"Unit-4&5 QB SOLUTION"

Prepared by: Prof. Swati patel

1. Check if a string is palindrome:

In [2]:
```
s= "radar"
if s == s[::-1]:
    print("string is palindrome")
else:
    print("string is not palindrome")
```

```
string is palindrome
```

2. Find length of a string in python:

In [2]:
```
string_length = len("Hello, World!")
print("Length of the string:", string_length)
```

```
Length of the string: 13
```

3. Find length of a string without using len function:

In [3]:
```
s="Hello, World!"
count = 0
for char in s:
    count += 1
print( "Length of the string:",count)
```

```
Length of the string: 13
```

4. Count uppercase and lowercase letters:

In [18]:
```
u=0
l=0
s = "Hello, World!"
for char in s:
    if char.isupper():
        u=u+1
    if char.islower():
        l=l+1
print("Uppercase count:", u)
print("Lowercase count:", l)
```

```
Uppercase count: 2
Lowercase count: 8
```

5. Demonstrate negative index in a Tuple:

In [5]:
```
my_tuple = (1, 2, 3, 4, 5)
print("Last element:", my_tuple[-1])
print("Second to last element:", my_tuple[-2])
```

```
Last element: 5
Second to last element: 4
```

6. Remove I'th character from string:

In [6]:
```
s = "Hello, World!"
i = 7
print(s[:i] + s[i+1:])
```

```
Hello, orld!
```

7. Create a string made of first, middle, and last character:

In [5]:
```
s= "Hello"
print(s[0] + s[len(s)//2] + s[-1])
```

```
Hlo
```

8. Find all occurrences of a substring ignoring case:

In [8]:
```
s = "Hello, hello, hEllo"
sub = "h"
for i in range(len(s)):
    if s[i:i+len(sub)].lower() == sub.lower():
        print(i)
```

```
0
7
14
```

9. Calculate the sum and average of digits in a string:

In [9]:
```
s=(input("enter string: "))
sum=0
c=0
for i in s:
    if i.isdigit()==True:
        sum+=int(i)
        c=c+1
print("sum",sum)
print(c)
print("avg=",sum/c)
```

```
enter string: asd123s
sum 6
3
avg= 2.0
```

10. Reverse a given string:

In [10]:
```
original_string = "Hello, World!"
reversed_string = original_string[::-1]
print("Original String:", original_string)
print("Reversed String:", reversed_string)
```

```
Original String: Hello, World!
Reversed String: !dlroW ,olleH
```

11. Print even length words in a string:

In [10]:
```
s=input("enter string: ")
n=s.split(" ")
for i in n:
    if len(i)%2==0:
        print(i,end=" ")
```

```
enter string: swati patel lj
lj
```

12. Uppercase Half String:

In [11]:
```
s=input("enter string: ")
new_string1=s[0:len(s)//2:1].upper()
new_string2=s[len(s)//2:len(s)+1:1]
print(new_string1+new_string2)
```

```
enter string: hello world
HELLO world
```

13. Capitalize first and last character of each word:

In [12]:
```
s=input("enter a string: ")
s = s.title()
result = ""
for word in s.split():
    result += word[:-1] + word[-1].upper() + " "
print(result)
```

```
enter a string: swati patel
SwatI PateL
```

14. Create a string made of the middle three characters:

In [13]:
```
s=input("enter string: ")
new_string=s[(len(s)//2)-1:(len(s)//2)+2]
print(new_string)
```

```
enter string: hello
ell
```

15. Check if two strings are balanced:

In [14]:
```
"""Write a program to check if two strings are balanced. For example,
strings s1 and s2 are balanced if all the characters in the s1 are present in s2.
The character’s position doesn’t matter."""
s1=input("enter string: ")
s2= input("enter string: ")
flag=0
for i in s1:
    if i in s2:
        continue
    else:
        flag=1
if(flag ==1):
    print("not balanced")
else:
    print("balanced")
```

```
enter string: swati
enter string: pswatih
balanced
```

16. Split a string on hyphens:

In [16]:
```
input_string = "hello-world-how-are-you"
split_string = input_string.split('-')
print("Split String:", split_string)
```

```
Split String: ['hello', 'world', 'how', 'are', 'you']
```

