## Python Exercises- Dictionary

<https://www.w3resource.com/python-exercises/dictionary/>

'''1. Write a Python script to sort (ascending and descending)

a dictionary by value

'''

dict1={'one':1,'two':2,'three':3}

print(sorted(dict1)) #['one','three','two']

print(sorted(dict1,key=lambda x: dict1[x],reverse=True)) #['three','two','one']

#Textbook-1

import operator

d = {1: 2, 3: 4, 4: 3, 2: 1, 0: 0}

print('Original dictionary : ',d)

sorted\_d = sorted(d.items(), key=operator.itemgetter(1))

print('Dictionary in ascending order by value : ',sorted\_d)

sorted\_d = dict( sorted(d.items(), key=operator.itemgetter(1),reverse=True))

print('Dictionary in descending order by value : ',sorted\_d)

#warmup

dict2={'one':1,'two':2,'three':3,'four':4,'five':5}

lst2=[1,2,3,4,5]

get\_idx1=operator.itemgetter(1) #透過itemgetter get\_idx1變成callable func

get\_two=operator.itemgetter('two')

print(get\_idx1(lst2)) #2

print(get\_two(dict2)) #2

print(dict2.items())

#dict\_items([('one', 1), ('two', 2), ('three', 3), ('four', 4), ('five', 5)])

#Textbook-2

def sort\_dict\_by\_value(d, reverse = False):

return dict(sorted(d.items(), key = lambda x: x[1], reverse = reverse))

print("Original dictionary elements:")

colors = {'Red': 1, 'Green': 3, 'Black': 5, 'White': 2, 'Pink': 4}

print(colors)

print("\nSort (ascending) the said dictionary elements by value:")

print(sort\_dict\_by\_value(colors))

print("\nSort (descending) the said dictionary elements by value:")

print(sort\_dict\_by\_value(colors, True))

'''2. Write a Python script to add a key to a dictionary.

Sample Dictionary : {0: 10, 1: 20}

Expected Result : {0: 10, 1: 20, 2: 30}

'''

dict1={0: 10, 1: 20}

dict1[2]=30

print(dict1)

#Textbook

d = {0:10, 1:20}

print(d)

d.update({2:30})

print(d)

d.update({2:[40,50]})

print(d)

d.update({2:60})

print(d) #clean out: {0: 10, 1: 20, 2: 60}

'''3. Write a Python script to concatenate the following dictionaries to create a new one

Sample Dictionary :

dic1={1:10, 2:20}

dic2={3:30, 4:40}

dic3={5:50,6:60}

Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

'''

dic1={1:10, 2:20}

dic2={3:30, 4:40}

dic3={5:50,6:60}

# print(dic1+dic2+dic3) #unsupported operand type +

combine=list(dic1.items())+list(dic2.items())+list(dic3.items())

print(dict(combine))

#Textbook

dic4 = {}

for d in (dic1, dic2, dic3): dic4.update(d)

print(dic4)

#timeit

from timeit import timeit

code1='''

dic1={1:10, 2:20}

dic2={3:30, 4:40}

dic3={5:50,6:60}

combine=list(dic1.items())+list(dic2.items())+list(dic3.items())

'''

code2='''

dic1={1:10, 2:20}

dic2={3:30, 4:40}

dic3={5:50,6:60}

dic4 = {}

for d in (dic1, dic2, dic3): dic4.update(d)

'''

print('1M execution of my code takes %.2f sec' % timeit(code1,number=1000000))

print('1M execution of TB code takes %.2f sec' % timeit(code2,number=1000000))

print('TB takes {:.0%} of my code time'.format(timeit(code2,number=1)/timeit(code1,number=1)))

'''4. Write a Python script to check whether a given key already exists

in a dictionary

'''

#warmup

dict1={1:4,2:5,3:6}

print(2 in dict1.keys()) #True

print(9 in dict1.keys()) #False

def gkeid(dic,k):

print(k in dic.keys())

gkeid({'a':'apple','b':'banana','c':'cat'},'c')

gkeid({'a':'apple','b':'banana','c':'cat'},1)

#Textbook

d = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

def is\_key\_present(x):

if x in d:

print('Key is present in the dictionary')

else:

print('Key is not present in the dictionary')

is\_key\_present(5)

is\_key\_present(9)

is\_key\_present(50) #not present

#5. Write a Python program to iterate over dictionaries using for loops

def iodufl(dicts):

return [(i,v) for i,v in dicts.items()]

print(iodufl({1:2,2:3,3:4,4:5}))

#Textbook

d = {'x': 10, 'y': 20, 'z': 30}

for dict\_key, dict\_value in d.items():

print(dict\_key,'->',dict\_value)

'''

6. Write a Python script to generate and print a dictionary that

contains a number (between 1 and n) in the form (x, x\*x).

Sample Dictionary ( n = 5) :

Expected Output : {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

'''

def square(n):

dict1={}

for i in range(1,n+1):

dict1[i]=i\*i

return dict1

print(square(5))

print(square(9))

'''

7. Write a Python script to print a dictionary where the keys are

numbers between 1 and 15 (both included) and the values are the

square of the keys

Sample Dictionary

{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}

'''

lst1=[x\*x for x in range(1,16)]

dict1={}

for i in range(1,16):

dict1[i]=lst1[i-1]

print(dict1)

#Textbook

d=dict()

for x in range(1,16):

d[x]=x\*\*2

print(d)

#8. Write a Python script to merge two Python dictionaries

def merge\_d(dict1,dict2):

dict1.update(dict2) # in-place execution

return dict1

dic1={1:2,2:3,3:4}

dic2={5:6,6:7}

print(merge\_d(dic1,dic2))

#Textbook

d1 = {'a': 100, 'b': 200}

d2 = {'x': 300, 'y': 200}

d = d1.copy()

d.update(d2)

print(d)

#10. Write a Python program to sum all the items in a dictionary

def saiid(dict1):

print(sum(dict1.values()))

dic1={1:2,2:3,3:4,4:5}

saiid(dic1)

#11. Write a Python program to multiply all the items in a dictionary

def maiid(dict1):

result=1

for i in dict1.values():

result\*=i

return result

dic1={1:2,2:3,3:4,4:5}

print(maiid(dic1))

#12. Write a Python program to remove a key from a dictionary

dict1={1:2,2:3,3:4}

print(dict1.pop(1)) #2 (return key-1 value :2)

print(dict1) # {2: 3, 3: 4}

#Textbook

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

print(myDict)

if 'a' in myDict:

del myDict['a']

print(myDict)

#del

lst1=[1,2,3,4]

del lst1[1]

print(lst1) #[1,3,4]

a=1.53

b=2.4

print(a+b)

del a

# print(a+b) # a is not defined

#13. Write a Python program to map two lists into a dictionary

#warmup

dict1={}

lst1=[1,2,3,4]

lst2=['a','b','c','d']

for i in range(len(lst1)):

dict1[lst1[i]]=lst2[i]

print(dict1)

def m2l(lst1,lst2):

dict2={}

for i in range(len(lst1)):

dict2[lst1[i]]=lst2[i]

print(dict2)

m2l(['bal','cin','pit','cle'],['ravens','bengals','steelers','browns'])

# m2l([1,2,3,4,5],[11,12,13,14]) #error if two lists length different

#Textbook

keys = ['red', 'green', 'blue']

values = ['#FF0000','#008000', '#0000FF']

color\_dictionary = dict(zip(keys, values))

print(color\_dictionary)

#網友:

d={key:value for key,value in zip(keys,values)}

print(d)

#14. Write a Python program to sort a given dictionary by key

dict1={'Jack':85,'Diane':76,'Kim':87,'Jeff':96}

print(sorted(dict1)) #['Diane', 'Jack', 'Jeff', 'Kim']

print(sorted(dict1.items()))

#[('Diane', 76), ('Jack', 85), ('Jeff', 96), ('Kim', 87)]

# dict-view, content is tuple-like

print(dict(sorted(dict1.items())))

# {'Diane': 76, 'Jack': 85, 'Jeff': 96, 'Kim': 87}

#Textbook-1

color\_dict = {'red':'#FF0000',

'green':'#008000',

'black':'#000000',

'white':'#FFFFFF'}

for key in sorted(color\_dict):

print("%s: %s" % (key, color\_dict[key]))

#Textbook-2

def sort\_dict\_by\_key(d, reverse = False):

return dict(sorted(d.items(), reverse = reverse))

students = { 'name1': 'Theodore', 'name2': 'Mathew', 'name4': 'Roxanne', 'name3': 'David' }

print("Original dictionary:")

print(students)

print("\nSort the said dictionary by key (Ascending order):")

print(sort\_dict\_by\_key(students))

print("\nSort the said dictionary by key (Descending order):")

print(sort\_dict\_by\_key(students, True))

'''

15. Write a Python program to get the maximum and minimum values of

a dictionary

'''

def max\_min\_dic(dicts):

print("max : %d" % max(dicts.values()))

print("min : %d" % min(dicts.values()))

max\_min\_dic({1:24,2:56,3:18,4:74,5:22})

max\_min\_dic({1:2.35,2:5.6,3:1.8,4:4.66,5:2.21})

# max : 5 min : 1

#Textbook

my\_dict = {'x':500, 'y':5874, 'z': 560}

key\_max = max(my\_dict.keys(), key=(lambda k: my\_dict[k]))

key\_min = min(my\_dict.keys(), key=(lambda k: my\_dict[k]))

print('Maximum Value: ',my\_dict[key\_max])

print('Minimum Value: ',my\_dict[key\_min])

#16. Write a Python program to get a dictionary from an object's fields

#Textbook

class dictObj(object):

def \_\_init\_\_(self):

self.x = 'red'

self.y = 'Yellow'

self.z = 'Green'

def do\_nothing(self):

pass

test = dictObj()

print(test.\_\_dict\_\_)

