Python coding questions for practice.

Lists

Strings

Tuple

Dictionary

sets

Map, filter, reduce and lambda functions

OOP

Exception Handling

LIST Questions

1.)

In [8]:

```
def find_sum(arr,i,j):
    total = 0
    for num in range(i,j+1):
        total += arr[num]

    print(total)



arr = [2,4,5,10]
i = 1
j = 3

find_sum(arr,i,j)

arr = [4,10,5,3,3]
i = 3
j = 3

find_sum(arr,i,j)
```

```
19
3
```

2.)

In [12]:

```
def rotate_arr(arr,d):
    rotate = arr[d:] + arr[:d]
    print(rotate)


arr = [1,2,3,4,5,6,7]
d = 2

rotate_arr(arr,d)

arr = [3,4,5,6,7,1,2]
d = 2

rotate_arr(arr,d)
```

```
[3, 4, 5, 6, 7, 1, 2]
[5, 6, 7, 1, 2, 3, 4]
```

3.)

In [12]:

```
from collections import Counter
def find_second_most_repeated_word(arr):
    count_dict = {}

    #use dictionary to store the word and it's frequency
    for i in range(len(arr)):
        count_dict[arr[i]] = count_dict.get(arr[i],0)+1

    # find second max value
    second_max_value = sorted(count_dict.values(),reverse=True)[1]

    #loop through the values and find second max value.
    for key,value in count_dict.items():
        if value == second_max_value:
            print(key)
            break


# list of strings
arr = ["aaa","bbb","ccc","bbb","aaa","aaa"]

#calling function by passing array
find_second_most_repeated_word(arr)
```

```
bbb
```

4.)

In [37]:

```
list1= [10,15,20,25,30,35,40]
list2 = [25,40,35]

res = []
for num in list1:
    if num not in list2:
        res.append(num)

print(res)
```

```
[10, 15, 20, 30]
```

5.)

In [38]:

```
list1 = [12, -7, 5, 64, -14]

for num in list1:
    if num >= 0:
        print(num,end=" ")
```

```
12 5 64
```

6.)

In [48]:

```
list1 = [0,10,[20,30],40,50,[60,70,80],[90,100,110,120]]
new_list = []

#loop through the given list
#isinstance() returns true if element is a object of mentioned type
#else false
for element in list1:
    if isinstance(element,list):
        new_list.extend(element)
    else:
        new_list.append(element)

print(new_list)
```

```
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]
```

7.)

In [59]:

```
from itertools import combinations

#combinations is a function in itertools used to
#find all the combinations of mentioned size
def find_triplet(arr,total):

    triplets = combinations(arr,3)
    for grp in triplets:
        if sum(grp) == total:
            return grp



array = [12,3,4,1,6,9]
total = 24
print(find_triplet(array,total))

array2 = [1,2,3,4,5]
total = 9
print(find_triplet(array2,total))
```

```
(12, 3, 9)
(1, 3, 5)
```

Strings

1.)

In [64]:

```
def find_missing_characters(string):

    '''
    lower() converts string into lower case.
    ascii values for lower case letters are from 97-122
    chr() converts ascii to char
    '''
    for i in range(97,123):
        if chr(i) not in string.lower():
            print(chr(i),end="")




string1 = "welcome to geeksforgeeks"
find_missing_characters(string1)


print()
string2 = "The quick brown fox jumps"
find_missing_characters(string2)
```

```
abdhijnpquvxyz
adglvyz
```

2.)

In [68]:

```
def count_of_substrings(string):
    length = len(string)
    print((length * (length +1))//2)


count_of_substrings("abc")

count_of_substrings("abcd")
```

```
6
10
```

3.)

