Perfect eLearning

Learn & Grow become leaders of tomorrow

Problem 1 : Python program to interchange first and last elements in a list

Examples:

Input : [12, 35, 9, 56, 24]

Output : [24, 35, 9, 56, 12]

Input : [1, 2, 3]

Output : [3, 2, 1]

Solution 1 : Python program to interchange first and last elements in a list

```
# Python3 program to swap first
# and last element of a list

# Swap function
def swapList(newList):
        size = len(newList)

        # Swapping
        temp = newList[0]
        newList[0] = newList[size - 1]
        newList[size - 1] = temp

        return newList

# Driver code
newList = [12, 35, 9, 56, 24]

print(swapList(newList))
```

Solution 2 : Python program to interchange first and last elements in a list

```
# Python3 program to swap first
# and last element of a list

# Swap function
def swapList(newList):

    newList[0], newList[-1] = newList[-1], newList[0]

    return newList

# Driver code
newList = [12, 35, 9, 56, 24]
print(swapList(newList))
```

Solution 3 : Python program to interchange first and last elements in a list

```
# Python3 program to swap first
# and last element of a list

# Swap function
def swapList(list):

    # Storing the first and last element
    # as a pair in a tuple variable get
    get = list[-1], list[0]

    # unpacking those elements
    list[0], list[-1] = get

    return list

# Driver code
newList = [12, 35, 9, 56, 24]
print(swapList(newList))
```

Solution 4 : Python program to interchange first and last elements in a list

```
# Python3 program to swap first
# and last element of a list

# Swap function
def swapList(list):

        first = list.pop(0)
        last = list.pop(-1)

        list.insert(0, last)
        list.append(first)

        return list

# Driver code
newList = [12, 35, 9, 56, 24]

print(swapList(newList))
```

Problem 2 : Python program to check if a string is palindrome or not

Examples:

Input : malayalam
Output : Yes

Input : geeks
Output : No

Solution 1 : Python program to check if a string is palindrome or not

```
# function which return reverse of a string

def isPalindrome(s):
        return s == s[::-1]


# Driver code
s = "malayalam"
ans = isPalindrome(s)

if ans:
        print("Yes")
else:
        print("No")
```

Solution 2 : Python program to check if a string is palindrome or not

```
# function to check string is
# palindrome or not
def isPalindrome(str):

        # Run loop from 0 to len/2
        for i in xrange(0, len(str)/2):
                if str[i] != str[len(str)-i-1]:
                        return False
        return True

# main function
s = "malayalam"
ans = isPalindrome(s)

if (ans):
        print("Yes")

else:
        print("No")
```

Solution 3 : Python program to check if a string is palindrome or not

```
# function to check string is
# palindrome or not
def isPalindrome(s):

        # Using predefined function to
        # reverse to string print(s)
        rev = ''.join(reversed(s))

        # Checking if both string are
        # equal or not
        if (s == rev):
                return True
        return False

# main function
s = "malayalam"
ans = isPalindrome(s)

if (ans):
        print("Yes")
else:
        print("No")
```

Solution 3 : Python Program for factorial of a number

```
# Python 3 program to find
# factorial of given number

def factorial(n):

        # single line to find factorial
        return 1 if (n==1 or n==0) else n * factorial(n - 1)


# Driver Code
num = 5
print ("Factorial of",num,"is",
        factorial(num))
```

Problem 3 : Program to create grade calculator in Python

Given different scored marks of students.

We need to find grades. The test score is an average of the respective marks scored in assignments, tests and lab-works. The final test score is assigned using below formula.

```
10 % of marks scored from submission of Assignments
70 % of marks scored from Test
 20 % of marks scored in Lab-Works
```

Grade will be calculated according to :

```
1. score >= 90 : "A"
2. score >= 80 : "B"
3. score >= 70 : "C"
4. score >= 60 : "D"
```

Also, calculate the total class average and letter grade of class.

