1. Two Sum

Easy 12570 440 Favorite Share

Given an array of integers, return **indices** of the two numbers such that they add up to a specific target.

You may assume that each input would have ***exactly*** one solution, and you may not use the *same* element twice.

Example:

```
Given nums = [2, 7, 11, 15], target = 9,

Because nums[0] + nums[1] = 2 + 7 = 9,
return [0, 1].
```

```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:

        for i in range(len(nums)-1):
            for j in range(i+1, len(nums)):
                if nums[i] + nums[j] == target:
                    return([i, j])
```

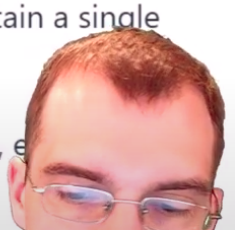
```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:

        seen = {}

        for i, num in enumerate(nums):
            if target - num in seen:
                return([seen[target - num], i])
            elif num not in seen:
                seen[num] = i
```

2. Add Two Numbers

Medium 8757 2209 Add to List Share

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order** and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.

You may assume the two numbers do not contain any leading zero, e
number 0 itself.



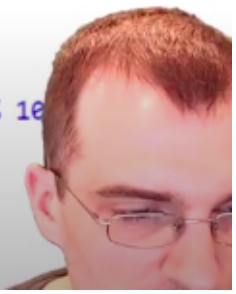
Example:

```
Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)
Output: 7 -> 0 -> 8
Explanation: 342 + 465 = 807.
```

```
class Solution:
    def addTwoNumbers(self, l1: ListNode, l2: ListNode) -> ListNode:

        added = ListNode(val = (l1.val + l2.val) % 10)
        carry_over = (l1.val + l2.val) // 10
        current_node = added

        while(l1.next and l2.next):

            l1 = l1.next
            l2 = l2.next

            current_node.next = ListNode(val = (carry_over + l1.val + l2.val) % 10)
            carry_over = (carry_over + l1.val + l2.val) // 10
            current_node = current_node.next
```

Processing math: 100%

```
while(l1.next and l2.next):

    l1 = l1.next
    l2 = l2.next

    current_node.next = ListNode(val = (carry_over + l1.val + l2.val) % 10)
    carry_over = (carry_over + l1.val + l2.val) // 10
    current_node = current_node.next

while(l1.next):
    l1 = l1.next
    current_node.next = ListNode(val = (carry_over + l1.val) % 10)
    carry_over = (carry_over + l1.val) // 10
    current_node = current_node.next

while(l2.next):
    l2 = l2.next
    current_node.next = ListNode(val = (carry_over + l2.val) % 10
    carry_over = (carry_over + l2.val) // 10
    current_node = current_node.next

if (carry_over) > 0:
    current_node.next = ListNode(val = 1)
```



```
return(added)
```

Palindrome

What is a Palindrome? A palindrome is nothing but any number or a string which remains unaltered when reversed.

Example: 12321 Output: Yes, a Palindrome number

Example: RACECAR Output: Yes, a Palindrome string

In [2]:
```
n=int(input("Enter the number: "))
rev=0
temp=n
while n>0:
    rev=(rev*10 + n%10)
    n=n//10
if temp==rev:
    print(True)
else:
    print(False)
```

```
Enter the number: 123
False
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Ques 1: Write a program to find sum of digits of a given number.

```
i=int(input("Enter Number to find sum of digits"))
sum=0
While (i>0):
        sum=sum+i%10
        i=i//10
Print("Sum of digits=",sum)
```

Ques 2: Write a program to find sum of square of digits of a given number.

```
i=int(input("Enter Number to find sum of square of digits"))
sum=0
while (i>0):
    sum=sum+(i%10)*(i%10)
    i=i//10
print("Sum of Square of each digits=",sum)
```

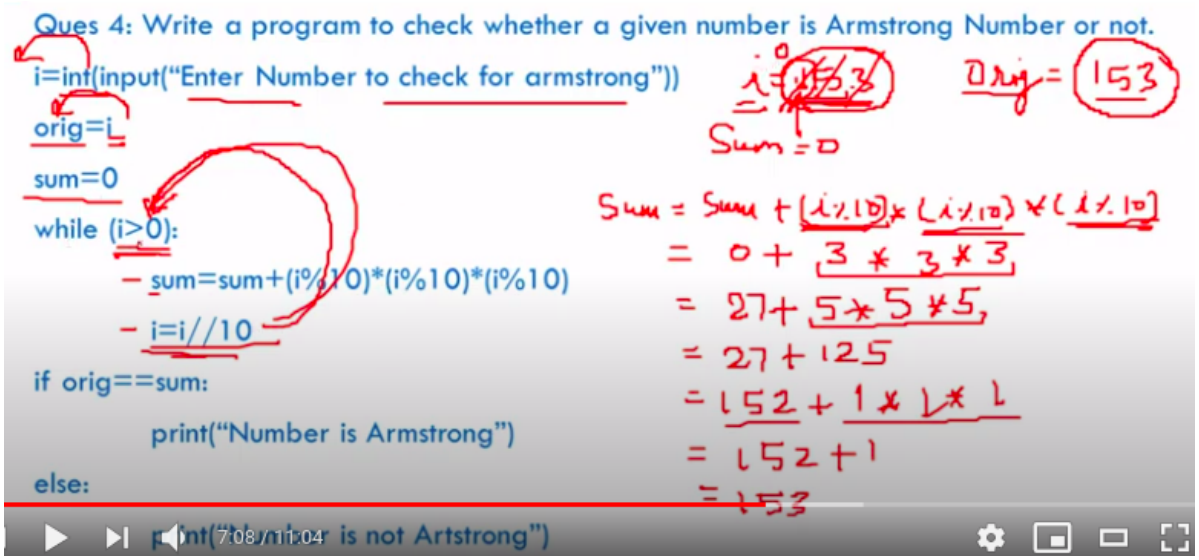
Ques 3: Write a program to find sum of cube of digits of a given number.

