Day17_List and dict methods

10. reverse(self):Reverse *IN PLACE*: This method is used for reversing the list and assign the reversed elements to the list

```
print(lst)
#[56, 98.36, None, (6-9j), True, 56, 'Hello']

lst.reverse()
print(lst)
#['Hello', 56, True, (6-9j), None, 98.36, 56]
```

11. sort(key=None, reverse=False):sorts the same type of elements *IN PLACE* and optionally can use the key .
By default sorting takes place in ascending order since reverse=False
If reverse = True descending order sorting takes place

```
lst1 = [56,12,96,-63,0]
lst1.sort()
print(lst1)
#[-63, 0, 12, 56, 96]

lst1 = [56,12,96,-63,0]
lst1.sort(reverse=True)
print(lst1)
#[96, 56, 12, 0, -63]

lst2 = ['Pfizer','Moderna','Johnson and Johnson','Covaxin','Covishield','Sputnik']
lst2.sort()
print(lst2)

def lensort(e):
    return len(e)

lst2 = ['Pfizer','Moderna','Johnson and Johnson','Covaxin','Covishield','Sputnik']
lst2.sort(key=lensort)
print(lst2)
#['Pfizer', 'Moderna', 'Covaxin', 'Sputnik', 'Covishield', 'Johnson and Johnson']
```

```
lst2 = ['Pfizer','Moderna','Johnson and Johnson','Covaxin','Covishield','Sputnik']
lst2.sort(key=lensort,reverse=True)
print(lst2)
#['Johnson and Johnson', 'Covishield', 'Moderna', 'Covaxin', 'Sputnik',
# 'Pfizer']
```

```
lst1 = ['Hello', 56, True, (6-9j), None, 98.36, 56]
lst1.sort()

#TypeError: '<' not supported between instances of 'int' and 'str'
```

```
lst1 = [56, True, -2,False,True]
lst1.sort()
print(lst1)
#[-2, False, True, True, 56]

False --> 0
True  --> 1
 list : [] --> mutable
tuple : () -->  immutable

tpl = ('Hello', 56, True, (6-9j), None, 98.36, 56)
dir(tpl)

 'count'
 'index'
help(tpl)

count(value)
    Return number of occurrences of value.

 index( value, start=0, stop=9223372036854775807)
    Return first index of value.
    Raises ValueError if the value is not present.
```

dict methods

```
dct = {
       1:"Pfizer",
       2:"moderna",
       3:"johnson and johnson"
      }
help(dct)
```

1. clear(): D.clear() -> None. Remove all items from D.

```
print(dct)
dct.clear()
print(dct)
```

```
{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson'}
{}
```

```
#deletes the dct structure
del dct
```

2. copy(...): D.copy() -> a shallow copy of D

```
dct = {
       1:"Pfizer",
       2:"moderna",
       3:"johnson and johnson"
      }

dct1 = dct.copy()
print(dct1)
#{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson'}

dct[4] = "Sputnik"

print(dct)
print(dct1)
```

```
{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson', 4: 'Sputnik'}
{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson'}
```

3. get(key, default=None)

Return the value for key if key is in the dictionary,
else default.

```
dct.get(1)
#'Pfizer'

dct.get(2,"Covaxin")
#'moderna'

dct.get(5,"Covaxin")
#'Covaxin'

print(dct.get(5))
#None
```

4. items(): D.items()->a set-like object providing a view on D's items

```
dct.items()
#dict_items([(1, 'Pfizer'), (2, 'moderna'), (3, 'johnson and johnson'),
#           (4, 'Sputnik')])

#looping
for k,v in dct.items():
   print("key=",k)
   print("value=",v)
```

```
key= 1
value= Pfizer
key= 2
value= moderna
key= 3
value= johnson and johnson
key= 4
value= Sputnik
```

5. keys(): D.keys() -> a set-like object providing a view on D's keys

```
dct.keys()
#dict_keys([1, 2, 3, 4])

for k in dct.keys():
   print(dct[k])
```

```
Pfizer
moderna
johnson and johnson
Sputnik
```

6. values(): D.values() -> an object providing a view on D's values

```
dct.values()
#dict_values(['Pfizer', 'moderna', 'johnson and johnson', 'Sputnik'])
for v in dct.values():
   print(v)
```

```
Pfizer
moderna
johnson and johnson
Sputnik
```

7. pop(): D.pop(k[,d]) -> v, remove specified key and return the corresponding value.
If key is not found, d is returned if given,
otherwise KeyError is raised

```
dct = {
       1:"Pfizer",
       2:"moderna",
       3:"johnson and johnson",
        4: 'Sputnik'
      }
print(dct)
```

```
#{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson', 4: 'Sputnik'}

print(dct.pop(3))
print(dct)
#johnson and johnson
#{1: 'Pfizer', 2: 'moderna', 4: 'Sputnik'}

print(dct.pop(2,"Vaccine"))
print(dct)
#moderna
#{1: 'Pfizer', 4: 'Sputnik'}


print(dct.pop(5,"Vaccine") )
print(dct)
#Vaccine
#{1: 'Pfizer', 4: 'Sputnik'}

print(dct.pop(6))
print(dct)
#KeyError: 6
```

8. popitem(): D.popitem() -> (k, v), remove and return last (key, value) pair as a 2-tuple; but raise KeyError if D is empty.

```
dct = {
       1:"Pfizer",
       2:"moderna",
       3:"johnson and johnson",
        4: 'Sputnik'
      }
print(dct)
#{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson', 4: 'Sputnik'}
print(dct.popitem())
#(4, 'Sputnik')
print(dct)
#{1: 'Pfizer', 2: 'moderna', 3: 'johnson and johnson'}
```

```
dct.clear()
print(dct.popitem())

#KeyError: 'popitem(): dictionary is empty'
```