Solution : Program to create grade calculator in Python (on IDE)

```
# Python code for the Grade
# Calculator program in action

# Creating a dictionary which
# consists of the student name,
# assignment result test results
# and their respective lab results

# 1. Jack's dictionary
jack = { "name":"Jack Frost",
                "assignment" : [80, 50, 40, 20],
                "test" : [75, 75],
                "lab" : [78.20, 77.20]
        }

# 2. James's dictionary
james = { "name":"James Potter",
                "assignment" : [82, 56,
44, 30],
                "test" : [80, 80],
                "lab" : [67.90, 78.72]
                }

# 3. Dylan's dictionary
dylan = { "name" : "Dylan Rhodes",
                "assignment" : [77, 82, 23,
39],
                "test" : [78, 77],
                "lab" : [80, 80]
                }

# 4. Jessica's dictionary
jess = { "name" : "Jessica Stone",
                "assignment" : [67, 55, 77,
21],
                "test" : [40, 50],
                "lab" : [69, 44.56]
        }
```

```
# 5. Tom's dictionary
tom = { "name" : "Tom Hanks",
                "assignment" : [29, 89,
60, 56],
                "test" : [65, 56],
                "lab" : [50, 40.6]
        }

# Function calculates average
def get_average(marks):
        total_sum = sum(marks)
        total_sum = float(total_sum)
        return total_sum /
len(marks)

# Function calculates total
average
def
calculate_total_average(students):
        assignment =
get_average(students["assignme
nt"])
        test =
get_average(students["test"])
        lab =
get_average(students["lab"])

        # Return the result based
        # on weightage supplied
        # 10 % from assignments
        # 70 % from test
        # 20 % from lab-works
        return (0.1 * assignment +
                        0.7 * test + 0.2 *
lab)


# Calculate letter grade of each
student
def assign_letter_grade(score):
        if score >= 90: return "A"
        elif score >= 80: return "B"
        elif score >= 70: return "C"
        elif score >= 60: return "D"
        else : return "E"
```

```
# Function to calculate the total
# average marks of the whole
class
def class_average_is(student_list):
	result_list = []

	for student in student_list:
		stud_avg =
calculate_total_average(student)
		result_list.append(stud
_avg)
		return
get_average(result_list)

# Student list consisting the
# dictionary of all students
students = [jack, james, dylan, jess,
tom]

# Iterate through the students list
# and calculate their respective
# average marks and letter grade
for i in students :
	print(i["name"])
print("=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+=+=")
	print("Average marks of %s is
: %s " %(i["name"],
																						c
alculate_total_average(i)))

	print("Letter Grade of %s is :
%s" %(i["name"],
	assign_letter_grade(calculat
e_total_average(i))))

	print()


# Calculate the average of whole
class
class_av =
class_average_is(students)

print( "Class Average is %s"
%(class_av))
print("Letter Grade of the class is
%s "
			%(assign_letter_grade(
class_av)))
```

Problem 4 : Python program to create a list of tuples from given list having number and its cube in each tuple

Example:

```
Input: list = [1, 2, 3]
Output: [(1, 1), (2, 8), (3, 27)]

Input: list = [9, 5, 6]
 Output: [(9, 729), (5, 125), (6,

 216)]
```

Solution : Python program to create a list of tuples from given list having number and its cube in each tuple

```
# Python program to create a list of tuples
# from given list having number and
# its cube in each tuple

# creating a list
list1 = [1, 2, 5, 6]

# using list comprehension to iterate each
# values in list and create a tuple as specified
res = [(val, pow(val, 3)) for val in list1]

# print the result
print(res)
```

Problem 5 : Python | Print an Inverted Star Pattern

Examples:

```
Below is the inverted star pattern of size
n=5
  (Because there are 5 horizontal lines
   or rows consist of stars).
```

```
*****
 ****
  ***
   **
    *
```

Solution : Python | Print an Inverted Star Pattern

```
# python 3 code to print inverted star
# pattern

# n is the number of rows in which
# star is going to be printed.
n=11

# i is going to be enabled to
# range between n-i t 0 with a
# decrement of 1 with each iteration.
# and in print function, for each
iteration,
# ” ” is multiplied with n-i and ‘*’ is
# multiplied with i to create correct
# space before of the stars.
for i in range (n, 0, -1):
        print((n-i) * ' ' + i * '*')
```

