Introduction

What is n-gram? => It is essentially a string of words that appear in the same window at the same time.

Use of n-grams? =>

1. In natural language processing
2. Text mining.

Google and Microsoft => have created web-scale grammar models => Uses are : 1.Spelling correction 2.Hyphenation 3.Text summarization

Que1 : Implement n-grams in Python with NLTK

```
from nltk import ngrams
```

Implementing n-grams in Pythong

```
sentence= "After mastering various skills, you became a connoisseur of knowledge, gracefully navigating the and boundless creativity "
```

```
def generate_ngrams(sentence,n):
  n_grams= ngrams(sentence.split(),n)
  for grams in n_grams:
    print(grams)
```

```
generate_ngrams(sentence,5)
```

```
('After', 'mastering', 'various', 'skills,', 'you')
('mastering', 'various', 'skills,', 'you', 'became')
('various', 'skills,', 'you', 'became', 'a')
('skills,', 'you', 'became', 'a', 'connoisseur')
('you', 'became', 'a', 'connoisseur', 'of')
('became', 'a', 'connoisseur', 'of', 'knowledge,')
('a', 'connoisseur', 'of', 'knowledge,', 'gracefully')
('connoisseur', 'of', 'knowledge,', 'gracefully', 'navigating')
('of', 'knowledge,', 'gracefully', 'navigating', 'the')
('knowledge,', 'gracefully', 'navigating', 'the', 'and')
('gracefully', 'navigating', 'the', 'and', 'boundless')
('navigating', 'the', 'and', 'boundless', 'creativity')
```

```
generate_ngrams(sentence,3)
```

```
('After', 'mastering', 'various')
('mastering', 'various', 'skills,')
('various', 'skills,', 'you')
('skills,', 'you', 'became')
('you', 'became', 'a')
('became', 'a', 'connoisseur')
('a', 'connoisseur', 'of')
('connoisseur', 'of', 'knowledge,')
('of', 'knowledge,', 'gracefully')
('knowledge,', 'gracefully', 'navigating')
('gracefully', 'navigating', 'the')
('navigating', 'the', 'and')
('the', 'and', 'boundless')
('and', 'boundless', 'creativity')
```

Que 2: Generate key phrases from a given text where length of key phrases varies from 1 to 5.

```
from nltk import word_tokenize
from nltk.util import ngrams
```

```
import nltk

# Download the 'punkt' resource
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
def generate_key_phrases(text):
  key_phrases = []# this is to initialize the empty list where you will save your answers
  tokens = word_tokenize(text.lower()) # tokenize the text
  max_length = min(5, len(tokens)) # maximum length of the key phrases ...we have specified over here is 5

  # generate key phrases of lengths varying from 1 to 5
  for n in range(1, max_length+1):
    temp=[]
    for ngram in ngrams(tokens, n):
      temp.append(' '.join(ngram))
    key_phrases.append(temp)

  return key_phrases
```

```
generate_key_phrases("ONE TWO THREE FOUR FIVE")
```

```
[['one', 'two', 'three', 'four', 'five'],
 ['one two', 'two three', 'three four', 'four five'],
 ['one two three', 'two three four', 'three four five'],
 ['one two three four', 'two three four five'],
 ['one two three four five']]
```