```
i=int(input("Enter Number to find sum of cube of digits"))
sum=0
while (i>0):
        sum=sum+(i%10)*(i%10)*(i%10)
        i=i//10
print("Sum of cube of each digits=",sum)
```

Processing math: 100%

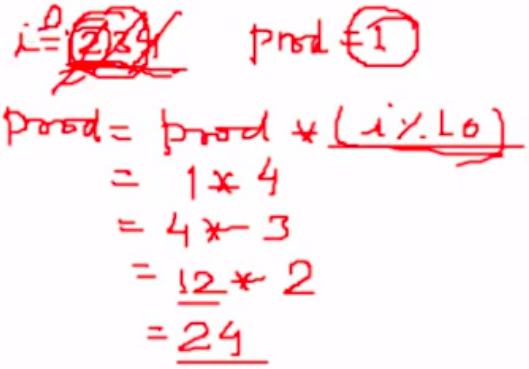
Ques 4: Write a program to check whether a given number is Armstrong Number or not.

```
i=int(input("Enter Number to check for armstrong"))
orig=i
sum=0
while (i>0):
    sum=sum+(i%10)*(i%10)*(i%10)
    i=i//10
if orig==sum:
    print("Number is Armstrong")
else:
    print("Number is not Artstrong")
```



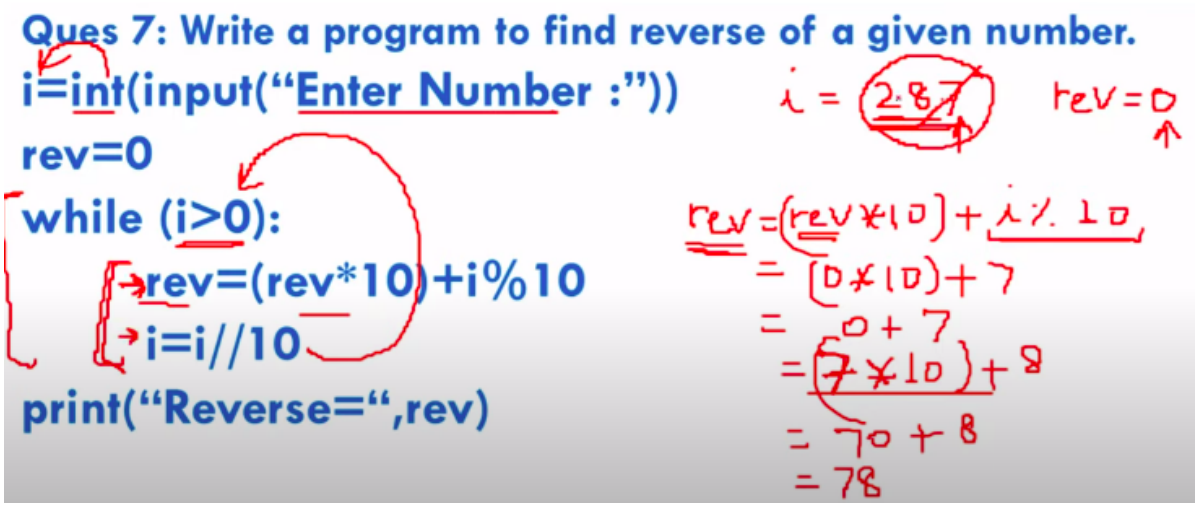
Ques 5: Write a program to find product of digits of a given number.

```
i=int(input("Enter Number :"))
prod=1
while (i>0):
    prod=prod*(i%10)
    i=i//10
print("Product of digits=",prod)
```



Ques 7: Write a program to find reverse of a given number.

```
i=int(input("Enter Number :"))
rev=0
while (i>0):
    rev=(rev*10)+i%10
    i=i//10
print("Reverse=",rev)
```



Processing math: 100%

Ques 8: Write a program to check whether a number is palindrome or not?

