

## **WEEK – 1 ( Input ,Print and Numbers )**

### **Statement :**

Write a program that takes three numbers and prints their sum. Every number is given on a separate line.

### **Objective :**

To get familiar with basics of python and using it for printing sum of given 3 numbers.

#### **Code :**

```
a=int(input())  
b=int(input())  
c=int(input())  
print(a+b+c)
```

#### **Input :**

2  
3  
6

#### **Output :**

11

### **Inference :**

3 numbers are given through user input and sum of 3 numbers is printed.

**Statement :**

Write a program that greets the person by printing the word "Hi" and the name of the person. See the examples below.

**Objective :**

To understand basics of python and Print a name with prefix "Hi".

**Code :**

```
name=input()  
print("Hi "+name)
```

**Input :**

John

**Output :**

Hi John

**Inference :**

When given a name, the word "Hi" is displayed along with the word.

**Statement :**

Write a program that takes a number and print its square.

**Objective :**

To get familiar with basics of python and print square of a given number.

**Code :**

```
a=input()
print(int(a)*int(a))
```

**Input :**

5

**Output :**

25

**Inference :**

We multiply the number by itself ,thus the square of the number is printed.

**Statement :**

Write a program that reads the length of the base and the height of a right-angled triangle and prints the area. Every number is given on a separate line.

**Objective :**

To understand the basics of python and print area of a right angled triangle.

**Code :**

```
b=int(input())
h=int(input())
area=(h*b)/2
print(area)
```

**Input :**

3  
5

**Output :**

7.5

**Inference :**

When given base and height of the triangle ,we use the formula  $(\text{base} \times \text{height}) / 2$  and thus print the area of the right angled triangle.

**Statement :**

Write a program that greets the user by printing the word "Hello", a comma, the name of the user and an exclamation mark after it. See the examples below.

**Objective :**

To get familiar with basics of python and print with prefix "Hello" and suffix "!" with given name in the middle.

**Code :**

```
name=input()
print("Hello,",name+"!")
```

**Input :**

Harry

**Output :**

Hello, Harry!

**Inference :**

We use string concatenation with "+" sign and add the "!" sign at the end.

**Statement :**

N students take K apples and distribute them among each other evenly. The remaining (the undivisible) part remains in the basket. How many apples will each single student get? How many apples will remain in the basket?

The program reads the numbers N and K. It should print the two answers for the questions above.

**Objective :**

To understand the basics of python and print how many apples each student gets and how many remain at the end.

**Code :**

```
n=int(input())
k=int(input())
print(k//n)
print(k%n)
```

**Input :**

6  
50

**Output :**

8  
2

**Inference :**

We are given two numbers ,where we use “//” and “%” operators to print the apples shared by each student and the apples remaining.

**Statement :**

Write a program that reads an integer number and prints its previous and next numbers. See the examples below for the exact format your answers should take. There shouldn't be a space before the period.

Remember that you can convert the numbers to strings using the function `str`.

**Objective :**

To get familiar with basics of python and print the next and previous of given number.

**Code :**

```
n=int(input())
print("The next number for the number "+str(n)+" is "+str(n+1)+".")
print("The previous number for the number "+str(n)+" is "+str(n-1)+".")
```

**Input :**

179

**Output :**

The next number for the number 179 is 180.  
The previous number for the number 179 is 178.

**Inference :**

We convert the number to string as `str(n)` and use string functions to print the previous and next number as `str(n-1)` and `str(n+1)`.

**Statement :**

A timestamp is three numbers: a number of hours, minutes and seconds. Given two timestamps, calculate how many seconds is between them. The moment of the first timestamp occurred before the moment of the second timestamp.

**Objective :**

To understand the basics of python and print the difference in seconds between two timestamps.

**Code :**

```
n1=int(input())
m1=int(input())
s1=int(input())
n2=int(input())
m2=int(input())
s2=int(input())
print((n2*3600)+(m2*60)+(s2)-(n1*3600)-(m1*60)-(s1))
```

**Input :**

1  
1  
1  
2  
2  
2

**Output :**

3661

**Inference :**

We use basic calculations and convert the timestamp to seconds and use “-” operator to get the difference between the two timestamps.



**Statement :**

A school decided to replace the desks in three classrooms. Each desk sits two students. Given the number of students in each class, print the smallest possible number of desks that can be purchased.

The program should read three integers: the number of students in each of the three classes, a, b and c respectively.

In the first test there are three groups. The first group has 20 students and thus needs 10 desks. The second group has 21 students, so they can get by with no fewer than 11 desks. 11 desks is also enough for the third group of 22 students. So we need 32 desks in total.

**Objective :**

To get familiar with basics of python and print the number of desks needed for a given classroom of students.

**Code :**

```
a=int(input())
b=int(input())
c=int(input())
desks=a//2+b//2+c//2+a%2+b%2+c%2
print(desks)
```

**Input :**

20  
21  
22

**Output :**

32

**Inference** : We are given 3 groups and benches which sit 2 students. We use basic calculations to print the total number of desks needed for the school.

## **WEEK-2 (Integer and Float Numbers)**

### **Statement:**

Given an integer number, print its last digit.

### **Objective:**

To get familiar with integer and float numbers

### **Code:**

```
a=int(input())  
print(a%10)
```

### **Input:**

179

### **Output :**

9

### **Inference:**

If an integer is divided by 10 (i.e., modulus operator ).The remainder gives last digit.

**Statement:**

Given an integer. Print its tens digit.

**Objective:**

To get familiar with integer and float numbers.

**Code:**

```
a=int(input())  
a=a//10  
a=a%10  
print(a)
```

**Input:**

73

**Output:**

7

**Inference:**

We type-casted the result into string to concatenate the digits obtained.

**Statement:**

Given a three-digit number. Find the sum of its digits.

**Objective:**

To get familiar with integer and float numbers

**Code:**

```
a=int(input())
sum=0
sum=sum+(a%10)
a=a//10
sum=sum+(a%10)
a=a//10
sum=sum+(a%10)
print(sum)
```

**Input:**

179

**Output :**

17

**Inference:**

Here, when we use division with 100, 17 gives 1<sup>st</sup> digit and division followed by modulus division by 10 gives tens digit and modulus division by 10 gives remainder so if we add those it gives us sum of each individual of a 3-digit number.

**Statement:**

Given a positive real number, print its fractional part.