17. Print maximum and minimum elements in given Tuple:

In [17]:
```
my_tuple = (3, 1, 4, 1, 5, 9, 2, 6, 5, 3)
max_element = max(my_tuple)
min_element = min(my_tuple)
print("Maximum element:", max_element)
print("Minimum element:", min_element)
```

```
Maximum element: 9
Minimum element: 1
```

18. Print even numbers from given Tuple:

In [18]:
```
my_tuple = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
even_numbers = [num for num in my_tuple if num % 2 == 0]
print("Even numbers:", even_numbers)
```

```
Even numbers: [2, 4, 6, 8, 10]
```

19. Print sum of even numbers and sum of odd numbers from elements in Tuple:

In [1]:
```
my_tuple = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
even_numbers_sum=0
odd_numbers_sum=0
for num in my_tuple:
    if num % 2 == 0:
        even_numbers_sum+=num
    else:
        odd_numbers_sum+=num
print("Sum of even numbers:", even_numbers_sum)
print("Sum of odd numbers:", odd_numbers_sum)
```

```
Sum of even numbers: 30
Sum of odd numbers: 25
```

Write a Python program using function to shift the decimal digits n places to the left, wrapping the extra digits around. If shift > the number of digits of n, then reverse the string.

> Note: Function will take two parameters: 1. The number 2. How much shift user want

```
Example:
Input: n=12345 shift=1
Output: Result=23451

Input: n=12345 shift=3
Output: Result=45123

Input: n=12345 shift=5
Output: Result=12345

Input: n=12345 shift=6
Output: Result=54321
```

In [20]:
```
def shift_decimal_digits(number, shift):
    str_number = str(number)
    num_digits = len(str_number)
    # Reverse the string if shift > number of digits
    if shift > num_digits:
        result = str_number[::-1]
    else:
        # Shift the digits to the left, wrapping around
        result = str_number[shift:] + str_number[:shift]
    return int(result)
n1 = 12345
shift1 = 1
result1 = shift_decimal_digits(n1, shift1)
print(f"Input: n={n1} shift={shift1}\nOutput: Result={result1}")

n2 = 12345
shift2 = 3
result2 = shift_decimal_digits(n2, shift2)
print(f"\nInput: n={n2} shift={shift2}\nOutput: Result={result2}")

n3 = 12345
shift3 = 5
result3 = shift_decimal_digits(n3, shift3)
print(f"\nInput: n={n3} shift={shift3}\nOutput: Result={result3}")

n4 = 12345
shift4 = 6
result4 = shift_decimal_digits(n4, shift4)
print(f"\nInput: n={n4} shift={shift4}\nOutput: Result={result4}")
```

```
Input: n=12345 shift=1
Output: Result=23451

Input: n=12345 shift=3
Output: Result=45123

Input: n=12345 shift=5
Output: Result=12345

Input: n=12345 shift=6
Output: Result=54321
```

Write a Python programme that accepts a string and calculate the number of uppercase letters, lowercase letters and number of digits.

> For example,

```
Input: Hello Pyth@n is 100% easy

Output:

Uppercase letters : 2
Lowercase letters : 14
Digits : 3
```

In [17]:
```
u=0
l=0
d=0
s= "Hello, World123!"
for char in s:
    if char.isupper():
        u=u+1
    if char.islower():
        l=l+1
    if char.isdigit():
        d+=1
print("Uppercase count:", u)
print("Lowercase count:", l)
print("Digits count:", d)
```

```
Uppercase count: 2
Lowercase count: 8
Digits count: 3
```

Write a python program to check the validity of a Password.

Primary conditions for password validation:

1. Minimum 8 characters.
2. The alphabet must be between [a-z]
3. At least one alphabet should be of Upper Case [A-Z]
4. At least 1 number or digit between [0-9]
5. At least 1 character from [_ or @ or $]

```
Examples:
Input: Ram@_f1234
Output: Valid Password

Input: Rama_fo$ab
Output: Invalid Password
Explanation: Number is missing

Input: Rama#fo9c
Output: Invalid Password
Explanation: Must consist from _ or @ or \$
```

In [19]:
```
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
        # 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)):
            print("valid")
        else:
            print("invalid")
    else:
        print("invalid")
password_check("Ram@_f1234")
```

```
valid
```

Write a Python program to return another string similar to the input string, but with its case inverted.

> For example, input of “Mr. Ed” will result in “mR. eD” as the output string. Note: Use of built in swapcase function is prohibited.