```
i=int(input("Enter Number :"))
rev=0
x=i
while (i>0):
        rev=(rev*10)+i%10
        i=i//10
If(x==rev):
        print("Palindrome Number")
else:
        print("Not Palindrome");
```

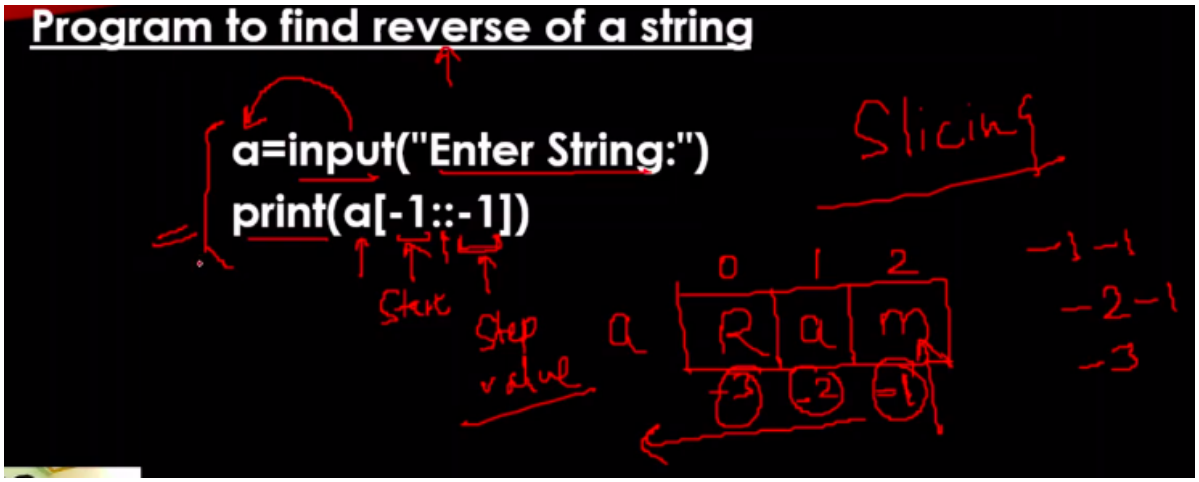
Ques 8: Write a program to check whether a number is prime or not?

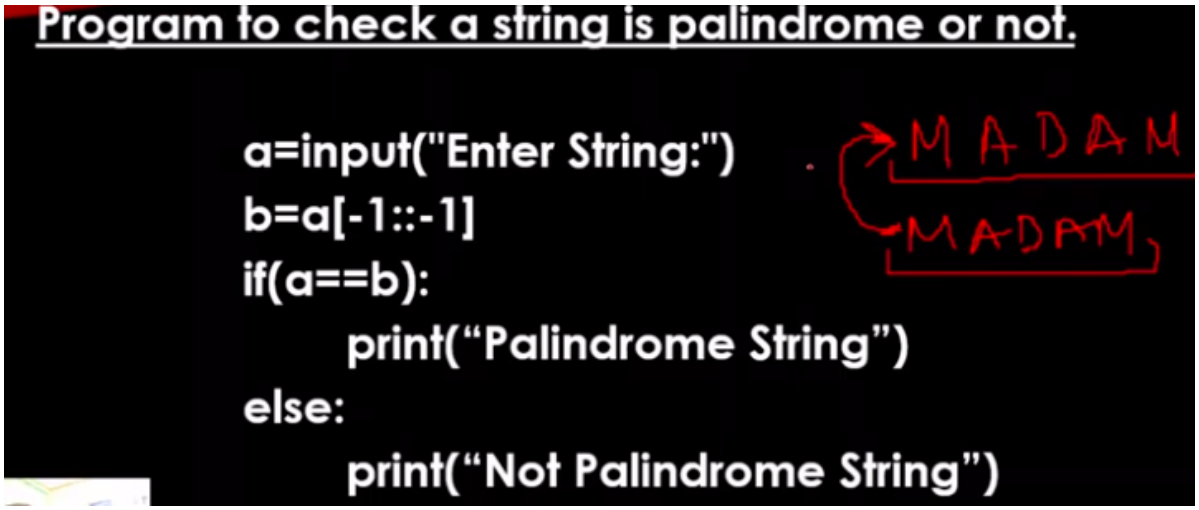
```
n=int(input("Enter Number :"))
count=0
i=1
while (i<=n):
        if (n%i==0):
                count=cont+1
        i=i+1
If(count==2):
        print("Prime Number")
else:
        print("Composite Number");
```

Ques 8: Write a program to print factorial of a given number?

```
i=int(input("Enter Number :"))
fac=1
while (i>0):
        fac=fac*i
        i=i-1
print("Factorial =",fac)
```

Processing math: 100%





In [21]:
```
a= input("Enter String: ")
# x=print(a)
y=(a[-1::-1])
if a==y:
    print(True)
else:
    print(False)
```

```
Enter String: nan
True
```

Processing math: 100%

2. Program to count total number of odd and even numbers in the List.

```
a=[]
size=int(input("How Many Elements You Want to Enter?"))
for i in range(size):
    val=int(input("Enter Number:"))
    a.append(val)
even=0
odd=0
for i in range(size):
    if(a[i]%2==0):
        even=even+1
    else:
        odd=odd+1
int("Total Even=", even, "Total Odd=",odd)
```

4. Program to search a number in the List. #Sequential search

```
a=[]
size=int(input("Enter Size of the List:"))
for i in range(size):
    val=int(input("Enter Number:"))
    a.append(val)
key=int(input("Enter Number to Search:"))
flag=0
for i in range(size):
    if(a[i]==key):
        flag=1
        pos=i+1
        break
if(flag==1):
    print("Element found at:",pos,"position.")
else:
    print("Element not found.")
```

5. Program to count frequency of a given number.