#warmup

lst1=list()

dict1={1:2,2:3}

print(lst1) # []

print(lst1.\_\_class\_\_) # <class 'list'>

print(dict1.\_\_class\_\_) # <class 'dict'>

print(dict1.\_\_doc\_\_) #documentation

'''

dict() -> new empty dictionary

dict(mapping) -> new dictionary initialized from a mapping object's

(key, value) pairs

dict(iterable) -> new dictionary initialized as if via:

d = {}

for k, v in iterable:

d[k] = v

dict(\*\*kwargs) -> new dictionary initialized with the name=value pairs

in the keyword argument list. For example: dict(one=1, two=2)

'''

#16. Write a Python program to get a dictionary from an object's fields

#Textbook

class dictObj(object):

def \_\_init\_\_(self):

self.x = 'red'

self.y = 'Yellow'

self.z = 'Green'

def do\_nothing(self):

pass

test = dictObj()

print(test.\_\_dict\_\_)

#warmup

lst1=list()

dict1={1:2,2:3}

print(lst1) # []

print(lst1.\_\_class\_\_) # <class 'list'>

print(dict1.\_\_class\_\_) # <class 'dict'>

# print(lst1.\_\_dict\_\_) #list object has no attribute:\_\_dict\_\_

print(dict1.\_\_doc\_\_) #documentation

'''

dict() -> new empty dictionary

dict(mapping) -> new dictionary initialized from a mapping object's

(key, value) pairs

dict(iterable) -> new dictionary initialized as if via:

d = {}

for k, v in iterable:

d[k] = v

dict(\*\*kwargs) -> new dictionary initialized with the name=value pairs

in the keyword argument list. For example: dict(one=1, two=2)

'''

#17. Write a Python program to remove duplicates from the dictionary

dict1={1:2,2:3,3:4,1:2,5:6}

print(dict1) # {1: 2, 2: 3, 3: 4, 5: 6

print(dict(set(dict1.items())))

# Are you serious? dictionaries don't even accept duplicate keys already. Your solution literally does nothing

#Textbook

student\_data = {'id1':

{'name': ['Sara'],

'class': ['V'],

'subject\_integration': ['english, math, science']

},

'id2':

{'name': ['David'],

'class': ['V'],

'subject\_integration': ['english, math, science']

},

'id3':

{'name': ['Sara'],

'class': ['V'],

'subject\_integration': ['english, math, science']

},

'id4':

{'name': ['Surya'],

'class': ['V'],

'subject\_integration': ['english, math, science']

},

}

result = {}

for key,value in student\_data.items():

if value not in result.values():

result[key] = value

print(result)

#18. Write a Python program to check if a dictionary is empty or not

dict\_empty={}

dict1={1:2,2:3,3:4}

dict2={}

print(dict1==dict\_empty)

print(dict2==dict\_empty)

print(bool(dict1==dict\_empty))

print(bool(dict1)) #True

print(bool(dict2)) #False

#Textbook

my\_dict = {}

if not bool(my\_dict):

print("Dictionary is empty")

'''

19. Write a Python program to combine two dictionary by adding values

for common keys

d1 = {'a': 100, 'b': 200, 'c':300}

d2 = {'a': 300, 'b': 200, 'd':400}

Sample output: Counter({'a': 400, 'b': 400, 'd': 400, 'c': 300})

'''

#warmup

d1 = {'a': 100, 'b': 200, 'c':300}

d2 = {'a': 300, 'b': 200, 'd':400}

print({key:value for key in d1.keys() for value in d1.values()})

# {'a': 300, 'b': 300, 'c': 300}

print(d2['b']+d1['b'])

def c2dbavfc(dict1,dict2):

Counter={}

for i in dict1.keys():

Counter.update({i:0})

for j in dict2.keys():

Counter.update({j:0})

for k in Counter.keys():

Counter[k]=dict1[k]+dict2[k]

#there will be 'key error' since no d2['c']

return Counter

#print(c2dbavfc(d1,d2))

#Textbook

from collections import Counter

d1 = {'a': 100, 'b': 200, 'c':300}

d2 = {'a': 300, 'b': 200, 'd':400}

d = Counter(d1) + Counter(d2)

print(d)

#collections.Counter

lst1=[1,2,3,4,5]

lst2=[6,7,8,9,10]

print(Counter(lst1)+Counter(lst2))

# Counter({1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1, 9: 1, 10: 1})

# 計數器的概念

tup1=(1,2,3,4)

tup2=(5,6,7,8)

print(Counter(tup1)+Counter(tup2))

# Counter({1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1})

#返回都是字典形式 => 種類:個數

'''

20. Write a Python program to print all distinct values in a dictionary.

Sample Data : [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"},{"VIII":"S007"}]

Expected Output : Unique Values: {'S005', 'S002', 'S007', 'S001', 'S009'}

'''

def dvid(lst):

lst0=[]

for dic in lst:

# print(\*dic.values())

lst0.append(\*dic.values())

print('Unique Values: ',set(lst0))

# set是用 { } 包裝起來的

dvid([{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"},{"VIII":"S007"}])

dvid([{"Soccer":"Tim"},{"Soccer":"Kenny"},{"Art":"Enya"},{"Art":"Kenny"},{"Lego":"Andy"},{"Lego":"Kenny"}])

#Textbook

L = [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"},{"VIII":"S007"}]

print("Original List: ",L)

u\_value = set( val for dic in L for val in dic.values())

print("Unique Values: ",u\_value)

#dict.values()是被packed起來的 所以要再用一個for loop去解它

'''

21. Write a Python program to create and display all combinations

of letters, selecting each letter from a different key in a dictionary.

Sample data : {'1':['a','b'], '2':['c','d']}

Expected Output:

ac

ad

bc

bd

'''

#warmup

from itertools import product as pr

print(\*pr(['a','b'],['c','d']))

def acol(dicts):

v1=[v for k,v in dicts.items()]

print(v1)

print(len(v1))

for cg in pr(v1[0],v1[1]):

print(cg[0],cg[1],sep='')

acol({'1':['a','b'], '2':['c','d']})

acol({'1':['k','s','r'], '2':['a','i','u','e','o']})

print()

#Textbook

import itertools

d ={'1':['a','b'], '2':['c','d'],'3':['r','t','x']}

for combo in itertools.product(\*[d[k] for k in sorted(d.keys())]):

print(''.join(combo))

#can adopt more than 2 dict.keys() !

'''

22. Write a Python program to find the highest 3 values of

corresponding keys in a dictionary

'''

from collections import Counter

def h3vck(dicts):

c=Counter(dicts)

print(c.most\_common(3))

h3vck({'cin':12,'bal':10,'cle':6,'pit':9,'ne':8,'buf':13,'mia':9,'nyj':7})

#Textbook

from heapq import nlargest

my\_dict = {'a':500, 'b':5874, 'c': 560,'d':400, 'e':5874, 'f': 20}

three\_largest = nlargest(3, my\_dict, key=my\_dict.get)

print(three\_largest)

'''

23. Write a Python program to combine values in a list of dictionaries.

Sample data: [{'item': 'item1', 'amount': 400}, {'item': 'item2', 'amount': 300}, {'item': 'item1', 'amount': 750}]

Expected Output: Counter({'item1': 1150, 'item2': 300})

'''

from collections import Counter

#mine: failed

l1= [{'item': 'item1', 'amount': 400}, {'item': 'item2', 'amount': 300}, {'item': 'item1', 'amount': 750}]

c=Counter()

for i in l1:

c.update(i)

print(c) #Counter({'item': 'item1item2item1', 'amount': 1450})

# not right! all stacked together!

#mine: success

l2=[{j['item']:j['amount']} for j in l1]

print(l2)

c=Counter()

for j in l2:

c.update(j)

print(c)

#Textbook

item\_list = [{'item': 'item1', 'amount': 400}, {'item': 'item2', 'amount': 300}, {'item': 'item1', 'amount': 750}]

result = Counter()

for d in item\_list:

result[d['item']] += d['amount']

print(result)

'''

24. Write a Python program to create a dictionary from a string

Note: Track the count of the letters from the string.

Sample string : 'w3resource'

Expected output: {'w': 1, '3': 1, 'r': 2, 'e': 2, 's': 1, 'o': 1, 'u': 1, 'c': 1}

'''

from collections import Counter

def cdfs(strs):

return Counter(strs)

print(cdfs('w3resource'))

print(cdfs('pittsburghpenguins'))

#Textbook

from collections import defaultdict, Counter

str1 = 'w3resource'

my\_dict = {}

for letter in str1:

my\_dict[letter] = my\_dict.get(letter, 0) + 1

print(my\_dict)

#dict.get(key, default=None)

#25. Write a Python program to print a dictionary in table format

def pditf(dicts):

for k,v in dicts.items():

print(k,':\t',v)

pditf({'k1':'v1','key2':'v2','k3':'v3','4th':'value4'})

#Textbook

my\_dict = {'C1':[1,2,3],'C2':[5,6,7],'C3':[9,10,11]}

for row in zip(\*([key] + (value) for key, value in sorted(my\_dict.items()))):

print(\*row)

'''

26. Write a Python program to count the values associated with a key

in a dictionary

Expected Output:

6

2

'''

#warmup

dict1={'k1':3,'k2':5,'k3':7,'k1':9}

print(dict1) #{'k1': 9, 'k2': 5, 'k3': 7}

#not quite understand 題意 since value cover the previous one for same key

#Textbook- dict of dict, then count the value

student = [{'id': 1, 'success': True, 'name': 'Lary'},

{'id': 2, 'success': False, 'name': 'Rabi'},

{'id': 3, 'success': True, 'name': 'Alex'}]

print(sum(d['id'] for d in student))

print(sum(d['success'] for d in student))

'''

27. Write a Python program to convert a list into a nested dictionary

of keys

nested dictionary: dictionary inside dictionary

'''