Problem 6 : Python program to swap two elements in a list

Examples:

```
Input : List = [23, 65, 19, 90], pos1 = 1, pos2=3

Output : [19, 65, 23, 90]
```

```
Input : List = [1, 2, 3, 4, 5], pos1 = 2, pos2 = 5

Output : [1, 5, 3, 4, 2]
```

Solution : Python program to swap two elements in a list

```
# Python3 program to swap elements
# at given positions

# Swap function
def swapPositions(list, pos1, pos2):

        list[pos1], list[pos2] = list[pos2],
list[pos1]
        return list

# Driver function
List = [23, 65, 19, 90]
pos1, pos2 = 1, 3

print(swapPositions(List, pos1-1, pos2-1))
```

Solution : Python program to swap two elements in a list

```
# Python3 program to swap elements
# at given positions

# Swap function
def swapPositions(list, pos1, pos2):

        # popping both the elements from list
        first_ele = list.pop(pos1)
        second_ele = list.pop(pos2-1)

        # inserting in each others positions
        list.insert(pos1, second_ele)
        list.insert(pos2, first_ele)

        return list

# Driver function
List = [23, 65, 19, 90]
pos1, pos2 = 1, 3

print(swapPositions(List, pos1-1, pos2-1))
```

Solution : Python program to swap two elements in a list

```
# Python3 program to swap elements at
# given positions

# Swap function
def swapPositions(list, pos1, pos2):

        # Storing the two elements
        # as a pair in a tuple variable get
        get = list[pos1], list[pos2]

        # unpacking those elements
        list[pos2], list[pos1] = get

        return list

# Driver Code
List = [23, 65, 19, 90]

pos1, pos2 = 1, 3
print(swapPositions(List, pos1-1, pos2-1))
```

Solution : Python program to swap two elements in a list

```
# Python3 program to swap elements
# at given positions

def swapPositions(list, pos1, pos2):
        list[pos1],list[pos2] = list[pos2],list[pos1]
        return list

# Driver Code
List = [23, 65, 19, 90]
pos1, pos2 = 1, 3
print(swapPositions(List, pos1-1, pos2-1))
```

Problem 7 : Python program to remove Nth occurrence of the given word

Given a list of words in Python, the task is to remove the Nth occurrence of the given word in that list.

Examples:

```
Input: list - ["perfect", "plan", "perfect"]
       word = perfect, N = 2
Output: list - ["perfect", "plan"]
```

```
Input: list - ["can", "you",  "can", "a", "can" "?"]
       word = can, N = 1
 Output: list - ["you",  "can", "a", "can" "?"]
```

Solution : Python program to remove Nth occurrence of the given word

```
# Python3 program to remove Nth
# occurrence of the given word

# Function to remove Ith word
def RemoveIthWord(lst, word, N):
        newList = []
        count = 0

        # iterate the elements
        for i in lst:
                if(i == word):
                        count = count + 1
                        if(count != N):
                                newList.append(i)
                else:
                        newList.append(i)

        lst = newList
        if count == 0:
                print("Item not found")
        else:
                print("Updated list is: ",
lst)

        return newList

# Driver code
list = ["geeks", "for", "geeks"]
word = "geeks"
N = 2
RemoveIthWord(list, word, N)
```

Solution : Python program to remove Nth occurrence of the given word

```
# Python3 program to remove Nth
# occurrence of the given word

# Function to remove Ith word
def RemoveIthWord(list, word, N):
        count = 0

        for i in range(0, len(list)):
                if (list[i] == word):
                        count = count + 1

                        if(count == N):
                                del(list[i])
                                return True

        return False
```

```
# Driver code
list = ['geeks', 'for', 'geeks']
word = 'geeks'
N = 2

flag = RemoveIthWord(list, word, N)

if (flag == True):
        print("Updated list is: ", list)
else:
        print("Item not Updated")
RemoveIthWord(list, word, N)
```

