1 440 Write a python code to demonstrate any three methods of list

[21]:
```
# Method 1: append()
my_list = eval(input("enter list:"))
my_list.append(4)
print("Method 1 - append:", my_list)
# Method 2: extend()
my_list = eval(input("enter list:"))
my_list.extend([5, 6])
print("Method 2 - extend:", my_list)
# Method 3: remove()
my_list = eval(input("enter list:"))
my_list.remove(3)
print("Method 3 - remove:", my_list)
```

```
enter list:[1, 2, 3, 4]
Method 1 - append: [1, 2, 3, 4, 4]
enter list:[1, 2, 3, 4]
Method 2 - extend: [1, 2, 3, 4, 5, 6]
enter list:[1, 2, 3, 4]
Method 3 - remove: [1, 2, 4]
```

2 441 Write a python code to explain map in Python program.

[22]:
```
# Example 1: Doubling each element in a list using map
numbers = eval(input("enter list:"))
doubled_numbers = list(map(lambda x: x * 2, numbers))
print("Doubled Numbers:", doubled_numbers)
# Example 2: Converting strings to uppercase using map
words = eval(input("enter list:"))
uppercase_words = list(map(str.upper, words))
print("Uppercase Words:", uppercase_words)
```

```
enter list:[2, 4, 6, 8]
Doubled Numbers: [4, 8, 12, 16]
enter list:['banana', 'vishal10', 'elderberry']
Uppercase Words: ['BANANA', 'VISHAL10', 'ELDERBERRY']
```

3 442 Write a python code to explain filter in Python program.

[23]:
```
# Example 1: Filtering even numbers from a list
numbers =eval(input("enter list:"))
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print("Even Numbers:", even_numbers)
# Example 2: Filtering words with length greater than 5
words = eval(input("enter list:"))
long_words = list(filter(lambda word: len(word) > 5, words))
print("Long Words:", long_words)
```

```
enter list:[1, 2, 3, 4, 5, 6, 7, 8, 9]
Even Numbers: [2, 4, 6, 8]
enter list:["apple", "banana", "vishal10", "date", "elderberry"]
Long Words: ['banana', 'vishal10', 'elderberry']
```

4 443 Write a Python program to print the even numbers from a given list.

[24]:
```
numbers = eval(input("enter list:"))
l=[]
for num in numbers:
    if num % 2 == 0:
        l.append(num)
print("Even Numbers:", l)
```

```
enter list:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Even Numbers: [2, 4, 6, 8, 10]
```

5 444 Write a Python Program to print the largest even number in a list.

[25]:
```
numbers =  eval(input("enter list:"))
l=[]
for num in numbers:
    if num % 2 == 0:
        l.append(num)
print("largest Even Number:", max(l))
```

```
enter list:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
largest Even Number: 10
```

6 445 Write a Python Program to print the largest odd number in a list.

[27]:
```
numbers =eval(input("enter list:"))
l=[]
for num in numbers:
    if num % 2 != 0:
        l.append(num)
print("largest odd Number:", max(l))
```

```
enter list:[1, 5, 8, 3, 12, 6, 10, 7]
largest odd Number: 7
```

7 446 Write a Python program to swap first and last element of the list.

[29]:
```
my_list =eval(input("enter list:"))
my_list[0], my_list[-1] = my_list[-1], my_list[0]
print("Swapped List:", my_list)
```

```
enter list:[1, 5, 8, 3, 12, 6, 10, 7]
Swapped List: [7, 5, 8, 3, 12, 6, 10, 1]
```

8 447 Write a Python program to find the sum of all the elements in the list.

[32]:
```
numbers = eval(input("enter list:"))
sum_of_elements = 0
for i in numbers:
    sum_of_elements+=i
print("Sum of Elements:", sum_of_elements)
```

```
enter list:[1,2,3,4,5]
Sum of Elements: 15
```

9 448 Write a Python function to sum all the numbers in a list

[31]:
```
numbers = eval(input("enter list:"))
sum_of_elements = sum(numbers)
print("Sum of Elements:", sum_of_elements)
```

```
enter list:[1, 2, 3, 4, 5]
Sum of Elements: 15
```

10 449 Write a Python program of Reversing a List.