In [1]:

```
def alternate_sort(string):

    # store upper and lower case letters into seperate lists
    upper = []
    lower = []
    for char in string:
        if char.isupper():
            upper.append(char)
        else:
            lower.append(char)

    #sort the lists in reverse order
    upper.sort(reverse=True)
    lower.sort(reverse=True)

    #iterate over the lists and add charaters altenatively
    #until anyone of the list becomes empty
    new_string = ""
    while(len(upper)> 0 and len(lower)>0):
        new_string += (upper.pop() + lower.pop())


    #now add the remaining characters to the new string
    new_string += "".join(upper[::-1])
    new_string += "".join(lower[::-1])


    print(new_string)




string1 = "bAwutndekWEdkd"
alternate_sort(string1)

string2 = "abbfDDhGFBvdFDGBNDasZVDFjkb"
alternate_sort(string2)
```

```
AbEdWddekkntuw
BaBaDbDbDbDdDfFhFjFkGsGvNVZ
```

4.)

In [4]:

```
words_list= list(set(input().split(", ")))
words_list.sort()
new_str = ", ".join(words_list)
print(new_str)
```

```
red, white, black, red, green, black
black, green, red, white
```

5.)

In [6]:

```
string = input()
count = {}

for char in string:
    count[char] = count.get(char,0)+1

print(count)
```

```
google.com
{'g': 2, 'o': 3, 'l': 1, 'e': 1, '.': 1, 'c': 1, 'm': 1}
```

6.)

In [13]:

```
string = input()

freq_dict = {}
for char in string:
    freq_dict[char] = freq_dict.get(char,0)+1


min_value = min(freq_dict.values())

print(f"The minimum of all characters in {string} is:",min_value)
```

```
iNeuronNet.com
The minimum of all characters in iNeuronNet.com is: 1
```

7.)

In [13]:

```
string = input()

freq_dict = {}
for char in string:
    freq_dict[char] = freq_dict.get(char,0)+1

odd_keys = []
for key,value in freq_dict.items():
    if value % 2 == 1:
        odd_keys.append(key)


print("The Odd Frequency Characters are:",odd_keys)
```

```
geekforgeeks is best for geeks
The Odd Frequency Characters are: ['g', 'e', 'k', 'i', 'b', 't']
```

Dictionary

1.)

In [18]:

```
def check_pattern(string,pattern):
    char_list = []

    #loop through the string
    for char in string:

        #if char found in pattern
        if char in pattern:

            #if char_list length > 0 and last appended char is not
            #same as current one then append the char to list
            if len(char_list) > 0 and char_list[-1] != char:
                char_list.append(char)
            #if length of char_list is 0 then append
            #char to list
            elif len(char_list) == 0:
                char_list.append(char)

    if pattern == "".join(char_list):
        print(True)
    else:
        print(False)


string1 = input()
pattern1 = input()

check_pattern(string1,pattern1)

string2 = input()
pattern2 = input()

check_pattern(string2,pattern2)
```

```
engineers rock
er
True
engineers rock
gsr
False
```

2.)

In [5]:

```
test_dict = {'Gfg' : 4, 'best' : 9}
test_list = [8, 2]

final_dict = {}
i = 0
for key,value in test_dict.items():
    final_dict[test_list[i]] = {key:value}
    i+=1
print(final_dict)
```

```
{8: {'Gfg': 4}, 2: {'best': 9}}
```

3.)

In [39]:

```
test_dict = {'c': [3], 'b': [12, 10], 'a': [19, 4]}

new_dict = {}
for key in sorted(test_dict.keys()):
    new_dict[key] = sorted(test_dict[key])

print(new_dict)
```

```
{'a': [4, 19], 'b': [10, 12], 'c': [3]}
```

4 .)

In [14]:

```
string = "Python is great and Java is also great"
unique_str = " ".join(list(set(string.split(" "))))

print(unique_str)
```

```
Java and great also is Python
```

5.)