Solution : Python program to remove Nth occurrence of the given word

```
# Python3 program to remove Nth
# occurrence of the given word

# Function to remove Ith word
def RemoveIthWord(list, word, N):
        count = 0

        for i in range(0, len(list)):
                if (list[i] == word):
                        count = count + 1

                        if(count == N):
                                del(list[i])
                                return True

        return False
```

```
# Driver code
list = ['geeks', 'for', 'geeks']
word = 'geeks'
N = 2

flag = RemoveIthWord(list, word,
N)

if (flag == True):
        print("Updated list is: ", list)
else:
        print("Item not Updated")
```

Solution : Python program to remove Nth occurrence of the given word

```
# Python3 program to remove Nth
# occurrence of the given word

# Function to remove nth word
def omit(list1,word,n1):
        #for counting the occurence of word
        count=0
        #for counting the index number where we are
at present
        index=0

        for i in list1:
                index+=1
                if i==word:
                        count+=1
                        if count==n1:
                                #(index-1) because in list
indexing start from 0th position
                                list1.pop(index-1)
        return list1
```

```
# Driver code
list1 = ["he", "is", "ankit", "is", "raj", "is","ankit raj"]

word="is"
n1=3

print("new list is :",omit(list1,word,n1))
```

Problem 8 : Python | Ways to find length of list

List being an integral part of Python day-day programming has to be learned by all the python users and having a knowledge of its utility and operations is essential and always a plus.

Show various methods to find length of a list

Solution : Python | Ways to find length of list

```
# Python code to demonstrate
# length of list
# using naive method

# Initializing list
test_list = [ 1, 4, 5, 7, 8 ]

# Printing test_list
print ("The list is : " + str(test_list))

# Finding length of list
# using loop
# Initializing counter
counter = 0
for i in test_list:

        # incrementing counter
        counter = counter + 1

# Printing length of list
print ("Length of list using naive method is : " +
str(counter))
```

Solution : Python | Ways to find length of list

```
# Python program to demonstrate working
# of len()
a = []
a.append("Hello")
a.append("Perfect")
a.append("Plan")
a.append("B")
print("The length of list is: ", len(a))
```

Solution : Python | Ways to find length of list

```
# Python code to demonstrate
# length of list
# using len() and length_hint
from operator import length_hint

# Initializing list
test_list = [ 1, 4, 5, 7, 8 ]

# Printing test_list
print ("The list is : " + str(test_list))

# Finding length of list
# using len()
list_len = len(test_list)

# Finding length of list
# using length_hint()
list_len_hint = length_hint(test_list)

# Printing length of list
print ("Length of list using len() is : " + str(list_len))
print ("Length of list using length_hint() is : " +
str(list_len_hint))
```

Problem 9 : Python | Ways to check if element exists in list

List being an integral part of Python day-day programming has to be learned by all the python users and having a knowledge of its utility and operations is essential and always a plus.

Show various methods to find if the element exists in the list or not.

Solution : Python | Ways to check if element exists in list

```
# Python code to demonstrate
# checking of element existence
# using loops and in

# Initializing list
test_list = [ 1, 6, 3, 5, 3, 4 ]

print("Checking if 4 exists in list ( using loop ) : ")

# Checking if 4 exists in list
# using loop
for i in test_list:
        if(i == 4) :
                print ("Element Exists")

print("Checking if 4 exists in list ( using in ) : ")

# Checking if 4 exists in list
# using in
if (4 in test_list):
        print ("Element Exists")
```

Solution : Python | Ways to check if element exists in list

```
# Python code to demonstrate
# checking of element existence
# using set() + in
# using sort() + bisect_left()
from bisect import bisect_left

# Initializing list
test_list_set = [ 1, 6, 3, 5, 3, 4 ]
test_list_bisect = [ 1, 6, 3, 5, 3, 4 ]

print("Checking if 4 exists in list ( using set() + in) : ")

# Checking if 4 exists in list
# using set() + in
test_list_set = set(test_list_set)
if 4 in test_list_set :
        print ("Element Exists")

print("Checking if 4 exists in list ( using sort() +
bisect_left() ) : ")

# Checking if 4 exists in list
# using sort() + bisect_left()
test_list_bisect.sort()
if bisect_left(test_list_bisect, 4):
        print ("Element Exists")
```