[34]:
```
my_list = eval(input("enter list:"))
reversed_list = my_list[::-1]
print("Reversed List:", reversed_list)
```

```
enter list:[1, 2, 3, 4, 5]
Reversed List: [5, 4, 3, 2, 1]
```

[35]:
```
my_list = eval(input("enter list:"))
my_list.reverse()
print("Reversed List:", my_list)
```

```
enter list:[1, 2, 3, 4, 5]
Reversed List: [5, 4, 3, 2, 1]
```

11 450 Write a Python program to Merging two Dictionaries

[36]:
```
dict1 =eval(input("enter dict1:"))
dict2 =eval(input("enter dict2:"))
dict1.update(dict2)
print("Merged Dictionary: ", dict1)
```

```
enter dict1:{'a': 1, 'b': 2}
enter dict2:{'b': 3, 'c': 4}
Merged Dictionary:  {'a': 1, 'b': 3, 'c': 4}
```

12 451 Write a Python program to calculate the sum of the positive and negative numbers of a given list of numbers using lambda function.

[37]:
```
nums =eval(input("enter list:"))
print("Original list:",nums)
total_negative_nums = list(filter(lambda nums:nums<0,nums))
total_positive_nums = list(filter(lambda nums:nums>0,nums))
print("Sum of the positive numbers: ",sum(total_negative_nums))
print("Sum of the negative numbers: ",sum(total_positive_nums))
```

```
enter list:[2, 4, -6, -9, 11, -12, 14, -5, 17]
Original list: [2, 4, -6, -9, 11, -12, 14, -5, 17]
Sum of the positive numbers:  -32
Sum of the negative numbers:  48
```

13 452 Write a Python program to rearrange positive and negative numbers in a given array using Lambda.

[38]:
```
array_nums = eval(input("enter list:"))
print("Original arrays:")
print(array_nums)
result = sorted(array_nums, key = lambda i: 0 if i == 0 else -1 / i)
print("Rearrange positive and negative numbers of the said array:")
print(result)
```

```
enter list:[-1, 2, -3, 5, 7, 8, 9, -10]
Original arrays:
[-1, 2, -3, 5, 7, 8, 9, -10]
Rearrange positive and negative numbers of the said array:
[2, 5, 7, 8, 9, -10, -3, -1]
```

14 453 Write a Python program to find numbers divisible by nineteen or thirteen from a list of numbers using Lambda.

[39]:
```
nums =eval(input("enter list:"))
print("Orginal list:")
print(nums)
result = list(filter(lambda x: (x % 19 == 0 or x % 13 == 0), nums))
print("Numbers of the above list divisible by nineteen or thirteen:")
print(result)
```

```
enter list:[19, 65, 57, 39, 152, 639, 121, 44, 90, 190]
Orginal list:
[19, 65, 57, 39, 152, 639, 121, 44, 90, 190]
Numbers of the above list divisible by nineteen or thirteen:
[19, 65, 57, 39, 152, 190]
```

15 454 Write a Python function to implement linear search algorithm.

Linear search is also called as sequential search algorithm. It is the simplest searching algorithm. In Linear search, we simply traverse the list completely and match each element of the list with the item whose location is to be found. If the match is found, then the location of the item is returned; otherwise, the algorithm returns NULL.

The following is linear search algorithm:

Given a list L of n elements with values or records L0 …. Ln−1, and target value T, the following subroutine uses linear search to find the index of the target T in L.

1. Set i to 0.
2. If Li = T, the search terminates successfully; return i.

3. Increase i by 1.
4. If i < n, go to step 2. Otherwise, the search terminates unsuccessfully.

Input:

Enter the list of numbers: 5 4 3 2 1 10 11 2

The number to search for: 1

Output:

1 was found at index 4.

[40]:
```
def linearSearch(array, n, x):

    # Going through array sequencially
    for i in range(0, n):
        if (array[i] == x):
            return i
    return -1
array=eval(input("enter list:"))
x = int(input("enter find number:"))
n = len(array)
result = linearSearch(array, n, x)
if(result == -1):
    print("Element not found")
else:
    print("Element found at index: ", result)
```

```
enter list:[4, 6, 6, 4, 2, 2, 4, 8, 5, 8]
enter find number:2
Element found at index:  4
```

16 455 Given a list of elements, write a python program to perform grouping of similar elements, as different key-value list in dictionary. Print the dictionary sorted in descending order of frequency of the elements.

- Note: To perform the sorting, use the sorted function by converting the dictionary into a list of tuples. After sorting, convert the list of tuples back into a dictionary and print it.