In [21]:
```
input_string = "Mr. Ed"
print(input_string)
inverted_string = ""
for char in input_string:
        if char.isupper():
            inverted_string += char.lower()
        elif char.islower():
            inverted_string += char.upper()
        else:
            inverted_string += char
print(inverted_string)
```

```
Mr. Ed
mR. eD
```

Write a Python program to create a Caesar encryption.

> Note: In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a right shift of 3, A would be replaced by D, E would become H, and so on. The method is named after Julius Caesar, who used it in his private correspondence.

```
For Example:

Input Text  : LJIET ENG
Shift : 3
Cipher:  OMLHW HQJ
```

In [23]:
```
def encrypt(message,key):
    message = message.upper()
    alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    result = ""
    for letter in message:
        if letter in alpha: #if the letter is actually a letter
            #find the corresponding ciphertext letter in the alphabet
            letter_index = (alpha.find(letter) + key) % len(alpha)
            result = result + alpha[letter_index]
        else:
            result = result + letter
    return result
input_text = "LJIET ENG"
shift_value = 3
cipher_text = encrypt(input_text, shift_value)
print("Input Text  :", input_text)
print("Shift        :", shift_value)
print("Cipher       :", cipher_text)
```

```
Input Text  : LJIET ENG
Shift        : 3
Cipher       : OMLHW HQJ
```

Write a program to check if two strings are balanced. For example, strings s1 and s2 are balanced if all the characters in the s1 are present in s2 and length of s1 & s2 should be same. The character's position doesn't matter.

> Example :

```
s1 = hello
s2 = olleh
Balanced
```

In [25]:
```
s1 = "hello"
s2 = "olleh"
    # Check if lengths are equal
if len(s1) != len(s2):
        print("not balanced")
# Check if all characters in s1 are present in s2
for char in s1:
    if char not in s2:
        print("not balanced")
print("balanced")
```

```
balanced
```

UNIT 5

Write a python code to demonstrate any three methods of list

In [25]:
```
# Method 1: append()
my_list = [1, 2, 3]
my_list.append(4)
print("Method 1 - append:", my_list)
# Method 2: extend()
my_list.extend([5, 6])
print("Method 2 - extend:", my_list)
# Method 3: remove()
my_list.remove(3)
print("Method 3 - remove:", my_list)
```

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

Write a python code to explain map in Python program.

In [26]:
```
# Example 1: Doubling each element in a list using map
numbers = [1, 2, 3, 4, 5]
doubled_numbers = list(map(lambda x: x * 2, numbers))
print("Doubled Numbers:", doubled_numbers)
# Example 2: Converting strings to uppercase using map
words = ["apple", "banana", "cherry"]
uppercase_words = list(map(str.upper, words))
print("Uppercase Words:", uppercase_words)
```

```
Doubled Numbers: [2, 4, 6, 8, 10]
Uppercase Words: ['APPLE', 'BANANA', 'CHERRY']
```

Write a python code to explain filter in Python program.

In [27]:
```
# Example 1: Filtering even numbers from a list
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
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 = ["apple", "banana", "cherry", "date", "elderberry"]
long_words = list(filter(lambda word: len(word) > 5, words))
print("Long Words:", long_words)
```

```
Even Numbers: [2, 4, 6, 8]
Long Words: ['banana', 'cherry', 'elderberry']
```

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

In [2]:
```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
l=[]
for num in numbers:
    if num % 2 == 0:
        l.append(num)
print("Even Numbers:", l)
```

```
Even Numbers: [2, 4, 6, 8, 10]
```

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

In [4]:
```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
l=[]
for num in numbers:
    if num % 2 == 0:
        l.append(num)
print("largest Even Number:", max(l))
```

```
largest Even Number: 10
```

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

In [5]:
```
numbers = [1, 5, 8, 3, 12, 6, 10, 7]
l=[]
for num in numbers:
    if num % 2 != 0:
        l.append(num)
print("largest odd Number:", max(l))
```

```
largest odd Number: 7
```

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

In [31]:
```
my_list = [1, 2, 3, 4, 5]
my_list[0], my_list[-1] = my_list[-1], my_list[0]
print("Swapped List:", my_list)
```

```
Swapped List: [5, 2, 3, 4, 1]
```

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

In [32]:
```
numbers = [1, 2, 3, 4, 5]
sum_of_elements = sum(numbers)
print("Sum of Elements:", sum_of_elements)
```

```
Sum of Elements: 15
```

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

In [1]:
```
numbers = [1, 2, 3, 4, 5]
result = sum(numbers)
print("Sum of Elements :", result)
```

```
Sum of Elements : 15
```

Write a Python program of Reversing a List.