In [30]:

```
#given dictionary
test_dict = {"a" : {"b" : {}},"d" : {"e" : {}},
"f" : {"g" : {}}}

#take out all the outer_keys into a list
outer_keys = list(test_dict.keys())

#empty list to store the inner keys
inner_keys = []

#loop through the test_dict and append the inner key to list
for key,value in test_dict.items():
    inner_keys.extend(list(value.keys()))

#empty dictionary
new_dict = {}

#index to iterate over outer and inner lists
i=0

#loop through the test_dict and add items to new dict
#so that int the result dictionary the inner and outer keys are swapped.
for key,value in test_dict.items():
    new_dict[inner_keys[i]] = {outer_keys[i]:list(value.values())[0]}
    i+=1



print(new_dict)
```

```
{'b': {'a': {}}, 'e': {'d': {}}, 'g': {'f': {}}}
```

6.)

In [74]:

```
def find_size_of_largest_subset_of_string(str_list):
    largest_subset = 0

    #loop through the str_list
    for string in str_list:

        char_list = list(string)
        length = len(string)

        subset = 0

        #check each word in list if it is anagram or not
        #if yes increment the subset count.
        for word in str_list:
            if length == len(word):
                if sorted(char_list) == sorted(list(word)):
                    subset += 1

        if largest_subset < subset:
            largest_subset = subset


    return largest_subset



str_list = ["ant", "magenta", "magnate", "tan", "gnamate"]
print(find_size_of_largest_subset_of_string(str_list))

str_list = ["cars", "bikes", "arcs","steer"]
print(find_size_of_largest_subset_of_string(str_list))
```

```
3
2
```

sets

1.)

In [81]:

```
def check_for_common(a,b):
    a = set(a)
    b = set(b)

    #intersection() is in-built function to find
    #common elements in both sets.
    if len(a.intersection(b)) > 0:
        print(True)
    else:
        print(False)

a = [1, 2, 3, 4, 5]
b = [5, 6, 7, 8, 9]

check_for_common(a,b)

a=[1, 2, 3, 4, 5]
b=[6, 7, 8, 9]

check_for_common(a,b)
```

```
True
False
```

2.)

In [82]:

```
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}

print(set1.intersection(set2))
```

```
{40, 50, 30}
```

3.)

In [89]:

```
def find_max(s):
    s_list = list(s)
    maximum = s_list[0]
    for num in s_list:
        if num > maximum:
            maximum = num
    print("max is",maximum)

def find_min(s):
    s_list = list(s)
    minimum = s_list[0]
    for num in s_list:
        if num < minimum:
            maximum = num
    print("min is",minimum)


set1 = {8, 16, 24, 1, 25, 3, 10, 65, 55}
find_max(set1)

set2 = {4, 12, 10, 9, 4, 13}
find_min(set2)
```

```
max is 65
min is 4
```

4.)

In [94]:

```
x = {'mango', 'apple'}
y = {'mango', 'orange'}
z = {'mango'}

#issubset() checks if one set is subset of other
print("x is subset of y:",x.issubset(y))
print("x is subset of z:",x.issubset(z))
print("y is subset of x:",y.issubset(x))
print("z is subset of y:",z.issubset(y))
print("z is subset of x:",z.issubset(x))
```

```
x is subset of y: False
x is subset of z: False
y is subset of x: False
z is subset of y: True
z is subset of x: True
```

5.)

In [95]:

```
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

#intersection() and difference() are inbuilt functions of set
intersect_set = set1.intersection(set2)
set1 = set1.difference(intersect_set)

print(set1)
print(set2)
```

```
{1, 2, 3}
{4, 5, 6, 7, 8}
```

6.)

In [98]:

```
"""
If we pass dictionary to a set constructor, then the values in set are keys
of the dictionary
"""
test_dict = {'a':1,'b':2}
s = set(test_dict)

print(s)
```

```
{'a', 'b'}
```

Tuple

1.)