#warmup

d1={}

d1.update({1:2,2:3})

d1.update({3:4,4:5})

print(d1)

d2={1:6,2:7,3:8,4:9}

d1.update(d2)

print(d1) #{1:6,2:7,3:8,4:9}

#how to append while dict2 has same key as dict1? probably into list first?

lst1=[('Mary','32','F','Yes'),('Tom','25','M','No'),('Regina','44','F','Yes')]

d={}

lst0=[]

for person in lst1:

d0={}

for i,info in enumerate(person):

if i==0: d0['name']=info

if i==1: d0['age']=info

if i==2: d0['sex']=info

if i==3: d0['License']=info

lst0.append(d0)

for i in range(1,len(lst0)+1):

d[i]=lst0[i-1]

print(d)

#Textbook

num\_list = [1, 2, 3, 4]

new\_dict = current = {}

for name in num\_list:

current[name] = {}

current = current[name]

print(new\_dict)

#網友

num\_list = [1, 2, 3, 4]

tree\_dict = {}

for key in num\_list[::-1]:

tree\_dict = {key: tree\_dict}

print(tree\_dict)

#28. Write a Python program to sort a list alphabetically in a dictionary

dict1={'afc\_south':['ind','jax','hou','ten'],'afc\_north':['cle','pit','bal','cin']}

print(sorted(sorted(teams) for div,teams in dict1.items()))

# [['bal', 'cin', 'cle', 'pit'], ['hou', 'ind', 'jax', 'ten']]

#Textbook

num = {'n1': [2, 3, 1], 'n2': [5, 1, 2], 'n3': [3, 2, 4]}

sorted\_dict = {x: sorted(y) for x, y in num.items()}

print(sorted\_dict)

#網友

d= {'n1': [2, 3, 1], 'n2': [5, 1, 2], 'n3': [3, 2, 4]}

for i in d:

d[i]=sorted(d[i])

print(d)

#29. Write a Python program to remove spaces from dictionary keys

dict1={'teams':['steeler s','bengal s','brown s','raven s'],'cities':['pit','cin','cle','bal']}

dict1['teams']=[team.replace(' ','') for team in dict1['teams']]

print(dict1)

#Textbook

student\_list = {'S 001': ['Math', 'Science'], 'S 002': ['Math', 'English']}

print("Original dictionary: ",student\_list)

student\_dict = {x.translate({32: None}): y for x, y in student\_list.items()}

print("New dictionary: ",student\_dict)

#str.translate

# https://www.runoob.com/python/att-string-translate.html

print(ord(' ')) #32

'''

30. Write a Python program to get the top three items in a shop.

Sample data: {'item1': 45.50, 'item2':35, 'item3': 41.30, 'item4':55, 'item5': 24}

Expected Output:

item4 55

item1 45.5

item3 41.3

'''

shop={'item1': 45.50, 'item2':35, 'item3': 41.30, 'item4':55, 'item5': 24}

sort\_item=sorted(shop,key=lambda x:shop[x],reverse=True)

top3=sort\_item[:3]

for i in top3:

print(i, shop[i], sep=' ')

#Textbook

from heapq import nlargest

from operator import itemgetter

items = {'item1': 45.50, 'item2':35, 'item3': 41.30, 'item4':55, 'item5': 24}

for name, value in nlargest(3, items.items(), key=itemgetter(1)):

print(name, value)

#nlargest is equivalent to sorted(iterable,key,reverse=True)[:n]

# nlargest(n,iterable,key)

#31. Write a Python program to get the key, value and item in a dictionary

def gkvi(dicts):

print(dicts.keys())

print(dicts.values())

print(dicts.items())

gkvi({'teams':['steeler s','bengal s','brown s','raven s'],'cities':['pit','cin','cle','bal']})

#Textbook

dict\_num = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

print("key value count")

for count, (key, value) in enumerate(dict\_num.items(), 1):

print(key,' ',value,' ', count)

#enumerate(iterable,start=0)

for i,count in enumerate(range(5),16):

print(i,count)

#但這裡是count放在前面

#32. Write a Python program to print a dictionary line by line

def plbl(dicts):

print(\*dicts.items(),sep='\n')

plbl({'teams':['steeler s','bengal s','brown s','raven s'],'cities':['pit','cin','cle','bal']})

#Textbook

students = {'Aex':{'class':'V',

'rolld\_id':2},

'Puja':{'class':'V',

'roll\_id':3}}

for a in students:

print(a)

for b in students[a]:

print (b,':',students[a][b])

#33. Write a Python program to check if multiple keys exist in a dictionary

from collections import Counter

students = {'Aex':{'class':'V',

'rolld\_id':2},

'Puja':{'class':'V',

'roll\_id':3}}

print(len(students.keys())>1)

#Textbook

student = {

'name': 'Alex',

'class': 'V',

'roll\_id': '2'

}

print(student.keys())

print(student.keys() >= {'class', 'name'}) #True

#meaning: check if keys{'class','name'} exist in student-dict or not

print(student.keys() >= {'name', 'Alex'}) #False

print(student.keys() >= {'roll\_id', 'name'}) #True

#warmup

d1={1:2,3:4,5:6,7:8}

print(d1.keys()>={3,7})

# dict.keys(): a set-like object

set1={1,2,3,4,5,6}

set2={1,1,2,4,4,6,6,6,6,6,6,6}

print(set1>set2) #True

set3={12,1}

print(set1>set3) #False

print(set3>set1) #False

print(set3==set1) #False

# so set 3 & set1 : not contained with each other, neither >, <, ==

set4={12,12,1,12,1,1,1,1,1}

print(set4==set3) #True, both sets have same elements

'''

34. Write a Python program to count the number of items in a

dictionary value that is a list

'''

#how to identify if an obj is a list? type!

unid1=[1,2,3,4]

unid2=(1,2,3,4)

unid3=[4,5,6,7]

print(type(unid1)==type(unid3)) #True

print(type(unid1)==type(unid2)) #False

unid4={7,8,9,0}

unid5={1:2,2:3,3:''}

print(type(unid4)==type(unid5)) #False

def cnidvisl(dicts):

lst\_sample=['this','is','list','sample']

count=0

for i in dicts.values():

if type(i)==type(lst\_sample):

count+=1

return count

print(cnidvisl({1:2,2:3,3:[4,5],6:[7,8,9],7:'wtf'})) # 2

#Textbook- different meaning

dict = {'Alex': ['subj1', 'subj2', 'subj3'], 'David': ['subj1', 'subj2']}

ctr = sum(map(len, dict.values()))

print(ctr) #5

print(\*map(len, dict.values())) # 3 2

'''

35. Write a Python program to sort Counter by value.

Sample data : {'Math':81, 'Physics':83, 'Chemistry':87}

Expected data: [('Chemistry', 87), ('Physics', 83), ('Math', 81)]

'''

from operator import itemgetter

def sbv(dicts):

return sorted(dicts.items(),key=itemgetter(1), reverse=True)

print(sbv({'Math':81, 'Physics':83, 'Chemistry':87}))

#Textbook

from collections import Counter

x = Counter({'Math':81, 'Physics':83, 'Chemistry':87})

print(x.most\_common())

'''

36. Write a Python program to create a dictionary from two lists

without losing duplicate values.

Sample lists: ['Class-V', 'Class-VI', 'Class-VII', 'Class-VIII'], [1, 2, 2, 3]

Expected Output: defaultdict(<class 'set'>, {'Class-V': {1}, 'Class-VI': {2}, 'Class-VII': {2}, 'Class-VIII': {3}})

'''

from collections import defaultdict

# defaultdict就是不用管原本的dict裡是否存在某個key

lst1=['Class-V', 'Class-VI', 'Class-VII', 'Class-VIII']

lst2=[1, 2, 2, 3]

lst12=list(zip(lst1,lst2))

print(lst12)

mydict=defaultdict(list) #default\_factory set to 'list'

for k,v in lst12:

mydict[k]=v

print(mydict)

# defaultdict(<class 'list'>, {'Class-V': 1, 'Class-VI': 2, 'Class-VII': 2, 'Class-VIII': 3})

#Textbook

class\_list = ['Class-V', 'Class-VI', 'Class-VII', 'Class-VIII']

id\_list = [1, 2, 2, 3]

temp = defaultdict(set)

for c, i in zip(class\_list, id\_list):

temp[c].add(i)

print(temp)

#set.add - only 'set' has submodule 'add'

set1={1,2,3,4}

set1.add(5)

print(set1) {1,2,3,4,5}

#37. Write a Python program to replace dictionary values with their sums

def rdvws(dicts):

return {k:sum(v) for k,v in dicts.items()}

print(rdvws({'a':[1,2,3],'b':[4,5,6],'c':[7,8,9]}))

#Textbook

def sum\_math\_v\_vi\_average(list\_of\_dicts):

for d in list\_of\_dicts:

n1 = d.pop('V')

n2 = d.pop('VI')

d['V+VI'] = (n1 + n2)/2

return list\_of\_dicts

student\_details= [

{'id' : 1, 'subject' : 'math', 'V' : 70, 'VI' : 82},

{'id' : 2, 'subject' : 'math', 'V' : 73, 'VI' : 74},

{'id' : 3, 'subject' : 'math', 'V' : 75, 'VI' : 86}

]

print(sum\_math\_v\_vi\_average(student\_details))

#dict.pop

d1={'a':0.765,'b':0.473,'c':0.651}

print(d1.pop('b')) # 0.473 (return 此k的value)

print(d1) # {'a': 0.765, 'c': 0.651}

'''

38. Write a Python program to match key values in two dictionaries.

Sample dictionary: {'key1': 1, 'key2': 3, 'key3': 2}, {'key1': 1, 'key2': 2}

Expected output: key1: 1 is present in both x and y

'''

x= {'key1': 1, 'key2': 3, 'key3': 2}

y= {'key1': 1, 'key2': 2}

for kx,vx in x.items():

for ky,vy in y.items():

if kx==ky and vx==vy:

print(kx,': ', vx,' is present in both x and y')

