In [1]:

```
#Qno.1
def rotate(l,n):
    for i in range(1,n+1):
        j = len(l)-1
        while j>0:
            temp = l[j]
            l[j] = l[j-1]
            l[j-1] = temp
            j = j-1
        print(i,' rotation ' , l)
    return
l = [1,2,3,4,5]
rotate(l,4)
```

```
1  rotation  [5, 1, 2, 3, 4]
2  rotation  [4, 5, 1, 2, 3]
3  rotation  [3, 4, 5, 1, 2]
4  rotation  [2, 3, 4, 5, 1]
```

In [6]:

```
#Qno.2
def squareRoot(n, l) :
    x = n
    count = 0
    while (1) :
        count += 1
        root = 0.5 * (x + (n / x))
        if (abs(root - x) < l) :
            break
        x = root

    return root
if __name__ == "__main__" :
    n = 4
    l = 0.000001

    print(squareRoot(n, l))
```

```
2.000000000000002
```

In [7]:

```
#Qno.3
def power(n,e):
    res=1
    for i in range(e):
        res *= n
    return res
x = int(input("Enter a number "))
y = int(input("Enter power "))
print(power(x,y))
```

```
Enter a number 12
Enter power 4
20736
```

In [8]:

```
#Qno.4
def linearSearch(l,n,x):
    for i in range(0,n):
        if(x==l[i]):
            return i
    return -1
l = [1,2,3,4,89,2,34,12,45]
y = linearSearch(l,9,34)
if(y==-1):
    print("The element in not in the list ")
else:
    print("The element is found at index ",y)
```

```
The element is found at index  6
```

In [9]:

```
#Qno.5
def binarySearch(l,n,x):
    low = 0
    high = n-1
    while low<high:
        mid = (low+high)//2
        if(l[mid]==x):
            return mid
        elif(l[mid]<x):
            low = mid + 1
        else:
            high = mid - 1
    return -1
l = [1,2,3,4,89,2,34,12,45]
l.sort()
y = binarySearch(l,9,34)
if(y==-1):
    print("The element in not in the list ")
else:
    print("The element is found at index ",y)
```

```
The element is found at index  6
```

In [10]:

```
#Qno.6
def selection(l,n,i):
    min = i
    for j in range(i,n):
        if l[min]>l[j]:
            min = j
    return min
l = [12,8,2,7,9,3,4,8,5]
for i in range (0,9):
    x = selection(l,9,i)
    l[i],l[x] = l[x],l[i]
print (l)
```

```
[2, 3, 4, 5, 7, 8, 8, 9, 12]
```

In [17]:

```
#Qno.7
def merge(ls,st,m,end):
    i=st
    j=m+1
    temp=[]
    while i<=m and j<=end:
        if ls[i]<ls[j]:
            temp.append(ls[i])
            i+=1
        else:
            temp.append(ls[j])
            j+=1
    while i<=m:
        temp.append(ls[i])
        i+=1
    while j<=end:
        temp.append(ls[j])
        j+=1
    k=0
    for i in range(st,end+1):
        ls[i]=temp[k]
        k+=1

def merge_sort(ls,st,end):
    if st<end:
        m=st+(end-st)//2
        merge_sort(ls,st,m)
        merge_sort(ls,m+1,end)
        merge(ls,st,m,end)

ls=[23,54,65,45,3,45,4,345,45,2]
merge_sort(ls,0,len(ls)-1)
print(ls)
```

```
[2, 3, 4, 23, 45, 45, 45, 54, 65, 345]
```

In [21]:

```
#Qno.8
def isPrime(n):
    if(n==1 or n==0):
        return False
    for i in range(2,n//2+1):
        if(n%i==0):
            return False
    return True

N = int (input("Enter the value of N "))
l = []
for i in range(1,N+1):
    if(isPrime(i)):
        l.append(i)
print (l)
```

```
Enter the value of N 13
[2, 3, 5, 7, 11, 13]
```

In [22]:

```
#Qno.9
X = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]
Y = [[5,8,1,2],
    [6,7,3,0],
    [4,5,9,1]]
result = [[0,0,0,0],
         [0,0,0,0],
         [0,0,0,0]]
for i in range(len(X)):
    for j in range(len(Y[0])):
        for k in range(len(Y)):
            result[i][j] += X[i][k] * Y[k][j]

for r in result:
    print(r)
```

```
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 112, 23]
```

In [23]:

```
#Qno.10
def linearSearch(l,n,mx):
    for i in range(0,n):
        if(mx<l[i]):
            mx = l[i]
    return mx
l = [1,2,3,4,89,2,34,12,45]
y = linearSearch(l,9,l[0])
print("The maximum element is ",y)
```

```
The maximum element is  89
```

In [27]:

```
#Qno.11
def caesar_cipher(message, key):
    encrypted_message = ""
    for character in message:
        if character.isalpha():
            shifted_character = chr((ord(character) - 65 + key) % 26 + 65)
            encrypted_message += shifted_character
        else:
            encrypted_message += character
    return encrypted_message
print(caesar_cipher("Bibek",5))
```

```
GTMPV
```

In [24]:

```
#Qno.12
str = input("Enter a string ")
c = input("Enter a character ")
l = len(str)
flag = False
for i in range(0,l):
    if(str[i]==c):
        flag = True
if(flag):
    print(c," is present in ",str)
else:
    print("Not present")
```

```
Enter a string BIBEK
Enter a character B
B  is present in  BIBEK
```

In [29]:

```
#Qno.13
mult = lambda a,b:a*b
print(mult(12,13))
```

```
156
```

In [30]:

```
#Qno.14
str = "This is reversing a string"
l = len(str)-1
s = ""
while l>=0:
    s += str[l]
    l-=1
print("The string in reverse is ",s)
```

```
The string in reverse is  gnirts a gnisrever si sihT
```

In [31]:

```
#Qno.15
def count(s):
    low=0
    up=0
    dg=0
    for i in s:
        if i.isalpha():
            if i.islower():
                low+=1
            else:
                up+=1
        else:
            dg+=1
    return [low,up,dg]
print(count("Bibek roll no 20051722"))
```

```
[10, 1, 11]
```