Input : test_list = [4, 6, 6, 4, 2, 2, 4, 8, 5, 8]

Output : {4: [4, 4, 4], 6: [6, 6], 2: [2, 2], 8: [8, 8], 5: [5]}

Explanation : Similar items grouped together on occurrences.

Input : test_list = [7, 7, 7, 7]

Output : {7 : [7, 7, 7, 7]}

Explanation : Similar items grouped together on occurrences.

[47]:
```
test_list = eval(input("enter list: "))
#  grouping of similar elements, as different key-value list in dictionary
# creating dictionary with empty lists as values
d={}
for i in test_list:
    d[i]=d.get(i,[])
    d[i].append(i)

print(d)
 # Sort the dictionary by frequency in descending order
sorted_list = sorted(d.items(), key=lambda x: len(x[1]), reverse=True)
d=dict(sorted_list)
print("Sort the dictionary by frequency in descending order",d)
```

```
enter list: [4, 6, 6, 4, 2, 2, 4, 8, 5, 8, 2, 2]
{4: [4, 4, 4], 6: [6, 6], 2: [2, 2, 2, 2], 8: [8, 8], 5: [5]}
Sort the dictionary by frequency in descending order {2: [2, 2, 2, 2], 4: [4, 4,
4], 6: [6, 6], 8: [8, 8], 5: [5]}
```

17 456 A digital image in a computer is represented by a pixels matrix. Each image processing operation in a computer may be observed as an operation on the image matrix. Suppose you are given an N x N 2D matrix A (in the form of a list) representing an image. Write a Python program to rotate this image by 90 degrees (clockwise) by rotating the matrix 90 degree clockwise. Write proper code to take input of N from the user and then to take input of an N x N matrix from the user. Rotate the matrix by 90 degree clockwise and then print the rotated matrix.

- Note: You are not allowed to use an extra iterable like list, tuple, etc. to do this. You need to make changes in the given list A itself. Your program should be able to handle any N x N matrix from N = 1 to N = 20.
- input: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
- output: [[7, 4, 1], [8, 5, 2], [9, 6, 3]]

[54]:
```
n=int(input("enter value of n (n*n matrix)"))
matrix=[]
for i in range(0,n):
    matrix.append([])
    for j in range(0,n):
        matrix[i].append(int(input(f"enter element of {i,j}: ")))
print(matrix)
for i in range(0,n):
```

```
    for j in range(0,n):
        if i>j:
            matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]


for r in range(len(matrix)):
    matrix[r].reverse()

print(matrix)
```

```
enter value of n (n*n matrix)3
enter element of (0, 0): 1
enter element of (0, 1): 2
enter element of (0, 2): 3
enter element of (1, 0): 4
enter element of (1, 1): 5
enter element of (1, 2): 6
enter element of (2, 0): 7
enter element of (2, 1): 8
enter element of (2, 2): 9
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
[[7, 4, 1], [8, 5, 2], [9, 6, 3]]
```

18 457 Write a Program to Print Longest Common Prefix from a given list of strings. The longest common prefix for list of strings is the common prefix (starting of string) between all strings. For example, in the given list [“apple”, “ape”, “zebra”], there is no common prefix because the 2 most dissimilar strings of the list “ape” and “zebra” do not share any starting characters. If there is no common prefix between all strings in the list than return -1.

For example,

Input list: [“lessonplan”, “lesson”,“lees”, “length”]

The longest Common Prefix is: le

Input list: [“python”,“pythonprogramming”,“pythonlist”]

The longest Common Prefix is: python

Input list: [“lessonplan”, “lesson”,“ees”, “length”]

The longest Common Prefix is: -1

[9]:
```
l=["lessonplan", "lesson","ees", "length"]
l.sort(reverse=True)
```

```
#print(l)
prefix=""
for i in range(0,len(l[-1])):

    if l[0][i]==l[-1][i]:
        prefix+=l[0][i]
        continue
    else:
        break
if prefix:
    print("The longest Common Prefix is: ",prefix)
else:
    print(-1)
```

```
-1
```

19 458 One of the ways to encrypt a string is by rearranging its characters by certain rules, they are broken up by threes, fours or something larger. For instance, in the case of threes, the string 'secret message' would be broken into three groups. The first group is sr sg, the characters at indices 0, 3, 6, 9 and 12. The second group is eemse, the characters at indices 1, 4, 7, 10, and 13. The last group is ctea, the characters at indices 2, 5, 8, and 11. The encrypted message is sr sgeemsectea.

```
If the string 'secret message' would be broken into four groups. The first group is seeg, the
```

```
(A). Write a program that asks the user for a string, and an integer determining whether to bre
```