In [99]:

```
def remove_tuple_with_length_k(test_list,k):
    for tup in test_list:
        if len(tup) == k:
            test_list.remove(tup)

    print(test_list)


test_list = [(4, 5), (4, ), (8, 6, 7), (1, ), (3, 4, 6, 7)]
K = 2
remove_tuple_with_length_k(test_list,K)
```

```
[(4,), (8, 6, 7), (1,), (3, 4, 6, 7)]
```

2.)

In [105]:

```
tup = (1, 3, 5, 2, 3, 5, 1, 1, 3)

new_tup = tuple()

for num in tup:
    if num not in new_tup:
        new_tup += (num,)

print(new_tup)
```

```
(1, 3, 5, 2)
```

3.)

In [116]:

```
def flatten_tuple(tup):
    new_tup = []
    for li in tup:
        if type(li)  == list:
            new_tup += tuple(li)
        elif type(li) == tuple:
            new_tup += li
        else:
            new_tup += (li,)


    print(tuple(new_tup))



test_tuple = ([5], [6], [3], [8])
flatten_tuple(test_tuple)

test_tuple = ([5, 7, 8])
flatten_tuple(test_tuple)
```

```
(5, 6, 3, 8)
(5, 7, 8)
```

4.)

In [117]:

```
def remove_nested_tuples(tup):
    new_tup = ()
    for i in tup:
        if type(i) == int:
            new_tup += (i,)

    print(new_tup)



tup =  (1, 5, 7, (4, 6), 10)
remove_nested_tuples(tup)
```

```
(1, 5, 7, 10)
```

5.)

In [126]:

```
x = (1,1,0,1,0,0,1)
i = 0
decimal = 0
for digit in x[::-1]:
    decimal += (2**i)*digit
    i+=1

print(decimal)
```

```
105
```

6.)

In [133]:

```
def count_digits(tup):
    return sum ([len(str(num)) for num in tup])


test_list = [(3, 4, 6, 723), (1, 2),(134, 234, 34)]

test_list.sort(key = count_digits)

print(test_list)
```

```
[(1, 2), (5, 6), (3, 4, 6, 723), (134, 234, 34)]
```

MAP & Lambda Function

1.)

In [3]:

```
n1 = [1, 2, 3]
n2 = [4, 5, 6]
n3 = [7, 8, 9]

lam_fun = lambda x,y,z: x + y + z

print(list(map(lam_fun,n1,n2,n3)))
```

```
[12, 15, 18]
```

2.)

In [6]:

```
tup_list = [('red', 'pink'), ('white', 'black'), ('orange', 'green')]
tup_to_str = lambda x: x[0]+" "+x[1]

print(list(map(tup_to_str,tup_list)))
```

```
['red pink', 'white black', 'orange green']
```

3.)

In [10]:

```
list_1 = [6, 5, 3, 9]
list_2 = [0, 1, 7, 7]

#lambda to add x and y
lam_sum = lambda x,y : x + y

#lambda to find difference
lam_sub = lambda x,y : x-y

#lambda function to combine both
lam_combine = lambda x,y : (lam_sum(x,y),lam_sub(x,y))

#call lam_combine and pass list_1 and list_2
print(list(map(lam_combine,list_1,list_2)))
```

```
[(6, 6), (6, 4), (10, -4), (16, 2)]
```

4.)

In [17]:

```
from functools import reduce

fib = lambda n: reduce(lambda x, _: x+[x[-1]+x[-2]],
                                range(n-2), [0, 1])

print(fib(5))
```

```
[0, 1, 1, 2, 3]
```

5.)

In [24]:

```
list_1 = [1, 2, 3, 5, 7, 8, 9, 10]
list_2 = [1, 2, 4, 8, 9]

check_lambda = lambda x :  x in list_1

print(list(filter(check_lambda,list_2)))
```

```
[1, 2, 8, 9]
```

6.)

In [1]:

```
str_list = ['php', 'w3r', 'Python', 'abcd', 'Java', 'aaa']

palindrome_lam = lambda string : string == string[::-1]

print(list(filter(palindrome_lam,str_list)))
```

```
['php', 'aaa']
```

7.)

