DICT BASICS

from math import sqrt

#creation

d={'k1':1,'k2':2,'k3':3,'k4':4}

d1=dict([('k1',10),('k2',20),('k3',30)])

print(d)

print(d1)

#assigning

d1['k1']=101        #-----d1 get modified

print(d1)           #modified d1 is printed

#extracting

print(d1['k1'])

print(d1['k2'])

#delete

del d1['k2']

print(d1)

#display all key value pairs

for k,v in d1.items():

    print(k,":",v, d1[k])

#display all keys

for k in d1.keys():

    print(k)

#display all values

for v in d1.values():

    print("vlaues: ",v)

#if all keys are numeric sort all keys

d3={1:89, 5:90, 2:91, 30:85, 20:89, 11:95}

print(sorted(d3))

#sort the dictionary keys using lambda

print(f"sort keys ascending order: {dict(sorted(d3.items(), key = lambda x : x[0], reverse=False))}")

print(f"sort keys decending order: {dict(sorted(d3.items(), key = lambda x : x[0], reverse=True))}")

#sort the dictionary Values using lambda

print(f"sort values ascending order: {dict(sorted(d3.items(), key = lambda x : x[1], reverse=True))}")

print(f"sort values descending order: {dict(sorted(d3.items(), key = lambda x : x[1], reverse=False))}")

#To form a dictionary from two list

l1=['a','b',15]

l2=[10,20,'c']

print(dict(zip(l1,l2)))

print(dict(zip(l2,l1)))

#To print dictionary

f=dict({'one': 1, 'three': 3}, two=2)

print(f)

#To pop an key from dictionary

d4=dict([('k1',10),('k2',20),('k3',30)])

print("d4: ", d4)

v=d4.pop('k2')

print(d4)

print("v: ", v)

print("-----------------------------------------------------------------------------------")

#To pop a values from dictionary

d4=dict([('k1',10),('k2',20),('k3',30)])

d5=dict([('k1',10),('k2',20),('k3',30)])

for k,v in d4.items():

    v=d5.popitem()

    print("v: ", v)

print("-----------------------------------------------------------------------------------")

#Copy

d6=dict([('k1',100),('k2',20),('k3',30)])

d7=d6.copy()

print("d6: ", d6)

print("d7: ", d7)

print("id(d6): ", id(d6))

print("id(d7): ", id(d7))

print("-----------------------------------------------------------------------------------")

#get

d8=dict([('k1',100),('k2',20),('k3',30)])

print("d8.get('k2'): ", d8.get('k2'))

print("-----------------------------------------------------------------------------------")

#reversed

d8=dict([('k1',100),('k2',20),('k3',30)])

print(reversed(d8))         #----> object returned hoga

print(list(reversed(d8)))

print(list(reversed(d8.keys())))

print("-----------------------------------------------------------------------------------")

#assign

d9={'k21':1, 'k2':2}

d9['k21'] = 45

print(d9)

d9['this key k3 does not exist in d9, therefore a key is created'] = 56

print(d9)

#iter(d)

print("-----------------------------------------------------------------------------------")

print(list(iter(d9.keys())))

print(list(iter(d9.values())))

print(list(iter(d9.items())))

print("-----------------------------------------------------------------------------------")

#max, min

print(max(d9))

print(min(d9))

print(max(d9.keys()))

print(min(d9.keys()))

print(max(d9.values()))

print(min(d9.values()))

print(max(d9.items()))

print(min(d9.items()))

print("-----------------------------------------------------------------------------------")

#d.update() ---- update values of a key

d11={'k1': 10, 'k2': 20, 'k3': 45}

d11.update(k1=41, k2=89)

print(d11)

print("-----------------------------------------------------------------------------------")

#concatenate two or more dictionaries ----> pipe symbol

d12={'k1':73}

d13={'k2': 78, 'k3': 56}

d14={'k1': 9999, 'k7': 48}

print(d12|d13|d14)

print("-----------------------------------------------------------------------------------")