**Objective:**

To get familiar with integer and float numbers

```
Code: import math
a=float(input())
print(a-math.floor(a))
```

**Input:**

17.9

**Output:**

0.89999999999999986

**Inference:**

Float type gives us the decimal value also and int when declared in decimal number it is rounded off to int number. So the difference between gives us the fractional part.

**Statement:**

Given a positive real number, print its first digit to the right of the decimal point.

**Objective:**

To get familiar with integer and float numbers

```
Code:  a=float(input())  
         b=(int)(a*10)  
         print(b%10)
```

**Input:**

1.79

**Output :**

7

**Inference:**

When the decimal is multiplied and 1<sup>st</sup> use modulus by 10 then we get a decimal remainder which int is used and results to an integer.

**Statement :**

A car can cover distance of N km/day. How many days will it take to cover a route of length M km .

**Objective :**

To get familiar with integer and float numbers using different types of functions.

**Code :**

```
from math import ceil
n=int(input())
m=int(input())
print(ceil(m/n))
```

**Input :**

700  
750

**Output :**

2

**Inference :**

In the above program , ceil() function is used to return the ceil of x i.e., the smallest integer greater than equal to x.

**Statement :**

The number of given the integer N - the number of minutes that is passed since midnight - how many hours and minutes are displayed on the 24h digital clock?

The program should print two numbers: the number of hours (between 0 and 23) and the number of minutes (between 0 and 59).

For example, if N = 150, then 150 minutes have passed since midnight - i.e. now is 2:30 am. So the program should print **2 30**.

**Objective :**

To get familiar with integer and float numbers using different types of functions.

**Code :**

```
n= int(input())
hours = n//60
minutes = n%60
print(hours , minutes)
```

**Input :**

150

**Output :**

2 30

**Inference :**

In the above program , comma operator is assigned between the hours and minutes in the print statement to print the digits in the output with gap.



**Statement :**

A cupcake costs A dollars and B cents. Determine, how many dollars and cents should one pay for N cupcakes. A program gets three numbers: A, B, N. It should print two numbers: total cost in dollars and cents.

**Objective :**

To get familiar with integer and float numbers using different types of functions.

**Code :**

```
a = int(input())
b = int(input())
n = int(input())
cost = n*(100*a+b)
print(cost // 100 , cost % 100)
```

**Input :**

2  
50  
4

**Output :**

10 0

**Inference :**

Here we need to use the modulus operator as the output should be printed in dollars and cents.

**Statement :**

H hours, M minutes and S seconds are passed since the midnight ( $0 \leq H < 12$ ,  $0 \leq M < 60$ ,  $0 \leq S < 60$ ). Determine the angle (in degrees) of the hour hand on the clock face right now.

**Objective :**

To get familiar with integer and float numbers using different types of functions.

**Code :**

```
h = int(input())
m = int(input())
s = int(input())
print(h * 30 + m * 30 / 60 + s * 30 / 3600)
```

**Input :**

1  
2  
6

**Output :**

31.05

**Inference :**

Here , to calculate the angle (in degrees) of the hour hand on the clock face , we need to convert minutes and seconds into hours and add them .

**Statement :**

Hour hand turned by  $\alpha$  degrees since the midnight. Determine the angle by which minute hand turned since the start of the current hour. Input and output in this problems are floating-point numbers.

**Objective :**

To get familiar with integer and float numbers using different types of functions.

**Code :**

```
alpha = float(input())  
print(alpha % 30 * 12)
```

**Input :**

190

**Output :**

120.0

**Inference :**

If we read a floating point number , then we need to cast the string to float using float().

## **WEEK – 3 ( if, else - Loops )**

### **Statement :**

Given two integers, print the smaller value.

**Objective** : To get familiar to tell the smallest element in 2 user inputs if-else loops.

### **Code :**

```
a=int(input())
b=int(input())
if a<b:
    print(a)
else:
    print(b)
```

### **Input :**

3  
7

### **Output :**

3

### **Inference :**

“ : ” allows the program to execute the if and else conditional statements if they are true.

**Statement :**

For the given integer X print 1 if it's positive, -1 if it's negative, or 0 if it's equal to zero. Try to use the cascade if-elif-else for it.

**Objective :**

To get familiar to tell whether a number is +ve or -ve using if-else loops.

**Code :**

```
a=int(input())
if a>0:
    print('1')
elif a<0:
    print('-1')
else:
    print('0')
```

**Input :**

-143

**Output :**

-1

**Inference :**

The way of style of if-else in python is similar to c but only format is different i.e., instead of else if we give elif.

**Statement :**

Given three integers, print the smallest value.

**Objective :**

To get familiar to tell the smallest element in 3 user inputs if-else conditional statements.

**Code :**

```
a=int(input())
b=int(input())
c=int(input())
if (a>b)&(c>b):
    print(b)
if (a>c)&(b>c):
    print(c)
if (c>a)&(b>a):
    print(a)
```

**Input :**

1  
4  
3

**Output :**

1

**Inference :**

Here ' & ' is used whereas in c we use ' && ' as an AND operation. Also we can use ' and ' as a AND operator.

**Statement :**

Given three integers, determine how many of them are equal to each other. The program must print one of these numbers: 3 (if all are the same), 2 (if two of them are equal to each other and the third is different) or 0 (if all numbers are different).

**Objective :**

To get familiar with if-elif-else conditional statements.

**Code :**

```
a=int(input())
b=int(input())
c=int(input())
if (a==b)&(a==c):
    print(3)
elif (a!=b)&(a!=c)&(b!=c):
    print(0)
else:
    print(2)
```

**Input :**

1  
4  
1

**Output :**

2

**Inference :**

Here we use 'elif' instead of 'else if' and let to print the alternative conditional statement.

**Statement :**

Chess rook moves horizontally or vertically. Given two different cells of the chessboard, determine whether a rook can go from the first cell to the second in one move. The program receives the input of four numbers from 1 to 8, each specifying the column and row number, first two - for the first cell, and then the last two - for the second cell. The program should output YES if a rook can go from the first cell to the second in one move, or NO otherwise.

**Objective :**

To get familiar with rook moves in chess using if-else conditional statements.

**Code :**

```
a1=int(input())
b1=int(input())
a2=int(input())
b2=int(input())
if (a1==a2)|(b1==b2):
    print('YES')
else:
    print('NO')
```

**Input :**

4  
4  
4  
5

**Output :**

YES

**Inference :**

Here while moving horizontally the y-coordinate is constant, vice-versa. But cannot move diagonally when both coordinates are constant.



