**AA\_LAB\_04\_Assignment**

**CE\_054**

**Aim :-** Implementation of Rabin-Karp String Matching Algorithm!

1. Implementation of Rabin-Karp String matching Algorithm using python.

Code: -

# -\*- coding: utf-8 -\*-

"""

Created on Fri Aug 7 19:49:35 2020

@author: DHRUV

"""

total\_char = 256

def search(pat, txt, q):

len\_pattern = len(pat)

len\_string = len(txt)

flag, j, value, temp, hash\_ = 0, 0, 0, 0, 1

for i in range(len\_pattern-1):

hash\_ = (hash\_ \* total\_char) % q

for i in range(len\_pattern):

value = (total\_char \* value + ord(pat[i])) % q

temp = (total\_char \* temp + ord(txt[i])) % q

for i in range(len\_string - len\_pattern + 1):

if value == temp:

for j in range(len\_pattern):

if txt[i + j] != pat[j]:

break

j += 1

if j == len\_pattern:

print (f"Pattern found at index {i}" )

flag = 1

if i < len\_string - len\_pattern:

temp = (total\_char \* (temp - ord(txt[i]) \* hash\_) + ord(txt[i +

len\_pattern])) % q

if temp < 0:

temp = temp + q

else:

if (flag == 0):

print(“pattern not found!”)

if \_\_name\_\_ == "\_\_main\_\_":

txt = input("Enter the string : ")

pat = input("Enter the pattern : ")

q = 103

search(pat,txt,q)

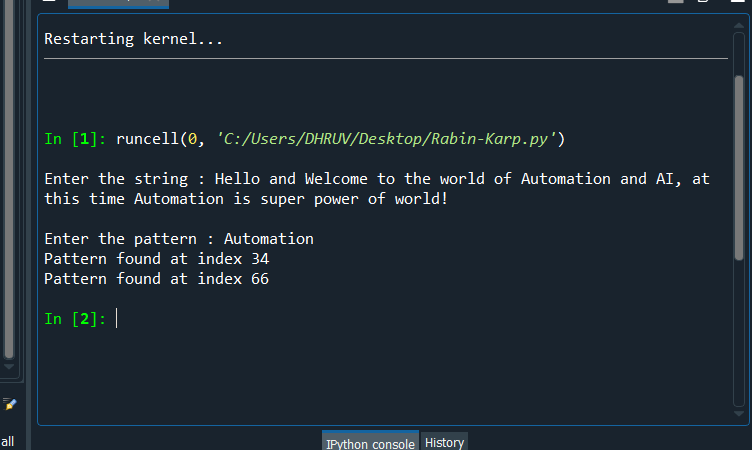
**Output :-**

Enter the string : Hello and Welcome to the world of Automation and AI, at this time Automation is super power of world!

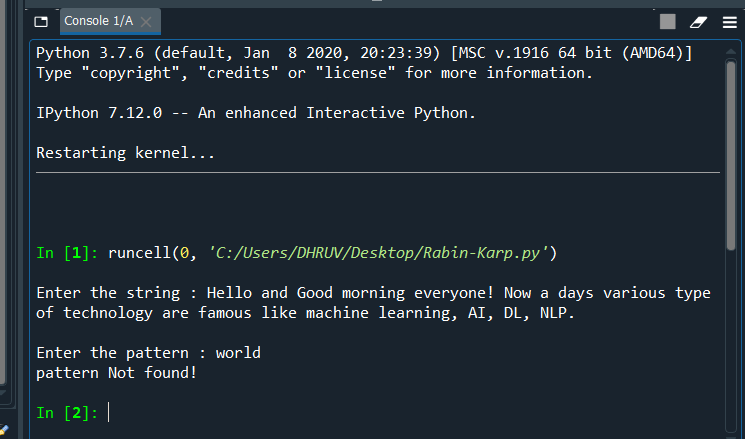
Enter the pattern : Automation

Pattern found at index 34

Pattern found at index 66



* If pattern is not present in String then :-



**About Algorithm:-**

The **Rabin-Karp** algorithm is a [string](https://brilliant.org/wiki/strings/)-searching algorithm that uses [hashing](https://brilliant.org/wiki/hash-based-data-structure/) to find patterns in strings. A string is an [abstract data type](https://brilliant.org/wiki/abstract-data-types/) that consists of a sequence of characters. Letters, words, sentences, and more can be represented as strings.

The Rabin-Karp algorithm makes use of [hash functions](https://brilliant.org/wiki/hash-based-data-structure/) and the [rolling hash](https://brilliant.org/wiki/rolling-hash/) technique. A hash function is essentially a function that maps one thing to a value. In particular, hashing can map data of arbitrary size to a value of fixed size.

Assume the text is length n and the length of the pattern is m. The best and average case running time of Rabin-Karp is O(m + n).

The worst-case running time of Rabin-Karp is O(nm). This would occur with an extremely awful hash function that resulted in a false positive at each step. Since whenever the algorithm thinks it found a match, it must verify each of the m letters in the pattern, if there is a collision at each step, m letters will be checked n times resulting in a running time of O(nm) This can be avoided with a good choice of hash function.