In [34]:
```
my_list = [1, 2, 3, 4, 5]
reversed_list = my_list[::-1]
print("Reversed List:", reversed_list)
```

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

Write a Python program to Merging two Dictionaries

In [51]:
```
dict1 = {'a': 1, 'b': 2}
dict2 = {'b': 3, 'c': 4}
dict1.update(dict2)
print("Merged Dictionary: ", dict1)
```

```
Merged Dictionary:  {'a': 1, 'b': 3, 'c': 4}
```

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

In [36]:
```
numbers = [1, -2, 3, -4, 5, -6]
sum_positive = lambda x: sum(num for num in x if num > 0)
sum_negative = lambda x: sum(num for num in x if num < 0)
print("Sum of Positive Numbers:", sum_positive(numbers))
print("Sum of Negative Numbers:", sum_negative(numbers))
```

```
Sum of Positive Numbers: 9
Sum of Negative Numbers: -12
```

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

In [37]:
```
numbers = [1, -2, 3, -4, 5, -6]
arrange_numbers = lambda x: sorted(x, key=lambda num: (num >= 0, num))
arranged_numbers = arrange_numbers(numbers)
print("Rearranged Numbers:", arranged_numbers)
```

```
Rearranged Numbers: [-6, -4, -2, 1, 3, 5]
```

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

In [2]:
```
numbers = [19, 26, 39, 52, 65, 13, 78]
divisible_by_nineteen_or_thirteen = lambda x: [num for num in x if num % 19 == 0 or num % 13 == 0]
result = divisible_by_nineteen_or_thirteen(numbers)
print("Numbers Divisible by Nineteen or Thirteen:", result)
```

```
Numbers Divisible by Nineteen or Thirteen: [19, 26, 39, 52, 65, 13, 78]
```

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.
```

In [8]:
```
l=[1,2,3,4]
i=0
target= int(input("The number to search for: "))
for num in l:
        if num != target:
            i+=1
        else:
            print(num,"found at index no",i)
```

```
The number to search for: 3
3 found at index no 2
```

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.

In [9]:
```
def group_elements(lst):
    grouped_dict = {}
    for elem in lst:
        if elem in grouped_dict:
            grouped_dict[elem].append(elem)
        else:
            grouped_dict[elem] = [elem]
    # Sort the dictionary by frequency in descending order
    sorted_list = sorted(grouped_dict.items(), key=lambda x: len(x[1]), reverse=True)
    # Convert the sorted list back to a dictionary
    sorted_dict = dict(sorted_list)
    return sorted_dict
test_list = [4, 6, 6, 4, 2, 2, 4, 8, 5, 8]
result_dict = group_elements(test_list)
print("Input List:", test_list)
print("Output Dictionary:", result_dict)
```

```
Input List: [4, 6, 6, 4, 2, 2, 4, 8, 5, 8]
Output Dictionary: {4: [4, 4, 4], 6: [6, 6], 2: [2, 2], 8: [8, 8], 5: [5]}
```

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.

In [4]:
```
def rotate_matrix_90_degrees(matrix):
    N = len(matrix)
    #print(N)
    # Transpose the matrix
    for i in range(N):
        for j in range(i, N):
            #print(i,j)
            matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]
    print(matrix)
    # Reverse each row
    for i in range(N):
        matrix[i].reverse()
# Get the size of the matrix from the user
N = int(input("Enter the size of the matrix (N): "))
# Get the matrix input from the user
matrix = []
print("Enter the entries rowwise:")

# For user input
for i in range(N): # A for loop for row entries
    a =[]
    for j in range(N): # A for loop for column entries
        a.append(int(input()))
    matrix.append(a)

# For printing the matrix
for i in range(N):
    for j in range(N):
        print(matrix[i][j], end = " ")
    print()
rotate_matrix_90_degrees(matrix)
# Print the rotated matrix
print("Rotated Matrix:")
for row in matrix:
    for i in row:
        print(i,end=" ")
    print()
```