**Statement :**

Given two cells of a chessboard. If they are painted in one color, print the word YES, and if in a different color - NO. The program receives the input of four numbers from 1 to 8, each specifying the column and row number, first two - for the first cell, and then the last two - for the second cell.

**Objective :**

To get familiar with different moves in chess using if-else conditional statements.

**Code :**

```
a1=int(input())
b1=int(input())
a2=int(input())
b2=int(input())
if (a1 + b1 + a2 + b2) % 2 == 0:
    print('YES')
else:
    print('NO')
```

**Input :**

1  
1  
2  
6

**Output :**

YES

**Inference :**

If both coordinates are even the corresponding move should be even, vice-versa for (OO->OO),(EO->EO) and (OE->OE). Hence sum of all coordinates is divisible by 2.

**Statement :**

Chess king moves horizontally, vertically or diagonally to any adjacent cell. Given two different cells of the chessboard, determine whether a king can go from the first cell to the second in one move. The program receives the input of four numbers from 1 to 8, each specifying the column and row number, first two - for the first cell, and then the last two - for the second cell. The program should output YES if a king can go from the first cell to the second in one move, or NO otherwise.

**Objective :**

To get familiar with king moves in chess using if-else conditional statements.

**Code :**

```
a1=int(input())
b1=int(input())
a2=int(input())
b2=int(input())
if (-1<=(a1-a2)<=1)&(-1<=(b1-b2)<=1):
    print('YES')
else:
    print('NO')
```

**Input :**

4  
4  
5  
5

**Output :**

YES

**Inference :**

As king can move in all directions but only moves in all directions but only moves to adjacent cells so, the difference of any coordinates should be unity.

**Statement :**

In chess, the bishop moves diagonally, any number of squares. Given two different squares of the chessboard, determine whether a bishop can go from the first to the second in one move. The program receives as input four numbers from 1 to 8, specifying the column and row numbers of the starting square and the column and row numbers of the ending square. The program should output YES if a Bishop can go from the first square to the second in one move, or NO otherwise.

**Objective :**

To get familiar with bishop moves in chess using if-else conditional statements.

**Code :**

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
if abs( x1 - x2 ) == abs( y1 - y2 ):
    print('YES')
else:
    print('NO')
```

**Input :**

1  
4  
4  
1

**Output :**

YES

**Inference :**

When bishop moves from one cell to another then the x & y-coordinates increases simultaneously.

**Statement :**

Chess queen moves horizontally, vertically or diagonally to any number of cells. Given two different cells of the chessboard, determine whether a queen can go from the first cell to the second in one move. The program receives the input of four numbers from 1 to 8, each specifying the column and row number, first two - for the first cell, and then the last two - for the second cell. The program should output YES if a queen can go from the first cell to the second in one move, or NO otherwise.

**Objective :**

To get familiar with queen moves in chess using if-else conditional statements.

**Code :**

```
a1=int(input())
b1=int(input())
a2=int(input())
b2=int(input())
if (abs(a1 - a2) == abs(b1 - b2)) | ((a1==a2) | (b1==b2)):
    print('YES')
else:
    print('NO')
```

**Input :**

2  
7  
6  
7

**Output :**

YES

**Inference :**

It is the combination of the codes of moves of rook and moves of bishop.

**Statement :**

Chess knight moves like the letter L. It can move two cells horizontally and one cell vertically, or two cells vertically and one cells horizontally. Given two different cells of the chessboard, determine whether a knight can go from the first cell to the second in one move. The program receives the input of four numbers from 1 to 8, each specifying the column and row number, first two - for the first cell, and then the last two - for the second cell. The program should output YES if a knight can go from the first cell to the second in one move, or NO otherwise.

**Objective :**

To get familiar with knight moves in chess using if-else conditional statements.

**Code :**

```
a1=int(input())
b1=int(input())
a2=int(input())
b2=int(input())
if (abs(a1 - a2)==1)&(abs(b1 - b2)==2)|(abs(a1 - a2)==2)&(abs(b1 - b2)==1):
    print('YES')
else:
    print('NO')
```

**Input :**

2  
5  
3  
2

**Output :**

YES

**Inference :**

As the knight moves 2 horizontal and 1 vertical in a single moves or 2 vertical and 1 horizontal moves so, we took their differences as 1 for vertical and 2 for horizontal.

**Statement :**

Chocolate bar has the form of a rectangle divided into  $n \times m$  portions. Chocolate bar can be split into two rectangular parts by breaking it along a selected straight line on its pattern. Determine whether it is possible to split it so that one of the parts will have exactly  $k$  squares. The program reads three integers:  $n$ ,  $m$ , and  $k$ . It should print YES or NO.

**Objective :**

To get familiar with splitting a chocolate bar into respected rows and columns using if-else conditional statements.

**Code :**

```
n = int(input())
m = int(input())
k = int(input())
if (k % n == 0 and k / n < m) or (k % m == 0 and k / m < n):
    print('YES')
else:
    print('NO')
```

**Input :**

4  
2  
6

**Output :**

YES

**Inference :**

We don't know whether  $m$  or  $n$  is greater, if  $m$  is smaller we can make squares of  $m \times m$  portions and  $k$  should be less than  $m \times n$  portions.

**Statement :**

Given the year number. You need to check if this year is a leap year. If it is, print LEAP, otherwise print COMMON. A year is a leap year if its number is exactly divisible by 4 and is not exactly divisible by 100. A year is always a leap year if its number is exactly divisible by 400

**Objective :**

To get familiar with leap years using if-else conditional statements.

**Code :**

```
y=int(input())
if (y%4==0)&(y%100!=0):
    print('LEAP')
elif (y%400==0):
    print('LEAP')
else:
    print('COMMON')
```

**Input :**

1492

**Output :**

LEAP

**Inference :**

A year is a leap year if its number is exactly divisible by 4 and is not exactly divisible by 100. A year is always a leap year if its number is exactly divisible by 400

## **WEEK – 4 ( For Loops )**

### **Statement :**

Given two integers A and B ( $A \leq B$ ). Print all numbers from A to B inclusively.

### **Objective :**

To get familiar with printing the numbers between 2 input numbers in ascending order using for loops.

#### **Code :**