For example,

Input message: This is python, a programming language

Input Key: 4

Output Encrypted Message: T poaomgnghiyn gm geist,prilus h ranaa

Input message: This is python, a programming language

Input Key: 7

Output Message: T ,ggahp r giyaalest ma hpmniorigsnonu

20 (B). If you get a message which is encoded by the method above then, Write a decryption program for the general case. Taking input of any encrypted string from user with key number used while breaking message apart during encryption.

For example,

Input Encrypted message: Hloe gl o sogrilw g epntstfii o yotay hee nnh aoiortiimreegehrun nhnse ne

Input Key used during encryption: 5

Output Decrypted Message: Hi hello how are you going to learn python in this semester of engineering

Input Encrypted message: Ig ntot oopid ys lt dehaaao yrn

Input Key used during encryption: 8

Output Decrypted Message: It is a good day to learn python

Input Encrypted message: istemoaa!t e ym ntt p eiohitlgs

Input Key used during encryption: 4

Output Decrypted Message: it is not the time to play games!

21 (C). From the output string (Output Decrypted Message) of above program (Part-B), create a Dictionary with Key as First Character and Value as list of words Starting with that Character from above string. And print that dictionary by sorting it based on the number of elements in a list of values in descending order.

Note: Consider capital and lower first character of words as same character in this program. For ex. 'Hi' and 'hello' both will be considered starting from 'h'.

For example,

Enter Decrypted Message: Hi hello how are you going to learn python in this semester of engineering

Output: {'h': ['Hi', 'hello', 'how'], 't': ['to', 'this'], 'a': ['are'], 'y': ['you'], 'g': ['going'], 'l': ['learn'], 'p': ['python'], 'i': ['in'], 's': ['semester'], 'o': ['of'], 'e': ['engineering']}

[68]:
```
decrypted_message="Hi hello how are you going to learn python in this semester␣
 ↪of engineering"
word_dict = {}
words = decrypted_message.split()
for word in words:
        first_char = word[0].lower()  # Considering capital and lower first␣
 ↪characters as the same
```

```
        if first_char in word_dict:
            word_dict[first_char].append(word)
        else:
            word_dict[first_char] = [word]
print(word_dict)
x=sorted(word_dict.items(),key=lambda x:len(x[1]),reverse=True)
print(x)
```

```
{'h': ['Hi', 'hello', 'how'], 'a': ['are'], 'y': ['you'], 'g': ['going'], 't':
['to', 'this'], 'l': ['learn'], 'p': ['python'], 'i': ['in'], 's': ['semester'],
'o': ['of'], 'e': ['engineering']}
[('h', ['Hi', 'hello', 'how']), ('t', ['to', 'this']), ('a', ['are']), ('y',
['you']), ('g', ['going']), ('l', ['learn']), ('p', ['python']), ('i', ['in']),
('s', ['semester']), ('o', ['of']), ('e', ['engineering'])]
```

part A

[10]:
```
s=input()
s1=""
key=int(input())
for i in range(key):
    s1+=s[i::key]
print(s1)
```

```
This is python, a programming language
5
Tit omlahshagiagi o rnnespnpagg y,rm u
```

part B

[17]:
```
string=input('what is your message: ')
key=int(input())
length=len(string)#to get the length of the input
part=length//key #half of the input
extra=length%key #extra characters after dividing string in equal parts
part1=string[:(part+1)*extra]
part2=string[(part+1)*extra:]
msg=''
for i in range(part+1):
    if i<part:
        msg+=part1[i::part+1]+part2[i::part]
    else:
        msg+=part1[i::part+1]
print(msg)
```

```
what is your message: Tit omlahshagiagi o rnnespnpagg y,rm u
5
This is python, a programming language
```

22 459 Write a python program to print all possible combinations from the three Digits and also count unique values inside a list and also find list product excluding duplicates and also find sum of list's elements excluding duplicates.

