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 L =[1,2,3,4] ; list(map(lambda x: x\*\*2, L))

**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 accoring jto key age and name

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

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 valu 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()}