```
a = int( input( ) )  
b = int( input( ) )  
for i in range( a , b+1 ):  
    print( i )
```

#### **Input :**

1  
4

#### **Output :**

1  
2  
3  
4

### **Inference :**

In c we give for(  $i = 0 ; i < n ; i++$  ) here we give for i in range( 0, n+1, 1 ) where i increments by 1 from 0 to n i.e.,  $0 \leq i < n+1$  .



**Statement :**

Given two integers A and B. Print all numbers from A to B inclusively, in ascending order, if  $A < B$ , or in descending order, if  $A \geq B$ .

**Objective :**

To get familiar with printing the numbers between 2 input numbers either in ascending or descending order for loops.

**Code :**

```
a=int(input())
b=int(input())
if a<b:
    for i in range(a,b+1,1):
        print(i)
else:
    for i in range(a,b-1,-1):
        print(i)
```

**Input :**

1  
5

**Output :**

1  
2  
3  
4  
5

**Inference :**

It is similar to C only the format matters

for (i = a ; i<b+1 ; i++) / for l in range(a,b+1,1) → ascending

for (i = a ; i>b-1 ; i - -) / for l in range(a,b-1,-1) → descending

**Statement :**

5 numbers are given in the input. Read them and print their sum. Use as few variables as you can.

**Objective :**

To get familiar with finding the sum of 5 input numbers using for loops.

**Code :**

```
s=0
for i in range(10):
    a=int(input())
    s+=a
print(s)
```

**Input :**

1  
2  
3  
4  
5

**Output :**

15

**Inference :**

We can also write range ( 0,10,1 ) but by default it increments by 1 and also repeats for n times if we write range ( n ).

**Statement :**

N numbers are given in the input. Read them and print their sum.

The first line of input contains the integer N, which is the number of integers to follow. Each of the next N lines contains one integer. Print the sum of these N integers.

**Objective :**

To get familiar with finding the sum of n random numbers using for loops.

**Code :**

```
n=int(input())
s=0
for i in range(n):
    a=int(input())
    s=s+a
print(s)
```

**Input :**

4  
3  
3  
3  
3

**Output :**

12

**Inference :**

It is similar to before problem but here we give the no. of input values also.

**Statement :**

For the given integer N calculate the following sum:

$$1^3 + 2^3 + 3^3 + ..... + N^3$$

**Objective :**

To get familiar with finding answers for different mathematical series using for loops.

**Code :**

```
n=int(input())
s=0
for i in range(1,n+1):
    s=s+(i**3)
print(s)
```

**Input :**

6

**Output :**

441

**Inference :**

Here to print the cubes we add and store its values in s and cube can also written as  $i**3$  or  $i*i*i$ . and this is added to a variable s for every iteration of the for loop.

**Statement :**

In mathematics, the factorial of an integer  $n$ , denoted by  $n!$  is the following product:

$$n! = 1 \times 2 \times \dots \times n$$

For the given integer  $n$  calculate the value  $n!$ . Don't use `math` module in this exercise.

**Objective :**

To get familiar with finding the factorial using for loops.

**Code :**

```
n=int(input())
p=1
for i in range(1,n+1):
    p=p*i
print(p)
```

**Input :**

4

**Output :**

24

**Inference :**

In the range (  $1, n+1$  ) we use 1 instead of 0 as  $p=p*i$  always becomes 0 as  $p=p*0=0$  we get factorial of any number is ' 0 '.

**Statement :**

Given N numbers: the first number in the input is N, after that N integers are given. Count the number of zeros among the given integers and print it. You need to count the number of numbers that are equal to zero, not the number of zero digits.

**Objective :**

To get familiar with counting the no. of zeros in the given sequence using for loops.

**Code :**

```
n=int(input())
c=0
for i in range(n):
    a=int(input())
    if a==0:
        c=c+1
print(c)
```

**Input :**

4  
0  
21  
0  
35

**Output :**

2

**Inference :**

Here we use C to convert the no. of zeros, if the a value becomes 0 the C value is incremented. Hence gives the no. of zeros.

**Statement :**

Given an integer n, print the sum  $1!+2!+3!+\dots+n!$ .

This problem has a solution with only one loop, so try to discover it. And don't use the math library :)

**Objective :**

To get familiar with adding the individual factorials of n numbers using for loops.

**Code :**

```
n=int(input())
p=1
s=0
for i in range(1,n+1):
    p=p*i
    s=s+p
print(s)
```

**Input :**

4

**Output :**

33

**Inference :**

We just combine the 4<sup>th</sup> and 6<sup>th</sup> program .sum of n numbers and factorial, the for loop iterates the factorial and adds to s.

**Statement :**

For given integer  $n \leq 9$  print a ladder of  $n$  steps. The  $k$ -th step consists of the integers from 1 to  $k$  without spaces between them.

To do that, you can use the `sep` and `end` arguments for the function `print ()`.

**Objective :**

To get familiar with printing different patterns using for loops.

**Code :**

```
n=int(input())
for i in range(1,n+1):
    k=1
    for j in range(1,i+1):
        print(k,end='')
        k=k+1
    print()
```

**Input :**

3

**Output :**

1  
12  
123

**Inference :**

'end' is used to join the elements in a row for single iteration of  $j$  for loop and to go to the next line after the  $j$  for loop we just give an empty `print()` statement.



**Statement :**

Given a number N, followed by N – 1 integers - representing the numbers on the remaining cards (distinct integers in the range from 1 to N). Find and print the number on the lost card.

**Objective :**

To get familiar with finding the missing card using for loops.

**Code :**

```
n=int(input())
s=0
for i in range(n-1):
    i=int(input())
    s+=i
t=n*(n+1)//2
print(t-s)
```

**Input :**

4  
2  
3  
4

**Output :**

1

**Inference :**

As there are cards with numbers less than or equal to the no.of cards so,  
[ the sum of all cards ] – [ sum of N-1 cards ] gives the lost card.

## **WEEK – 5 ( Strings )**

### **Statement :**

You are given a string.

In the first line, print the third character of this string.

In the second line, print the second to last character of this string.

In the third line, print the first five characters of this string.

In the fourth line, print all but the last two characters of this string.

In the fifth line, print all the characters of this string with even indices (remember indexing starts at 0, so the characters are displayed starting with the first).

In the sixth line, print all the characters of this string with odd indices (i.e. starting with the second character in the string).

In the seventh line, print all the characters of the string in reverse order.

In the eighth line, print every second character of the string in reverse order, starting from the last one.

In the ninth line, print the length of the given string.

### **Objective :**

To get familiar with strings and operations on strings.

### **Code :**