```
a=[]
size=int(input("Enter Size of the List:"))
for i in range(size):
    val=int(input("Enter Number:"))
    a.append(val)
key=int(input("Enter Number to Find Frequency:"))
count=0
for i in range(size):
    if(a[i]==key):
        count=count+1
print("Frequency=",count)
```

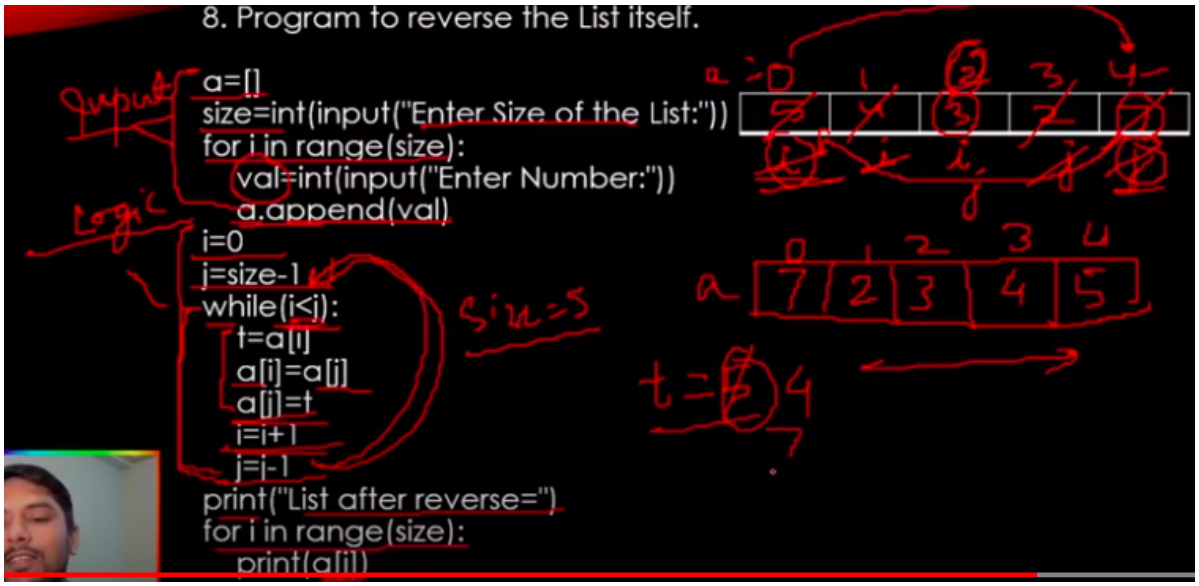
Processing math: 100%

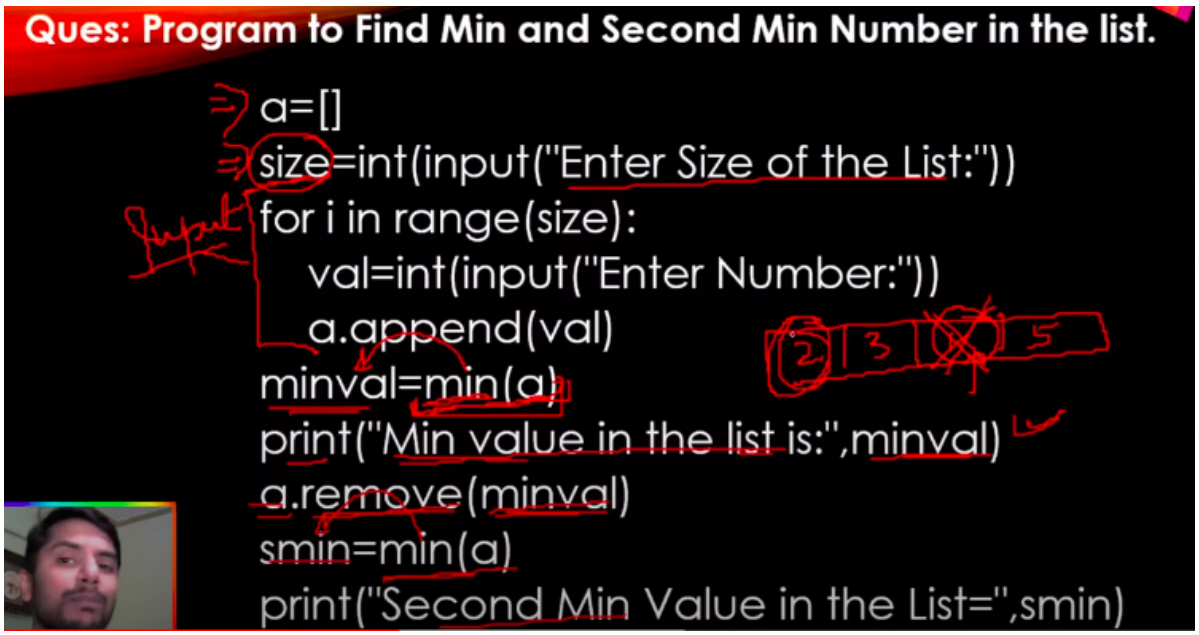
6. Program to find max number of the List.

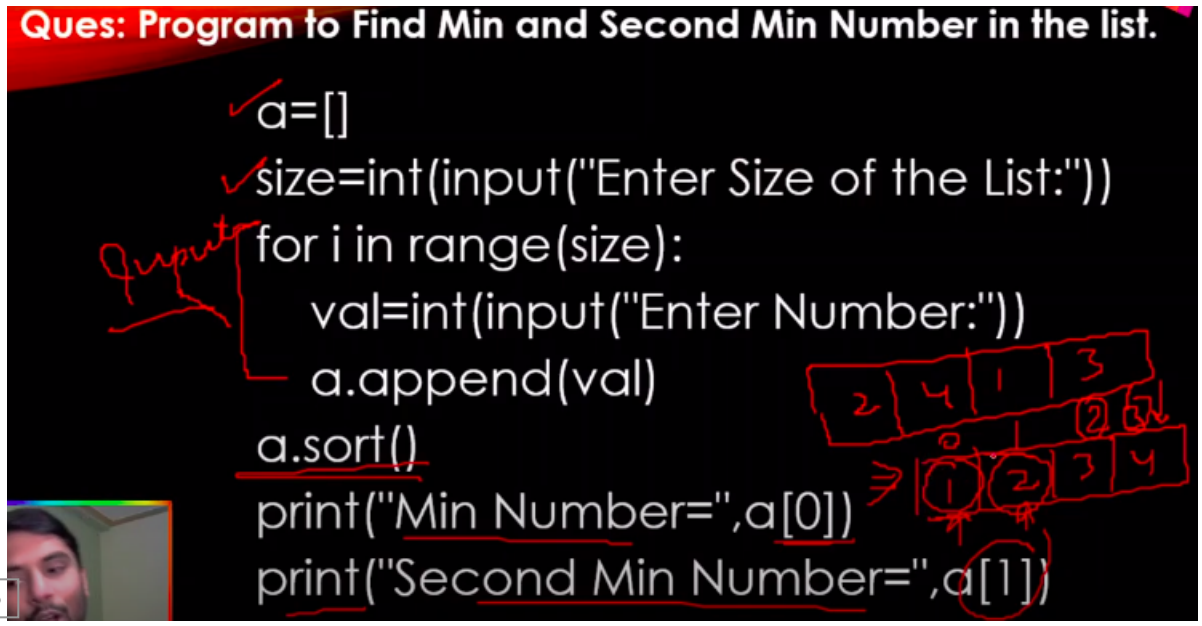
```
a=[]
size=int(input("Enter Size of the List:"))
for i in range(size):
    val=int(input("Enter Number:"))
