

Assignment-8

ELP - 718 Telecom Software Laboratory

Pardhi Chandan Waman

2018JTM2248

2018-2020

A report presented for the assignment on
Python and Github



Bharti School Of
Telecommunication Technology and Management
IIT Delhi
India
September 27, 2018

Contents

1	Problem Statement 1	4
1.1	Problem Statement	4
1.2	Program Structure	4
1.3	Algorithm and Implementation	5
1.4	Input and Output format	5
1.5	Test Cases	5
1.6	Difficulties/Issues Faced	6
1.7	Screen-shots	7
2	Problem Statement 2	8
2.1	Problem Statement	8
2.2	Assumptions	8
2.3	Program Structure	9
2.4	Algorithm and Implementation	10
2.5	Input and Output format	10
2.6	Test Cases	10
2.7	Difficulties/Issues Faced	10
2.8	Screen-shots	11
3	Appendix	12
3.1	Appendix-A: code for ps1	12
3.2	Appendix-B:code for ps2	15

List of Figures

1	flowchart for ps1	4
2	Output for ps1	7
3	flowchart for ps1	9
4	ps2 - Output on Terminal	11

1 Problem Statement 1

1.1 Problem Statement

IIT Delhi, has just got the strongest computer. The professors in charge wants to check the computational capacity of the computer. So, they decided to create the problem which is to be given as an assignment to students. Can you help the professor to check the computation capability of the computer?

A valid cross is defined here as the two regions (horizontal and vertical) of equal lengths crossing over each other. These lengths must be odd, and the middle cell of its horizontal region must cross the middle cell of its vertical region.

1.2 Program Structure

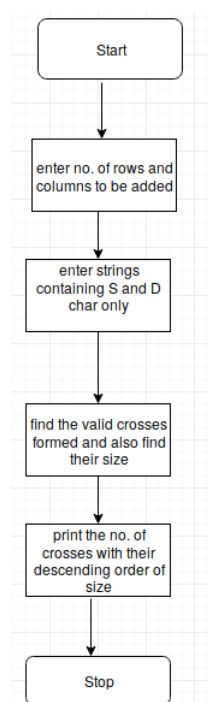


Figure 1: flowchart for ps1

1.3 Algorithm and Implementation

- create ps1.py to write a code for given problem statement.
- take m n i.e. no of rows and col from user
- take array of 'S' and 'D' from user.
- find the pattern using a code.
- output the largest two matching pattern diamentions
- stop.

1.4 Input and Output format

- **Input Format**

The first line contains two space-separated integers, n and m. Each of the next lines n contains a string of m characters where each character is either S (Smart) or D (Dull). These strings represent the rows of the grid. If the jth character in the ith line is S, then (i,j) is a cell smart. Otherwise it's a dull cell

Constraints

$$2 \leq n \leq 105$$

$$2 \leq m \leq 105$$

- **Output Format**

Find two valid crosses that can be drawn on smart cell of the grid, and return the dimension of both the crosses in the reverse sorted order(i.e. First Dimension should be the larger one and other should be smaller one).

1.5 Test Cases

1. 5 6
SSSSSS
SDDDS
SSSSSS

SSDDSD
SSSSSS

2. 6 6
DSDDSD
SSSSSS
DSDDSD
SSSSSS
DSDDSD
DSDDSD

1.6 Difficulties/Issues Faced

In building a logic for the problem statement. Actual implementation in code was difficult to achieve.

1.7 Screen-shots



```
chandan@pc13-OptiPlex-9020:~/Desktop/assignment-8$ ./ps1.py
Enter m and n
5 6
SSSSSS
SDDSD
SSSSSS
SDDSD
SSSSSS
Output
5 1
chandan@pc13-OptiPlex-9020:~/Desktop/assignment-8$ |
```

Figure 2: Output for ps1

2 Problem Statement 2

2.1 Problem Statement

After, getting mix results of valid crosses, professors decided to test the computation abilities on one more problem. This time professors wanted to test the decryption capabilities of the computer. Encryption of a message requires three keys, k_1 , k_2 , and k_3 . The 26 letters of English and underscore are divided in three groups, [a-i] form one group, [j-r] a second group, and everything else ([s-z] and underscore) the third group. Within each group the letters are rotated left by k_i positions in the message. Each group is rotated independently of the other two. Decrypting the message means doing a right rotation by k_i positions within each group.