#Textbook

x = {'key1': 1, 'key2': 3, 'key3': 2}

y = {'key1': 1, 'key2': 2}

for (key, value) in set(x.items()) & set(y.items()):

print('%s: %s is present in both x and y' % (key, value))

#set1 & set2 : 可作為迴圈的主體

set1={1,2,3,4}

set2={3,4,5,6}

print(set1 & set2) # {3,4}

'''

39. Write a Python program to store dictionary data in a JSON file.

Original dictionary:

{'students': [{'firstName': 'Nikki', 'lastName': 'Roysden'}, {'firstName': 'Mervin', 'lastName': 'Friedland'}, {'firstName': 'Aron ', 'lastName': 'Wilkins'}], 'teachers': [{'firstName': 'Amberly', 'lastName': 'Calico'}, {'firstName': 'Regine', 'lastName': 'Agtarap'}]}

<class 'dict'>

Json file to dictionary:

{'students': [{'firstName': 'Nikki', 'lastName': 'Roysden'}, {'firstName': 'Mervin', 'lastName': 'Friedland'}, {'firstName': 'Aron ', 'lastName': 'Wilkins'}], 'teachers': [{'firstName': 'Amberly', 'lastName': 'Calico'}, {'firstName': 'Regine', 'lastName': 'Agtarap'}]}

'''

import json

d={'students': [{'firstName': 'Nikki', 'lastName': 'Roysden'}, {'firstName': 'Mervin', 'lastName': 'Friedland'}, {'firstName': 'Aron ', 'lastName': 'Wilkins'}], 'teachers': [{'firstName': 'Amberly', 'lastName': 'Calico'}, {'firstName': 'Regine', 'lastName': 'Agtarap'}]}

j1=json.dumps(d)

print(j1,'\n',type(j1))

#Textbook

print("Original dictionary:")

print(d)

print(type(d))

import json

with open("dictionary", "w") as f:

json.dump(d, f, indent = 4, sort\_keys = True)

print("\nJson file to dictionary:")

with open('dictionary') as f:

data = json.load(f)

print(data)

# open(file, mode='r')

'''

40. Write a Python program to create a dictionary of keys x, y, and z

where each key has as value a list from 11-20, 21-30, and 31-40

respectively. Access the fifth value of each key from the dictionary.

{'x': [11, 12, 13, 14, 15, 16, 17, 18, 19],

'y': [21, 22, 23, 24, 25, 26, 27, 28, 29],

'z': [31, 32, 33, 34, 35, 36, 37, 38, 39]}

15

25

35

x has value [11, 12, 13, 14, 15, 16, 17, 18, 19]

y has value [21, 22, 23, 24, 25, 26, 27, 28, 29]

z has value [31, 32, 33, 34, 35, 36, 37, 38, 39]

'''

kl=['x','y','z']

vl=[list(range(11,20)),list(range(21,30)),list(range(31,40))]

d=dict(zip(kl,vl))

print(d)

print(d['x'][4],d['y'][4],d['z'][4], sep='\n')

#Textbook

from pprint import pprint

dict\_nums = dict(x=list(range(11, 20)), y=list(range(21, 30)), z=list(range(31, 40)))

pprint(dict\_nums) #在x,y,z可自動換行

print(dict\_nums["x"][4])

print(dict\_nums["y"][4])

print(dict\_nums["z"][4])

for k,v in dict\_nums.items():

print(k, "has value", v)

'''

41. Write a Python program to drop empty items from a given dictionary.

Original Dictionary:

{'c1': 'Red', 'c2': 'Green', 'c3': None}

New Dictionary after dropping empty items:

{'c1': 'Red', 'c2': 'Green'}

'''

def dei(dicts):

return {k:v for k,v in dicts.items() if v}

print(dei({'c1': 'Red', 'c2': 'Green', 'c3': None}))

#Textbook

dict1 = {'c1': 'Red', 'c2': 'Green', 'c3':None}

print("Original Dictionary:")

print(dict1)

print("New Dictionary after dropping empty items:")

dict1 = {key:value for (key, value) in dict1.items() if value is not None}

print(dict1)

'''

42. Write a Python program to filter a dictionary based on values.

Original Dictionary:

{'Cierra Vega': 175, 'Alden Cantrell': 180, 'Kierra Gentry': 165, 'Pierre Cox': 190}

Marks greater than 170:

{'Cierra Vega': 175, 'Alden Cantrell': 180, 'Pierre Cox': 190}

'''

def fdbov(dicts,ll):

return {k:v for k,v in dicts.items() if v>ll}

print(fdbov({'Cierra Vega': 175, 'Alden Cantrell': 180, 'Kierra Gentry': 165, 'Pierre Cox': 190},170))

print(fdbov({'Cierra Vega': 175, 'Alden Cantrell': 180, 'Kierra Gentry': 165, 'Pierre Cox': 190},185))

'''

43. Write a Python program to convert more than one list to a

nested dictionary

Original strings:

['S001', 'S002', 'S003', 'S004']

['Adina Park', 'Leyton Marsh', 'Duncan Boyle', 'Saim Richards']

[85, 98, 89, 92]

Nested dictionary:

[{'S001': {'Adina Park': 85}}, {'S002': {'Leyton Marsh': 98}}, {'S003': {'Duncan Boyle': 89}}, {'S004': {'Saim Richards': 92}}]

'''

lst1=['S001', 'S002', 'S003', 'S004']

lst2=['Adina Park', 'Leyton Marsh', 'Duncan Boyle', 'Saim Richards']

lst3=[85, 98, 89, 92]

d1=dict(zip(lst2,lst3))

d2={k:v for k,v in d1.items()}

print(\*zip(lst1,d2.items())) # ('S001', ('Adina Park', 85))...

# not fit the solution!

#Textbook

def nested\_dictionary(l1, l2, l3):

result = [{x: {y: z}} for (x, y, z) in zip(l1, l2, l3)]

return result

#要點1: dict-comprehension 可以是customized format !

#要點2: zip(...)的element形式是tuple

#要點3: 在一個for中表達數個變數 可用tuple框起來表示!

student\_id = ["S001", "S002", "S003", "S004"]

student\_name = ["Adina Park", "Leyton Marsh", "Duncan Boyle", "Saim Richards"]

student\_grade = [85, 98, 89, 92]

print("Original strings:")

print(student\_id)

print(student\_name)

print(student\_grade)

print("\nNested dictionary:")

print(nested\_dictionary(student\_id, student\_name, student\_grade))

'''

44. Write a Python program to filter the height and weight of students,

which are stored in a dictionary.

Original Dictionary:

{'Cierra Vega': (6.2, 70), 'Alden Cantrell': (5.9, 65), 'Kierra Gentry': (6.0, 68), 'Pierre Cox': (5.8, 66)}

Height >= 6ft and Weight>= 70kg:

{'Cierra Vega': (6.2, 70)}

'''

def fhw(dicts,hll,wll):

print({x:(y,z) for x,(y,z) in dicts.items() if y>=hll and z>=wll})

fhw({'Cierra Vega': (6.2, 70), 'Alden Cantrell': (5.9, 65), 'Kierra Gentry': (6.0, 68), 'Pierre Cox': (5.8, 66)},6,70)

#Textbook

def filter\_data(students):

result = {k: s for k, s in students.items() if s[0] >=6.0 and s[1] >=70}

return result

'''

45. Write a Python program to verify that all values in a dictionary

are the same.

Original Dictionary:

{'Cierra Vega': 12, 'Alden Cantrell': 12, 'Kierra Gentry': 12, 'Pierre Cox': 12}

Check all are 12 in the dictionary.

True

Check all are 10 in the dictionary.

False

'''

d={'Cierra Vega': 12, 'Alden Cantrell': 12, 'Kierra Gentry': 12, 'Pierre Cox': 12}

print('Check all are 12 in the dictionary:')

print( set([v for k,v in d.items()]) == {12})

print('Check all are 10 in the dictionary:')

print( set([v for k,v in d.items()]) == {10})

#Textbook

def value\_check(students, n):

result = all(x == n for x in students.values())

return result

'''

46. Write a Python program to create a dictionary grouping a sequence

of key-value pairs into a dictionary of lists

Original list:

[('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]

Grouping a sequence of key-value pairs into a dictionary of lists:

{'yellow': [1, 3], 'blue': [2, 4], 'red': [1]}

'''

def gsokvp(lsts):

return {ele[0]:ele[1] for ele in lsts}

print(gsokvp([('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]))

# {'yellow': 3, 'blue': 4, 'red': 1}

# don't know how to handle duplicated key with different value

#Textbook

def grouping\_dictionary(l):

result = {}

for k, v in l:

result.setdefault(k,[]).append(v)

return result

colors = [('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]

print("Original list:")

print(colors)

print("\nGrouping a sequence of key-value pairs into a dictionary of lists:")

print(grouping\_dictionary(colors))

#warmup - dict.setdefault(key,default=None)

#Returns the value of the item with the specified key

# Insert key with value of default if key is not in dictionary

d1= {'a':1,'b':2,'d':4}

d1.setdefault('c','x')

print(d1) #{'a': 1, 'b': 2, 'd': 4, 'c': 'x'}

d1.setdefault('e')

print(d1) #{'a': 1, 'b': 2, 'd': 4, 'c': 'x', 'e': None}

print(d1.setdefault('b','zz')) # 2

#因為b已經是既有key,故zz沒用, 還是回傳'2'

#除非本體用list 再用append, 如上例

'''

47. Write a Python program to split a given dictionary of lists

into lists of dictionaries.