In [22]:

```
# list of lists
nested_list = [[0], [1, 3], [5, 7], [9, 11], [13, 15, 17]]

#lambda function to find the size of list
find_size = lambda li : len(li)

#list which as sizes of all inner lists
size_list = list(map(find_size,nested_list))

#find min,max sizes of lists
max_size = max(size_list)

min_size = min(size_list)

#find the list with mi
find_list = lambda nested_list,size:(size,nested_list[size_list.index(size)])

print("List with maximum length of lists:")
print(find_list(nested_list,max_size))
print("List with minimum length of lists:")
print(find_list(nested_list,min_size))
```

```
List with maximum length of lists:
(3, [13, 15, 17])
List with minimum length of lists:
(1, [0])
```

8.)

In [24]:

```
num_list = [1, 2, 3, 4, 5, 6, 7]

find_triple = lambda x : x*3

print(list(map(find_triple,num_list)))
```

```
[3, 6, 9, 12, 15, 18, 21]
```

List Comprehension

1.)

In [30]:

```
range_nums = [x for x in range(1,1001)]
digits = [x for x in range(2,10)]

num_list = [num for num in range_nums if True in [True for d in digits if num % d == 0]]

print(num_list)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25, 26, 27,
28, 30, 32, 33, 34, 35, 36, 38, 39, 40, 42, 44, 45, 46, 48, 49, 50, 51, 52,
54, 55, 56, 57, 58, 60, 62, 63, 64, 65, 66, 68, 69, 70, 72, 74, 75, 76, 77,
78, 80, 81, 82, 84, 85, 86, 87, 88, 90, 91, 92, 93, 94, 95, 96, 98, 99, 100,
102, 104, 105, 106, 108, 110, 111, 112, 114, 115, 116, 117, 118, 119, 120, 1
22, 123, 124, 125, 126, 128, 129, 130, 132, 133, 134, 135, 136, 138, 140, 14
1, 142, 144, 145, 146, 147, 148, 150, 152, 153, 154, 155, 156, 158, 159, 16
0, 161, 162, 164, 165, 166, 168, 170, 171, 172, 174, 175, 176, 177, 178, 18
0, 182, 183, 184, 185, 186, 188, 189, 190, 192, 194, 195, 196, 198, 200, 20
1, 202, 203, 204, 205, 206, 207, 208, 210, 212, 213, 214, 215, 216, 217, 21
8, 219, 220, 222, 224, 225, 226, 228, 230, 231, 232, 234, 235, 236, 237, 23
8, 240, 242, 243, 244, 245, 246, 248, 249, 250, 252, 254, 255, 256, 258, 25
9, 260, 261, 262, 264, 265, 266, 267, 268, 270, 272, 273, 274, 275, 276, 27
8, 279, 280, 282, 284, 285, 286, 287, 288, 290, 291, 292, 294, 295, 296, 29
7, 298, 300, 301, 302, 303, 304, 305, 306, 308, 309, 310, 312, 314, 315, 31
6, 318, 320, 321, 322, 324, 325, 326, 327, 328, 329, 330, 332, 333, 334, 33
5, 336, 338, 339, 340, 342, 343, 344, 345, 346, 348, 350, 351, 352, 354, 35
5, 356, 357, 358, 360, 362, 363, 364, 365, 366, 368, 369, 370, 371, 372, 37
4, 375, 376, 378, 380, 381, 382, 384, 385, 386, 387, 388, 390, 392, 393, 39
4, 395, 396, 398, 399, 400, 402, 404, 405, 406, 408, 410, 411, 412, 413, 41