    a.append(val)
max=a[0]
for i in range(size):
    if(a[i]>max):
        max=a[i]
print("Max Number=",max)
```

7. Program to find min number of the List.

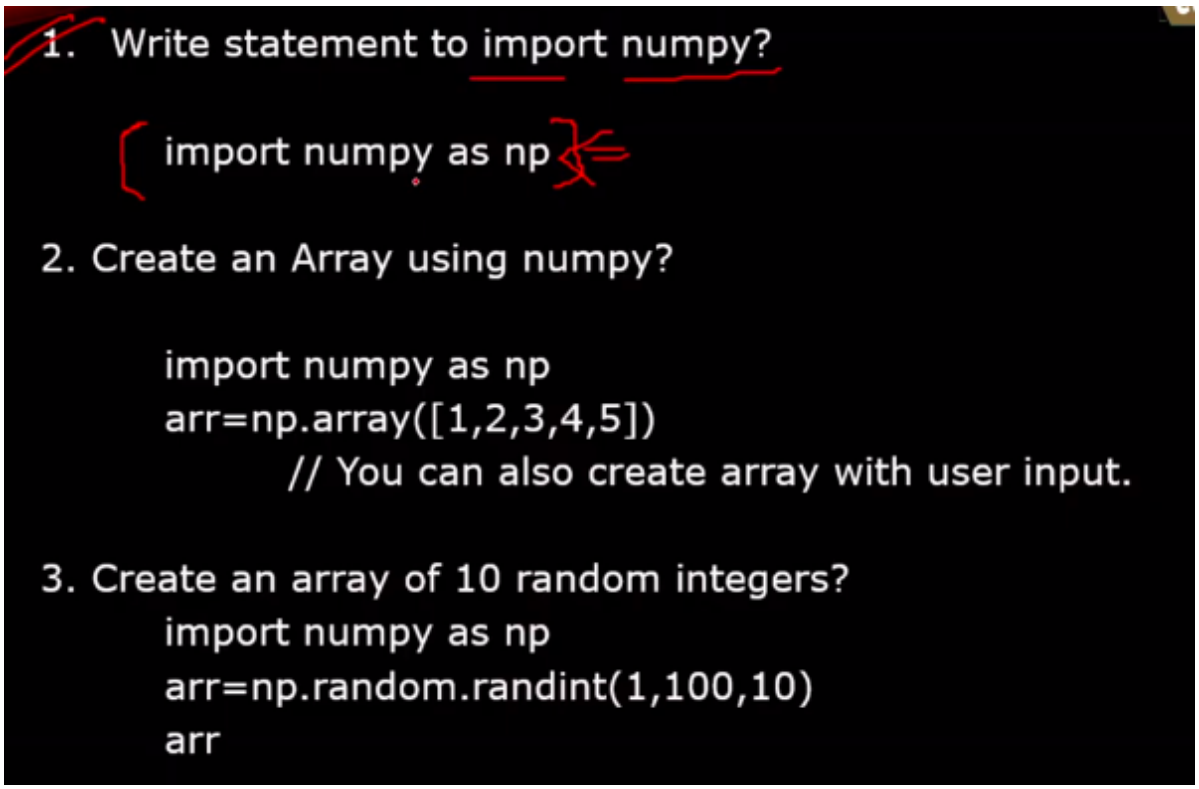
```
a=[]
size=int(input("Enter Size of the List:"))
for i in range(size):
    val=int(input("Enter Number:"))
    a.append(val)
min=a[0]
for i in range(size):
    if(a[i]<min):
        min=a[i]
print("Min Number=",min)
```

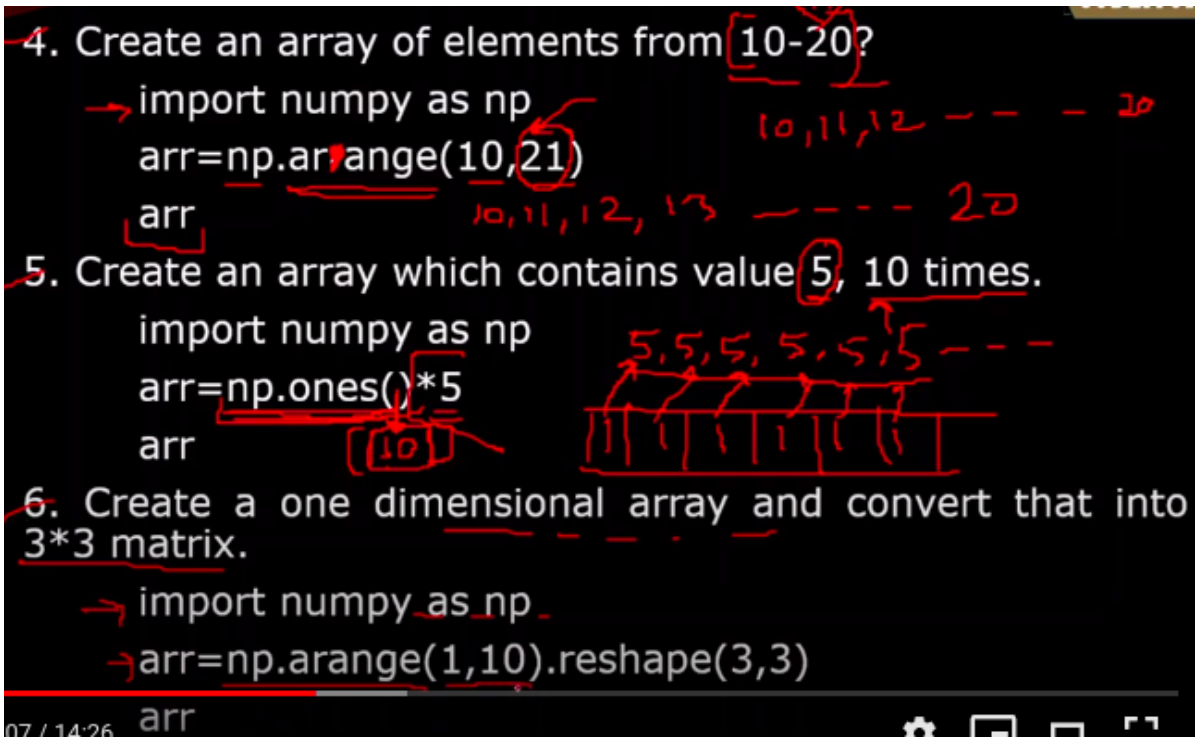






NumPy





Processing math: 100%

7. Create an array and then convert all the even numbers with 0.

