Counter – convert list to dictionary into single word based dictioanry

zip – zip two values in single

map- can apply different logic on list

itemgetter—list with nested dictioanry

Key with maximum unique values: using counter

Using max() + lambda function: lambda can be implemented in max funtion with key attribute

dict(zip(

OrderedDict(reversed(list( : reverse the order of dictionary

Lambda : can be used as function

<https://betterprogramming.pub/10-ways-to-convert-lists-to-dictionaries-in-python-d2c728d2aeb8>

https://www.geeksforgeeks.org/tag/python-dictionary-programs/

https://www.geeksforgeeks.org/python-dictionary-exercise/

https://docs.python.org/3.8/library/stdtypes.html#dict-views

https://docs.python.org/3/library/collections.html

**Map:-**

Lamdba always take parameter like Ls =[1,2,3,4] ;

list(map(lambda x: x\*\*2, Ls))

**map()** function returns a map object. Applied on the iterable list, set, dict

map(fun, iter)

---------------------------------------

dict = {'ravi': '10', 'rajnish': '9', 'sanjeev': '15', 'yash': '2', 'suraj': '32'}

dict1 = dict.items() --> list( tuple())

dict\_items([('ravi', '10'), ('rajnish', '9'), ('sanjeev', '15'), ('yash', '2'), ('suraj', '32')])

Dictionary has O(1) search time complexity whereas List has O(n) time complexity

----------------------------------------------

print(country\_code.get('Japan', 'Not Found'))

-------------------------------------------

defd = collections.defaultdict(lambda : 'Key Not found') # set defaukt key for all absent key

----------------------------------------------

sorted(lis, key=itemgetter('age', 'name')) # sort dict according to key age and name

https://www.geeksforgeeks.org/ways-sort-list-dictionaries-values-python-using-itemgetter/

-----------------------------------------------

sorted(lis, key = lambda i: i['age'],reverse=True)

-------------------------------------------------

**Three type of merging dictionaries:-**

dict2.update(dict1)

dict = dict1 | dict2

res = {\*\*dict1, \*\*dict2} # Store in a third dictionary

**Unpacking:-**

dict = {'a': 'Geeks', 'b': 'For', 'c': 'geeks'}

print([\*dict]) # Unpacking with \* works with any object that is iterable

**Itemgenerator:-**

from operator import itemgetter

dict = {'a': 'Geeks', 'b': 'For', 'c': 'geeks'}

ls = list(map(itemgetter(1), dict.items()))

print(ls)

**Map with Lambda:**

numbers = (1, 2, 3, 4)

result = map(lambda x: x + x, numbers) #Pick value one by one and adding

print(list(result))

-------------------------------------------------------------------

l = ['sat', 'bat', 'cat', 'mat']

# map() can listify the list of strings individually

test = list(map(list, l))

#output will be splited char in list

-------------------------------------------------------------------

# Itertools

# Using chain.from\_iterable()

from itertools import chain

# Get all tuple keys from dictionary

# Using chain.from\_iterable()

res = list(chain.from\_iterable(test\_dict))

**Tuple():-**

ls = tuple(x for x in dict.keys()) # for making tuple we have to mention tuple in the front of () unlike list

# Unpacking

# Using "=" operator and multiple variables

a, b, c = test\_dict # Unpacking dictionary keys into tuple

res = a, b, c

**2nd example of Tuple Unpacking:-**

z = (10, 100) # tuple with two values

print (result(\*z))# \* unpack two values and pass to result methods

Special Symbols Used for passing arguments:-

1.)\*args (Non-Keyword Arguments) #“wildcard” or “\*”

2.)\*\*kwargs (Keyword Arguments) # to unpack dictionary we mention \*\* with arguments

args = [0, 1, 4, 9]

def func(a, b, c):

func(\*args) # unpack and pass the values

**Key exists in tuple keys dictionary:**

1. using any() + generator expression

test\_dict = {(4, 5) : '1', (8, 9) : '2', (10, 11) : '3'}

res = any(key in sub for sub in test\_dict)