Problem 10 : Python | Reversing a List

Examples:

```
Input : list = [10, 11, 12, 13, 14, 15]
Output : [15, 14, 13, 12, 11, 10]
```

```
Input : list = [4, 5, 6, 7, 8, 9]
 Output : [9, 8, 7, 6, 5, 4]
```

Solution : Python | Reversing a List

```
# Reversing a list using reversed()
def Reverse(lst):
        return [ele for ele in reversed(lst)]

# Driver Code
lst = [10, 11, 12, 13, 14, 15]
print(Reverse(lst))
```

Solution : Python | Reversing a List

```
# Reversing a list using reverse()
def Reverse(lst):
        lst.reverse()
        return lst

lst = [10, 11, 12, 13, 14, 15]
print(Reverse(lst))
```

Solution : Python | Reversing a List

```
# Reversing a list using slicing
technique
def Reverse(lst):
        new_lst = lst[::-1]
        return new_lst

lst = [10, 11, 12, 13, 14, 15]
print(Reverse(lst))
```

Problem 11 : Reverse words in a given String in Python

We are given a string and we need to reverse words of

given string ?

Examples:

```
Input : str = "quiz practice code"
 Output : str = "code practice quiz"
```

Solution : Reverse words in a given String in Python

```
# Function to reverse words of string

def rev_sentence(sentence):

        # first split the string into words
        words = sentence.split(' ')

        # then reverse the split string list and
join using space
        reverse_sentence = '
'.join(reversed(words))

        # finally return the joined string
        return reverse_sentence

if __name__ == "__main__":
        input = 'geeks quiz practice code'
        print rev_sentence(input)
```

Problem 12 : Python | Check if a Substring is Present in a Given String

Given two strings, check if s1 is there in s2.

Examples:

```
Input : s1 = perfect s2=perfect plan b
Output : yes
```

```
Input : s1 = per s2=perfect plan b
 Output : yes
```

Solution : Python | Check if a Substring is Present in a Given String

```
# function to check if small string is
# there in big string
def check(string, sub_str):
        if (string.find(sub_str) == -1):
                print("NO")
        else:
                print("YES")

# driver code
string = "perfect plan b"
sub_str ="perfect"
check(string, sub_str)
```

Solution : Python | Check if a Substring is Present in a Given String

```
def check(s2, s1):
        if (s2.count(s1)>0):
                print("YES")
        else:
                print("NO")

s2 = "A geek in need is a geek indeed"
s1 ="geek"
check(s2, s1)
```

Solution : Python | Check if a Substring is Present in a Given String

```
# When you have imported the re module, you can start
using regular expressions.
import re

# Take input from users
MyString1 = "A geek in need is a geek indeed"
MyString2 ="geek"

# re.search() returns a Match object if there is a match
anywhere in the string
if re.search( MyString2, MyString1 ):
        print("YES,string '{0}' is present in string '{1}' "
.format(MyString2,MyString1))
else:
        print("NO,string '{0}' is not present in string {1} "
.format(MyString2, MyString1) )
```

Problem 13 : Python program to print even length words in a string

Given a string. The task is to print all words with even length in the given string.

Examples:

```
Input: s = "This is a python language"
Output: This
        is
        python
        language
```

```
Input: s = "i am muskan"
Output: am
         muskan
```

Solution : Python program to print even length words in a string

```
# Python3 program to print
# even length words in a string

def printWords(s):

        # split the string
        s = s.split(' ')

        # iterate in words of string
        for word in s:

                # if length is even
                if len(word)%2==0:
                        print(word)


# Driver Code
s = "i am muskan"
printWords(s)
```

Problem 14 : Python | Program to accept the strings which contains all vowels

Given a string, the task is check if every vowel is present or not. We consider a vowel to be present if it is present in upper case or lower case. i.e. ‘a’, ‘e’, ‘i’.’o’, ‘u’ or ‘A’, ‘E’, ‘I’, ‘O’, ‘U’ .