4, 415, 416, 417, 418, 420, 422, 423, 424, 425, 426, 427, 428, 429, 430, 43
2, 434, 435, 436, 438, 440, 441, 442, 444, 445, 446, 447, 448, 450, 452, 45
3, 454, 455, 456, 458, 459, 460, 462, 464, 465, 466, 468, 469, 470, 471, 47
2, 474, 475, 476, 477, 478, 480, 482, 483, 484, 485, 486, 488, 489, 490, 49
2, 494, 495, 496, 497, 498, 500, 501, 502, 504, 505, 506, 507, 508, 510, 51
1, 512, 513, 514, 515, 516, 518, 519, 520, 522, 524, 525, 526, 528, 530, 53
1, 532, 534, 535, 536, 537, 538, 539, 540, 542, 543, 544, 545, 546, 548, 54
9, 550, 552, 553, 554, 555, 556, 558, 560, 561, 562, 564, 565, 566, 567, 56
8, 570, 572, 573, 574, 575, 576, 578, 579, 580, 581, 582, 584, 585, 586, 58
8, 590, 591, 592, 594, 595, 596, 597, 598, 600, 602, 603, 604, 605, 606, 60
8, 609, 610, 612, 614, 615, 616, 618, 620, 621, 622, 623, 624, 625, 626, 62
7, 628, 630, 632, 633, 634, 635, 636, 637, 638, 639, 640, 642, 644, 645, 64
6, 648, 650, 651, 652, 654, 655, 656, 657, 658, 660, 662, 663, 664, 665, 66
6, 668, 669, 670, 672, 674, 675, 676, 678, 679, 680, 681, 682, 684, 685, 68
6, 687, 688, 690, 692, 693, 694, 695, 696, 698, 699, 700, 702, 704, 705, 70
6, 707, 708, 710, 711, 712, 714, 715, 716, 717, 718, 720, 721, 722, 723, 72
4, 725, 726, 728, 729, 730, 732, 734, 735, 736, 738, 740, 741, 742, 744, 74
5, 746, 747, 748, 749, 750, 752, 753, 754, 755, 756, 758, 759, 760, 762, 76
3, 764, 765, 766, 768, 770, 771, 772, 774, 775, 776, 777, 778, 780, 782, 78
3, 784, 785, 786, 788, 789, 790, 791, 792, 794, 795, 796, 798, 800, 801, 80
2, 804, 805, 806, 807, 808, 810, 812, 813, 814, 815, 816, 818, 819, 820, 82
2, 824, 825, 826, 828, 830, 831, 832, 833, 834, 835, 836, 837, 838, 840, 84
2, 843, 844, 845, 846, 847, 848, 849, 850, 852, 854, 855, 856, 858, 860, 86
1, 862, 864, 865, 866, 867, 868, 870, 872, 873, 874, 875, 876, 878, 879, 88
0, 882, 884, 885, 886, 888, 889, 890, 891, 892, 894, 895, 896, 897, 898, 90
0, 902, 903, 904, 905, 906, 908, 909, 910, 912, 914, 915, 916, 917, 918, 92
0, 921, 922, 924, 925, 926, 927, 928, 930, 931, 932, 933, 934, 935, 936, 93
8, 939, 940, 942, 944, 945, 946, 948, 950, 951, 952, 954, 955, 956, 957, 95
8, 959, 960, 962, 963, 964, 965, 966, 968, 969, 970, 972, 973, 974, 975, 97
6, 978, 980, 981, 982, 984, 985, 986, 987, 988, 990, 992, 993, 994, 995, 99
6, 998, 999, 1000]
```

2.)

In [32]:

```
num_list = [x+6 for x in range(1,11)]

print(num_list)
```

```
[7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
```

3.)

In [33]:

```
input_list = [1, 2, 3, 4, 5, 6, 7]
output = {}

for num in input_list:
    output[num] = num**3

print(output)
```

```
{1: 1, 2: 8, 3: 27, 4: 64, 5: 125, 6: 216, 7: 343}
```

4.)