Original dictionary of lists:

{'Science': [88, 89, 62, 95], 'Language': [77, 78, 84, 80]}

Split said dictionary of lists into list of dictionaries:

[{'Science': 88, 'Language': 77}, {'Science': 89, 'Language': 78}, {'Science': 62, 'Language': 84}, {'Science': 95, 'Language': 80}]

'''

from operator import itemgetter

d={'Science': [88, 89, 62, 95], 'Language': [77, 78, 84, 80]}

# d1={}

lst1=[]

for i in range(4):

ig=itemgetter(i)

# d1.update({k:ig(v) for k,v in d.items()})

lst1.append({k:ig(v) for k,v in d.items()})

print(lst1)

#Textbook

def list\_of\_dicts(marks):

keys = marks.keys()

vals = zip(\*[marks[k] for k in keys])

result = [dict(zip(keys, v)) for v in vals]

return result

marks = {'Science': [88, 89, 62, 95], 'Language': [77, 78, 84, 80]}

print("Original dictionary of lists:")

print(marks)

print("\nSplit said dictionary of lists into list of dictionaries:")

print(list\_of\_dicts(marks))

#網友

d={'Science': [88, 89, 62, 95], 'Language': [77, 78, 84, 80]}

res=[dict(zip(d.keys(), itm)) for itm in list(zip(\*d.values()))]

print(res)

'''

48. Write a Python program to remove a specified dictionary

from a given list.

Original list of dictionary:

[{'id': '#FF0000', 'color': 'Red'}, {'id': '#800000', 'color': 'Maroon'}, {'id': '#FFFF00', 'color': 'Yellow'}, {'id': '#808000', 'color': 'Olive'}]

Remove id #FF0000 from the said list of dictionary:

[{'id': '#800000', 'color': 'Maroon'}, {'id': '#FFFF00', 'color': 'Yellow'}, {'id': '#808000', 'color': 'Olive'}]

'''

#warmup

dict1={'id': '#FF0000', 'color': 'Red'}

print(dict1.get('color')) #Red

print(dict1.get('id')) # #FF0000

print(dict1.pop('id')) # #FF0000

print(dict1) # {'color': 'Red'}

dict2={'id': '#FF0000', 'color': 'Red'}

print(dict2.popitem()) #('color', 'Red')

print(dict2) #{'id': '#FF0000'}

#pop是指定key後 return 該key的value; popitem是return 整組item(以tuple表示)

def rmsd(lsts, rmid):

result=[]

for d in lsts:

if d['id']!=rmid:

result.append(d)

return result

print(rmsd([{'id': '#FF0000', 'color': 'Red'}, {'id': '#800000', 'color': 'Maroon'}, {'id': '#FFFF00', 'color': 'Yellow'}, {'id': '#808000', 'color': 'Olive'}],'#FF0000'))

#Textbook

def remove\_dictionary(colors, r\_id):

colors[:] = [d for d in colors if d.get('id') != r\_id]

return colors

'''

49. Write a Python program to convert string values of a given

dictionary into integer/float datatypes.

Original list:

[{'x': '10', 'y': '20', 'z': '30'}, {'p': '40', 'q': '50', 'r': '60'}]

String values of a given dictionary, into integer types:

[{'x': 10, 'y': 20, 'z': 30}, {'p': 40, 'q': 50, 'r': 60}]

Original list:

[{'x': '10.12', 'y': '20.23', 'z': '30'}, {'p': '40.00', 'q': '50.19', 'r': '60.99'}]

String values of a given dictionary, into float types:

[{'x': 10.12, 'y': 20.23, 'z': 30.0}, {'p': 40.0, 'q': 50.19, 'r': 60.99}]

'''

def csviif(lsts,dtypes):

result=[]

if dtypes==int:

for d in lsts:

d1={k:int(v) for k,v in d.items()}

result.append(d1)

elif dtypes==float:

for d in lsts:

d2={k:float(v) for k,v in d.items()}

result.append(d2)

return result

print(csviif([{'x': '10', 'y': '20', 'z': '30'}, {'p': '40', 'q': '50', 'r': '60'}],int))

print(csviif([{'x': '10.12', 'y': '20.23', 'z': '30'}, {'p': '40.00', 'q': '50.19', 'r': '60.99'}],float))

print(csviif([{'x': '10', 'y': '20', 'z': '30'}, {'p': '40', 'q': '50', 'r': '60'}],float))

#Textbook

def convert\_to\_int(lst):

result = [dict([a, int(x)] for a, x in b.items()) for b in lst]

return result

def convert\_to\_float(lst):

result = [dict([a, float(x)] for a, x in b.items()) for b in lst]

return result

nums =[{ 'x':'10' , 'y':'20' , 'z':'30' }, { 'p':'40', 'q':'50', 'r':'60'}]

print("Original list:")

print(nums)

print("\nString values of a given dictionary, into integer types:")

print(convert\_to\_int(nums))

nums =[{ 'x':'10.12', 'y':'20.23', 'z':'30'}, { 'p':'40.00', 'q':'50.19', 'r':'60.99'}]

print("\nOriginal list:")

print(nums)

print("\nString values of a given dictionary, into float types:")

print(convert\_to\_float(nums))

'''

50. A Python dictionary contains List as a value. Write a Python program

to clear the list values in the said dictionary

Original Dictionary:

{'C1': [10, 20, 30], 'C2': [20, 30, 40], 'C3': [12, 34]}

Clear the list values in the said dictionary:

{'C1': [], 'C2': [], 'C3': []}

'''

#warmup

dict1={'C1': [10, 20, 30]}

dict1.clear()

print(dict1) #{} that is all clear (incl. keys)

def clv(dicts):

return {k:[] for k,v in dicts.items()}

print(clv({'C1': [10, 20, 30], 'C2': [20, 30, 40], 'C3': [12, 34]}))

#Textbook

def test(dictionary):

for key in dictionary:

dictionary[key].clear()

return dictionary

'''

51. A Python Dictionary contains List as a value. Write a Python program

to update the list values in the said dictionary.

Original Dictionary:

{'Math': [88, 89, 90], 'Physics': [92, 94, 89], 'Chemistry': [90, 87, 93]}

Update the list values of the said dictionary:

{'Math': [89, 90, 91], 'Physics': [90, 92, 87], 'Chemistry': [90, 87, 93]}

'''

def upscore(dicts,newscore):

print(dict(zip(list(dicts.keys()),newscore)))

dict1={'Math': [88, 89, 90], 'Physics': [92, 94, 89], 'Chemistry': [90, 87, 93]}

new1=[[89, 90, 91],[90, 92, 87],[90, 87, 93]]

upscore(dict1,new1)

#Textbook-different meaning!

def test(dictionary):

dictionary['Math'] = [x+1 for x in dictionary['Math']]

dictionary['Physics'] = [x-2 for x in dictionary['Physics']]

return dictionary

'''

52. Write a Python program to extract a list of values from a

given list of dictionaries.

Original Dictionary:

[{'Math': 90, 'Science': 92}, {'Math': 89, 'Science': 94}, {'Math': 92, 'Science': 88}]

Extract a list of values from said list of dictionaries where subject = Science

[92, 94, 88]

Original Dictionary:

[{'Math': 90, 'Science': 92}, {'Math': 89, 'Science': 94}, {'Math': 92, 'Science': 88}]

Extract a list of values from said list of dictionaries where subject = Math

[90, 89, 92]

'''

#warmup

dict1={'Math': 90, 'Science': 92}

print(dict1.get('Science')) #92

def elov(lsts,subject):

return [d.get(subject) for d in lsts]

print(elov([{'Math': 90, 'Science': 92}, {'Math': 89, 'Science': 94}, {'Math': 92, 'Science': 88}],'Science'))

print(elov([{'Math': 90, 'Science': 92}, {'Math': 89, 'Science': 94}, {'Math': 92, 'Science': 88}],'Math'))

#Textbook

def test(lst, marks):

result = [d[marks] for d in lst if marks in d]

return result

'''

53. Write a Python program to find the length of a dictionary of values

Original Dictionary:

{1: 'red', 2: 'green', 3: 'black', 4: 'white', 5: 'black'}

Length of dictionary values:

{'red': 3, 'green': 5, 'black': 5, 'white': 5}

Original Dictionary:

{'1': 'Austin Little', '2': 'Natasha Howard', '3': 'Alfred Mullins', '4': 'Jamie Rowe'}

Length of dictionary values:

{'Austin Little': 13, 'Natasha Howard': 14, 'Alfred Mullins': 14, 'Jamie Rowe': 10}

'''

import cProfile

def lodv(dicts):

return {v:len(v) for v in dicts.values()}

print(lodv({1: 'red', 2: 'green', 3: 'black', 4: 'white', 5: 'black'}))

print(lodv({'1': 'Austin Little', '2': 'Natasha Howard', '3': 'Alfred Mullins', '4': 'Jamie Rowe'}))

#chatGPT told me cProfile can evaluate the performance of Func

cProfile.run('lodv')

#Textbook

def test(dictt):

result = {}

for val in dictt.values():

result[val] = len(val)

return result

'''

54. Write a Python program to get the depth of a dictionary.

Expected Output:

4

'''

#warmup

dict1={'a': 1, 'b': {'c': 2, 'd': {'e': 3}}} #depth=3

print([i for i in dict1.values()])

print(isinstance(dict1,dict)) #True

isdict\_cnt=0

for v in dict1.values():

if isinstance(v,dict): isdict\_cnt+=1

print(isdict\_cnt) #1

print()

# mycode:

def isdict(val):

return isinstance(val,dict)

def depth(dicts):

dpth=0

chk=dicts

if isdict(chk): dpth+=1

chk=list(chk.values())

# print(chk,len(chk))

while len(chk)>0:

chk=list(j.values() for j in chk if isinstance(j,dict))

# print(chk)

dpth+=1

# print(dpth)

return dpth

print(depth({'a': 1, 'b': {'c': 2, 'd': {'e': 3}}}))

print(depth({1:1,2:{21:211,22:{221:2211,222:2221,223:{2231:22312,2232:{'a':'a1','b':{'fin1','fin2'}}}}}}))

print(depth({'a': 1, 'b': {'c': 2, 'd': {'e': 3,'f':{'g':4}}}}))

print(depth({1:1,2:{21:1}}))

#there is some bugs, got the same answer '3'

#Textbook

print()

def dict\_depth(d):

if isinstance(d, dict):

return 1 + (max(map(dict\_depth, d.values())) if d else 0)

return 0

dic = {'a':1, 'b': {'c': {'d': {‘e’:5}}}}

print(dict\_depth(dic))

dict2={1:1,2:{21:211,22:{221:2211,222:2221,223:{2231:22312,2232:{'a':'a1','b':{'fin1','fin2'}}}}}}

print(dict\_depth(dict2))

#chatGPT

def dict\_depth(d, level=1):

"""

Recursively get the depth of a dictionary.