--------------------

Counter provide sorted dictionary

from collections import Counter

ar1 = [1, 5, 10, 20, 40, 80]

ar2 = [6, 7, 20, 80, 100]

ar3 = [3, 4, 15, 20, 30, 70, 80, 120]

ar1 = Counter(ar1)

ar2 = Counter(ar2)

ar3 = Counter(ar3)

resultDict = dict(ar1.items() & ar2.items() & ar3.items())

list(resultDict.keys())

-----------------

from itertools import repeat

res = dict(zip(range(4), repeat(test\_dict))) # return 4 same key value pair for test\_dic

------------------

<https://www.geeksforgeeks.org/dictionary-counter-python-find-winner-election/>

from collections import Counter

input =['john','johnny','jackie','johnny',

            'john','jackie','jamie','jamie',

            'john','johnny','jamie','johnny',

            'john']

vote\_count=Counter(input) #Counter({'jackie': 2, 'jamie': 3, 'john': 4, 'johnny': 4})

max\_votes=max(vote\_count.values()) # number 4 will response

st=[i for i in vote\_count.keys() if vote\_count[i]==max\_votes]

------------------

<https://www.geeksforgeeks.org/python-key-with-maximum-unique-values/>

Key with maximum unique values: test\_dict = {“Gfg” : [5, 7, 9, 4, 0], “is” : [6, 7, 4, 3, 3], “Best” : [9, 9, 6, 5, 5]}; output : Gfg

from collections import Counter

import numpy as np

test\_dict = {"Gfg" : [5, 7, 5, 4, 5], "is" : [6, 7, 4, 3, 3], "Best" : [9, 9, 6, 5, 5]}

counter = Counter(test\_dict)

index = np.argmax([len(set(x)) for x in counter.values()])

print(sorted(list(counter.keys()))[index])

2nd solution :

max\_key = sorted(test\_dict, key = lambda ele: len(set(test\_dict[ele])), reverse = True)[0]

pass test\_dict key one by one to lambda then reverse based on key in sorted function

----------------------------

<https://www.geeksforgeeks.org/python-find-duplicate-characters-string/>

input = 'geeksforgeeks'

counter = Counter(input)

for value , key in counter.items(): # counter will dic with number of variable as dic

-------------------------------

<https://www.geeksforgeeks.org/python-group-similar-items-to-dictionary-values-list/>

**First approach:-**  **test\_list = [4, 6, 6, 4, 2, 2, 4, 8, 5, 8] Output : {4: [4, 4, 4], 6: [6, 6], 2: [2, 2], 8: [8, 8], 5: [5]}**

from collections import Counter

test\_list = [4, 6, 6, 4, 2, 2, 4, 4, 8, 5, 8]

c = Counter(test\_list)

def k(l):

d = {}, k = []

for i in range(l[1]):

k.append(l[0])

d[l[0]] = k

return d

print(list(map(lambda x: k(x), c.items())))

**2nd approach:-**

**# using defaultdict for default list**

res = defaultdict(list)

for ele in test\_list:

    res[ele].append(ele)

**3rd approach:**

res = {key : [key] \* val for key, val in Counter(test\_list).items()}

------------------------------- **Example of dictionary get value from based on two value dict.get(wrd, wrd))**

-------------------------------

test\_list = ["Gfg", "is", "Best"]

# initializing subs. Dictionary

subs\_dict = {

"Gfg" : [5, 6, 7],

"is" : [7, 4, 2],

}

**1st program:- ( good example of if condition)**

K = 2

res = [ele if ele not in subs\_dict else subs\_dict[ele][K] for ele in test\_list]

**2nd example : not completed version (check map get with two key)**

for ele in test\_list:

print(subs\_dict.get(ele, ele))

output:-

[5, 6, 7]

[7, 4, 2]

Best

--------------

Good example of isinstance:-

completed version:-

res = [subs\_dict.get(ele, ele) for ele in test\_list]

es = [ele[K] if isinstance(ele, list) else ele for ele in res]

##Remember how element is saved in list in with if condition. Check above example in red.

