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:-**

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)