Examples :

```
Input : perfectplanb
Output : Not Accepted
'i,o,u' are not present
```

```
Input : ABeeIghiObhkUul
Output : Accepted
 All vowels are present
```

Solution : Python | Program to accept the strings which contains all vowels

```
# Python program to accept the strings
# which contains all the vowels

# Function for check if string
# is accepted or not
def check(string) :

    string = string.lower()

    # set() function convert "aeiou"
    # string into set of characters
    # i.e.vowels = {'a', 'e', 'i', 'o', 'u'}
    vowels = set("aeiou")

    # set() function convert empty
    # dictionary into empty set
    s = set({})

    # looping through each
    # character of the string
    for char in string :

        # Check for the character is present
inside
        # the vowels set or not. If present,
then
        # add into the set s by using add
method
        if char in vowels :
            s.add(char)
        else:
            pass

    # check the length of set s equal to length
    # of vowels set or not. If equal, string is
    # accepted otherwise not
    if len(s) == len(vowels) :
        print("Accepted")
    else :
        print("Not Accepted")


# Driver code
if __name__ == "__main__" :

    string = "SEEquoiaL"

    # calling function
    check(string)
```

Problem 15 : Remove all duplicates from a given string in Python

We are given a string and we need to remove all duplicates

from it ? What will be the output if order of character

matters ?

Examples:

```
Input : geeksforgeeks
 Output : efgkos
```

Solution : Remove all duplicates from a given string in Python

```
from collections import OrderedDict

# Function to remove all duplicates from string
# and order does not matter
def removeDupWithoutOrder(str):

        # set() --> A Set is an unordered collection
        #               data type that is iterable, mutable,
        #               and has no duplicate elements.
        # "".join() --> It joins two adjacent elements in
        #                       iterable with any symbol defined in
        #                       "" ( double quotes ) and returns a
        #                       single string
        return "".join(set(str))

# Function to remove all duplicates from string
# and keep the order of characters same
def removeDupWithOrder(str):
        return "".join(OrderedDict.fromkeys(str))
```

```
# Driver program
if __name__ == "__main__":
        str = "geeksforgeeks"
        print "Without Order =
",removeDupWithoutOrder(
str)
        print "With Order =
",removeDupWithOrder(str)
```

Solution : Remove all duplicates from a given string in Python

```
def removeDuplicate(str):
        s=set(str)
        s="".join(s)
        print("Without Order:",s)
        t=""
        for i in str:
                if(i in t):
                        pass
                else:
                        t=t+i
                print("With Order:",t)

str="geeksforgeeks"
removeDuplicate(str)
```

Problem 16 : Python program to find the sum of all items in a dictionary

Given a dictionary in Python, write a Python program to find the sum

of all Items in the dictionary.

Examples:

```
Input : {'a': 100, 'b':200, 'c':300}
Output : 600
```

```
Input : {'x': 25, 'y':18, 'z':45}
 Output : 88
```

Solution : Python program to find the sum of all items in a dictionary

```
# Python3 Program to find sum of
# all items in a Dictionary

# Function to print sum
def returnSum(myDict):

    sum = 0
    for i in myDict:
        sum = sum + myDict[i]

    return sum

# Driver Function
dict = {'a': 100, 'b':200, 'c':300}
print("Sum :", returnSum(dict))
```

Problem 17 : Python | Convert a list of Tuples into Dictionary

Sometimes you might need to convert a tuple to dict object to make it more readable.

In this article, we will try to learn how to convert a list of tuples into a dictionary.

Examples:

```
Input : [("akash", 10), ("gaurav", 12), ("anand", 14),
         ("suraj", 20), ("akhil", 25), ("ashish", 30)]
Output : {'akash': [10], 'gaurav': [12], 'anand': [14],
          'ashish': [30], 'akhil': [25], 'suraj': [20]}
```

```
Input : [('A', 1), ('B', 2), ('C', 3)]
 Output : {'B': [2], 'A': [1], 'C': [3]}
```

Solution : Python | Convert a list of Tuples into Dictionary

```
# Python code to convert into dictionary

def Convert(tup, di):
        for a, b in tup:
                di.setdefault(a, []).append(b)
        return di

# Driver Code
tups = [("akash", 10), ("gaurav", 12), ("anand", 14),
        ("suraj", 20), ("akhil", 25), ("ashish", 30)]
dictionary = {}
print (Convert(tups, dictionary))
```