2.2 Assumptions

Input string and keys to decrypt a string is available with the user decrypting a string.

2.3 Program Structure

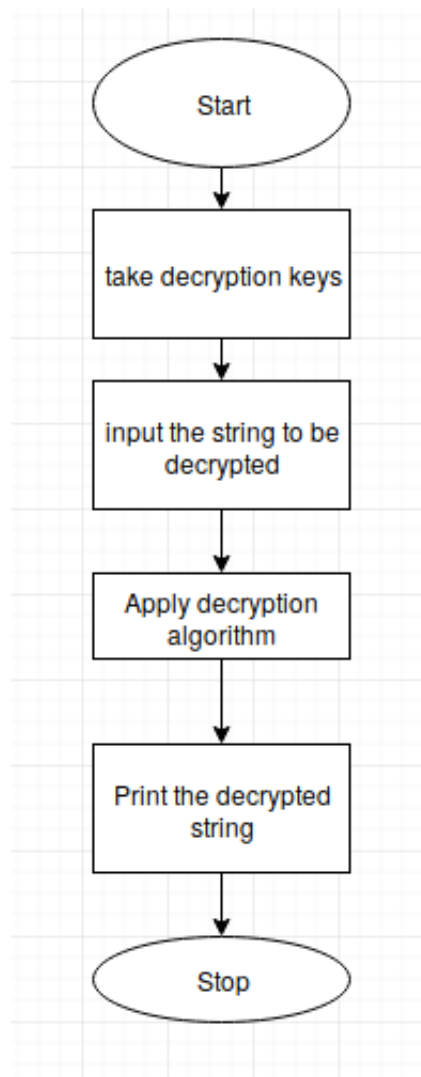


Figure 3: flowchart for ps1

2.4 Algorithm and Implementation

- Created ps2.py to write a code for decryption algorithm
- key and encrypted string is taken from user.
- Decryption algorithm is applied as given in the problem statement.
- input Encrypted string along with a key gives Original Decrypted string.

2.5 Input and Output format

- **Input Format**
Input is key and Encrypted string.
- **Output Format**
Output is Decrypted String

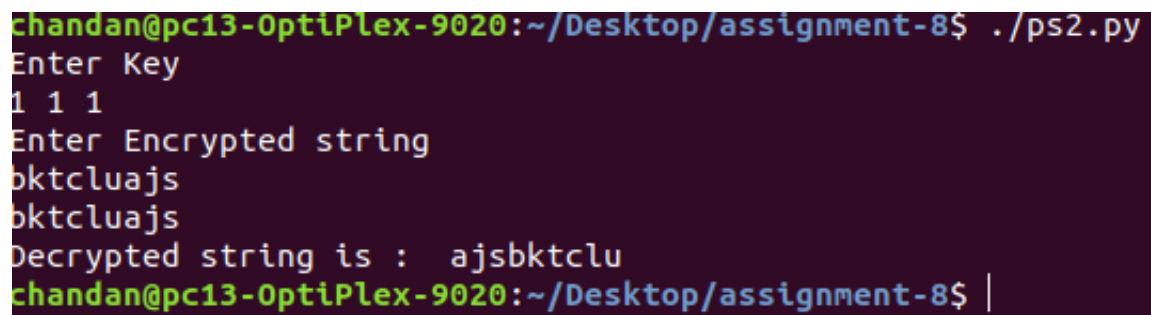
2.6 Test Cases

1. Input 1 1 1
bktcluaajs
2. Output
ajsbktclu

2.7 Difficulties/Issues Faced

Proper output is not obtained for some other random strings.

2.8 Screen-shots



```
chandan@pc13-OptiPlex-9020:~/Desktop/assignment-8$ ./ps2.py
Enter Key
1 1 1
Enter Encrypted string
bktcluajs
bktcluajs
Decrypted string is : ajsbktclu
chandan@pc13-OptiPlex-9020:~/Desktop/assignment-8$ |
```

