
Dict

Methods:

dict.keys -> return list of keys

dict.items -> return list of tuple of key and value

dict.values -> return list of values

```
import collections
dic = collections.defaultdict(lambda: "key not found")
```

-> This created a dict in which if new key is not key value then it appeared "key not found".

sys.getsizeof(dict) = Size of dict

sorted(dict) — sorted(dict, key=lambda)
sorted(dict, key=itemgetter

lambda kv = kv[1] [sort by value]

itemgetter [give key ace to which sort]

sorted(dict, key=itemgetter('age')

Use when { name="", age="" , name="", age="" }

Merge

→ dict1.update(dict2) → return none because, dict1 is update with dict2.

→ dict3 = dict1 | dict2 → return new merged dict

→ for loop and key → for i in dict2.keys():
dict1[i] = dict2[i]
return dict1

insert at beggining :-

from collections import OrderedDict

dict2 = OrderedDict(Dict1)
dict2.update({'manjot': '3'})
→ appended in the last
dict2.move_to_end('manjot', last=False)

Important)

-> from collections import OrderedDict

dict = OrderedDict.fromkeys("gorav")

-> OrderedDict([('g', None), ('a', None)...])

-> from collections import Counter

dict = Counter(list)

-> return a dict having key and value as its frequency.

->

-> from collection import defaultdict

Create a new dict

new_dict = defaultdict(list)

give list properties to value.

-> move_to_end()

-> move the key to the end

-> popitem()

dict = {"A": 2, 'B': 3}

1) Sort Python Dict by key and value

→ sorted(dict) → default is sorted acc. to keys.

→ sorted(dict.items() key=lambda kv: kv[1])

dict.items() → give both keys and value

kv[1] → give value instead of keys.

2) Missing a key in dict

→ dict.get('C', 'Not found'))

Return value of C else return not found

→ dict.setdefault('C', 'Not found')

→ import collections

dict = collections.defaultdict(lambda: 'key not found')

defaultdict is created

→ dict["Gaurav"] = 1

→ dict["gaurav"] → key not found