CH5: Problem on python Data-Structure

In section, we will discuss some example to clarify the concept of Python data structure. We will also do some problem solving question using python language and see how easily and quickly we can solve a problem in python.

Problem1: Given a String, find out if it is palindrome or not !

ANS:

myStr = raw\_input("Enter a String:")

print 'Is palindrome? Ans:',myStr == myStr[::-1]

Output:

Enter a String:MADAM

Is palindrome? Ans: True

Enter a String:PYTHON

Is palindrome? Ans: False

*Explanation:*

**Problem2: Take a line and reverse the line without reversing the word.**

*ANS:*

myStr = raw\_input("Enter a String:")

print 'Reversed Line:',' '.join(myStr.split(' ')[::-1])

OUTPUT:

Enter a String:I LOVE PYTHON

Reversed Line: PYTHON LOVE I

>>>

Enter a String:I AM DIPANKAR

Reversed Line: DIPANKAR AM I

>>>

*Explanation:*

**Problem3:** **Checking Whether a String Contains a Set of Characters,** **You need to check for the occurrence of any of a set of characters in a string.**

*ANS: This problem can be solved by multiple way ,code is givem as bewlow:*

myStr1 = raw\_input("Enter a String:")

myStr2 = raw\_input("Enter a ChracterSet:")

print 'Is all Chracter covered:', set(myStr1) >=set(myStr2)

**OUTPUT:**

>>>

Enter a String:dipankar

Enter a ChracterSet:dip

Is all Chracter covered: True

>>>

Enter a String:python

Enter a ChracterSet:djn

Is all Chracter covered: False

**Find Occurrences:**

myStr1 = raw\_input("Enter a String:")

myDict ={}

for i in myStr1:

if i in myDict:

myDict[i] +=1

else:

myDict[i] =1

print myDict

**Output:**

>>>

Enter a String:I LOVE PYTHON

{' ': 2, 'E': 1, 'I': 1, 'H': 1, 'L': 1, 'O': 2, 'N': 1, 'P': 1, 'T': 1, 'V': 1, 'Y': 1}

>>>

Enter a String:abbbcccaaaaaadddd

{'a': 7, 'c': 3, 'b': 3, 'd': 4}

>>>

*Explanation:*

**Problem 4.** **You have a string made up of multiple lines, and you need to build another string from it, adding or removing leading spaces on each line so that the indentation of each line is some absolute number of spaces.**

*ANS:*

myStr1 ="""

Hello Welcome!

Welcome to python!

I love python!

"""

indent\_spc = 4

print '\n'.join([ ' ' \* indent\_spc +line.strip() for line in myStr1.splitlines()])

OUTPUT

Hello Welcome!

Welcome to python!

I love python!

>>> When indent\_spc = 2

Hello Welcome!

Welcome to python!

I love python!

*Explanation:*

**Problem 5: You want to create a multidimensional list but want to avoid implicit reference sharing**

*ANS:*

>>> alist =[0]\*5

>>> alist

[0, 0, 0, 0, 0]

>>> dlist =[0]\*5 \*10

>>> dlist

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

>>> dlist [ [0] \*5 ] \*10]

>>> dlist = [ [0]\*5] \* 10 # Creating 2D list Wrongly

>>> dlist

[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]

>>> dlist[0][0]='hello' # If you change 0,0 Elemnet it will change all

>>> dlist

[['hello', 0, 0, 0, 0], ['hello', 0, 0, 0, 0], ['hello', 0, 0, 0, 0], ['hello', 0, 0, 0, 0], ['hello', 0, 0, 0, 0], ['hello', 0, 0, 0, 0], ['hello', 0, 0, 0, 0], ['hello', 0, 0, 0, 0], ['hello', 0, 0, 0, 0], ['hello', 0, 0, 0, 0]]

# Correct way as below

>>> dlist =[ [0 for i in range(5) ] for j in range(10)]

>>> dlist

[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]

>>> dlist[0][0] ='correct'

>>> dlist

[['correct', 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]

>>>

*Explanation:*

**Problem6: Transpose a 2D list**

*ANS:*

#Creating a 2D list

>>> myList = [[5\*i+j+1 for j in range(5)] for i in range(5)]

>>> myList

[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15], [16, 17, 18, 19, 20], [21, 22, 23, 24, 25]]

>>>

#Transpose It

>>> print [[r[col] for r in myList] for col in range(len(myList[0]))]

[[1, 6, 11, 16, 21], [2, 7, 12, 17, 22], [3, 8, 13, 18, 23], [4, 9, 14, 19, 24], [5, 10, 15, 20, 25]

*Explanation:*

**Problem 7: Given a dictionary, Find out the reverse dictionary. For example, it input is {a:1,b:2} output should be {b:2,1:a}**