Args:

d (dict): The dictionary to get the depth of.

level (int, optional): The current level of the dictionary. Defaults to 1.

Returns:

int: The depth of the dictionary.

"""

if not isinstance(d, dict) or not d:

return level

return max(dict\_depth(d[k], level + 1) for k in d)

# Example usage

my\_dict = {

'a': 1,

'b': {

'c': {

'd': {

'e': 2

}

}

}

}

print(dict\_depth(my\_dict)) # Output: 5

'''

55. Write a Python program to access dictionary key's element by index.

Expected Output:

physics

math

chemistry

'''

dict1={'physics':76,'math':85,'chemistry':91}

for i in dict1.keys():

print(i)

#Textbook

num = {'physics': 80, 'math': 90, 'chemistry': 86}

print(list(num)[0])

print(list(num)[1])

print(list(num)[2])

'''

56. Write a Python program to convert a dictionary into a list of lists

Original Dictionary:

{1: 'red', 2: 'green', 3: 'black', 4: 'white', 5: 'black'}

Convert the said dictionary into a list of lists:

[[1, 'red'], [2, 'green'], [3, 'black'], [4, 'white'], [5, 'black']]

Original Dictionary:

{'1': 'Austin Little', '2': 'Natasha Howard', '3': 'Alfred Mullins', '4': 'Jamie Rowe'}

Convert the said dictionary into a list of lists:

[['1', 'Austin Little'], ['2', 'Natasha Howard'], ['3', 'Alfred Mullins'], ['4', 'Jamie Rowe']]

'''

def dtlol(dicts):

return list([k,v] for k,v in dicts.items())

print(dtlol({1: 'red', 2: 'green', 3: 'black', 4: 'white', 5: 'black'}))

print(dtlol({'1': 'Austin Little', '2': 'Natasha Howard', '3': 'Alfred Mullins', '4': 'Jamie Rowe'}))

#Textbook

def test(dictt):

result = list(map(list, dictt.items()))

return result

'''

57. Write a Python program to filter even numbers from a dictionary

of values.

Original Dictionary:

{'V': [1, 4, 6, 10], 'VI': [1, 4, 12], 'VII': [1, 3, 8]}

Filter even numbers from said dictionary values:

{'V': [4, 6, 10], 'VI': [4, 12], 'VII': [8]}

Original Dictionary:

{'V': [1, 3, 5], 'VI': [1, 5], 'VII': [2, 7, 9]}

Filter even numbers from said dictionary values:

{'V': [], 'VI': [], 'VII': [2]}

'''

#warmup

lst1=[1,3,4,6,10]

print([i for i in lst1 if not i%2])

def fen(dicts):

return {k:[i for i in v if not i%2] for k,v in dicts.items()}

print(fen({'V': [1, 4, 6, 10], 'VI': [1, 4, 12], 'VII': [1, 3, 8]}))

print(fen({'V': [1, 3, 5], 'VI': [1, 5], 'VII': [2, 7, 9]}))

'''

58. Write a Python program to get all combinations of key-value pairs

in a given dictionary.

Original Dictionary:

{'V': [1, 4, 6, 10], 'VI': [1, 4, 12], 'VII': [1, 3, 8]}

Combinations of key-value pairs of the said dictionary:

[{'V': [1, 4, 6, 10], 'VI': [1, 4, 12]}, {'V': [1, 4, 6, 10], 'VII': [1, 3, 8]}, {'VI': [1, 4, 12], 'VII': [1, 3, 8]}]

Original Dictionary:

{'V': [1, 3, 5], 'VI': [1, 5]}

Combinations of key-value pairs of the said dictionary:

[{'V': [1, 3, 5], 'VI': [1, 5]}]

'''

from itertools import combinations

def ac(dicts):

# return list({k:v} for (k,v) in combinations(dicts.items(),2))

return [dict(pair) for pair in combinations(dicts.items(), 2)]

#chatGPT provide, probably I didn't notice the combination取2. 已經是1 element

print(ac({'V': [1, 4, 6, 10], 'VI': [1, 4, 12], 'VII': [1, 3, 8]}))

#TypeError: unhashable type:'list'

#Textbook

import itertools

def test(dictt):

result = list(map(dict, itertools.combinations(dictt.items(), 2)))

return result

students = {'V' : [1, 4, 6, 10], 'VI' : [1, 4, 12], 'VII' : [1, 3, 8]}

print("\nOriginal Dictionary:")

print(students)

print("\nCombinations of key-value pairs of the said dictionary:")

print(test(students))

'''

59. Write a Python program to find the specified number

of maximum values in a given dictionary.

Original Dictionary:

{'a': 5, 'b': 14, 'c': 32, 'd': 35, 'e': 24, 'f': 100, 'g': 57, 'h': 8, 'i': 100}

1 maximum value(s) in the said dictionary:

['f']

2 maximum value(s) in the said dictionary:

['f', 'i']

5 maximum value(s) in the said dictionary:

['f', 'i', 'g', 'd', 'c']

'''

from operator import itemgetter

num=itemgetter(1)

def snomv(dicts,maxv):

# return sorted(dicts.items(),key=num, reverse=True)[:maxv]

return [k for k,v in sorted(dicts.items(),key=num, reverse=True)[:maxv]]

print(snomv({'a': 5, 'b': 14, 'c': 32, 'd': 35, 'e': 24, 'f': 100, 'g': 57, 'h': 8, 'i': 100},2))

print(snomv({'a': 5, 'b': 14, 'c': 32, 'd': 35, 'e': 24, 'f': 100, 'g': 57, 'h': 8, 'i': 100},5))

#Textbook

def test(dictt, N):

result = sorted(dictt, key=dictt.get, reverse=True)[:N]

return result

'''

60. Write a Python program to find the shortest list of values

for the keys in a given dictionary.

Original Dictionary: {'V': [10, 12], 'VI': [10], 'VII': [10, 20, 30, 40], 'VIII': [20], 'IX': [10, 30, 50, 70], 'X': [80]}

Shortest list of values with the keys of the said dictionary:

['VI', 'VIII', 'X']

'''

dict1={'V': [10, 12], 'VI': [10], 'VII': [10, 20, 30, 40], 'VIII': [20], 'IX': [10, 30, 50, 70], 'X': [80]}

shortest=min(list(map(len,dict1.values())))

print([i[0] for i in dict1.items() if len(i[1])==shortest])

#Textbook

def test(dictt):

min\_value=1

result = [k for k, v in dictt.items() if len(v) == (min\_value)]

return result

'''

61. Write a Python program to count the frequency of a dictionary.

Original Dictionary:

{'V': 10, 'VI': 10, 'VII': 40, 'VIII': 20, 'IX': 70, 'X': 80, 'XI': 40, 'XII': 20}

Count the frequency of the said dictionary:

Counter({10: 2, 40: 2, 20: 2, 70: 1, 80: 1})

'''

from collections import Counter

from collections import defaultdict

#warmup

dict1={'V': 10, 'VI': 10, 'VII': 40, 'VIII': 20, 'IX': 70, 'X': 80, 'XI': 40, 'XII': 20}

print(Counter(dict1)) #just sort by value

swapped\_dict={v:k for k,v in dict1.items()}

print(swapped\_dict) #the duplicate values not included!

swapped\_dict1=defaultdict(list)

for k,v in dict1.items():

swapped\_dict1[v].append(k)

dict2=dict(swapped\_dict1)

print(dict2)

# {10: ['V', 'VI'], 40: ['VII', 'XI'], 20: ['VIII', 'XII'], 70: ['IX'], 80: ['X']}

#Textbook -more simple, just use dict.values()!

from collections import Counter

def test(dictt):

result = Counter(dictt.values())

return result

'''

62. Write a Python program to extract values from a given dictionary

and create a list of lists from those values.

Original Dictionary:

[{'student\_id': 1, 'name': 'Jean Castro', 'class': 'V'}, {'student\_id': 2, 'name': 'Lula Powell', 'class': 'V'}, {'student\_id': 3, 'name': 'Brian Howell', 'class': 'VI'}, {'student\_id': 4, 'name': 'Lynne Foster', 'class': 'VI'}, {'student\_id': 5, 'name': 'Zachary Simon', 'class': 'VII'}]

Extract values from the said dictionarie and create a list of lists using those values:

[[1, 'Jean Castro', 'V'], [2, 'Lula Powell', 'V'], [3, 'Brian Howell', 'VI'], [4, 'Lynne Foster', 'VI'], [5, 'Zachary Simon', 'VII']]

[[1, 'Jean Castro'], [2, 'Lula Powell'], [3, 'Brian Howell'], [4, 'Lynne Foster'], [5, 'Zachary Simon']]

[['Jean Castro', 'V'], ['Lula Powell', 'V'], ['Brian Howell', 'VI'], ['Lynne Foster', 'VI'], ['Zachary Simon', 'VII']]

'''

def clol(lsts):

return [v for d in lsts for v in d.values()]

print(clol([{'student\_id': 1, 'name': 'Jean Castro', 'class': 'V'}, {'student\_id': 2, 'name': 'Lula Powell', 'class': 'V'}, {'student\_id': 3, 'name': 'Brian Howell', 'class': 'VI'}, {'student\_id': 4, 'name': 'Lynne Foster', 'class': 'VI'}, {'student\_id': 5, 'name': 'Zachary Simon', 'class': 'VII'}]))

