 sparse matrix of
type '<class 'numpy.int64'>'
      with 17 stored elements in Compressed Sparse Row format>>
It is the result of xyz=pd.DataFrame
   _hashtag_  _num_  _url_  a  accessible  boss  com  followers  google
how  \
0          0      0      0  1           0     1    0          0       0
1
1          0      0      1  0           1     0    1          0       3
0
2          1      1      0  1           0     0    0          1       0
0

   is_  like  new  to  tokenize  via
0    0     1    0   1         1    0
1    1     0    0   0         0    1
2    0     0    1   0         0    0
```

17. (f) Program on CountVectorizer

```python
print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

import pandas as pd
lines=[
    'India is a great country',
    'Bharatha has rich divine culture'
   ]


def tokenizer(line):
    #line=line.lower()          #Changed
    #line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    #line=re.sub(r'#\w+','_hashtag_',line)
    #line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


from sklearn.feature_extraction.text import CountVectorizer
```

```
vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)  #Changed
x=vec.fit_transform(lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )

#pd.set_option("display.max_rows", None)
#pd.set_option("display.max_columns", None)

#print('It is the result of xyz=pd.DataFrame')
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
   a  bharatha  country  culture  divine  great  has  india  is  rich
0  1         0        1        0       0      1    0      1   1     0
1  0         1        0        1       1      0    1      0   0     1
```

17. (g) Program on CountVectorizer

```
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)
```

```
import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)



flight_delayed_lines=[
    'Flight was delayed , I am not happy much',
    'Flight was not delayed,I am happy'
    ]


x=vec.fit_transform(flight_delayed_lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
   am  delayed  flight  happy  i  much  not  was
0   1        1       1      1  1     1    1    1
1   1        1       1      1  1     0    1    1
```

17. (g) Program on CountVectorizer

```
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)

flight_delayed_lines=[
    'Flight was delayed , I am not happy'

    ]


x=vec.fit_transform(flight_delayed_lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print('x.todense() and its zize ',x.todense(),x.todense().shape)
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
x.todense() and its zize  [[1 1 1 1 1 1 1]] (1, 7)
   am  delayed  flight  happy  i  not  was
0   1        1       1      1  1    1    1
```

17. (h) Program on CountVectorizer

```
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print('x.todense = ',x.todense)
pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)

print('It is the result of xyz=pd.DataFrame')
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
x.todense =  <bound method spmatrix.todense of <3x16 sparse matrix of
type '<class 'numpy.int64'>'
      with 17 stored elements in Compressed Sparse Row format>>
It is the result of xyz=pd.DataFrame
   _hashtag_  _num_  _url_  a  accessible  boss  com  followers  google
how  \
0          0      0      0  1           0     1    0          0       0
1
1          0      0      1  0           1     0    1          0       3
0
2          1      1      0  1           0     0    0          1       0
0

   is_  like  new  to  tokenize  via
0    0     1    0   1         1    0
1    1     0    0   0         0    1
2    0     0    1   0         0    0
```

17. (i) Program on CountVectorizer

```python
print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

import pandas as pd
lines=[
    'India is a great country',
    'Bharatha has rich divine culture'
   ]


def tokenizer(line):
    #line=line.lower()         #Changed
    #line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    #line=re.sub(r'#\w+','_hashtag_',line)
    #line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)
```

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

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)  #Changed
x=vec.fit_transform(lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )

#pd.set_option("display.max_rows", None)
#pd.set_option("display.max_columns", None)

#print('It is the result of xyz=pd.DataFrame')
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
   a  bharatha  country  culture  divine  great  has  india  is  rich
0  1         0        1        0       0      1    0      1   1     0
1  0         1        0        1       1      0    1      0   0     1
```

17. (j) Program on CountVectorizer

```python
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)



flight_delayed_lines=[
    'Flight was delayed , I am not happy much',
    'Flight was not delayed,I am happy'
    ]


x=vec.fit_transform(flight_delayed_lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
   am  delayed  flight  happy  i  much  not  was
0   1        1       1      1  1     1    1    1
1   1        1       1      1  1     0    1    1
```

17. (k) Program on CountVectorizer

```
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')
print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)

flight_delayed_lines=[
    'Flight was delayed , I am not happy'

    ]


x=vec.fit_transform(flight_delayed_lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print('x.todense() and its zize ',x.todense(),x.todense().shape)
print(xyz)
```

OUTPUT

```
      Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
```

```
x.todense() and its zize  [[1 1 1 1 1 1 1]] (1, 7)
   am  delayed  flight  happy  i  not  was
0   1        1       1      1  1    1    1
```

17. (1) Program on CountVectorizer

```python
import pandas as pd

print(' Program on CountVectorizer to tokenize the given sentences')

print('THIS IS THE PROGRAM  ON CountVectorizer')
print('ONLY Count Vectorizer')

lines=[
    'How to tokenize?\nLike a boss.',
    'Google google is_ accessible via http://www.google.com',
   '1000 new followers! a #TwitterFamous'
   ]


def tokenizer(line):
    line=line.lower()
    line=re.sub(r'http[s]?://[\w\?]+','_url_',line)
    line=re.sub(r'#\w+','_hashtag_',line)
    line=re.sub(r'\d+','_num_',line)
    return token_pattern.findall(line)


import re
_token_pattern=r"\w+"
token_pattern=re.compile(_token_pattern)


import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

vec=CountVectorizer(lowercase=True,tokenizer=tokenizer)
x=vec.fit_transform(lines)


xyz=pd.DataFrame(
    x.todense(),
    columns=vec.get_feature_names_out()
    )
print('x.todense = ',x.todense)
pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)
```

```
print('It is the result of xyz=pd.DataFrame')
print(xyz)
```

OUTPUT

```
     Program on CountVectorizer to tokenize the given sentences
THIS IS THE PROGRAM  ON CountVectorizer
ONLY Count Vectorizer
x.todense =  <bound method spmatrix.todense of <3x16 sparse matrix of
type '<class 'numpy.int64'>'
     with 17 stored elements in Compressed Sparse Row format>>
It is the result of xyz=pd.DataFrame
   _hashtag_  _num_  _url_  a  accessible  boss  com  followers  google
how  \
0          0      0      0  1           0     1    0          0       0
1
1          0      0      1  0           1     0    1          0       3
0
2          1      1      0  1           0     0    0          1       0
0

   is_  like  new  to  tokenize  via
0    0     1    0   1         1    0
1    1     0    0   0         0    1
2    0     0    1   0         0    0
```