```
import numpy as np
arr=np.arange(1,11)
arr[arr%2==0]=0
arr
```

8. Create a 2D array of size 3*3 but all the elements should be between 0 and 1.

```
import numpy as np
arr=np.random.rand(9).reshape(3,3)
arr
```

9. Perform the following slicing:

Input

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

output

6	7
10	11
14	15

```
import numpy as np
arr1=np.arange(1,17).reshape(4,4)
arr2=arr[1:,1:3]
arr2
```

10. Concatenate 2D array horizontally and vertically.

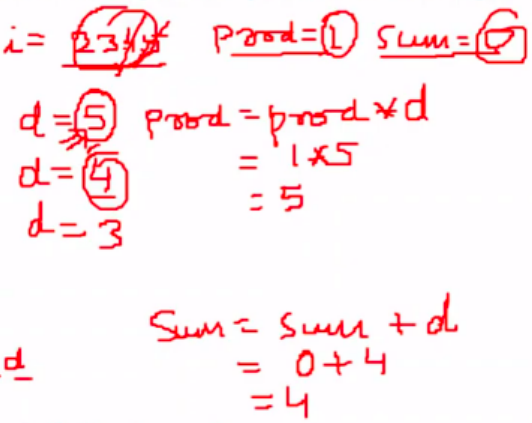
```
import numpy as np
arr1=np.arange(1,10).reshape(3,3)
arr2=np.arange(10,19).reshape(3,3)
np.hstack((arr1,arr2))
```

```
import numpy as np
arr1=np.arange(1,10).reshape(3,3)
arr2=np.arange(10,19).reshape(3,3)
np.vstack((arr1,arr2))
```

Processing math: 100%

Ques 6: Write a program to find sum of even digits and product of odd digits of a given number.