Figure 4: ps2 - Output on Terminal

3 Appendix

3.1 Appendix-A: code for ps1

```
#!/usr/bin/python3

##### this is the first .py file #####

##### write your code here #####

# ps1.py is to check the valid crosses
def find_pattern(list1 , i, j):
    countr = list()
    # keeping i and j stored in another variables to recover them
    afterwards.
    a=i
    b=j
    # initialising counters for different directions

    count1=0
    count2=0
    count3=0
    count4=0
    count=0

    #print(list1)
    #print(i)
    #print(j)
    #tracing around the given element in a list.
    if list1[i][j]=='S':
        count=1
        while (j<(m-1)):
            j=j+1
            if list1[i][j]=='S':
                count1=count1+1
            else:
                break

        i=a
        j=b

    while j>0:
```

```

        j=j-1
        if list1[i][j] == 'S':
            count2 = count2+1
        else:
            break
    i=a
    j=b
    while i<(n-1):
        i=i+1
        if list1[i][j] == 'S':
            count3 = count3 +1
        else:
            break
    i=a
    j=b
    while i>0:
        i=i-1
        if list1[i][j] == 'S':
            count4 = count4+1
        else:
            break
    i=a
    j=b

    else:
        count=0
        countr = [count1 ,count2 ,count3 ,count4] # list of all the four
        counters.
        min_count=min(countr)
        count = (min_count*4)+1
        return count
print("Enter m and n")
n1,n2=input().split(" ")
n=int(n1)
m=int(n2)
list1=[]
count_array = ""
# Input a Array of S and D from user adn store it in list1
for index in range(n):
    s=list(input())

```

```

list1 += [s]

for i in range(n):
    for j in range(m):
        #print(i,j)
        count = find_pattern(list1 , i , j)
        count_array=count_array + str(count)
#print(count_array)
final_count = list(count_array)
max1=max(final_count)
final_count.remove(max(final_count))
max2=max(final_count)
print("Output")
print(max1,max2)

```

3.2 Appendix-B:code for ps2

```
#!/usr/bin/python3
##### this is the second .py file #####

##### write your code here #####
#function definition to rotate a string d elemets to right
def rotate_right(array,d):
    r1=array[0:len(array)-d] # taking first n-d letters
    r2=array[len(array)-d:] # last d letters
    rotate = r2+r1 # reversed the order
    return rotate #return ststatement

decrypted="" # decrypted string will be stored
             here
#k1=int(input("Enter the amount by which key1 elemets to be rotated\n
            Decryption key1 = : "))
#k2=int(input("\nDecryption key2 = : "))
#k3=int(input("\nDecryption key3 = : "))
print("Enter _Key")
j1,j2,j3 =input().split("_")
k1=int(j1)
k2=int(j2)
k3=int(j3)
quer_str = input("Enter _Encrypted_string\n")
print(quer_str)
alphabets="abcdefghijklmnopqrstuvwxyz_"
alphabets1=alphabets[0:9]
alphabets2=alphabets[9:18]
alphabets3=alphabets[18:27]
# Declaring Strings to store different key characters
key1=""
key2=""
key3=""
# Seperating keys for different range
for i in quer_str :
    for j in alphabets1:
        if i==j :
            key1 = key1 + str(i)
```

```

for k in alphabets2:
    if i==k :
        key2 = key2 + str(i)

for l in alphabets3:
    if i==l:
        key3 = key3 + str(i)

# keys sorted according to input numbers by which they are to be
  shifted
new_k1=rotate_right(key1,k1)
new_k2=rotate_right(key2,k2)
new_k3=rotate_right(key3,k3)
index1=0
index2=0
index3=0
# Decrypting a string and printing original decrypted string
for i in quer_str:
    for j in new_k1 :
        if i==j:
            decrypted=decrypted+new_k1[index1]
            index1 = index1+1

    for k in new_k2 :
        if i==k :
            decrypted=decrypted+new_k2[index2]
            index2=index2+1

    for l in new_k3 :
        if i==l :
            decrypted=decrypted+new_k3[index3]
            index3=index3+1

print("Decrypted string is :",decrypted)

```


References

- [1] *Python 3.7 documentation*, <https://docs.python.org/3/>.
- [2] *Python3 tutorial*, <https://www.tutorialspoint.com/python3/index.htm>.
- [3] *Run python in linux(ubuntu)*, Oct 2015, <https://www.youtube.com/watch?v=3xp-ixFbDuE>.
- [4] Paul Barry, *Head first python*, <https://doc.lagout.org/programmation/python/Head%20First%20Python%2C%20First%20Edition%20%282010%29.pdf>.
- [5] Bitbucket, *Become a git guru*, <https://www.atlassian.com/git/tutorials>.
- [6] Microwave Sam, *How to Get Started with Github-Beginer*, <https://www.youtube.com/watch?v=73I5dRucCds&spfreload=5>.