MCKV INSTITUTE OF ENGINEERING

Computer Science and Engineering – Data Science

***Name - Abhinaba Sarkar***

***Roll No. - BTECH/CSE-DS/2020/47***

**Assignment Number:** 8

**Problem Statement:**

Take a string as input. Form a dictionary which will have each unique word present in the string as key and frequency of the word as value.Original string: Python is inspired by Monty Python.Printing count of each item {“Python”:2, “is”: 1, “inspired”: 1, “by”: 1, “Monty”:1}

**Source code:**

*str=input('enter a string:')*

*l1=[]*

*l2=[]*

*l3=str.split()*

*for n in l3:*

*if n.isalpha():*

*l1.append(n)*

*l2.append(l3.count(n))*

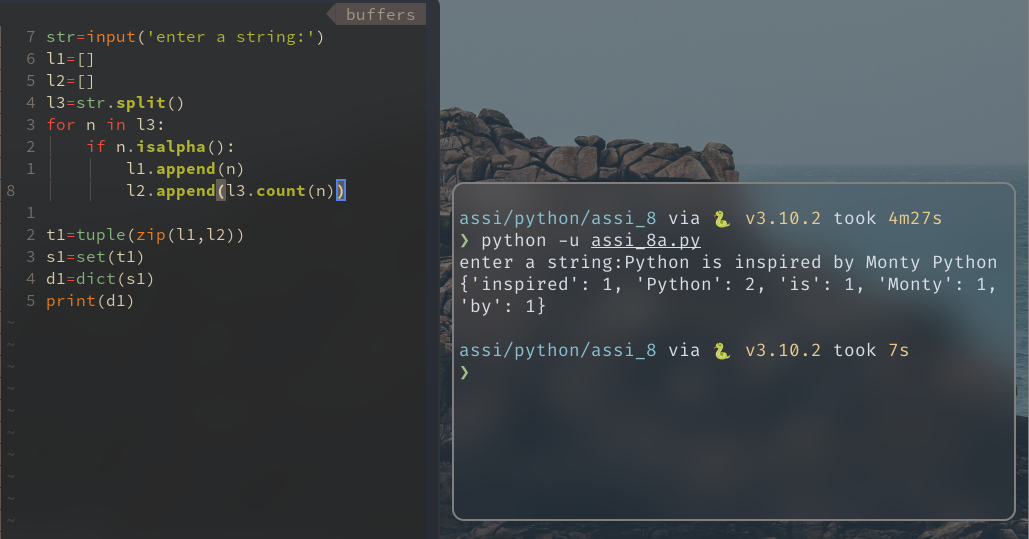
*t1=tuple(zip(l1,l2))*

*s1=set(t1)*

*d1=dict(s1)*

*print(d1)*

**OUTPUT:**



**Assignment Number: 8(B)**

**Problem Statement:**

A string with parentheses is well bracketed if all parentheses are matched: every opening bracket has a matching closing bracket and vice versa. Write a Python program using function to check an expression is well bracketed or not.

**Source code:**

*def wellbrackted(exp):*

*stck=[]*

*for i in exp:*

*if i =='(':*

*stck.append('(')*

*if i ==')':*

*try:*

*stck.pop()*

*except IndexError:*

*return False*

*if len(stck)== 0:*

*return True*

*else:*

*return False*

*exp=input('enter an expression:')*

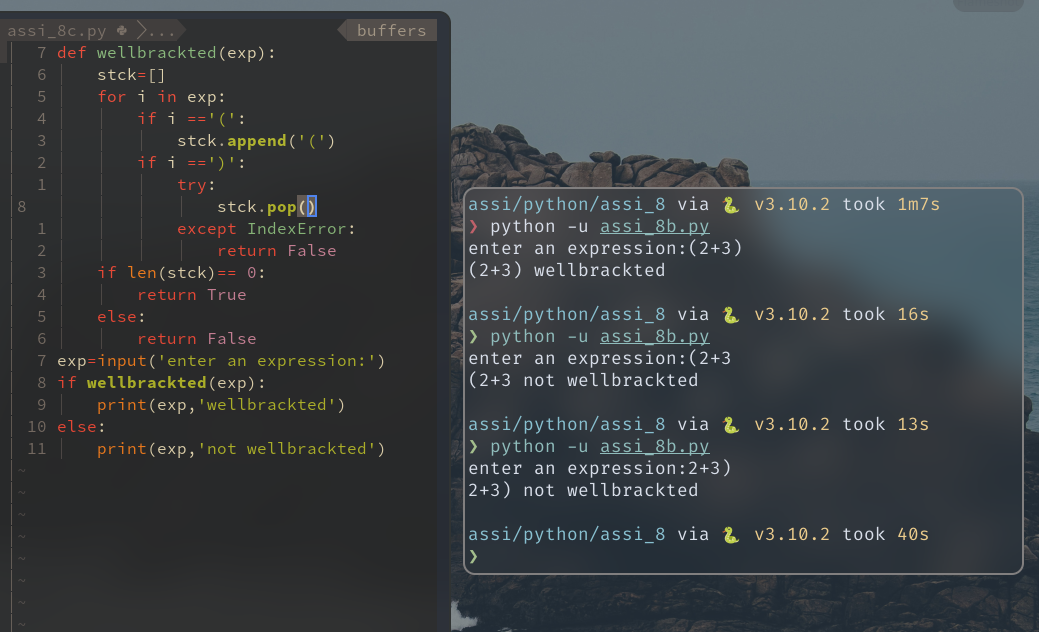
*if wellbrackted(exp):*

*print(exp,'wellbrackted')*

*else:*

*print(exp,'not wellbrackted')*

**OUTPUT:**



**Assignment Number: 8(C)**

**Problem Statement:**

Write ratotalist(), Python function which will take 3 paramaters as input . 1st parameter is a list which to be rotated, 2nd parameter will be the number times it will be rotated and 3rd one will the direction (“Left” / “Right”). The function will return a new rotated list. Number of rotation must be less than the number of element in the list, else it will throw a user define exception and return the original list. By default number of rotation will be 1 and direction of rotation will be “Left”.