Examples:

To print all possible combinations

Input: [1, 2, 3]

Output:

1 2 3

1 3 2

2 1 3

2 3 1

3 1 2

3 2 1

Count unique values inside a list

input = [1, 2, 3]

No of unique items are: 3

input = [1, 2, 2]

No of unique items are: 2

input = [2, 2, 2]

No of unique items are: 3

List product excluding duplicates

Input: [2, 3, 5]

Duplication removal list product: 30

Input: [2, 2, 3]

Duplication removal list product: 6

Sum of list's elements excluding duplicates

Input: [1, 3, 5]

Output: 9

Input: [1, 2, 2]

Output: 3

[20]:
```
L1=int(input("enter input1"))
L2=int(input("enter input1"))
L3=int(input("enter input1"))
L=[L1,L2,L3]
for i in range(n):
        for j in range(n):
            for k in range(n):

                # check if the indexes are not
                # same
                if (i!=j and j!=k and i!=k):
                    print(L[i], L[j], L[k])
```

```
enter input11
enter input12
enter input13
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

23 460 • Use appropriate comment lines to divide subprograms.

- Also demonstrate the program with one example test case. (Example test input and output are given)

PART - A

- Using map function, write a Python program to convert the given list into a tuple of strings. For the given input, the program must print the output as shown below -

Input – [1,2,3,4]

Output – ('1','2','3','4')

PART - B

- Write a Python program that multiply each number of the given list with 10 using lambda function. For the given input, the program must print the output as shown below -

Input – [1,2,3,4]

Output – [10,20,30,40]

PART - C

- Write a Python program that multiply all elements of the given list using reduce function and return the product. For the given input, the program must print the output as shown below -

Input – [1,2,3,4]

Output – 24 (which is 123*4)

PART - D

- Write a Python program satisfying following conditions -
- Create a python function countchar() that count the character of a string in a given string without using inbuilt functions. For the given input, the program must print the output as shown below -

Given input string – 'hello'

countchar('l')

Output : 2

- Create a python function findchar() that find the index of first occurrence of a character in a given string without using inbuilt functions. It should return -1 if it does not find the character. For the given input, the program must print the output as shown below -

Given input string – 'helloe'

findchar('e')

Output : 1

findchar('z')

Output : -1

24 part A

- Using map function, write a Python program to convert the given list into a tuple of strings. For the given input, the program must print the output as shown below -

Input – [1,2,3,4]

Output – ('1','2','3','4')

[28]:
```
l=eval(input("enter a list"))
t=tuple(map(lambda x:str(x),l))
print("output",t)
```

```
enter a list[1,2,3]
output ('1', '2', '3')
```

25 part B

- Write a Python program that multiply each number of the given list with 10 using lambda function. For the given input, the program must print the output as shown below -

Input – [1,2,3,4]

Output – [10,20,30,40]

[29]:
```
l=eval(input("enter a list"))
t=list(map(lambda x:x*10,l))
print("output",t)
```

```
enter a list[1,2,3,4]
output [10, 20, 30, 40]
```

PART - C

- Write a Python program that multiply all elements of the given list using reduce function and return the product. For the given input, the program must print the output as shown below -

Input – [1,2,3,4]

Output – 24 (which is 123*4)

[31]:
```
from functools import reduce
l=eval(input("enter list:"))
ans = reduce(lambda x, y: x * y, l, 1)
print(ans)
```

```
enter list:[1,2,3,4,5]
120
```

26 PART - D

- Write a Python program satisfying following conditions - Create a python function countchar() that count the character of a string in a given string without using inbuilt functions. For the given input, the program must print the output as shown below -

Given input string – 'hello'

countchar('l')

Output : 2

- Create a python function findchar() that find the index of first occurrence of a character in a given string without using inbuilt functions. It should return -1 if it does not find the character. For the given input, the program must print the output as shown below -

Given input string – 'helloe'

findchar('e')

Output : 1

findchar('z')

Output : -1

[33]:
```
def countchar(s, char):
    count = 0
    for c in s:
```

```
        if c == char:
            count += 1
    return count

def findchar(s, char):
    for i in range(len(s)):
        if s[i] == char:
            return i
    return -1

# Example usage:
input_string = input("enter string: ")
search_char = input("enter find char:")

# Count occurrences of a character
char_count = countchar(input_string, search_char)
print(f"Output count for '{search_char}': {char_count}")

# Find the index of the first occurrence of a character
char_index = findchar(input_string, search_char)
print(f"Output index for '{search_char}': {char_index}")
```