--------------------- <https://www.geeksforgeeks.org/python-replace-words-from-dictionary/>

**1st Example:-**

test\_str = 'geekforgeeks best for geeks'

lookp\_dict = {"best" : "good and better", "geeks" : "all CS aspirants"}

temp = test\_str.split()

res = []

for wrd in temp:

res.append(lookp\_dict.get(wrd, wrd))

res = ' '.join(res) # merge array to string

print(res) #Output : geekforgeeks best for all CS aspirants

**2nd Example : good short cut of joining element with dictionary get**

res = " ".join(lookp\_dict.get(ele, ele) for ele in test\_str.split())

<https://www.geeksforgeeks.org/python-remove-duplicate-values-across-dictionary-values/>

test\_dict = {'Manjeet': [1], 'Akash': [1, 8, 9]}

#{‘Manjeet’: [], ‘Akash’: [8, 9]}

from collections import Counter

cnt = Counter()

for idx in test\_dict.values():

cnt.update(idx) ###Counter({1: 2, 8: 1, 9: 1})

res = {idx: [key for key in j if cnt[key]==1] for idx, j in test\_dict.items()}

print(res)

**------------------------------find mirror characters in a string-------------------**

# Mirror characters

<https://www.geeksforgeeks.org/python-dictionary-find-mirror-characters-string/>

A good example of zip and dictionary combination:-

original = 'abcdefghijklmnopqrstuvwxyz'

reverse = 'zyxwvutsrqponmlkjihgfedcba'

dictChars = dict(zip(original,reverse)) # creating dictioanry with a to z as key

input = 'paradox'

k = 3

prefix = input[0:k-1]

suffix = input[k-1:]

mirror = ''

print(prefix, suffix)

for i in range(0, len(suffix)):

mirror = mirror + dictChars[suffix[i]]

print(prefix + mirror)

-------------------------------------------------------------Very Good exmaple of val.count(i) and dic.get(I,0) ---------------------

<https://www.geeksforgeeks.org/counting-the-frequencies-in-a-list-using-dictionary-in-python/>

val = [1, 1, 1, 5, 5, 3, 1, 3, 3, 1, 4, 4, 4, 2, 2, 2, 2]

from collections import Counter

counter = Counter()

dc = {}

for i in val:

dc[i] = val.count(i)

print(dc)

---------

2nd example

count = {}

for i in val:

count[i] = count.get(i, 0) + 1

print(count)

------------------------

# Counter

https://www.geeksforgeeks.org/python-counter-dictionary-intersection-example-make-string-using-deletion-rearrangement/

Without brute force:

s2 = "hellow"

dict1 = Counter(s1)

dict2 = Counter(s2)

result = dict1 & dict2 #give common of two dictionary

print(result == dict1) #two dictionaries are eqauls

Brute force:

flag = False

for k, v in c1.items():

if k in c2.keys() and v <= c2.get(k):

flag = “possible”

else:

flag = “not possible”

break

print(flag)

---------------------

Order(preserve order of dictionary but consume more memory) vs Regular dictionary **:**

d = {chr(k):k for k in range(ord('a'), ord('g'))}

for k, v in d.items():

    print(k, v)

# Creating an Ordered dictionary

print('\nOrderedDict:')

d = collections.OrderedDict()##create ordered dictioanry

[d.setdefault(chr(k), k) for k in range(ord('a'), ord('g'))]#set key val

-----------------------------

# Max order based on custom key with lambda

**Using**max()**+ lambda function**

res = max(test\_dict, key = lambda sub: test\_dict[sub][key])

------------------------------

# Zip with +=

res.append([key] + val) : val is list and key is string can be appended in final list

for idx in range(0, 10, 2): for loop within ternary condition

for idx, sub in enumerate(test\_list, start = 0): idx will start from o to length of the dic.