Here are some examples to show how your function should work.

>>> rotatelist([1,2,3,4,5], 1,”Right”)

[5, 1, 2, 3, 4]

>>> rotatelist([1,2,3,4,5], 3, “Left”)

[4, 5, 1, 2, 3]

**Source code:**

*class MyException(Exception):*

*pass*

*def rotatelist(l,n,d):*

*l1=[]*

*try:*

*if n >= len(l):*

*raise MyException*

*if d== "Left":*

*l1=l[n:len(l)]+l[0:n]*

*print('after rotation:',l1)*

*if d== "Right":*

*n=len(l)-n*

*l1=l[n:len(l)]+l[0:n]*

*print('after rotation:',l1)*

*except MyException:*

*print('The number of rotation is greater than the number of elememt')*

*l1=l.copy()*

*finally:*

*return l1*

*lst=[]*

*l=int(input('enter elements no:'))*

*for i in range(l):*

*l1=int(input('Enter element no {}:'.format(i+1)))*

*lst.append(l1)*

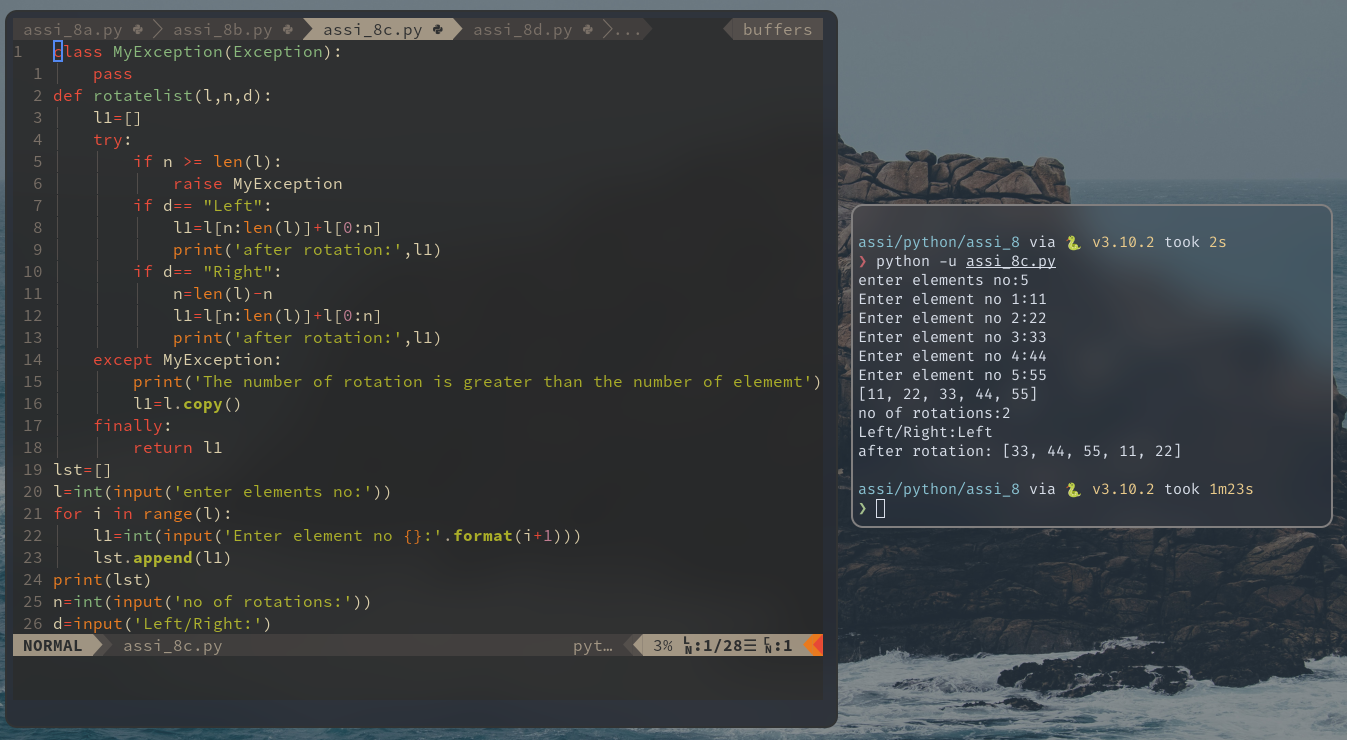
*print(lst)*

*n=int(input('no of rotations:'))*

*d=input('Left/Right:')*

*rotatelist(lst,n,d)*

**OUTPUT:**



**Assignment Number: 8(D)**

**Problem Statement:**

Write a recursive Python code to implement a soltions for Tower of Hanio for ‘n’ disk.

**Source code:**

*def towers(n,source,destination,auxiliary):*

*if n>0:*

*towers(n-1,source,auxiliary,destination)*

*print('move disk {} from peg {} to peg {}'.format(n,source,destination))*

*towers(n-1,auxiliary,destination,source)*

*n=int(input('enter disk no:'))*

*if n==0:*

*print('not possible!!!')*

*else:*

*towers(n,'A','C','B')*

**OUTPUT:**