Solution : Python | Convert a list of Tuples into Dictionary

```
# Python code to convert into dictionary
def Convert(tup, di):
        di = dict(tup)
        return di

# Driver Code
tups = [("akash", 10), ("gaurav", 12), ("anand", 14),
        ("suraj", 20), ("akhil", 25), ("ashish", 30)]
dictionary = {}
print (Convert(tups, dictionary))
```

Solution : Python | Convert a list of Tuples into Dictionary

```
# Python code to convert into dictionary

print (dict([('Sachin', 10), ('MSD', 7), ('Kohli', 18),
('Rohit', 45)]))
```

Problem 18 : Python Program for Tower of Hanoi

Tower of Hanoi is a mathematical puzzle where we have three rods and n disks. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1) Only one disk can be moved at a time.

2) Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.

3) No disk may be placed on top of a smaller disk.

Note: Transferring the top n-1 disks from source rod to Auxilliary rod can again be thought of as a fresh problem and can be solved in the same manner.

Solution : Python | Convert a list of Tuples into Dictionary

```
# Recursive Python function to solve tower of hanoi

def TowerOfHanoi(n , source, destination, auxilliary):
        if n==1:
                print "Move disk 1 from source",source,"to
destination",destination
                return
        TowerOfHanoi(n-1, source, auxilliary,
destination)
        print "Move disk",n,"from source",source,"to
destination",destination
        TowerOfHanoi(n-1, auxilliary, destination,
source)

# Driver code
n = 4
TowerOfHanoi(n,'A','B','C')
# A, C, B are the name of rods
```

Problem 19 : Python program to copy odd lines of one file to other

Write a python program to read contents of a file and copy only the content of odd lines into new file.

Examples:

```
Input : Hello
        World
        Python
        Language
Output : Hello
         Python
```

```
Input : Python
        Language
        Is
        Easy
Output : Python
          Is
```

Solution : Python program to copy odd lines of one file to other

```
# open file in read mode
fn = open('bcd.txt', 'r')

# open other file in write mode
fn1 = open('nfile.txt', 'w')

# read the content of the file line by line
cont = fn.readlines()
type(cont)
for i in range(0, len(cont)):
        if(i % 2 ! = 0):
                fn1.write(cont[i])
        else:
                pass

# close the file
fn1.close()
fn1.close()
```

```
# open file in read mode
fn1 = open('nfile.txt', 'r')

# read the content of the file
cont1 = fn1.read()

# print the content of the file
print(cont1)

# close all files
fn.close()
```

Problem 20 : Python program to check if a string contains all unique characters

To implement an algorithm to determine if a string contains all unique characters.

Examples:

Input : s = “abcd”

Output: True

“abcd” doesn’t contain any duplicates. Hence the output is True.

Input : s = “abbd”

Output: False

“abbd” contains duplicates. Hence the output is False.

Solution : Python program to check if a string contains all unique characters

```
def isUniqueChars(st):

        # String length cannot be more than
        # 256.
        if len(st) > 256:
                return False

        # Initialize occurrences of all characters
        char_set = [False] * 128

        # For every character, check if it exists
        # in char_set
        for i in range(0, len(st)):

                # Find ASCII value and check if it
                # exists in set.
                val = ord(st[i])
                if char_set[val]:
                        return False

                char_set[val] = True

        return True

# driver code
st = "abcd"
print(isUniqueChars(st))
```

About Piazza

Please watch this video about how to use Piazza:

https://youtu.be/ndzsh5ShhhA

Social Media Links:

Facebook: https://www.facebook.com/MyPerfectеLearning

Twitter: https://twitter.com/Perfectelearn

Linkedin: https://www.linkedin.com/company/myperfectelearning/

Instagram: https://www.instagram.com/myperfectelearning/

Quora: https://perfectelearning.quora.com/

Youtube: https://www.youtube.com/c/PerfecteLearning

Pinterest: https://pin.it/4CjOr3R