# IMPLEMENTATION OF CUT-UP METHOD-PYTHON

## PROBLEM IDENTIFICATION :

To implement the “cut-up” method using python programming language and create a python application which does the following task: getting input data and split it into columns with lines containing equal number of characters, then the split columns are rearranged according to the given input format of the columns and it should be joined to form a new rearranged text which is the required cut-up text and this is the required output of the python application. There are many ways by which this cut-up method can be implemented using python also merits and demerits always exist, we have to choose the best suitable method to implement the cut-up method.

## ANALYSIS OF THE PROBLEM :

1. The input should be fed into the program by an external input text file using open() function in read mode and the data in the file should be stored in a variable using file.readline() .
2. As per the user input the given text should be cut into equal and multiple columns(A,B,C,D,E,F) with lines having equal number of characters, as per the user input and the split columns are stored in a list .
3. As per the user given input of the order of columns the split columns should be iterated over and rearranged then it should be stored in a list.
4. There are two logics to combine the spliced columns. If Input is “ACB” directly join the text in each column. If input is “A+BC”, write the whole “A” column first and combine the rest of the columns.
5. The rearranged columns should be iterated over and joined using string concatination method.
6. The output file should be opened in writing mode and the final output that is the newly formed cut-up text should be written in the output file , after writing close the output file and terminate the program.

## POTENTIAL SOLUTIONS:

1. Use Dictionary to store the columns and respective number ,the lines in the input data is split into n columns and then textwrap.wrap() function is used to shape the split string such that each and every line contains the exact and equal number of characters,then the splitted columns are rearranged in the required order and written in the output text file.
2. Get the input data and then split it using “string methods” - string.split(“\n”) cutting lines and dividing the lines into needed number of columns and storing it as strings and as per the required order the strings can be rearranged and the joined using string concatination producing a the required output .
3. Get the input as a list and spit the list into number of columns needed and rearrange the columns as per the requirement and store it in a variable and open the output file and write the result in the output file and close the file .

## IMPLEMENTING AS PSEUDOCODE:

Here I used the first approach of using the wrap() function in the “textwrap” library to wrap the split text with required number of characters as per our requirement that is a metric of how many characters should be there in a line, and then the columns are iterated over and as per the required order of columns they can be joined and it can be written in the output file.

### Base algorithm:

1. Define the column-dictionary with column(A,B,C,D,E,F) as keys and their respective numbers as their values.

1. Open the input txt file in read mode and fetch the data from the file and store it in a list using readline() function. .
2. Iterate over the lines in the input and split the lines into n-columns as per the user given input data of number of columns during the program runtime.
3. While splitting, each line is applied a wrap- using the textwrap.wrap() that is every line must contain specified number of characters this is a metric for number of characters.
4. Open the output file with write permission and combine the splitted columns as per the user given input order of columns and write the final output in the output text file.
5. Close the output file after writing the output and terminate the program.

### Pseudocode:-

IMPORT textwrap

DEFINE FUNCTION main():

SET switcher TO {

'A': 1,

'B': 2,

'C': 3,

'D': 4,

'E': 5,

'F': 6

}

#Reading the INPUT values

SET INPUTdoc TO open(r"input.txt", 'rt')

SET splitexp TO INPUT('Enter the INPUT')

n=len(splitexp)

splitexpchararr=list(splitexp)

splittedstrings=[]

#Reading lines from file

SET doclines TO INPUTdoc.readlines()

#Split the lines IN the document into n columns

consol=[]

Nol=1

FOR line IN doclines:

SET noofchars TO int(len(line)/len(splitexp))

SET splittedstrings TO textwrap.wrap(line, noofchars)

if(len(splittedstrings) > len(splitexp)):

SET splittedstrings[n-1:n+1] TO [' '.join(splittedstrings[n-1:n+1])]

WHILE len(splittedstrings) < n:

splittedstrings.append(' ')

#============================================================

consol.append(splittedstrings)