In [36]:

```
state = ['Gujarat', 'Maharashtra', 'Rajasthan']
capital = ['Gandhinagar', 'Mumbai', 'Jaipur']

print(dict(zip(state,capital)))
```

```
{'Gujarat': 'Gandhinagar', 'Maharashtra': 'Mumbai', 'Rajasthan': 'Jaipur'}
```

5.)

In [48]:

```
string = '2459a09b'
num_list = []
for char in string:
    if char.isdigit():
        num_list.append(int(char))

print(num_list)
```

```
[2, 4, 5, 9, 0, 9]
```

6.)

In [49]:

```
names = ['Ch','Dh','Eh','cb','Tb','Td','Chb','Tdb']

filter_lambda = lambda string : string[-1] == 'b' and len(string)>2


print(list(filter(filter_lambda,names)))
```

```
['Chb', 'Tdb']
```

7.)

In [50]:

```
string_list = ['Hello', 'Analytics', 'Vidhya']

rev_strings = lambda string : string[::-1]

print(list(map(rev_strings,string_list)))
```

```
['olleH', 'scitylanA', 'ayhdiV']
```

8.)

Object Oriented Programming

1.)

In [52]:

```
class Vehicle:
    def __init__(self,name,speed,mileage):
        self.name = name
        self.speed =  speed
        self.mileage = mileage
        self.color = "White"



v1 = Vehicle("School Volvo",180,12)
v2 = Vehicle("Audi Q5",240, 18)

print(f"Color: {v1.color}, Vehicle name: {v1.name}, Speed: {v1.speed}, Mileage: {v1.mileage
print(f"Color: {v2.color}, Vehicle name: {v2.name}, Speed: {v2.speed}, Mileage: {v2.mileage
```

```
Color: White, Vehicle name: School Volvo, Speed: 180, Mileage: 12
Color: White, Vehicle name: Audi Q5, Speed: 240, Mileage: 18
```

2.)

In [53]:

```
class Vehicle:
    def __init__(self,name,speed,mileage):
        self.name = name
        self.speed =  speed
        self.mileage = mileage
        self._color = "White"

    @property
    def get_color(self):
        return self._color

    @get_color.setter
    def get_color(self,a):
        self._color = a



v1 = Vehicle("School Volvo",180,12)
v2 = Vehicle("Audi Q5",240, 18)

v1.color = "Blue"
print(v1.color)
```

```
Blue
```

3.)

In [55]:

```
class topClass:
    def top(self):
        print("This is top class")

class middleClass(topClass):
    def middle(self):
        print("This is middle class")

class bottomClass(middleClass):
    def bottom(self):
        print("This is bottom class")


#create object for bottomClass and try to access
#methods from upper classes in the hierarchy
obj  = bottomClass()

obj.bottom()
obj.middle()
obj.top()
```

```
This is bottom class
This is middle class
This is top class
```

4.)

In [57]:

```
class Student:
    def __init__(self):
        print("Student object is created.")

    def __del__(self):
        print("Student object is deleted.")


s =  Student()

del s
```

```
Student object is created.
Student object is deleted.
```

5.)

In [59]:

```
class MyDecorator:
    def __init__(self, function):
        self.function = function

    def __call__(self):

        # We can add some code
        # before function call

        self.function()

        # We can also add some code
        # after function call.


# adding class decorator to the function
@MyDecorator
def function():
    print("function called")

function()
```

```
function called
```

6.) Stack using a List

In [62]:

```
#stack class
class Stack:

    #initialize the stack
    def __init__(self):
        self.stack = []

    #function to add values into stack
    def push(self,value):
        self.stack.append(value)

    #return the top element
    def peek(self):
         return self.stack[-1]

    #delete and return top element
    def pop(self):
        return self.stack.pop()

    #function to return size of stack
    def size(self):
        return len(self.stack)

    #function to check if stack is is_empty
    def is_empty(self):
        return len(self.stack) == 0

    #function to print the stack
    def display(self):
        print(list(self.stack))


#main function
if __name__ =="__main__":

    #stack object
    s = Stack()

    #do some operations on stack
    s.push(5)
    s.push(10)
    s.push(15)
    s.push(20)
    print("After inserting values: ")
    s.display()
    s.is_empty()
    print("Remove and return top element of Stack:",s.pop())
    s.display()
    print("Return top element of Stack:",s.peek())
    print("Size of the stack:",s.size())
```

```
After inserting values:
[5, 10, 15, 20]
Remove and return top element of Stack: 20
[5, 10, 15]
Return top element of Stack: 15
Size of the stack: 3
```