```
enter string: hello
enter find char:k
Output count for 'k': 0
Output index for 'k': -1
```

27 461 Write a Python program to calculate the sum of the positive and negative numbers of the below given list of numbers using lambda function.

Input : m = [2, 4, -6, -9, 11, -12, 14, -5, 17]

Output : Sum of the positive numbers: -32

```
        Sum of the negative numbers:  48
```

[35]:
```
m=eval(input("enter list: "))
sum_posative=(sum(list(filter(lambda x:x>0,m))))
sum_negative=(sum(list(filter(lambda x:x<0,m))))
print("Sum of the positive numbers:",sum_posative)
print("Sum of the negative numbers:",sum_negative)
```

```
enter list: [2, 4, -6, -9, 11, -12, 14, -5, 17]
Sum of the positive numbers: 48
Sum of the negative numbers: -32
```

28 462 Create a python program which takes password as input and a function which checks whether the given password is valid or not under following conditions without using the RegEx module in Python language.

Conditions required for a valid password:

1. Password strength should be at least 8 characters long
2. Password should contain at least one uppercase and one lowercase character.
3. Password must have at least one number.

[36]:
```
def password_check(password):
    l, u, p, d = 0, 0, 0, 0
    if len(password) >= 8:
        for i in password:
        # counting lowercase alphabets
            if (i.islower()):
                l+=1
        # counting uppercase alphabets
            if (i.isupper()):
                u+=1
        # counting digits
            if (i.isdigit()):
                d+=1
        if l>=1 and u>=1 and d>=1 :
            print("valid")
        else:
            print("invalid")
    else:
        print("invalid")
password=input("enter password:")
password_check(password)
```

```
enter password:lk#$$55G
valid
```

29 463 Write a python program with user defined function that reads the words from paragraph and stores them as keys in a dictionary and count the frequency of it as a value .

For Example:

Input string: “Dog the quick brown fox jumps over the lazy dog”

Output: {‘the’: 2, ‘jumps’: 1, ‘brown’: 1, ‘lazy’: 1, ‘fox’: 1, ‘over’: 1, ‘quick’: 1, ‘dog’: 2}

[45]:
```
s=input("enter a string: ")
l=s.split()
d={}
for i in l:
    d[i]=d.get(i,0)+1
print(d)
```

```
enter a string: Dog the quick brown fox jumps over the lazy dog"
{'Dog': 1, 'the': 2, 'quick': 1, 'brown': 1, 'fox': 1, 'jumps': 1, 'over': 1,
'lazy': 1, 'dog"': 1}
```

[42]:
```
d={1:2,3:4}
d['dog']=d.get("dog",0)+1
d
```

[42]: `{1: 2, 3: 4, 'dog': 1}`

30 464 Write a python Program to check entered password by user is correct or not. Entered password is correct if it has upper character, lower character , digits (but not more than 3 digits) ,special character and length is greater than or equal to eight and less than equal to fifteen. Get the digits from entered password and convert it in to number and then convert it in to English Word .

Example:

case 1

pw= R@m@3fa1tu9e$

Valid Password

num= 319

three hundred and nineteen

case 2

pw= S@m@6a1tue$

Valid Password

num= 61

sixty-one

case 3

pw= S@m@6a26u8$

Invalid Password

[55]:
```
l, u, p, d = 0, 0, 0, 0
s = input("enter string: ")
print("pw=",s)
if (len(s) >= 8):
    for i in s:
 # counting lowercase alphabets
        if (i.islower()):
            l+=1
 # counting uppercase alphabets
        if (i.isupper()):
            u+=1
 # counting digits
        if (i.isdigit()):
            d+=1
 # counting the mentioned special characters
        if(i=='@'or i=='$' or i=='_'):
            p+=1
if (l>=1 and u>=1 and p>=1 and d>=1 and d<4 and l+p+u+d==len(s)):
    print("Valid Password")
    no=""
    for i in s:
        if i.isdigit():
            no=no+i
    num=int(no)
    print("num=",num)

    def int_to_en(num):
        d = { 0 : 'zero', 1 : 'one', 2 : 'two', 3 : 'three', 4 : 'four', 5 :␣
 ↪'five',
     6 : 'six', 7 : 'seven', 8 : 'eight', 9 : 'nine', 10 : 'ten',
     11 : 'eleven', 12 : 'twelve', 13 : 'thirteen', 14 : 'fourteen',
     15 : 'fifteen', 16 : 'sixteen', 17 : 'seventeen', 18 : 'eighteen',
     19 : 'nineteen', 20 : 'twenty',
     30 : 'thirty', 40 : 'forty', 50 : 'fifty', 60 : 'sixty',