Nol+=1

SET splitedexpression TO splitexp.split('+')

SET oFile TO open(r"op.txt",'w+')

SET output TO ''

# Combine the splited column as per INPUT and write it IN output text file

FOR i IN splitedexpression:

FOR row IN range(0,Nol-1):

FOR col IN i:

#OUTPUT(col)

temp=(consol[row][switcher[col]-1])

#temp1=switcher[col]-1

#temp=consol[row][temp1]

#OUTPUT(temp)

output=' '.join([output,temp])

oFile.write(output+'\n')

SET output TO ' '

oFile.close()

#============================================================#OUTPUT([' '.join(x) FOR x IN zip(splittedstrings[splitexparrindex[0]], splittedstrings[1])])

main()

## PYTHON CODE-IMPLEMENTATION:

#below code is the implementation of the pseudocode

import textwrap

def main():

switcher = {

'A': 1,

'B': 2,

'C': 3,

'D': 4,

'E': 5,

'F': 6

}

#Reading the input values

inputdoc = open(r"input.txt", 'rt')

splitexp = input('Enter the input')

n=len(splitexp)

splitexpchararr=list(splitexp)

splittedstrings=[]

#Reading lines from file

doclines = inputdoc.readlines()

#Split the lines in the document into n columns

consol=[]

Nol=1

for line in doclines:

noofchars = int(len(line)/len(splitexp))

splittedstrings = textwrap.wrap(line, noofchars)

if(len(splittedstrings) > len(splitexp)):

splittedstrings[n-1:n+1] = [' '.join(splittedstrings[n-1:n+1])]

while len(splittedstrings) < n:

splittedstrings.append(' ')

consol.append(splittedstrings)

Nol+=1

splitedexpression = splitexp.split('+')

oFile = open(r"op.txt",'w+')

output = ''

# Combine the splited column as per input and write it in output text file

for i in splitedexpression:

for row in range(0,Nol-1):

for col in i:

#print(col)

temp=(consol[row][switcher[col]-1])

#temp1=switcher[col]-1

#temp=consol[row][temp1]

#print(temp)

output=' '.join([output,temp])

oFile.write(output+'\n')

output = ' '

oFile.close()

#============================================================

#Merge the columns into given forms to make a new paragraph

#print([' '.join(x) for x in zip(splittedstrings[splitexparrindex[0]], splittedstrings[1])])

main()

## EVALUATION:

This program uses text-wrap function so the lines are exactly splitted into required number of characters so the output looks even and neat. This program is efficient in the sense of any given input it produces a good and desired output below is an demo for evaluation.

For the purpose of evaluation lets take a random text,

#### INPUT:

“The [real number](https://en.wikipedia.org/wiki/Real_number) denoted by the [recurring decimal](https://en.wikipedia.org/wiki/Recurring_decimal) [**0.999…**](https://en.wikipedia.org/wiki/0.999%E2%80%A6) is exactly [equal](https://en.wikipedia.org/wiki/Equality_(mathematics)) to [1](https://en.wikipedia.org/wiki/1_(number)). In other words, "0.999…" represents the same number as the symbol "1". Various [proofs](https://en.wikipedia.org/wiki/Mathematical_proof) of this identity have been formulated with varying [rigour](https://en.wikipedia.org/wiki/Rigour" \l "Mathematical_rigour" \o "Rigour), preferred development of the real numbers, background assumptions, historical context, and target audience. The equality has long been taught in textbooks, and in the last few decades, researchers of [mathematics education](https://en.wikipedia.org/wiki/Mathematics_education) have studied the reception of this [equation](https://en.wikipedia.org/wiki/Equation) among students, who often reject the equality. The students' reasoning is typically based on one of a few common erroneous intuitions about the real numbers; for example, a belief that each unique [decimal expansion](https://en.wikipedia.org/wiki/Decimal_expansion) must correspond to a unique number, an expectation that [infinitesimal](https://en.wikipedia.org/wiki/Infinitesimal) quantities should exist, that [arithmetic](https://en.wikipedia.org/wiki/Arithmetic) may be broken, an inability to understand [limits](https://en.wikipedia.org/wiki/Limit_(mathematics)) or simply the belief that 0.999… should have a last 9. These ideas are false with respect to the real numbers, which can be proven by explicitly constructing the reals from the [rational numbers](https://en.wikipedia.org/wiki/Rational_number), and such constructions can also prove that 0.999… = 1 directly.”