#dictionary comprehension

d15 = {x: x\*\*2 for x in range(1,10)}

print(d15)

d16 = {y: sqrt(y) for y in range(1,10)}

print(d16)

d\_odd = {k:v for k,v in d15.items() if v%2 != 0}

print(d\_odd)

d\_even = {k:v for k,v in d15.items() if v%2 == 0}

print(d\_even)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

#dictionary packing and unpacking + concatenation

d17={'one':1, 'two':2, 'three':3}

d18={'four':4, 'five':5, 'six':6}

combination={\*\*d17, \*\*d18}

print(combination)

print(d17|d18)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

#how to create a dictionary from two list

d23=dict()

\_keys=['k1','k2','k3']

\_values=[10,20,30]

for k in \_keys:

    for v in \_values:

        d23[k] = v

        \_values.remove(v)

        break

print("create a dictionary from two list: ",d23)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

#creation of a dictionary

d24=dict({'one': 1, 'three': 3}, two=2)

print(d24)

#map function

d25=dict({'k1':25, 'k2':36}, two = 49)

print(d25)

r=map(lambda x: x\*\*0.5, d25.values())

print(list(r))

**LIST BASICS**

l1=[1,2,13,-9,121,33,-25,5,8]

l2=[e for e in l1 if e > 3]

print(l2)

\_11 = [e for e in l1 if e % 11 == 0]

print(\_11)

l3 = [e if e > 3 else None for e in l1 ]

print(l3)

print("---------------------------insert---------------------------")

l2=[10,20,30,40,50]

print(l2)

l2.insert(3,-50)

print(l2)

print("---------------------------update---------------------------")

l2[3]=-60

print(l2)

print("---------------------------remove---------------------------")

l2.remove(-60)

print(l2)

print("---------------------------slicing---------------------------")

l4=[10,20,30,40,50,60]

print(f"l4[2:4]: {l4[2:4]}")

print(f"l4[-5:-1]: {l4[-5:-1]}")

print(f"l4[3:5:2]: {l4[3:5:2]}")

print(f"l4[0:5:2]: {l4[0:5:2]}")

print("---------------------------shallow copy---------------------------")

l5=[10,20,30,40]

print(l5)

l6=l5[::]   #shallow copy

print(l6)

l5.remove(30)

print(l5)

print(l6)

print("---------------------------shallow copy---------------------------")

l6=[10,[20,30],[40,50]]

l7=l6[::]   #shallow copy

print("l6: ", l6)

print("l7: ", l7)

print("after removing 10")

l6.remove(10)   #list l6 length get changed

print("l6: ", l6)

print("l7: ", l7)

#l6[2][0] = 400 ------- IndexError

l6[1][0] = 400      #modify list element

print("l6: ", l6)

print("l7: ", l7)

print("---------------------------copy by assignment---------------------------")

l8 = [1,2,3,4]

l9 = l8 #copy by assignment

print("l8: ", l8)

print("l9: ", l9)

l8.remove(2)

print("l8: ", l8)

print("l9: ", l9)

print("---------------------------search an element---------------------------")

l10=["a","b","c",1,2]

if "a" in l10:

    print("yes")

else:

    print("no")

print("---------------------------concat---------------------------")

l11=[10,20,-30]

l12=['a','b',30]

l13 = l11 + l12

print("l13: ", l13)

print("---------------------------multiply---------------------------")

l14=[1,2,'a']

l14 = l14 \* 2

print("l14: ", l14)

print("---------------------------access element---------------------------")

print(l14[2])

print(l14.index(2))

# print("---------------------------replace element through slice---------------------------")

# l15=['a',11,'b',10,20,30,40]

# l15[::2]=[500,500]

# print(l15)

print("---------------------------delete element through slice---------------------------")

l15=['a','b',10,[10,'a'],20,30,40]

del l15[2:3:1]

print(l15)

print("---------------------------delete every alternate element through slice---------------------------")

l15=['a','b',10,[10,'a'],20,30,40]

del l15[::2]

print(l15)

print("---------------------------pop---------------------------")

l16=['a','b',10,[10,'a'],20,30,40]

l16.pop()

l16.pop()

l16.pop()

l16.pop()

print(l16)

print("---------------------------reverse---------------------------")

l18=['a','b',10,[10,'a'],20,30,40]

l18.reverse()

print(l18)

print("---------------------------reverse through slice---------------------------")

l18=['a','b',10,[10,'a'],20,30,40]