```
i=int(input("Enter Number :"))
Sum=0
Prod=1
while (i>0):
    d=i%10
    if d%2==0:
        sum=sum+d
    else:
        prod=prod*d
    i=i//10
print("Sum of digits=",sum,"Product of digits=",prod)
```



HackerRank

Task

Given a string, S, of length N that is indexed from 0 to $N-1$, print its even-indexed and odd-indexed characters as 2 space-separated strings on a single line (see the Sample below for more detail).

Note: 0 is considered to be an even index.

Input Format

The first line contains an integer, T (the number of test cases).

Each line i of the T subsequent lines contain a String, S.

Constraints

- $1 \leq T \leq 10$
- $2 \leq \text{length of } S \leq 10000$

Output Format

For each String S_j (where $0 \leq j \leq T-1$), print S_j's even-indexed characters, followed by a space, followed by S_j's odd-indexed characters.

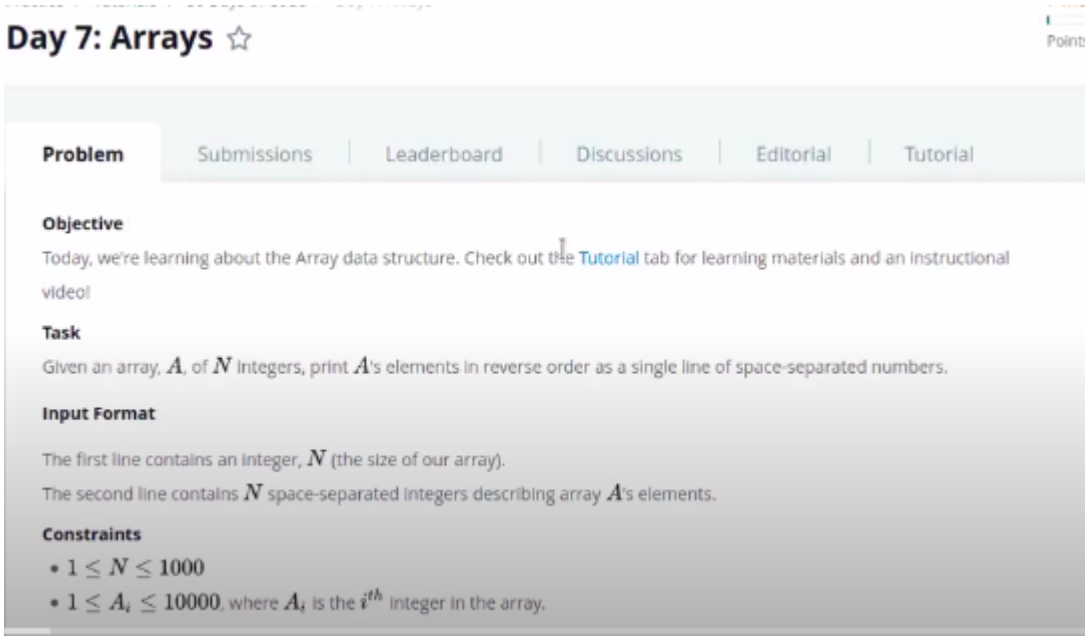
Sample Input

```
2
Hacker
Rank
```

Sample Output

```
Hce akr
Rn ak
```

```
for z in range(int(input())):
    strlist = list(input())
    for i in range(len(strlist)):
        if i%2 == 0:
            print(strlist[i],end="")
    print(" ",end="")
    for i in range(len(strlist)):
        if i%2 != 0:
            print(strlist[i],end="")
    print()
```



Day 7: Arrays ☆

Problem | Submissions | Leaderboard | Discussions | Editorial | Tutorial

Objective

Today, we're learning about the Array data structure. Check out the Tutorial tab for learning materials and an instructional video!

Task

Given an array, A, of N integers, print A's elements in reverse order as a single line of space-separated numbers.

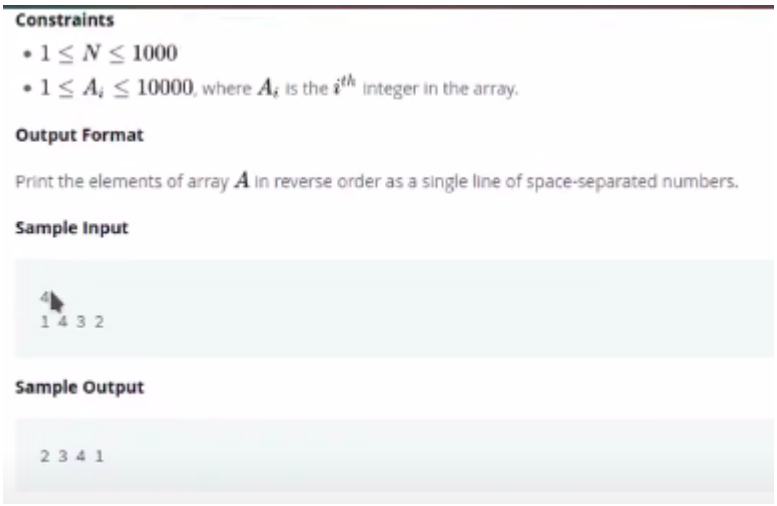
Input Format

The first line contains an integer, N (the size of our array).

The second line contains N space-separated integers describing array A's elements.

Constraints

- $1 \leq N \leq 1000$
- $1 \leq A_i \leq 10000$, where A_i is the i^{th} integer in the array.



Constraints

- $1 \leq N \leq 1000$
- $1 \leq A_i \leq 10000$, where A_i is the i^{th} integer in the array.

Output Format

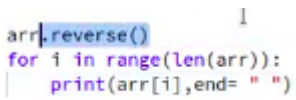
Print the elements of array A in reverse order as a single line of space-separated numbers.

Sample Input