The column order input is given as : “ADBCFE”

#### OUTPUT:

“ The real number denoted by the recurring decimal 0.999… is exactly equal to 1. In other words, "0.999…" represents the same number as the symbol "1". Various proofs of this students' reasoning is typically based on one of a few common erroneous intuitions about the real numbers; for example, a belief that each unique decimal expansion must correspond to a identity have been formulated with varying rigour, preferred development of the real numbers, background assumptions, historical context, and target audience. The equality has long been taught in textbooks, and in the last few decades, researchers of mathematics education have studied the reception of this equation among students, who often reject the equality. The a last 9. These ideas are false with respect to the real numbers, which can be proven by explicitly constructing the reals from the rational numbers, and such constructions can also prove that 0.999… = 1 directly unique number, an expectation that infinitesimal quantities should exist, that arithmetic may be broken, an inability to understand limits or simply the belief that 0.999… should have”

## PROBLEMS AND ISSUES:

1. **Issues-** At first I did not implement the text-wrap function When I did directly I faced a problem with uneven splitting of the lines in the input text and so the columns split will have uneven lines with difference in the number of characters in a line .

**Solution -** That’s why I have used Text-wrap () function to exactly wrap the line as per the requirement.

1. **Issues-** I used dictionary to store the splitted columns and then when I have to write the output in the output text file I have to call the key’s in the dictionary in the order of the users input this seems to be not simple so I implemented a simpler method like direct application with the list itself.

**Inference:**List can also be implemented instead of dictionary instead of key in case of the dictionary we can easily pass in the index in the case of the list.

## DEMONSTRATION:

## TEST-CASE-:- recommended test case.

## Input.txt:-Alice in Wonderland a paragraph from chapter 1

Suddenly she came upon a little three-legged table, all made of solid glass; there was nothing on it except a tiny golden key, and Alice's first thought was that it might belong to one of the doors of the hall; but, alas! either the locks were too large, or the key was too small, but at any rate it would not open any of them. However, on the second time round, she came upon a low curtain she had not noticed before, and behind it was a little door about fifteen inches high: she tried the little golden key in the lock, and to her great delight it fitted!

## Runtime column order input:-”FABECD”

### Output.txt:-

fifteen inches high: she tried the little golden key in the lock, and to her great delight it fitted! Suddenly she came upon a little three-legged table, all made of solid glass; there was nothing on it except a tiny golden key, and Alice's first thought was that it might belong to came upon a low curtain she had not noticed before, and behind it was a little door about one of the doors of the hall; but, alas! either the locks were too large, or the key was too small, but at any rate it would not open any of them. However, on the second time round, she

#### Conclusion:

we can see that the given input text file is splitted into the six columns(A,B,C,D,E,F) and then it is rearranged as per the given input order of columns(FABECD) and then it is written in the output text file also note that the lines in the columns have equal number of characters since it used textwrap.wrap() function.

## Libraries used:

This are the libraries and modules which are used in the program,

### Textwrap:

The textwrap python library provides some cool functions which can be used to deal with wrapping the text in a paragraph so as to get well shaped and clean and neat text.

**Functions-used: Textwrap.wrap(para,width)**

The wrap function can be used to reshape the lines in a paragraph so that every line in the paragraph looks even and has equal number of words in every line. It needs two arguments to be passed, they are para-the text or paragraph which needs to be reshaped , width-This argument represents the maximum number of character that a line must contain in the paragraph. This function returns a text with all the lines having equal number of words . It also removes the newline at the end.