|  |  |
| --- | --- |
| **Ex. 12** | **EXPLORING COMMAND-LINE ARGUMENTS** |
| **Date: 22/04/24** | |
|  | |

**Aim:**

To explore command-line arguments in Python by writing programs for the following:

1. Given a sentence as a command-line argument, compute the probability of the sentence being valid based on bigram probabilities computed in the previous exercise.
2. Given a file name as a command line argument, check if the file exists. If the file exists, read the contents of the file and display it. If the file does not exist, create the file and add some content to it. (Use try ... except ... else blocks.)
3. Given an integer as a command-line argument, print the list of prime numbers less than the given integer.

**Algorithm:**

(a)

STEP 1: Remove punctuations from the entire text file to normalize the data. Split the text into individual words.

STEP 2: Create a list of bigrams, which are consecutive word pairs, from the list of words.

STEP 3: Calculate the frequency of each bigram relative to the appearance of its first word, and store these probabilities in a dictionary. Each bigram probability is calculated as the count of the bigram divided by the count of its first word.

STEP 4: Read the input sentence, strip it of punctuations, and split it into words.

STEP 5: Initialize a variable `prob` to 1. For each consecutive word pair (bigram) in the sentence, multiply `prob` by the bigram's probability from the dictionary if it exists. If a bigram does not exist in the dictionary, multiply `prob` by 0, effectively setting the probability of the entire sequence to zero.

STEP 6: Output the computed probability of the sentence. This probability represents the likelihood of the sentence's word sequence based on the bigram frequencies in the provided text.

(b)

STEP 1: Retrieve the filename from the command-line arguments.

STEP 2: Attempt to open and read the contents of the specified file.

STEP 3: If the file does not exist (FileNotFoundError), create the file and write "some contents" to it.

STEP 4: Display the file contents if the file is read successfully.

(c)

STEP 1: Retrieve the upper limit N from the command-line arguments.

STEP 2: Iterate through each number from 2 to N .

STEP 3: Assume each number is prime.

STEP 4: Check if the current number is divisible by any smaller number (from 2 to the number minus one). If divisible, mark the number as not prime and break the loop.

STEP 5: If the number is prime, print it.

**Program:**

**(a)**

import sys

bigram=[]

bi\_prob={}

punctuations = ['.', ',', '?','!', ':', ';', "'", '"', '(', ')', '[', ']', '{', '}', '-', '—', '/', '&', '\*', '$', '%' ,'#', '@', '+', '=', '~', '^', '\\', '|', '•', '†', '‡', '“', '’', '”']

with open("text.txt",encoding="utf8") as text:

    file=text.read()

    for p in punctuations:

        file=file.replace(p,'')

words= file.split()

bigram=[(words[i].lower(),words[i+1].lower()) for i in range(len(words)-1)]

bi\_prob={(w1,w2):format(bigram.count((w1,w2))/words.count(w1),'.3f') for w1,w2 in set(bigram)}

prob=1

sentence=sys.argv[1]

for p in punctuations:

    sentence=sentence.replace(p,"")

sentence\_words=sentence.split()

for i in range(len(sentence\_words) - 1):

    pair = (sentence\_words[i].lower(), sentence\_words[i+1].lower())

    if pair in bi\_prob.keys():

        prob \*= bi\_prob[pair]

print(prob)

**(b)**

import sys

read\_file=sys.argv[1]

try:

    with open(read\_file,"r") as file:

        print(file.read())

except FileNotFoundError:

    with open(read\_file,"w") as file:

        file.write("some contents")

**(c)**

import sys

N = int(sys.argv[1])

for num in range(2, N + 1):

    prime = True

    for i in range(2, num):

        if num % i == 0:

            prime = False

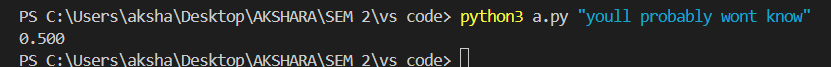
            break

    if prime:

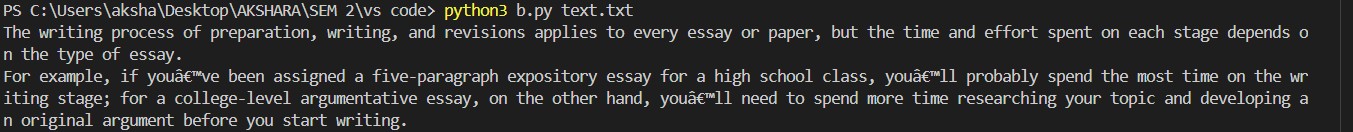
        print(num, end=' ')

**Screenshot of Output:**

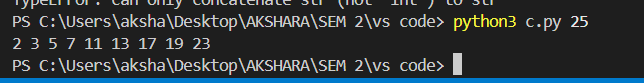
(a)

****

(b)



(c)



**Result:**

Thus, programs have been written and executed to explore command-line arguments in Python.