```
s = input()
print(s[2])
print(s[-2])
print(s[:5])
```

```
print(s[:-2])
print(s[::-2])
print(s[1::2])
print(s[::-1])
print(s[::-2])
print(len(s))
```

**Input :**

Abrakadabra

**Output :**

r  
r  
Abrak  
Abrakadab  
Arkdba  
baaar  
arbadakarbA  
abdkrA  
11

**Inference :**

In the above program we need to print above statements using strings.

**Statement :**

Given a string consisting of words separated by spaces. Determine how many words it has. To solve the problem, use the method `count`.

**Objective :**

To count number of words in a given string.

**Code :**

```
print(input().count(' ') + 1)
```

**Input :**

Hello world

**Output :**

2

**Inference :**

using in built function “count” we count the number of words in a given statement.

**Statement :**

Given a string. Cut it into two "equal" parts (If the length of the string is odd, place the center character in the first string, so that the first string contains one more character than the second). Now print a new string on a single row with the first and second half's interchanged (second half first and the first half second)  
Don't use the statement `if` in this task.

**Objective :**

To make the string into two halves and print.

**Code :**

```
s = input()
l = int(len(s)/2)
print(s[-l:]+s[:l])
```

**Input :**

Hi

**Output :**

iH

**Inference :**

Here we used in built "`len()`" function to find the length of string and indexing for printing.

**Statement :**

Given a string consisting of exactly two words separated by a space. Print a new string with the first and second word positions swapped (the second word is printed first).

This task should not use loops and if.

**Objective :**

To swap the words separated by space.

**Code :**

```
s = input()
n = s.find(' ')
print(s[n:], s[:n])
```

**Input :**

Hello, world!

**Output :**

world! Hello,

**Inference :**

Inbuilt function “find()” is used to find space, and indexing used from first and last.

**Statement :**

Given a string that may or may not contain a letter of interest. Print the index location of the first and last appearance of f. If the letter f occurs only once, then output its index. If the letter f does not occur, then do not print anything.

Don't use loops in this task.

**Objective :**

to print first and last occurrences.

**Code :**

```
s = input()
c = s.count('f')
if c == 1:
    print(s.find('f'))
if c > 1:
    print(s.find('f'),s.rfind('f'))
```

**Input :**

comfort

**Output :**

3

**Inference :**

Inbuilt function find() and rfind() are used,and count() is used.

**Statement :**

Given a string that may or may not contain the letter of interest. Print the index location of the second occurrence of the letter f. If the string contains the letter f only once, then print -1, and if the string does not contain the letter f, then print -2.

**Objective :**

to find the index of second occurrence of desired letter.

**Code :**

```
s = input()
c = s.count('f')
if c == 0:
    print(-2)
if c == 1:
    print(-1)
if c > 1:
    print(s.find('f', s.find('f') + 1))
```

**Input :**

comfort

**Output :**

-1

**Inference :**

Here inbuilt function find () is used and count() is used



**Statement :**

Given a string in which the letter h occurs at least twice. Remove from that string the first and the last occurrence of the letter h, as well as all the characters between them.

**Objective :**

To remove all characters in between the two letters of desired letter.

**Code :**

```
s = input()
print(s[:s.find('h')] + s[s.rfind('h')+1:])
```

**Input :**

In the hole in the ground there lived a hobbit

**Output :**

In tobbit

**Inference :**

Here find() and rfind() inbuilt functions are used to find desired letter and the first and last parts of strings are printed by using string slicings and concatenated by "+".

**Statement :**

Given a string in which the letter h occurs at least two times, reverse the sequence of characters enclosed between the first and last appearances.

**Objective :**

To reverse the characters in between the two letters of desired letter.

**Code :**

```
s = input()
i = s[s.find('h'):(s.rfind('h'))]
print(s[:s.find('h') + 1] + i[::-1] + s[s.rfind('h')+1:])
```

**Input :**

In the hole in the ground there lived a  
hobbit

**Output :**

In th a devil ereht dnuorg eht ni eloh  
ehobbit

**Inference :**

Here indexing of [::-1] is crucial logic and find is used with +1 of desired letter, rfind() also used to find from last.

**Statement :**

Given a string. Remove from this string all the characters @.

**Objective :**

To remove desired characters

**Code :**

```
print(input().replace('@',''))
```

**Input :**

Bilbo.Baggins@bagend.hobbiton.shire.  
me

**Output :**

Bilbo.Bagginsbagend.hobbiton.shire.  
me

**Inference :**

here simply inbuilt function replace() is used as “ to remove desired characters at desired places.

**Statement :**

Given a string. Replace in this string all the numbers 1 by the word one.

**Objective :**

To learn how to replace the one word with another word.

**Code :**

```
print(input().replace('1','one'))
```

**Input :**

1+1=2

**Output :**

one+one=2

**Inference :**

Here very simply inbuilt function keyword replace is used to replace one word with another word.

## **Week – 6 ( While Loops )**

### **Statement :**

For a given integer N, print all the squares of integer numbers where the square is less than or equal to N, in ascending order.

### **Objective :**

To get familiar to print the squares up to a given range using while loops.

### **Code :**

```
n=int(input())
i=1
while (i*i)<=n:
    print(i*i,end=' ')
    i+=1
```

### **Input :**

50

### **Output :**

1 4 9 16 25 36 49

### **Inference :**

When  $i*i$  crosses the given n value, the while loop will be terminated. Also as i was incremented by 1 all the consecutive squares will be printed.

**Statement :**

Given an integer not less than 2. Print its smallest integer divisor greater than 1.

**Objective :**

To get familiar with finding the smallest divisor using while loops.

**Code :**

```
n=int(input())
i=2
while(n%i!=0):
    i=i+1
print(i)
```

**Input :**

27

**Output :**

3

**Inference :**

i starts from 2 and increments by 1 if the while loop executes and also checks whether it is divisor or not.

**Statement :**

For a given integer N, find the greatest integer x where  $2^x$  is less than or equal to N. Print the exponent value and the result of the expression  $2^x$ .

Don't use the operation `**`.

**Objective :**

To get familiar to find least power of 2 that is less than or equal to given number.

**Code :**