#cannot make as list of list

#Textbook

def test(dictt,keys):

return [list(d[k] for k in keys) for d in dictt]

students = [

{'student\_id': 1, 'name': 'Jean Castro', 'class': 'V'},

{'student\_id': 2, 'name': 'Lula Powell', 'class': 'V'},

{'student\_id': 3, 'name': 'Brian Howell', 'class': 'VI'},

{'student\_id': 4, 'name': 'Lynne Foster', 'class': 'VI'},

{'student\_id': 5, 'name': 'Zachary Simon', 'class': 'VII'}

]

print("\nOriginal Dictionary:")

print(students)

print("\nExtract values from the said dictionarie and create a list of lists using those values:")

print("\n",test(students,('student\_id', 'name', 'class')))

print("\n",test(students,('student\_id', 'name')))

print("\n",test(students,('name', 'class')))

'''

63. Write a Python program to convert a given list of lists to a dictionary.

Original list of lists:

[[1, 'Jean Castro', 'V'], [2, 'Lula Powell', 'V'], [3, 'Brian Howell', 'VI'], [4, 'Lynne Foster', 'VI'], [5, 'Zachary Simon', 'VII']]

Convert the said list of lists to a dictionary:

{1: ['Jean Castro', 'V'], 2: ['Lula Powell', 'V'], 3: ['Brian Howell', 'VI'], 4: ['Lynne Foster', 'VI'], 5: ['Zachary Simon', 'VII']}

'''

#desired structure: i[0]:[i[1],i[2]] for i in lsts

lst1=[[1, 'Jean Castro', 'V'], [2, 'Lula Powell', 'V'], [3, 'Brian Howell', 'VI'], [4, 'Lynne Foster', 'VI'], [5, 'Zachary Simon', 'VII']]

print({i[0]:[i[1],i[2]] for i in lst1})

#Textbook

def test(lst):

result = {item[0]: item[1:] for item in lst}

return result

'''

64. Write a Python program that creates key-value list pairings within

a dictionary.

Original dictionary:

{1: ['Jean Castro'], 2: ['Lula Powell'], 3: ['Brian Howell'], 4: ['Lynne Foster'], 5: ['Zachary Simon']}

A key-value list pairings of the said dictionary:

[{1: 'Jean Castro', 2: 'Lula Powell', 3: 'Brian Howell', 4: 'Lynne Foster', 5: 'Zachary Simon'}]

'''

#desired structure:list({k:v[0]})

dict1={1: ['Jean Castro'], 2: ['Lula Powell'], 3: ['Brian Howell'], 4: ['Lynne Foster'], 5: ['Zachary Simon']}

print(list({k:v[0]} for k,v in dict1.items()))

#Textbook

from itertools import product

def test(dictt):

result = [dict(zip(dictt, sub)) for sub in product(\*dictt.values())]

return result

'''

65. Write a Python program to get the total length of all values

in a given dictionary with string values.

Original dictionary:

{'#FF0000': 'Red', '#800000': 'Maroon', '#FFFF00': 'Yellow', '#808000': 'Olive'}

Total length of all values of the said dictionary with string values:

20

'''

def tloav(dicts):

return sum([len(v) for v in dicts.values()])

print(tloav({'#FF0000': 'Red', '#800000': 'Maroon', '#FFFF00': 'Yellow', '#808000': 'Olive'}))

'''

65. Write a Python program to get the total length of all values

in a given dictionary with string values.

Original dictionary:

{'#FF0000': 'Red', '#800000': 'Maroon', '#FFFF00': 'Yellow', '#808000': 'Olive'}

Total length of all values of the said dictionary with string values:

20

'''

def tloav(dicts):

return sum([len(v) for v in dicts.values()])

print(tloav({'#FF0000': 'Red', '#800000': 'Maroon', '#FFFF00': 'Yellow', '#808000': 'Olive'}))

'''

66. Write a Python program to check if a specific key and a value

exist in a dictionary.

Original dictionary:

[{'student\_id': 1, 'name': 'Jean Castro', 'class': 'V'}, {'student\_id': 2, 'name': 'Lula Powell', 'class': 'V'}, {'student\_id': 3, 'name': 'Brian Howell', 'class': 'VI'}, {'student\_id': 4, 'name': 'Lynne Foster', 'class': 'VI'}, {'student\_id': 5, 'name': 'Zachary Simon', 'class': 'VII'}]

Check if a specific Key and a value exist in the said dictionary:

True

True

True

False

False

False

'''

def skave(lsts,attr,val):

for d in lsts:

if d[attr]==val:

return True

return False

print(skave([{'student\_id': 1, 'name': 'Jean Castro', 'class': 'V'}, {'student\_id': 2, 'name': 'Lula Powell', 'class': 'V'}, {'student\_id': 3, 'name': 'Brian Howell', 'class': 'VI'}, {'student\_id': 4, 'name': 'Lynne Foster', 'class': 'VI'}, {'student\_id': 5, 'name': 'Zachary Simon', 'class': 'VII'}], 'name','Zachary Simon'))

print(skave([{'student\_id': 1, 'name': 'Jean Castro', 'class': 'V'}, {'student\_id': 2, 'name': 'Lula Powell', 'class': 'V'}, {'student\_id': 3, 'name': 'Brian Howell', 'class': 'VI'}, {'student\_id': 4, 'name': 'Lynne Foster', 'class': 'VI'}, {'student\_id': 5, 'name': 'Zachary Simon', 'class': 'VII'}], 'class','XI'))

print(skave([{'student\_id': 1, 'name': 'Jean Castro', 'class': 'V'}, {'student\_id': 2, 'name': 'Lula Powell', 'class': 'V'}, {'student\_id': 3, 'name': 'Brian Howell', 'class': 'VI'}, {'student\_id': 4, 'name': 'Lynne Foster', 'class': 'VI'}, {'student\_id': 5, 'name': 'Zachary Simon', 'class': 'VII'}], 'name','Amy Grant'))

#Textbook

def test(dictt, key, value):

if any(sub[key] == value for sub in dictt):

return True

return False

'''

67. Write a Python program to invert a given dictionary with

non-unique hashable values.

Sample Output:

{8: ['Ora Mckinney', 'Mathew Gilbert'], 7: ['Theodore Hollandl', 'Mae Fleming', 'Ivan Little']}

'''

# from collections import defaultdict

#warmup

d1={8: ['Ora Mckinney', 'Mathew Gilbert']}

# d=defaultdict(list) #當值不存在時 以defaultdict生成一個list放預設值([])

# print(d['Ora McKinney']) # []

lst1=[v[i] for k,v in d1.items() for i in range(len(v))]

lst2=[k for k,v in d1.items() for i in range(len(v))]

print(lst1, lst2)

print(dict(zip(lst1,lst2)))

print()

def invert(dicts):

resd={}

for dictk,dictv in dicts.items():

lst1=[v for v in dictv]

lst2=[dictk] \* len(dictv)

print(lst1,lst2)

resd.update(dict(zip(lst1,lst2)))

return resd

print(invert({8: ['Ora Mckinney', 'Mathew Gilbert'], 7: ['Theodore Hollandl', 'Mae Fleming', 'Ivan Little']}))

# {'Ora Mckinney': 8, 'Mathew Gilbert': 8, 'Theodore Hollandl': 7, 'Mae Fleming': 7, 'Ivan Little': 7}

print()

#Textbook- different meaning

from collections import defaultdict

def test(students):

obj = defaultdict(list)

for key, value in students.items():

obj[value].append(key)

# 此defaultdict的value已經設為list,故可以append

#print(obj)

return dict(obj)

students = {

'Ora Mckinney': 8,

'Theodore Hollandl': 7,

'Mae Fleming': 7,

'Mathew Gilbert': 8,

'Ivan Little': 7,

}

print('Textbook:')

print(test(students))

'''

68. Write a Python program to combine two or more dictionaries,

creating a list of values for each key.

Sample Output:

Original dictionaries:

{'w': 50, 'x': 100, 'y': 'Green', 'z': 400}

{'x': 300, 'y': 'Red', 'z': 600}

Combined dictionaries, creating a list of values for each key:

{'w': [50], 'x': [100, 300], 'y': ['Green', 'Red'], 'z': [400, 600]}

'''

from collections import defaultdict

combd=defaultdict(list)

dict1={'w': 50, 'x': 100, 'y': 'Green', 'z': 400}

dict2={'x': 300, 'y': 'Red', 'z': 600}

for k,v in dict1.items():

combd[k].append(v)

print(dict(combd))

for k,v in dict2.items():

combd[k].append(v)

print(dict(combd))

#Textbook

from collections import defaultdict

def test(\*dicts):

result = defaultdict(list)

for el in dicts:

for key in el:

result[key].append(el[key])

return dict(result)

d1 = {'w': 50, 'x': 100, 'y': 'Green', 'z': 400}

d2 = {'x': 300, 'y': 'Red', 'z': 600}

print("Original dictionaries:")

print(d1)

print(d2)

print("\nCombined dictionaries, creating a list of values for each key:")

print(test(d1, d2))

#warmup- \*: take arbitrary number of arguments

def test1(\*num):

return sum(num)

print(test1(1,2,3,4)) #10

print(test1(67,34.71,-22,980,52.3,-113,20)) #1019.01

'''

69. Write a Python program to group the elements of a given list

based on the given function.