7.) Queue using List

In [64]:

```
#Queue class
class Queue:

    #declare queue
    def __init__(self):
        self.queue = []

    #add values to queue
    def enqueue(self,value):
        self.queue.append(value)

    #remove value from queue
    def dequeue(self):
        return self.queue.pop(0)

    #return front value
    def front_value(self):
        return self.queue[0]

    #return back_value
    def back_value(self):
        return self.queue[-1]

     #return size
    def size(self):
        return len(self.queue)

    #return the total queue
    def display(self):
        return self.queue


#main function
if __name__ == "__main__":

    #create object for Queue class
    q = Queue()

    #insert values into queue
    q.enqueue(10)
    q.enqueue(20)
    q.enqueue(30)
    q.enqueue(40)
    q.enqueue(50)

    #do some other operationsa
    print("Queue after inserting values:",q.display())
    q.dequeue()
    q.dequeue()
    print("Queue after some deletions:",q.display())
    print("Front value:",q.front_value())
    print("Back value:",q.back_value())
    print("Size of queue:",q.size())
```

```
Queue after inserting values: [10, 20, 30, 40, 50]
Queue after some deletions: [30, 40, 50]
Front value: 30
Back value: 50
Size of queue: 3
```

8.) LinkedList

In [67]:

```
class Node:
    def __init__(self,data):
        self.data = data
        self.next = None



class LinkedList:
    def __init__(self):
        self.head = None


    def insert(self,data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
        else:
            a =  self.head
            while a.next is not None:
                a = a.next
            a.next = new_node

    def traverse(self):
        temp = self.head
        while temp is not None:
            print(temp.data)
            temp = temp.next

    def insert_after_specific_node(self,check_node, data):
        temp=self.head
        new_node = Node(data)

        while temp.data != check_node:
            temp =  temp.next

        if temp is None:
            print("There is no mentioned node in LinkedList")
        new_node.next = temp.next
        temp.next = new_node

    def delete_at_end(self):
        temp = self.head
        while temp.next.next is not None:
            temp = temp.next

        temp.next = None

    def delete_a_node(self,data):
        pass

    def detect_loop(self):
        slow_p =  self.head
        fast_p = self.head
        while(slow_p and fast_p and fast_p.next):
            slow_p = slow_p.next
            fast_p = fast_p.next.next
            if slow_p == fast_p:
                return 1
        return 0;
```

```
ll = LinkedList()

ll.insert(10)
ll.insert(20)
ll.insert(30)
ll.insert_after_specific_node(20,25)
ll.delete_at_end()
ll.traverse()
ll.detect_loop()
```

```
10
20
25
```

Out[67]:

```
0
```

9.) Explain What happens you create a object of a class (Inside python) ?

1.) Creating a object for a class means making a model based on the blueprint of the class. Using that object we can access the components of the class. 2.) Memory will be allocated to that object. 3.) Every object is unique means properties are same, but the values can be different.

10.)

In [68]:

```
#
```

Exception Handling

1.)

In [70]:

```
try:
    print(10/0)
except ZeroDivisionError:
    print("Cannot divide with zero")
```

```
Cannot divide with zero
```

2.)

In [74]:

```
try:
    l = [1,2,3]
    print(li[5])
except  ValueError:
    print("Value not found")
except IndexError:
    print("Index not found")
```

```
Index not found
```

In [ ]:

```

```