```
n=int(input())
x=0
e=1
while(n>=e):
    x=x+1
    e=e*2
print(x-1,e//2)
```

**Input :**

50

**Output :**

5 32

**Inference :**

Here  $e = 2^x$ , where  $x = 0$ ,  $e = 1$  and for every incrementation of x, e is multiplied by 2. We want the x and e values before executing the true condition of the loop so we print  $x - 1$  and  $e // 2$ .

**Statement :**

As a future athlete you just started your practice for an upcoming event. Given that on the first day you run x miles, and by the event you must be able to run y miles.

Calculate the number of days required for you to finally reach the required distance for the event, if you increase your distance each day by 10% from the previous day.

Print one integer representing the number of days to reach the required distance.

**Objective :**

To get familiar with calculating the time taken to cover the given distance.

**Code :**

```
x=int(input())
y=int(input())
d=1
while(x<y):
    x=x*1.1
    d=d+1
print(d)
```

**Input :**

10  
20

**Output :**

9

**Inference :**

In 1 day it takes x miles, next day it becomes  $(x + x*10\%) = x*(1+0.1) = x*1.1$  and for every execution of while loop we increment number of days.



**Statement :**

Given a sequence of non-negative integers, where each number is written in a separate line. Determine the length of the sequence, where the sequence ends when the integer is equal to 0. Print the length of the sequence (not counting the integer 0). The numbers following the number 0 should be omitted.

**Objective :**

To get familiar to calculate the length of an given sequence ending with 0.

**Code :**

```
c=0
while(int(input())!=0):
    c=c+1
print(c)
```

**Input :**

1  
4  
3  
0

**Output :**

3

**Inference :**

For every execution of while we give an input, if it is not equal to 0 we increment the count which gives the number of elements. Finally we print the count.

**Statement :**

Determine the sum of all elements in the sequence, ending with the number 0.

**Objective :**

To get familiar with finding the sum of elements in a sequence ending with 0.

**Code :**

```
s=0
i=int(input())
while(i!=0):
    s=s+i
    i=int(input())
print(s)
```

**Input :**

1  
4  
3  
0

**Output :**

8

**Inference :**

The while loop adds the input values to s until  $i \neq 0$ , so to enter into the while loop we had gave an input value outside the while loop to give a base condition.

**Statement :**

Determine the average of all elements of the sequence ending with the number 0.

**Objective :**

To get familiar with finding the average of the given sequence ending with 0.

**Code :**

```
s=0
n=0
i=int(input())
while(i!=0):
    s=s+i
    i=int(input())
    n=n+1
print(s/n)
```

**Input :**

1  
4  
3  
0

**Output :**

2.6666666666666665

**Inference :**

Similar to previous 2 questions, we find the sum by adding the each element to an random variable simultaneously we count the number of elements in the sequence. Finally we divide the sum by count and print it.

**Statement :**

A sequence consists of integer numbers and ends with the number 0. Determine the largest element of the sequence.

**Objective :**

To get familiar to find the largest element in an given sequence ending with 0.

**Code :**

```
i=int(input())
m=0
while(i!=0):
    if(i>m):
        m=i
    i=int(input())
print(m)
```

**Input :**

1  
4  
3  
0

**Output :**

4

**Inference :**

The ' if ' loop checks whether  $i > m$  so then for every input value if the condition comes true we assign i value to m.

**Statement :**

A sequence consists of integer numbers and ends with the number 0. Determine the index of the largest element of the sequence. If the highest element is not unique, print the index of the first of them.

**Objective :**

To get familiar with finding the index of maximum element in a sequence.

**Code :**

```
i=int(input())
m=0
l=1
while(i!=0):
    if(i>m):
        m=i
        c=l
    i=int(input())
    l=l+1
print(c)
```

**Input :**

1  
4  
3  
0

**Output :**

2

**Inference :**

Here l should be incremented for every execution of while loop at the same time we check the maximum element and initiate the l value 'at that point' to c.

**Statement :**

Determine the number of even elements in the sequence ending with the number 0.

**Objective :**

To get familiar with finding the no. of even numbers in the sequence ending with 0.

**Code :**

```
i=int(input())
e=0
while(i!=0):
    if(i%2==0):
        e=e+1
    i=int(input())
print(e)
```

**Input :**

1  
4  
3  
0

**Output :**

1

**Inference :**

The 'if' loop checks whether the input value is even or not, if it is even we increment the count and finally print the count value.

**Statement :**

A sequence consists of integer numbers and ends with the number 0. Determine how many elements of this sequence are greater than their neighbors above.

**Objective :**

To get familiar with finding no. of elements before the largest element in a sequence ending with 0.

**Code :**

```
i=int(input())
g=0
while(i!=0):
    j=int(input())
    if(i<j):
        g=g+1
    i=j
print(g)
```

**Input :**

1  
4  
3  
0

**Output :**

1

**Inference :**

Here we need 2 elements to check the greatest among the neighbors, so we take j inside the while loop if  $i < j$  we increment g and assign  $j = i$  to check the next element.

**Statement :**

The sequence consists of distinct positive integer numbers and ends with the number 0. Determine the value of the second largest element in this sequence. It is guaranteed that the sequence has at least two elements.

**Objective :**

To get familiar with finding the second maximum element in a sequence.

**Code :**

```
i = int(input( ))
m = 0
sm = 0
while ( i != 0 ):
    if ( i > m ):
        sm = m
        m = i
    elif ( i < m ):
        if ( i > sm ):
            sm = i
    i = int(input( ))
print( sm )
```

**Input :**

1  
4  
3  
0

**Output :**

3

**Inference :**

Initially we consider max ( m ), and second max ( sm ). Now there are 2 cases if i > m, then i is max and assume m as second max, if i < m we check i > sm and then we consider i as second maximum.



**Statement :**

A sequence consists of integer numbers and ends with the number 0. Determine how many elements of this sequence are equal to its largest element.

**Objective :**

To get familiar with counting no. of second max elements in the sequence.

**Code :**

```
i=int(input())
m=0
c=0
while(i!=0):
    if(i>m):
        m=i
        c=1
    elif(i==m):
        c=c+1
    i=int(input())
print(c)
```

**Input :**

1  
4  
3  
0

**Output :**

1

**Inference :**

The if loop searches for max term, if found it was assigned to m and 'elif' loop counts the max terms after if loop.

**Statement :**

The Fibonacci sequence is defined as follows:

$$\emptyset_0=0, \emptyset_1=1, \emptyset_n = \emptyset_{n-1} + \emptyset_{n-2}.$$

Given a non-negative integer n, print the nth Fibonacci number  $\emptyset_n$

This problem can also be solved with a **for** loop.

**Objective :**

To get familiar to find the nth Fibonacci number.

**Code :**