     70 : 'seventy', 80 : 'eighty', 90 : 'ninety' }
        k = 1000

        if (num < 20):
            return d[num]
        if (num < 100):
            if num % 10 == 0:
                return d[num]
            else:
                return d[num // 10 * 10] + '-' + d[num % 10]
        if (num < k):
            if num % 100 == 0:
```

```
                return d[num // 100] + ' hundred'
            else:
                return d[num // 100] + ' hundred and ' + int_to_en(num % 100)
    print(int_to_en(num))
else:
       print("Invalid Password")
```

```
enter string: 999SSdd@@@
pw= 999SSdd@@@
Valid Password
num= 999
nine hundred and ninety-nine
```

31 465 Write a Python Program using function to count number of strings where the string length is 3 or more and the first and last character are same from a given list of string.

Example :

Input: [‘abc’,‘xyz’,‘aba’,‘2112’,‘123451’,‘12345’]

Output: 3

[56]:
```
strings=eval(input("enter list: "))
count = 0
for string in strings:
        if len(string) >= 3 and string[0] == string[-1]:
            count += 1
print("count=",count)
```

```
enter list: ['abc', 'xyz', 'aba', '2112', '123451', '12345']
count= 3
```

32 466 Given a list L of size N. You need to count the number of special elements in the given list. An element is special if removal of that element makes the list balanced. The list will be balanced if sum of even index elements is equal to the sum of odd elements. Also print the updated lists after removal of special elements.

Example 1:

Input:

L=[5, 5, 2, 5, 8]

Output:

Original List: [5, 5, 2, 5, 8]

Index to be removed is: 0

List after removing index 0 : [5, 2, 5, 8]

Original List: [5, 5, 2, 5, 8]

Index to be removed is: 1

List after removing index 1 : [5, 2, 5, 8]

Total number of special elements: 2

Explaination:

If we delete L[0] or L[1], list will be balanced.

[5, 2, 5, 8]

(5+5) = (2+8)

So L[0] and L[1] are special elements, So Count is 2.

After removal of the special elements, list will be: [5, 2, 5, 8]

Example 2:

Input:

L=[2,1,6,4]

Output:

Original List: [2, 1, 6, 4]

Index to be removed is: 1

List after removing index 1 : [2, 6, 4]

Total Number of Special elements: 1

Explaination:

If we delete L[1] from list : [2,6,4]

(2+4) = (6)

Here only 1 special element. So Count is 1.

After removal of special element, list will be : [2,6,4]

[57]:
```
l=eval(input("enter list"))
print("Original List:",l)
count=0
for i in range(0,len(l)):
    c=l.copy()
    c.pop(i)
    sum1=sum(c[0::2])
    sum2=sum(c[1::2])
```

```
    if sum1==sum2:
        print("index to be removed is: ",i)
        count+=1
        print("List after removing index ",i,"is ",l[0:i]+l[i+1:])
print("Total Number of Special elements:",count)
```

```
enter list[2, 1, 6, 4]
Original List: [2, 1, 6, 4]
index to be removed is:  1
List after removing index  1 is  [2, 6, 4]
Total Number of Special elements: 1
```

33 467 Write Python Program to create a dictionary with the key as the first character and value as a list of words starting with that character.

Example:

Input: Don't wait for your feelings to change to take the action. Take the action and your feelings will change

Output:

{'D': ['Don't'], 'w': ['wait', 'will'], 'f': ['for', 'feelings', 'feelings'], 'y': ['your', 'your'], 't': ['to', 'to', 'take', 'the', 'the'], 'c': ['change', 'change'], 'a': ['action.', 'action', 'and'], 'T': ['Take']}

[58]:
```
s=input("enter a string: ")
l=s.split()
d={}
for i in l:
    d[i[0]]=d.get(i[0],[])
    d[i[0]].append(i)
print(d)
```