Sample Output:

Original list & function:

[7, 23, 3.2, 3.3, 8.4] Function name: floor:

Group the elements of the said list based on the given function:

{7: [7], 23: [23], 3: [3.2, 3.3], 8: [8.4]}

Original list & function:

['Red', 'Green', 'Black', 'White', 'Pink'] Function name: len:

Group the elements of the said list based on the given function:

{3: ['Red'], 5: ['Green', 'Black', 'White'], 4: ['Pink']}

'''

from collections import defaultdict

from math import floor

#warmup

lst1=[7, 23, 3.2, 3.3, 8.4]

dd=defaultdict(list)

for i in lst1:

dd[floor(i)].append(i)

print(dict(dd))

def gebf(lsts,func):

dd=defaultdict(list)

for i in lsts:

dd[func(i)].append(i)

return dict(dd)

print(gebf([7, 23, 3.2, 3.3, 8.4],floor))

print(gebf(['Red', 'Green', 'Black', 'White', 'Pink'],len))

print(gebf([43,72,66,14,97,132,11,26],lambda x: x%7))

'''

70. Write a Python program to map the values of a given list to

a dictionary using a function, where the key-value pairs consist of

the original value as the key and the result of the function

as the value.

Sample Output:

{1: 1, 2: 4, 3: 9, 4: 16}

'''

import math

#warmup

lst1=[(i,i\*\*2) for i in range(1,5)]

print(dict(lst1))

def mvuf(lsts,func):

lsts1=[(i,func(i)) for i in lsts]

return (dict(lsts1))

print(mvuf([1,2,3,4],math.sqrt))

print(mvuf([30,60,90,180],math.radians))

print(mvuf(['nothing','but','a','G','thang'],str.capitalize))

#Textbook

def test(itr, fn):

return dict(zip(itr, map(fn, itr)))

print(test([1, 2, 3, 4], lambda x: x \* x))

'''

71. Write a Python program to retrieve the value of the nested key

indicated by the given selector list from a dictionary or list.

Sample Output:

Russell

2

'''

#Textbook

from functools import reduce

from operator import getitem

def test(d, selectors):

return reduce(getitem, selectors, d)

users = {

'Carla ': {

'name': {

'first': 'Carla ',

'last': 'Russell'

},

'postIds': [1, 2, 3, 4, 5]

}

}

print(test(users, ['Carla ', 'name', 'last']))

print(test(users, ['Carla ', 'postIds', 1]))

#warmup- functools.reduce

# apply the lambda(or similar) function to the list in a cumulative way

#syntax: reduce(function, sequence, initial)

print(reduce(lambda x,y:x+y, [1,2,3,4,5])) #15

print(reduce(lambda x,y:x+y, [1,2,3,4,5], 100)) #115(起始值設為100)

#有點 利用函數 將一個序列收合成一個值 的味道

#warmup- operator.getitem

'''operator.getitem returns a function that can be used to retrieve a single item from a sequence or mapping object, given a key or index. The key or index is passed as an argument when the function is called. operator.getitem can be used with any object that supports indexing or key lookup, including sequences, lists, tuples, and mappings.

operator.itemgetter returns a function that can be used to retrieve multiple items from a sequence or mapping object, given a list of keys or indices. The keys or indices are passed as arguments when the function is called. operator.itemgetter can be used with any object that supports indexing or key lookup, including sequences, lists, tuples, and mappings.

'''

lst1=[0,1,2,3,4]

print(getitem(lst1,2)) #2

# 相當於 print(lst1[2])

'''

72. Write a Python program to invert a dictionary with unique hashable

values.

Sample Output:

{10: 'Theodore', 11: 'Mathew', 9: 'Roxanne'}

'''

lst1=['Theodore', 'Mathew', 'Roxanne']

idx=(10,11,9)

d1=dict(zip(idx,lst1))

print(d1)

#Textbook

def test(students):

return { value: key for key, value in students.items() }

students = {

'Theodore': 10,

'Mathew': 11,

'Roxanne': 9,

}

print(test(students))

'''

73. Write a Python program to convert a list of dictionaries into

a list of values corresponding to the specified key.

Sample Output:

Original list of dictionaries:

[{'name': 'Theodore', 'age': 18}, {'name': 'Mathew', 'age': 22}, {'name': 'Roxanne', 'age': 20}, {'name': 'David', 'age': 18}]

Convert a list of dictionaries into a list of values corresponding to the specified key:

[18, 22, 20, 18]

'''

#warmup

lst1=[{'name': 'Theodore', 'age': 18}, {'name': 'Mathew', 'age': 22}, {'name': 'Roxanne', 'age': 20}, {'name': 'David', 'age': 18}]

print([d['age'] for d in lst1])

def cilov(lsts,skey):

return [d[skey] for d in lsts]

print(cilov([{'name': 'Theodore', 'age': 18}, {'name': 'Mathew', 'age': 22}, {'name': 'Roxanne', 'age': 20}, {'name': 'David', 'age': 18}],'age'))

#Textbook

def test(lsts, key):

return [x.get(key) for x in lsts]

'''

74. Write a Python program to create a dictionary with the same keys

as the given dictionary and values generated by running the

given function for each value.

Sample Output:

Original dictionary elements:

{'Theodore': {'user': 'Theodore', 'age': 45}, 'Roxanne': {'user': 'Roxanne', 'age': 15}, 'Mathew': {'user': 'Mathew', 'age': 21}}

Dictionary with the same keys:

{'Theodore': 45, 'Roxanne': 15, 'Mathew': 21}

'''

d1={'Theodore': {'user': 'Theodore', 'age': 45}, 'Roxanne': {'user': 'Roxanne', 'age': 15}, 'Mathew': {'user': 'Mathew', 'age': 21}}

print({d:v['age'] for d,v in d1.items()})

#Textbook

def test(obj, fn):

return dict((k, fn(v)) for k, v in obj.items())

users = {

'Theodore': { 'user': 'Theodore', 'age': 45 },

'Roxanne': { 'user': 'Roxanne', 'age': 15 },

'Mathew': { 'user': 'Mathew', 'age': 21 },

}

print("\nOriginal dictionary elements:")

print(users)

print("\nDictionary with the same keys:")

print(test(users, lambda u : u['age']))

'''

75. Write a Python program to find all keys in a dictionary that have the given value. Go to the editor

Sample Output:

Original dictionary elements:

{'Theodore': 19, 'Roxanne': 20, 'Mathew': 21, 'Betty': 20}

Find all keys in the said dictionary that have the specified value:

['Roxanne', 'Betty']

'''

#Textbook

def test(dict, val):

return list(key for key, value in dict.items() if value == val)

students = {

'Theodore': 19,

'Roxanne': 20,

'Mathew': 21,

'Betty': 20

}

print("\nOriginal dictionary elements:")

print(students)

print("\nFind all keys in the said dictionary that have the specified value:")

print(test(students, 20))

'''

76. Write a Python program to combine two lists into a dictionary.

The elements of the first one serve as keys and the elements of the

second one serve as values. Each item in the first list must be

unique and hashable.

Sample Output:

Original lists:

['a', 'b', 'c', 'd', 'e', 'f']

[1, 2, 3, 4, 5]

Combine the values of the said two lists into a dictionary:

{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}

'''

def c2lid(lst1,lst2):

tup1=zip(lst1,lst2)

return dict(list(tup1))

print(c2lid(['a', 'b', 'c', 'd', 'e', 'f'],[1, 2, 3, 4, 5]))

#Textbook

def test(keys, values):

return dict(zip(keys, values))

'''

77. Write a Python program to transform a dictionary into a list of tuples.

Sample Output:

Original Dictionary:

{'Red': 1, 'Green': 3, 'White': 5, 'Black': 2, 'Pink': 4}

Convert the said dictionary to a list of tuples:

[('Red', 1), ('Green', 3), ('White', 5), ('Black', 2), ('Pink', 4)]

'''

dict1={'Red': 1, 'Green': 3, 'White': 5, 'Black': 2, 'Pink': 4}

print(list(tuple(dict1.items())))

#Textbook

def test(d):

return list(d.items())

d = {'Red': 1, 'Green': 3, 'White': 5, 'Black': 2, 'Pink': 4}

print("Original Dictionary:")

print(d)

print("\nConvert the said dictionary to a list of tuples:")

print(test(d))

'''

78. Write a Python program to create a flat list of all the keys in

a flat dictionary.

Sample Output:

Original dictionary elements:

{'Theodore': 19, 'Roxanne': 20, 'Mathew': 21, 'Betty': 20}

Create a flat list of all the keys of the said flat dictionary:

['Theodore', 'Roxanne', 'Mathew', 'Betty']

'''

dict1={'Theodore': 19, 'Roxanne': 20, 'Mathew': 21, 'Betty': 20}

print(list(dict1))

#Textbook

def test(flat\_dict):

return list(flat\_dict.keys())

'''

79. Write a Python program to create a flat list of all the values

in a flat dictionary.

Sample Output:

Original dictionary elements:

{'Theodore': 19, 'Roxanne': 20, 'Mathew': 21, 'Betty': 20}

Create a flat list of all the values of the said flat dictionary:

[19, 20, 21, 20]

'''

dict1={'Theodore': 19, 'Roxanne': 20, 'Mathew': 21, 'Betty': 20}

print([i for i in dict1.values()])

#Textbook

def test(flat\_dict):

return list(flat\_dict.values())

'''

80. Write a Python program to find the key of the maximum and minimum value

in a dictionary.

Sample Output:

Original dictionary elements:

{'Theodore': 19, 'Roxanne': 22, 'Mathew': 21, 'Betty': 20}

Finds the key of the maximum and minimum value of the said dictionary:

('Roxanne', 'Theodore')

'''

def komam(dicts):

lst1=list(dicts.items())

age\_seq=sorted([t[1] for t in lst1])

return [k for k,v in dicts.items() if v==age\_seq[0] or v==age\_seq[-1]]

print(komam({'Theodore': 19, 'Roxanne': 22, 'Mathew': 21, 'Betty': 20}))

#Textbook

def test(d):

return max(d, key = d.get), min(d, key = d.get)

students = {

'Theodore': 19,

'Roxanne': 22,

'Mathew': 21,

'Betty': 20

}

print("\nOriginal dictionary elements:")

print(students)

print("\nFinds the key of the maximum and minimum value of the said dictionary:")

print(test(students))