print("l18[::-1]: ", l18[::-1])

print("l18: ", l18) #original list do not get reversed because shallow copy me reverse ho raha h

print("---------------------------> , <---------------------------")

l19=[10,20,30,40]

l20=[10,20,30,60]

print("l19>l20: ", l19>l20)

print("l19<l20: ", l19<l20)

l20.pop()

print(l20)

print("l19<l20: ", l19<l20)

print("---------------------------unpacking---------------------------")

l21=[10,20,40,-60]

a,\*b,c=l21

print(f"a: {a}, b: {b}, c: {c}")

print("---------------------------sort numbers---------------------------")

l22=[1,6,-7,0]

l22.sort(reverse=True)

print("l22.sort(): ", l22)

print("---------------------------sort words---------------------------")

l22=['rahul','mitu','paras']

l22.sort(reverse=True)

print("l22.sort(): ", l22)

print("---------------------------sort through lambda---------------------------")

l22=['rahul','mitu','paras']

print("Ascending order: ", sorted(l22, key = lambda x: x, reverse=False))

print("Descending order: ", sorted(l22, key = lambda x: x, reverse=True))

print("---------------------------------------------------------------------")

l51=[10,[1,2,[15,25], 78, 96], 99, [101,102]]

print("l51[1][2][1]: ", l51[1][2][0])

print("l51[3][0]: ", l51[3][0])

print("l51[1][4]: ", l51[1][4])

**TUPLE BASICS**

#create a tuple

t=tuple()

print(type(t))

print("-----------------------------------------------------------------------------")

#create a tuple

t1=(1,2,3)

print(t1)

print(type(t1))

print("-----------------------------------------------------------------------------")

#to create tuple with a single value

single\_value\_tuple=(100,)

print(single\_value\_tuple)

print("-------------------------------slicing----------------------------------------------")

#slicing

t2=(1,2,3,4,5,6,7,8,9)

print(t2[0:2:1])

print("odd: ", t2[0:len(t2):2])

print("even: ", t2[1:len(t2):2])

print(t2[-1:-5:-1])

print(t2[-1:-8:-2])

print(t2[::-1])

print("-------------------------------comparison of tuples----------------------------------------------")

#comparison of tuples

t3=(1,2,50,8)

t4=(6,8,3)

print(t3>t4)    #false---->6<1 then stop

print(t3<t4)   #true--->6>1 then stop

t3=(1,2,50,8,9)

t4=(1,2,50,8)

print(t3>t4)    #true---->since first 4 elements are same then look length of tuple len(t3)>len(t4)

print(t3<t4)   #false--->since first 4 elements are same then look length of tuple len(t3)>len(t4)

print("-------------------------------packing and unpacking tuples----------------------------------------------")

#packing and unpacking tuples

t3=(1,2,50,8,9)

a,\*b,c=t3

print("a: ", a, "\*b: ", \*b, "c: ", c)

print("-------------------------------x in tuple----------------------------------------------")

#x in t3

if 2 in t3:

 print("yes")

print("-------------------------------x not in tuple----------------------------------------------")

#x in t3

if 20 not in t3:

 print("yes")

print("-------------------------------concat tuple----------------------------------------------")

#concat

t6=(71,72,73,74)

t7=(71,78,79,75)

t8=t7+t6

print(t8)

print("-------------------------------i th element of tuple----------------------------------------------")

#concat

t6=(71,72,73,74)

print(t6[3])

print(t6[-1])

print("-------------------------------len max min tuple----------------------------------------------")

#len max min

t6=(-171,-72,-89,-96,-74)

print(len(t6))

print(max(t6))

print(min(t6))

print("-------------------------------to find index of element of a tuple------------------------------")

#to find index of element of a tuple

t6=(-171,-72,-89,-96,-74)

t7=(9,8,6,8,9,6)

print(t6.index(-171))           #-----> 0

print(t6.index(-89))            #-----> 2

print(t7.index(8))              #-----> 1 ----> first occurrence de ga

#print(t6.index(-1710))          #-----> since element is present in the tuple, therefore ValueError comes

print("-------------------------------count occurrence------------------------------")