Good example of zip:

test\_dict = {'month' : [1, 2, 3], 'name' : ['Jan', 'Feb', 'March']}

print(dict(zip(test\_dict['month'], test\_dict['name'])))

Good example of dict as tuple:

from collections import defaultdict

res = defaultdict(tuple)

for key, val in test\_dict.items():

for x in val:

res[x] += (val[x],)

------------------

# defaultdict

# initializing dictionary

test\_dict = {'Gfg': { 'a' : [1, 3], 'b' : [3, 6], 'c' : [6, 7, 8]},

'Best': { 'a' : [7, 9], 'b' : [5, 3, 2], 'd' : [0, 1, 0]}}

from collections import defaultdict

#{‘d’: {‘Best’: [0, 1, 0]}, ‘a’: {‘Gfg’: [1, 3], ‘Best’: [7, 9]}, ‘c’: {‘Gfg’: [6, 7, 8]}, ‘b’: {‘Gfg’: [3, 6], ‘Best’: [5, 3, 2]}}

a = defaultdict(dict)

for i, v in test\_dict.items():

for i1, v1 in v.items():

a[i1][i] = v1

print(str(dict(a)))

---------------

# Reverse dictionary order

# initializing dictionary

from collections import OrderedDict

# initializing dictionary

test\_dict = {'gfg' : 4, 'is' : 2, 'best' : 5}

# printing original dictionary

print("The original dictionary : " + str(test\_dict))

# Reverse Dictionary Keys Order

# Using OrderedDict() + reversed() + items()

res = OrderedDict(reversed(list(test\_dict.items())))

print(test\_dict.items())

--------------very very good example ------------------------

# Set intersection

if K in set(test\_list).intersection(test\_dict): #return keys which are available in both list and dictionary

    res = test\_dict[K]

------------------------------ isinstance example---------------

# isinstance

if not (isinstance(test\_dict[key], int)

--------------max based on key length------------

# max

res = max(test\_list, key = len)

--------merge all keys and values in single list--------

list(test\_dict.keys()) + list(test\_dict.values())

--------- # chain() is used for concatenation--------

# Chain

from itertools import chain

res = list(chain(test\_dict.keys(), test\_dict.values()))

------------------------ defaultdict example---------------

# Defaultdict example

from collections import defaultdict

# initializing dictionary

test\_dict = {'gfg' : [1, 2, 3], 'is' : [1, 4], 'best' : [4, 2]}

# Using defaultdict() + loop

res = defaultdict(list)

for key, val in test\_dict.items():

for ele in val:

res[ele].append(key)

print(res)

output: {1: ['gfg', 'is'], 2: ['gfg', 'best'], 3: ['gfg'], 4: ['is', 'best']}

----------------good example of dictionary with nested list as values

# Dictionary nested list values

arr = ['cat', 'dog', 'tac', 'god', 'act']

#'cat tac act dog god'

dict = {}

for strVal in arr:

key = ''.join(sorted(strVal ))

if key in dict.keys():

dict[key].append(strVal)

else:

dict[key] = []

dict[key].append(strVal)

print(dict)

Output : 'cat tac act dog god'

# Lambda function with if but without else.

square = lambda x : x\*x if(x > 0) #this will throw an error

print(square(6))

<https://www.geeksforgeeks.org/lambda-with-if-but-without-else-in-python/>

# Example of lambda function using if-else

max = lambda a, b : a if(a > b) else b

print(max(1, 2))

print(max(10, 2))

print(list(c1 & c2)[::-1]) reverse the string

-----------------

# Reverse order with pattern

Reverse order according to the given pattern

<https://www.geeksforgeeks.org/python-sorting-string-using-order-defined-by-another-string/>

from collections import Counter

pat = "sasbcklfdmegnot"

str1 = "ekgse"

c1 = Counter(pat)

s1 = list(str1)

s1.sort( key = lambda ele : c1[ele])

s1.reverse()

new\_str = ''.join(s1)

print(new\_str)

# len(list(filter(lambda

odd even count of list

# list of numbers

list1 = [10, 21, 4, 45, 66, 93, 11]

odd\_count = len(list(filter(lambda x: (x%2 != 0) , list1)))

print(odd\_count)

----------

print (reduce(lambda a, b: a ^ b, input))??????????????