```
p=int(input())
n1,n2=0,1
i=0
while(i<p):
    n1,n2=n2,n1+n2
    i=i+1
print(n1)
```

**Input :**

4

**Output :**

3

**Inference :**

According to Fibonacci series n1 and n2 are assigned as 0 and 1. For every incrementation of i the successive Fibonacci numbers forms until i reaches the given position ' p '.

**Statement :**

The Fibonacci sequence is defined as follows:

$$\emptyset_0=0, \emptyset_1=1, \emptyset_n = \emptyset_{n-1} + \emptyset_{n-2}.$$

Given an integer a, determine its index among the Fibonacci numbers, that is, print the number n such that  $\emptyset_n=a$ . If a is not a Fibonacci number, print -1.

**Objective :**

To get familiar to know whether the given number is present in the Fibonacci series or not.

**Code :**

```
n=int(input())
n1,n2=0,1
i=0
while(n1<n):
    n1,n2=n2,n1+n2
    i+=1
if n==n1:
    print(i)
else:
    print('-1')
```

**Input :**

13

**Output :**

7

**Inference :**

The while loop gives Fibonacci series up to n, and if  $n == n1$  that means n exists in the Fibonacci series and else loop is executed if  $n2 > n$ .

**Statement :**

Given a sequence of integer numbers ending with the number 0. Determine the length of the widest fragment where all the elements are equal to each other.

**Objective :**

To get familiar to find the no. of greatest no. of successive elements in a sequence

**Code :**

```
i=int(input())
j=0
m,c=1,1
while i!=0:
    j=int(input())
    if i==j:
        c=c+1
    else:
        if c>m:
            m=c
        c=1
    i=j
print(m)
```

**Input :**

1  
4  
3  
3  
0

**Output :**

2

**Inference :**

The ' if ' loop counts the similar elements and if any different element was given , ' else ' loop will execute and assigns the count to m and again to start counting for other similar elements, c is again initialized to 1.

## WEEK – 7 ( LISTS )

### **Statement :**

Given a list of numbers, find and print all the list elements with an even index number. (i.e. `A[0]`, `A[2]`, `A[4]`, ...).

### **Objective :**

To get familiar with Lists using an in built functions i.e., `split()`, `join()`, etc.

### **Code :**

```
a = input().split()
for i in range(0, len(a), 2):
    print(a[i])
```

<b><u>Input :</u></b>	<b><u>Output :</u></b>
1 2 3 4 5	1 3 5

### **Inference :**

The method `split()` has an optional parameter that determines which string will be used as the separator between list items

**Statement :**

Given a list of numbers, find and print all elements that are an even number. In this case use a for-loop that iterates over the list, and not over its indices! That is, don't use range()

**Objective :**

To get familiar with Lists using an in built functions i.e., split(),join(),etc.

**Code :**

```
a = [int(i) for i in input().split()]
for elem in a:
    if elem % 2 == 0:
        print(elem)
```

**Input :**

1 2 2 3 3 3 4

**Output :**

2 2 4

**Inference :**

Here the entire list can be read using input().We can use a string method split(),which returns a list of string separated by spaces.

**Statement:**

Given a list of numbers, find and print all the elements that are greater than the previous element.

**Objective:**

To get familiar with the lists to store to store the multiple data types in a single variable.

**Code:**

```
a=[int(i) for i in input().split()]
for i in range(1,len(a)):
    if a[i]>a[i-1]:
        print(a[i])
```

**Input:**

1 5 2 4 3

**Output:**

5 4

**Inference:**

The method split() has an optional parameter that determines which string will be used as the separator between the list items.

**Statement:**

Given a list of numbers, find and print the first adjacent elements which have the same sign. If there is no such pair, leave the output blank.

**Objective:**

To get familiar with lists using an in built functions i.e,split(),join(),etc.

**Code:**

```
a=[int(i) for i in input().split()]
for i in range(1,len(a)):
    if a[i]*a[i-1]>0:
        print(a[i-1],a[i])
        break
```

**Input:**

-1 2 3 -1 -2

**output:**

2 3

**Inference:**

Here the entire list can be read using input().Using for loop and range of length of the string we can solve the given problem.



**Statement :**

Given a list of numbers, determine and print the quantity of elements that are greater than both of their neighbors.

**Objective :**

To get familiar with the Lists to store the multiple data types in a single variable.

**Code :**

```
a=[int(i) for i in input().split()]
count=0
for i in range(1,len(a)-1):
    if(a[i-1]<a[i]&a[i]<a[i+1]):
        count+=1
print(count)
```

**Input :**

1 2 3 4 5

**Output :**

0

**Inference :**

The method split() has an optional parameter that determines which string will be used as the separator between the list items .

**Statement :**

Given a list of numbers ,determine the element in the list with the largest value.Print the value of the largest element and then the index number.If the highest element is not unique ,print the index of the first instance.

**Objective :**

To get familiar with the lists to store the multiple data types in a single variable.

**Code :**

```
a=[int(i) for i in input().split()]
index=0
for i in range(1,len(a)):
    if a[i]>a[index]:
        index=i
print(a[index],index)
```

**Input :**

1 2 3 2 1

**Output :**

3 2

**Inference :**

Here a.index(x) returns the index of the first occurrence of element x in the list . It is one of the operation of the list.

**Statement :**

Given a list of numbers with all of its elements sorted in ascending order, determine and print the quantity of distinct elements in it.

**Objective :**

To get familiar with lists using an in built function i.e., split( ) .

**Code :**

```
a = [int(i) for i in input().split()]
num_distinct = 1
for i in range(0, len(a) - 1):
    if a[i] != a[i + 1]:
        num_distinct += 1
print(num_distinct)
```

**Input :**

1 2 2 3 3 3

**Output :**

3

**Inference :**

In this program we used split() to take input and make it into a 1-D list . We also used an in-built function len() to find the length of an array .

**Statement :**

Given a list of numbers, swap adjacent items in pairs (A[0] with A[1], A[2] with A[3], etc.). Print the resulting list. If a list has an odd number of elements, leave the last element in place.

**Objective :**

To get familiar with lists using some in built function i.e., split( ) , join( ).

**Code :**