#count occurrence

t6=(-171,-72,-89,-96,-74)

t7=(9,8,6,8,9,6,9,9,-9)

print(t7.count(9))          #-----> 4

print(t7.count(-9))         #-----> 1

print("-------------------------------all------------------------------")

#all --- if all values are non zero return true else false

t8=(9,8,6,8,'z',6,9,'a',-9)

t9=(9,8,'',8,'z',6,9,'a',-9)

print(t8)

print(all(t8))      # true

print(all(t9))      # false

print("-------------------------------any------------------------------")

#any --- if any values are non zero return true else false

t8=(9,8,6,8,'z',6,9,'a',-9)

t9=()

print(t8)

print(all(t8))      # true

print(all(t9))      # false

print("-------------------------------tuple comprehension------------------------------")

#tuple comprehension

t8=(9,8,6,8,'z',6,9,'a',-9)

r=tuple(t8[i] for i in range(0,len(t8),2))

print(r)

t9=(1,2,3,4,5,6,7,8,9)

r1=tuple(t9[i] for i in range(0,len(t9),1) if t9[i] % 2 == 0)

print("even: ", r1)

r2=tuple(t9[i] for i in range(0,len(t9),1) if t9[i] % 2 != 0)

print("odd: ", r2)

r3=tuple(t9[i] for i in range(0,len(t9),1) if t9[i] > 6)

print(r3)

print("-------------------------------add element to tuple------------------------------")

#add element to tuple

t9=(1,2,3,4,5,6,7,8,9)

t10=(10,)

print(t10)

t9 = t9 + t10 #tuple is immutable but here we are concatinating two tuples and storing the tuple in previous tuple

print(t9)

print("-------------------------------modifying mutable part of tuple------------------------------")

#modifying mutable part of tuple

t11=(1,2,3,[4,5,6,7],8,9)

# t11[0]=25       #error--- TypeError: 'tuple' object does not support item assignment

t11[3][0]=25    #we are here modifying mutable part of tuple

print(t11)  #(1, 2, 3, [25, 5, 6, 7], 8, 9)

print("-------------------------------zip and tuple------------------------------")

#zip

t12=(1,2,3)

t13=([4,5,6,7],8,9)

print(tuple(zip(t12,t13)))

print("-------------------------------iterate over tuple------------------------------")

#iterate over tuple

t12=(1,2,3,[4,5,6,7],8,9)

for e in t12:

 print(e)

print("-------------------------------sort + tuple------------------------------")

#iterate over tuple

t12=(10,2,30,8,9)

print(tuple(sorted(t12, key= lambda x : x, reverse=True)))

print(tuple(sorted(t12, key= lambda x : x, reverse=False)))

print("-------------------------------convert tuple to string------------------------------")

#convert tuple to string

t12=('p','a','r','a','s')

print(type(t12))                #<class 'tuple'>

print(str(t12), type(str(t12))) #('p', 'a', 'r', 'a', 's') <class 'str'>

print("-------------------------------join elements of tuple------------------------------")

#join elements of tuple

t12=('p','a','r','a','s')

result="".join(t12)

print(result)   #output -------> paras

print("-------------------------------4th elements from last of tuple------------------------------")

#4th elements from last of tuple

t12=('p','a','r','a','s',1,5,6,7)

print(t12[-4:-5:-1])

print("-------------------------------find repeated items in a tuple------------------------------")

#find repeated items in a tuple

list1=list()

t12 = ('p','a','r','a','s',1,5,7,6,7)

for e in t12:

    r=t12.count(e)

    if r >= 2:

        if e not in list1:

            list1.append(e)

        r=0

print("repeated items in a tuple: ",list1)

                                #by using index

list2=list()

for idx in range(0,len(t12),1):

    r1=t12.count(t12[idx])

    if r1 >= 2:

        if t12[idx] not in list2:

            list2.append(t12[idx])

print("repeated items using index in a tuple: ",list2)

print("-------------------------------to delete element in a tuple------------------------------")

#to delete element in a tuple

#We can not delete element in a tuple since tuple is immutable

#but i have a JUGAAD

#create empty tuple -> convert tuple(from which element to be deleted) into set -> iterate over set