```
enter a string: Don't wait for your feelings to change to take the action. Take
the action and your feelings will change
{'D': ['Don't'], 'w': ['wait', 'will'], 'f': ['for', 'feelings', 'feelings'],
'y': ['your', 'your'], 't': ['to', 'to', 'take', 'the', 'the'], 'c': ['change',
'change'], 'a': ['action.', 'action', 'and'], 'T': ['Take']}
```

34 468

- d={"student0":'Student@0',"student1":'Student@11',"student2":'Student@121', "student3":'Student@052',"student4":'Student@01278',"student5":'Student@0125', "student6":'Student@042', "student7":'Student@07800',"student8":'Student@012', "student9":'Student@04789'} ### Write a python program to update the password of any user given the above dictionary(d) which stores the username as the key of the dictionary and the username's password as the value of the dictionary. print the updated dictionary

and print the username and password according to ascending order of password length of the updated dictionary.

- For the password updating of that username follow some instructions.

- Give the three chances to user enter the correct username and password. If the user does not enter the correct username and password then display “enter correct password and username”. if the user does not enter the correct username and password in a given three chances then display “enter correct password and username” and “try after 24h”
- If the user enters the correct username and password in a given three chances. Give the three chances to user enter a new password to update the password of that username. If the user enters a new password not follow the below format, then display “follow the password format”. if the user does not enter the password in a given format in a given three chances, then display “follow the password format” and “try after 24h”
- The check, of whether the new password format is correct or wrong makes the user define a function. That user define a function to return True or False for password valid or not. That user define function return value used in this program for new password validation.

o New password must have the below format: 1. at least 1 number between 0 and 9

2. at least 1 upper letter (between a and z)
3. at least 1 lower letter (between A and Z)
4. at least 1 special character out of @$_
5. minimum length of the password is 8 and the maximum length is 15
6. Do not use space and other special characters. Only uses @$_

- If the new password follows the format of the password in a given three chances. then print the updated dictionary and print the username and password according to ascending order of password length of an updated dictionary. If the dictionary is not updated then take the old dictionary

[59]:
```
def password_check(password):
    l, u, p, d = 0, 0, 0, 0
    if (len(password) >= 8 and len(password) <= 15):
        for i in password:
        # counting lowercase alphabets
            if (i.islower()):
                l+=1
        # counting uppercase alphabets
            if (i.isupper()):
                u+=1
        # counting digits
            if (i.isdigit()):
                d+=1
        # counting the mentioned special characters
            if(i=='@'or i=='$' or i=='_'):
                p+=1
```

```
        if (l>=1 and u>=1 and p>=1 and d>=1 and l+p+u+d==len(password)):
            return True
        else:
            return False
    else:
        return False
d={"student0":'Student@0',"student1":'Student@11',"student2":
 ↪'Student@121',"student3":'Student@052',"student4":'Student@01278',
  "student5":'Student@0125',   "student6":'Student@042',"student7":
 ↪'Student@07800',"student8":'Student@012',
  "student9":'Student@04789'}
first_password_username_count=0
s="  "
while first_password_username_count<3:
    if s=="stop":
        break
    username=input("enter correct username:")
    password=input("enter correct password:")
    if ((username not in d.keys()) or (d[username]!=password)):
        print("enter correct username and password")
        first_password_username_count+=1
        continue
    else:
        #update password
        update_count=0
        while(update_count<3):
            update_password=input("enter update password: ")
            if password_check(update_password):
                d[username]=update_password
                s="stop"
                break
            else:
                update_count+=1
                print(""" # The Password must have:
         1. at least 1 number between 0 and 9
         2. at least 1 upper letter (between a and z)
         3. at least 1 lower letter (between A and Z)
         4. at least 1 special character out of @$_:
         5. minimum length of password is 8 and maximum length is 15""")
                continue
        else:
            print("try after 24h")
            s="stop"
            break
else:
    print("try after 24h")
```

```
print(d)
#Find and print longest and shortest password with its username
sorted_list=sorted(d.items(),key=lambda x:len(x[1]))
print("longest password",sorted_list[-1][0],"--",sorted_list[-1][1])
print("shortest password",sorted_list[0][0],"--",sorted_list[0][1])
```

```
enter correct username:student0
enter correct password:Student@0
enter update password: Student@88888
{'student0': 'Student@88888', 'student1': 'Student@11', 'student2':
'Student@121', 'student3': 'Student@052', 'student4': 'Student@01278',
'student5': 'Student@0125', 'student6': 'Student@042', 'student7':
'Student@07800', 'student8': 'Student@012', 'student9': 'Student@04789'}
longest password student9 -- Student@04789
shortest password student1 -- Student@11
```