```
4
1 4 3 2
```

Sample Output

```
2 3 4 1
```



```
arr.reverse()
for i in range(len(arr)):
    print(arr[i],end= " ")
```

In [30]:
```
n=int(input("Enter No.: "))
rev=0
while (n>0):
    rev=(rev*10)+ n%10
    n=n//10
print("Reverse",rev)
```

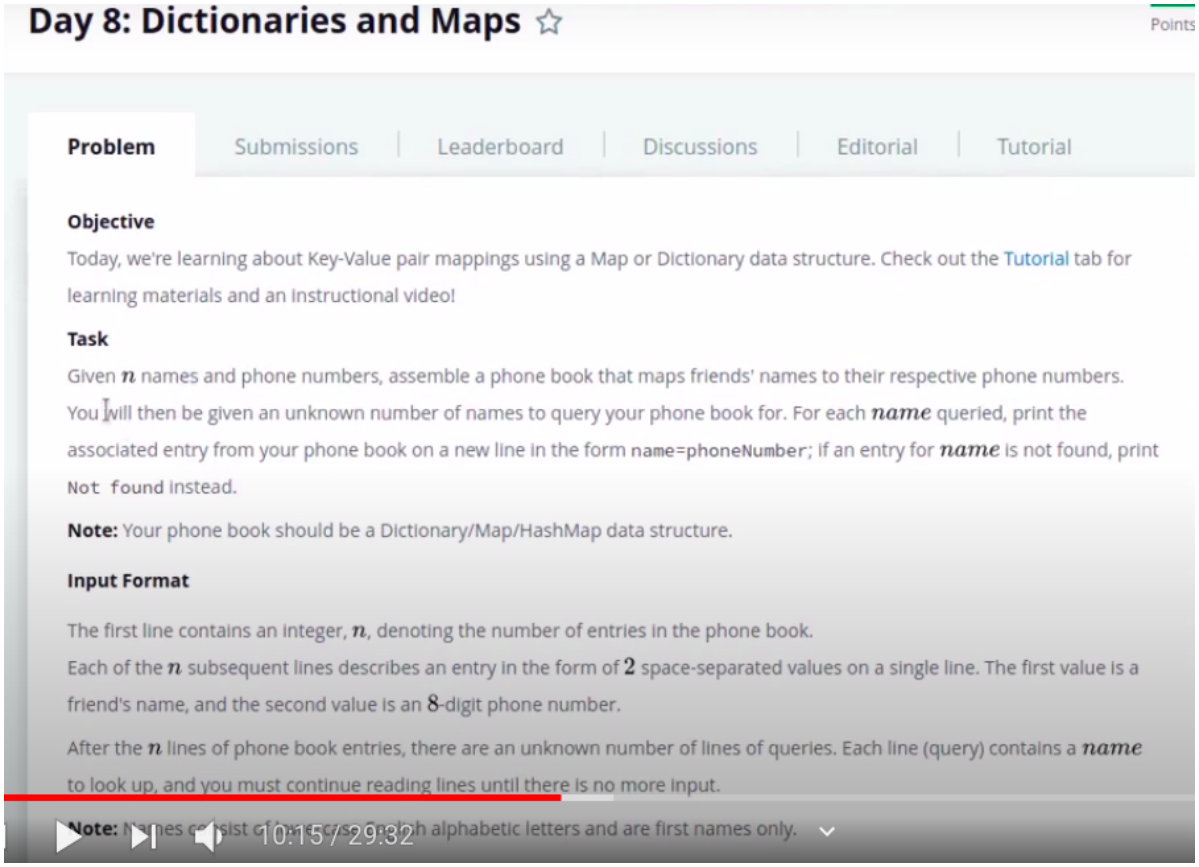
```
Enter No.: 123
Reverse 321
```

Processing math: 100%

In [37]:
```
n=int(input("Enter No.: "))
x=str(n)[::-1]
int(x)
```

```
Enter No.: 123
```

Out[37]: 321



Note: Names consist of lowercase English alphabetic letters and are first names only.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq queries \leq 10^5$

Output Format

On a new line for each query, print `Not found` if the name has no corresponding entry in the phone book; otherwise, print the full $name$ and $phoneNumber$ in the format `name=phoneNumber`.

Sample Input

```
3
sam 99912222
tom 11122222
harry 12299933
sam
edward
harry
```

Sample Output

```
sam=99912222
Not found
harry=12299933
```

Explanation

We add the following $n = 3$ (Key,Value) pairs to our map so it looks like this:

$$phoneBook = \{(sam, 99912222), (tom, 11122222), (harry, 12299933)\}$$

We then process each query and print `key=value` if the queried key is found in the map; otherwise, we print `Not found`.

Query 0: `sam`

Sam is one of the keys in our dictionary, so we print `sam=99912222`.

Query 1: `edward`

Edward is not one of the keys in our dictionary, so we print `Not found`.

Query 2: `harry`

Harry is one of the keys in our dictionary, so we print `harry=12299933`.

```
# Enter your code here. Read input from STDIN. Print

n = int(input())
phoneBook = {}
for i in range(n):
    l1 = list(map(str, input().split()))
    key = l1[0]
    value = l1[1]
    phoneBook[key] = value
```

```
while True:
    try:
        name = input()
        if name in phoneBook:
            print(name + "=" + phoneBook[name])
        else:
            print("Not found")
    except:
        break
```

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2