#-> add all elememt from set into empty tuple except that which need to be deleted

list1=list()

t18 = ('p','a','r','a','s',1,5,7,6,7)

delete\_element=7

empty\_tuple=tuple()

for e in set(t18):

   if e != delete\_element:

      empty\_tuple = empty\_tuple + (e,)

print(empty\_tuple)

print("-------------------------------find index of element in tuple------------------------------")

#find index of element in tuple

t18 = ('p','a','r','a','s',1,5,7,6,7)

for idx in range(0,len(t18),1):

   if t18[idx] == 7:

      print("index of 7 :", idx)

print("-------------------------------reversed(list,tuple,dict,etc)------------------------------")

#reverse

t18 = ('p','a','r','a','s',1,5,7,6,7)

print(t18[::-1])

print(tuple(reversed(t18)))

#join

m = ('10', '20', '40', '5', '70')

r="".join(m)

print("r: " , r)     #output -------> 102040570

str1=str()

for e in m:

    str1=str1+e

print(str1)

**STRING BASICS**

# str1="Manthan is a good guy"

# print(" ".join(str1.split()[::-1]))         #guy good a is Manthan

# str2=input("Enter a string: ")

# first\_char=str2[0:1:1]

# rest\_char=str2[1:len(str2):1]

# #print(first\_char, rest\_char)

# for e in rest\_char:

#     if e == first\_char:

#         re=rest\_char.lower().replace(e,"$")

# # print("re: ", re)

# final\_string = first\_char + re

# print(final\_string)             #rohit $ohit $ohit

# #create a string

#s1="Yulia"

# s2=str("James")

# print(s1 ,":", type(s1))

# print(s2 ,":", type(s2))

# #Print every character of a string

# for char in s1:

#      print(char)

# print("\n")

# for idx in range(0,len(s2),1):

#     print(s2[idx])

# #convert string to a list of character

# s1="Yulia"

# print(list(s1))         #['Y', 'u', 'l', 'i', 'a']

#take 123456789 as input as string and seperate each digit

# s1=input("enter a number: ")

# print(type(s1))

# for char in s1:

#     print(char)

# #       OR method 2

# l=list(s1)

# print(l)

# "is" operator to check if both string are same object

# s1=s2="ram"

# s3="ram"

# if s1 is s2:

#     print("s1 is s2 reference to same object.")

# if s1 is s3:

#     print("s1 is s3 reference to different object.")

# if s2 is s3:

#     print("s2 is s3 reference to different object.")

#lexicographical ordering

# s1="radhey"

# s2="Radhey"

# if s1 == s2:

#     print("True")

# else:

#     print("radhey is not same as Radhey. As python is a case sensitive.") #this one is output

# s3="radhey"

# s4="radhey1"

# if s3 == s4:

#     print("True")

# else:

#     print("False")

# s5="radhey"

# s6="radhey1"

# if s5 > s6:

#     print("True")

# else:

#     print("False")      #False  first check the character if they are same then check the length

# s7="radhey"

# s8="radhey1"

# if s7 < s8:

#     print("True")       #True

# else:

#     print("False")

# s9="radhey"

# s10="rzadhey1"

# if s9 < s10:

#     print("True")       #True

# else:

#     print("False")

# s11="radhey"

# s12="Radhey"

# if s11 < s12:

#     print("radhey < than Radhey")

# else:

#     print("radhey > than Radhey")       #radhey > than Radhey small char > big char

#string index

# s1="Yulia"

# print(s1[0:len(s1):1])

# print(s1[-1:-(len(s1)+1):-1])

# #reverse the string character wise

# str1="Manthan is a good guy"

# print(str1[::-1])

# #reverse the string word wise

# str1="Manthan is a good guy"

# print(str1.split()[::-1])       #['guy', 'good', 'a', 'is', 'Manthan']

# print(" ".join(str1.split()[::-1])) #guy good a is Manthan

#When complete sting is selected using slicing then copt by assignment happens.