*ANS:*

>>> a ={ 'a':1,'b':2,'c':3,'d':4}

>>> a

{'a': 1, 'c': 3, 'b': 2, 'd': 4}

>>> print dict([ tuple(list(i)[::-1]) for i in a.items()])

{1: 'a', 2: 'b', 3: 'c', 4: 'd'}

>>>

*Explanation:*

**Problem 8 : How can u use printf in python**

*ANS:*

>>> import sys

>>> def printf(format, \*args):

sys.stdout.write(format % args)

>>> name = 'python'

>>> year =2003

>>> printf('I Love %s from year %d',name,year);

I Love python from year 2003

>>>

*Explanation:*

**Problem 9 : How to sort a dictionary ?**

*ANS:*

>>> for i in myDict:

print myDict[i] # Output is not sorted

1

9

3

2

>>> myDict.keys()

['a', 'x', 'c', 'b']

>>> sorted(myDict.keys())

['a', 'b', 'c', 'x']

>>> [myDict[i] for i in sorted(myDict.keys())]

[1, 2, 3, 9]

>>> for x in [myDict[i] for i in sorted(myDict.keys())]:

print x

1

2

3

9

>>>

*Explanation:*

**Problem 10: How to short a list of string when case-sensitivity is not taking in account.**

*ANS:*

>>> a =[ 'apple','Apple','ball','BALL']

>>> sorted(a)

['Apple', 'BALL', 'apple', 'ball']

>>> def mySort(a):

aux = [(x.lower( ), x) for x in a]

aux.sort()

return [x[1] for x in aux]

>>> mySort(a)

['Apple', 'apple', 'BALL', 'ball']

>>>

*Explanation:*

**Problem 11: How to sort a list of tuple based on the sum of the tuple.?**

*ANS:*

>>> l = [list(range(i, i+4)) for i in range(10,1,-1)]

>>> l

[[10, 11, 12, 13], [9, 10, 11, 12], [8, 9, 10, 11], [7, 8, 9, 10], [6, 7, 8, 9], [5, 6, 7, 8], [4, 5, 6, 7], [3, 4, 5, 6], [2, 3, 4, 5]]

>>> sorted(l, key=sum)

[[2, 3, 4, 5], [3, 4, 5, 6], [4, 5, 6, 7], [5, 6, 7, 8], [6, 7, 8, 9], [7, 8, 9, 10], [8, 9, 10, 11], [9, 10, 11, 12], [10, 11, 12, 13]]

>>>

*Explanation:*

**Problem 12 Getting the First Few Smallest Items of a Sequence:**

*ANS:*

>>> import heapq

>>> def getmin(data):

data=list(data)

heapq.heapify(data)

while data:

yield heapq.heappop(data)

>>> a =[1,4,6,9,3,2,0,120,44]

>>> x = getmin(a)

>>> x.next()

0

>>> x.next()

1

>>> x.next()

2

>>> x.next()

3

>>> x.next()

4

>>> x.next()

6

>>> x.next()

9

>>> x.next()

44

>>> x.next()

120

>>> x.next()

Traceback (most recent call last):

File "<pyshell#156>", line 1, in <module>

x.next()

StopIteration

>>>

*Explanation:*

**Problem 13 Implement Quick Sort in Python:**

*ANS:*

def qsort(L):

if len(L) <= 1: return L

return qsort([lt for lt in L[1:] if lt < L[0]]) + L[0:1] + qsort([ge for ge in L[1:] if ge >= L[0]])

a = [1,4,7,0,8,4,3,2,9]

print qsort(a) #0, 1, 2, 3, 4, 4, 7, 8, 9]

*Explanation:*

**Problem 14: How to generate all permutations of a list in Python ?**

*ANS:*

def permute(start, rest):

res = []

if len(rest) <= 1:

res += [start + rest, rest + start]

else:

for i, c in enumerate(rest):

s = rest[:i] + rest[i+1:]

for perm in permute(c, s):

res += [start + perm]

return res

print permute('', 'abc') #['abc', 'acb', 'acb', 'abc', 'bac', 'bca', 'bca', 'bac', 'cab', 'cba', 'cba', 'cab']

*Explanation:*

**Problem 15: Define a custom sort function such that it can sort a list of string based on their length.**

*ANS:*

# sort a list on your own criteria

# define your own method for sorting (must return 1, 0, -1)

def mysort(x,y):

x = len(x)

y = len(y)

if x>y:

return 1

elif x==y:

return 0

else:

return -1

alist = ['Here', 'is', 'a', 'list',

'of', 'small', 'and', 'gybungusly',

'big', 'words']

alist.sort(mysort)

# the list is now sorted from smallest to largest word

print alist

#output: ['a', 'is', 'of', 'and', 'big', 'Here', 'list','small', 'words', 'gybungusly']

*Explanation:*