```
Enter the size of the matrix (N): 3
Enter the entries rowwise:
1
2
3
4
5
6
7
8
9
1 2 3
4 5 6
7 8 9
[[1, 4, 7], [2, 5, 8], [3, 6, 9]]
Rotated Matrix:
7 4 1
8 5 2
9 6 3
```

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
```

In [10]:
```
a= ["lessonplan", "lesson","ees", "length"]
size = len(a) # if size is 0, return empty string
if (size == 0):
        print("nothing")
if (size == 1):
        print(a[0])
a.sort()# sort the list of strings
    # find the minimum length from first and last string
end = min(len(a[0]), len(a[size - 1]))
 # find the common prefix between the first and last string
i = 0
while (i < end and a[0][i] == a[size - 1][i]):
        i += 1
pre = a[0][0: i]
if pre:
    print(pre)
else:
    print(-1)
```

```
-1
```

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 characters at indices 0, 4, 8 and 12. The second group is etse, the characters at indices 1, 5, 9 and 13. The third group is c s, the characters at indices 2, 6 and 10. The fourth group is rma, the characters at indices 3, 7 and 11. The encrypted message is seegetsec srma.

In [11]:
```
s=input('what is your message to encrypt: ')
s1=""
key=int(input("enter key: "))
for i in range(key):
    s1+=s[i::key]
print("encrypted_message:",s1)
```

```
what is your message to encrypt: secret message
enter key: 3
encrypted_message: sr sgeemsectea
what is your message to decrypt: sr sgeemsectea
enter key: 3
decrypted_message:  secret message
```

(A). Write a program that asks the user for a string, and an integer determining whether to break things up by threes, fours, or whatever user inputs. Encrypt the string using above method.

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

(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 It 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!

(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']}

Enter Decrypted Message: It is a good day to learn python Output: {'i': ['It', 'is'], 'a': ['a'], 'g': ['good'], 'd': ['day'], 't': ['to'], 'l': ['learn'], 'p': ['python']}

In [13]:
```
# Task A
s=input('what is your message to encrypt: ')
s1=""
key=int(input("enter key: "))
for i in range(key):
    s1+=s[i::key]
print("encrypted_message:",s1)
```

```
what is your message to encrypt: This is python, a programming language
enter key: 4
encrypted_message: T poaomgnghiyn gm geist,prilus h ranaa
```

In [9]:
```
# Task B
string=input('what is your message to decrypt: ')
key=int(input("enter key: "))
length=len(string)#to get the length of the input
#print(length)
part=length//key #half of the input
print(part)
extra=length%key #extra characters after dividing string in equal parts
print(extra)
#print((part+1)*extra)
part1=string[:(part+1)*extra]
#print(p1)
part2=string[(part+1)*extra:]
#print(part1,part2)
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)
print("decrypted_message: ",msg)
```

```
what is your message to decrypt: T poaomgnghiyn gm geist,prilus h ranaa
enter key: 4
9
2
decrypted_message:  This is python, a programming language
```

In [15]:
```
# Task C
decrypted_message="This is python, a programming language"
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)
```

```
{'t': ['This'], 'i': ['is'], 'p': ['python,', 'programming'], 'a': ['a'], 'l': ['language']}
```

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: 1
```

In [12]:
```
#To print all possible combinations without duplications
my_list = [1,1,3]
l=[]
for i in range(0,3):
   for j in range(0,3):
      for k in range(0,3):
         if(i!=j and j!=k and k!=i):
              l.append(list((my_list[i],my_list[j],my_list[k])))
print(l)
final_list = []
for num in l:
    if num not in final_list:
        final_list.append(num)
print("To print all possible combinations without duplications")
print(final_list)
for i in final_list:
    for j in range(len(i)):
        print(i[j],end=" ")
    print()

unique_items = set(my_list)
print("Count unique values inside a list",len(unique_items))

unique_items = set(my_list)
product = 1
for num in unique_items:
        product *= num
print("List product excluding duplicates",product)

unique_items = set(my_list)
print("Sum of list's elements excluding duplicates: ",sum(my_list))
```