# s1="hi\_this\_is\_great!"

# print(s1[::])   #hi\_this\_is\_great!

# if s1 is s1[::]:

#     print("both s1 and s1[::] refer to same object.") #both s1 and s1[::] refer to same object.

# else:

#     print(" s1 and s1[::] refer to different object.")

#count(substr,start,end)

# s1="Paras is arara"

# substr="ara"

# print(s1.count(substr,0,len(s1))) #2 return non-overlapping occurrences of substring

#endswith(substr,start,end)

# s1="Paras is arara"

# substr="ara"

# print(s1.endswith(substr,0,len(s1)))            #True

# substr1="12"

# print(s1.endswith(substr1,0,len(s1)))           #False

#find(substr,start,end) # return -1 if substr is not found, if found return first occurrence (left most occurrence)

# s1="Paras is arara"

# substr="ara"

# print(s1.find(substr, 0, len(s1)))      #1  first occurrence (left most occurrence)

# substr1="z"

# print(s1.find(substr1,0,len(s1)))       # return -1 as z is not found in s1

#rfind(substr, start, end) # return -1 if substr is not found, if found return (right most occurrence)

# s1="Paras is arara"

# substr="ara"

# print(s1.rfind(substr, 0, len(s1))) #11

#index(substr, start, end)

# s1="Paras is arara"

# substr="ara"

# print(s1.index(substr,0,len(s1)))   #1 first occurrence leftmost occurrence

# substr1="z"

# print(s1.index(substr1,0,len(s1)))  #ValueError: substring not found

#rindex(substr, start, end)

# s1 = "Paras is arara hello"

# substr1 = "ara"

# print(s1.rindex(substr1,0,len(s1)))     #11 last occurrence rightmost occurrence

#s1.format\_map(d1--->dict hai)

# profession = { 'name':['Barry', 'Bruce'],

#                'profession':['Engineer', 'Doctor'],

#                'age':[30, 31] }

# name1 = "{name[0]}".format\_map(profession)

# print(name1)

# #print("{name[0]} {name[1]}".format\_map(profession))

# #print("{name} {profession} {age}".format\_map(profession))

#isalnum()

# s1="P@aras123"

# s2="123"

# s3="Paras"

# print(s1.isalnum())

# print(s2.isalnum())

# print(s3.isalnum())

#islower()

# s1="PRaja"

# print(s1.islower())

#isdigit() isnumeric()

# s1="123456"

# print(s1.isdigit())

# print(s1.isnumeric())

#lstrip(char)

# s1="    e   Pranjal"

# print(s1.lstrip(" "))

#partition(sep) rpartition(sep)

# s1="Pranjal \*\*\* Pratap Singh"

# print(s1.partition("\*\*"))       #('Pranjal ', '\*\*', '\* Pratap Singh')

# print(s1.partition("%"))        #when separator not found ('Pranjal \*\*\* Pratap Singh', '', '')

# print(s1.rpartition("a"))       #('Pranjal \*\*\* Prat', 'a', 'p Singh')

#replace(old, new, no. of times)    #do not modify original string, return a copy of string

# s1="This is a beautiful day"

# print(s1.replace("a", "^"))     #replace all the occurrence         This is ^ be^utiful d^y

# print(s1.replace("a","^",2))    #replace first 2 occurrence         This is ^ be^utiful day

# print(s1)                       #original string is not modified    This is a beautiful day

#to remove spaces from a string

# s1="Pr an ja l"

# s2=s1.replace(" ","")

# print(s2)                   #Pranjal    return a copty of a string

# print(s1)                   #Pr an ja l original string is not modified

# print(s1.strip())

#lstrip(char) , rstrip(char), strip(char)

# txt = "####banana...###"

# x = txt.lstrip("#")

# y=txt.rstrip("#")

# z=txt.strip("#")

# print("of all fruits", x, "is my favorite") #of all fruits banana...### is my favorite

# print("of all fruits", y, "is my favorite") #of all fruits ####banana... is my favorite

# print("of all fruits", z, "is my favorite") #of all fruits banana... is my favorite

#split(char,maxsplit\_number)

# s1="This is monday. This in weekend"

# print(s1.split(" "))        #['This', 'is', 'monday.', 'This', 'in', 'weekend']

# print(s1.split(" ", 2))     #['This', 'is', 'monday. This in weekend']

#print(s1.split("e"))         #['This is ', 'nd monday. This in w', '', 'k', 'nd']

#rsplit(char, maxsplit\_number)

# s1="This is end monday. This in weekend"

# print(s1.rsplit("e"))