```
[[1, 1, 3], [1, 3, 1], [1, 1, 3], [1, 3, 1], [3, 1, 1], [3, 1, 1]]
To print all possible combinations without duplications
[[1, 1, 3], [1, 3, 1], [3, 1, 1]]
1 1 3
1 3 1
3 1 1
Count unique values inside a list 2
List product excluding duplicates 3
Sum of list's elements excluding duplicates:  5
```

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.  Fo
r 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

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
```

In [50]:
```
from functools import reduce
# PART - A
l=[1,2,3,4]
print("tuple: ",tuple(l))
# PART - B
multiply_by_10 = list(map(lambda x: x * 10,l))
print("multiply_by_10: ",multiply_by_10)
# PART - C
print("multiplication of all elements: ",reduce(lambda x, y: x * y, l, 1))

# PART - D
print()
def countchar():
    input_string = 'hello'
    char_to_count="l"
    count = 0
    for char in input_string:
        if char == char_to_count:
            count += 1
    print("count of char: ",count)
countchar()
print()
def findchar():
    input_string = 'hello'
    char_to_find="e"
    li=[]
    for char in range(len(input_string)):
        if input_string[char] == char_to_find:
            print("char index no.: ",char)
            li.append(char)
    if len(li)==0:
        print(-1,"no char found")
findchar()
```

```
tuple:  (1, 2, 3, 4)
multiply_by_10:  [10, 20, 30, 40]
multiplication of all elements:  24

count of char:  2

char index no.:  1
```

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
```

In [45]:
```python
# Given list of numbers
m = [2, 4, -6, -9, 11, -12, 14, -5, 17]
# Lambda function to filter positive numbers
positive_numbers = lambda x: x > 0
# Lambda function to filter negative numbers
negative_numbers = lambda x: x < 0
# Calculate the sum of positive numbers using lambda function
sum_positive = sum(filter(positive_numbers, m))
# Calculate the sum of negative numbers using lambda function
sum_negative = sum(filter(negative_numbers, m))
# Output the results
print(f"Input: {m}")
print(f"Sum of the positive numbers: {sum_positive}")
print(f"Sum of the negative numbers: {sum_negative}")
```

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

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.

In [28]:
```
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_check("Ram1234asd")
```

```
valid
```

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}
```

In [29]:
```
def word_count(str):
    counts = dict()
    words = str.split()
    for word in words:
        if word in counts:
            counts[word] += 1
        else:
            counts[word] = 1
    return counts
print( word_count('the quick brown fox jumps over the lazy dog.'))
```

```
{'the': 2, 'quick': 1, 'brown': 1, 'fox': 1, 'jumps': 1, 'over': 1, 'lazy': 1, 'dog.': 1}
```

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
```

In [30]:
```
l, u, p, d = 0, 0, 0, 0
s = "S@m@6a1tue$"
print("pw=",s)
if (len(s) >= 8):
    for i in s:
 # counting lowercase alphabets
       if (i.islower()):
           l+=1
 # counting uppercase alphabets
       if (i.isupper()):
           u+=1
 # counting digits
       if (i.isdigit()):
           d+=1
 # counting the mentioned special characters
       if(i=='@'or i=='$' or i=='_'):
           p+=1
if (l>=1 and u>=1 and p>=1 and d>=1 and l+p+u+d==len(s)):
       print("Valid Password")
else:
       print("Invalid 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))
```

```
pw= S@m@6a1tue$
Valid Password
num= 61
sixty-one
```

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

In [34]:
```
strings=['abc', 'xyz', 'aba', '2112', '123451', '12345']
count = 0
for string in strings:
        if len(string) >= 3 and string[0] == string[-1]:
            count += 1
print("count=",count)
```

```
count= 3
```

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]
```

In [40]:
```
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
```

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:

In [44]:
```
sentence = "Don’t wait for your feelings to change to take the action."
words = sentence.split()
result_dict = {}
for word in words:   # Use the first character of the word as the key
        key = word[0]     # Check if the key is already present in the dictionary
        if key in result_dict:
            result_dict[key].append(word)
        else:
            result_dict[key] = [word]
print(result_dict)
```

```
{'D': ['Don’t'], 'w': ['wait'], 'f': ['for', 'feelings'], 'y': ['your'], 't': ['to', 'to', 'take', 'the'], 'c': ['cha
nge'], 'a': ['action.']}
```

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.

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'}
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.

```
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

In [41]:
```
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
   "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', 'studen
t4': '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
```