```
a = [int(i) for i in input().split()]
for i in range(1, len(a), 2):
    a[i - 1], a[i] = a[i], a[i - 1]
print(' '.join([str(i) for i in a]))
```

**Input :**

1 2 3 4 5

**Output :**

2 1 4 3 5

**Inference :**

In this program we used the range( ) with range of 2 , so that we can exchange alternative positions in lists . We also used the join( ) to print the output in a single line .

**Statement :**

Given a list of unique numbers, swap the minimal and maximal elements of this list. Print the resulting list.

**Objective :**

To get familiar to swap the min and max elements.

**Code :**

```
index_of_max = 0
index_of_min = 0
a = [int(i) for i in input().split()]
for i in range(1, len(a)):
    if a[i] > a[index_of_max]:
        index_of_max = i
    if a[i] < a[index_of_min]:
        index_of_min = i
a[index_of_max], a[index_of_min] = a[index_of_min], a[index_of_max]
print(' '.join([str(i) for i in a]))
```

**Input :**

3  
4  
5  
2  
1

**Output :**

3  
4  
1  
2  
5

**Inference :**

In these program i have learnt how to swap the minimum and max terms of a given list, it will just check the previous and next number.

**Statement :**

Given a list of numbers, count how many element pairs have the same value (are equal). Any two elements that are equal to each other should be counted exactly once.

**Objective :**

To get familiar to print the pair of same values at once.

**Code :**

```
a = [int(s) for s in input().split()]
counter = 0
for i in range(len(a)):
    for j in range(i + 1, len(a)):
        if a[i] == a[j]:
            counter += 1
print(counter)
```

**Input :**

1  
2  
3  
2  
3

**Output :**

2

**Inference :**

In these program i have learnt how to count the element pairs of same values and it is considered as exactly ones , in these we have used split function and length function.

**Statement :**

Given a list of numbers, find and print the elements that appear in the list only once. The elements must be printed in the order in which they occur in the original list

**Objective :**

To get familiar to print the elements that are in the list only ones.

**Code :**

```
a = [int(s) for s in input().split()]
unqs = [ ]
for elem in a:
    if a.count(elem) == 1:
        unqs.append(elem)
print(' '.join([str(i) for i in unqs]))
```

**Input :**

1  
2  
2  
3  
3  
3

**Output :**

1

**Inference :**

In these program i have learnt how to print the number that appears only one's in the list,the count function will count the number and it will append the number into another list.

### **Statement :**

In chess it is known that it is possible to place 8 queens on an 8×8 chess board such that none of them can attack another. Given a placement of 8 queens on the board, determine if there is a pair of queens that can attack each other on the next move. Print the word **NO** if no queen can attack another, otherwise print **YES**. The input consists of eight coordinate pairs, one pair per line, with each pair giving the position of a queen on a standard chess board with rows and columns numbered starting at 1.

### **Objective :**

To get familiar to print the exact place of the 8 queens no queen would be attacked by other.

#### **Code :**

```
N = 8
result = 'NO'
x = [0] * N
y = [0] * N
for i in range(N):
    x[i], y[i] = [int(j) for j in input().split()]
for i in range(N):
    for j in range(i+1, N):
        if x[i] == x[j] or y[i] == y[j] or abs(x[i] - x[j]) == abs(y[i] - y[j]):
            result = 'YES'
print(result)
```

#### **Input :**

```
1 7
2 4
3 2
4 8
5 6
```

#### **Output :**

```
No
```

### **Inference :**

In these program i have learnt how to print the 8 queens in the chess board without attacking the one queen to the another ,in these we have used range function.



### **Statement :**

In bowling, the player starts with 10 pins at the far end of a lane. The object is to knock all the pins down. For this exercise, the number of pins and balls will vary. Given the number of pins  $N$  and then the number of balls  $K$  to be rolled, followed by  $K$  pairs of numbers (one for each ball rolled), determine which pins remain standing after all the balls have been rolled. The balls are numbered from 1 to  $N$  (inclusive) for this situation. The subsequent number pairs, one for each  $K$  represent the start to stop (inclusive) positions of the pins that were knocked down with each role. Print a sequence of  $N$  characters, where "I" represents a pin left standing and "." represents a pin knocked down.

### **Objective :**

To get familiar to print "I" represents a pin left standing and "." represents a pin knocked down.

#### **Code :**

```
N, K = [int(i) for i in input().split()]
kegels = ['I'] * N
for j in range(K):
    l, r = [int(i) for i in input().split()]
    kegels[l-1:r] = ['.'] * (r-l+1)
print("".join([str(i) for i in kegels]))
```

#### **Input :**

```
10 3
8 10
2 5
3 6
```

#### **Output :**

```
I.....I...
```

### **Inference :**

In these program i have learnt how to print how many balls rolled and how many pins has been dropped,we have used split function and join function.

## **WEEK – 8 (Functions and recursion)**

### **Statement:**

Given four real numbers representing cartesian coordinates: (x1,y1),(x2,y2)(x1,y1),(x2,y2). Write a function distance (x1, y1, x2, y2) to compute the distance between the points (x1,y1)(x1,y1) and (x2,y2)(x2,y2). Read four real numbers and print the resulting distance calculated by the function.

The formula for distance between two points can be found at wolfram.

### **Objective:**

To understand the concept of functions and finding the distance between 2 points using a function in python.

#### **code :**

```
a=float(input())
b=float(input())
c=float(input())
d=float(input())
def segment(a,b,c,d):
    d=((a-c)**2 + (b-d)**2)**(1/2)
    print(d)
segment(a,b,c,d)
```

#### **Input :**

0  
0  
1  
1

#### **Output :**

1.4142135623730951

### **Inference :**

Distance between 2 points (x1,y1) & (x2,y2) is  $\sqrt{\sqrt{x1-x2}+\sqrt{y1-y2}}$   $\Rightarrow ((x1-x2)**2 + (y1-y2)**2)$  as also input should be given in float as in co-ordinate axes have all type of values.

**Statement :**

Given a positive real number aa and **integer** nn.

Compute  $a^n$ . Write a function `power (a,n)` to calculate the results using the function and print the result of the expression. Dont use the same function from the standard library.

**Objective :**

To understand the concept of recursion and finding the , a power b using a function in python.  
where a and b integers.

**Code :**

```
def power(a,n):  
    res=1  
    for i in range (abs(n)):  
        res*=a  
    if n>=0:  
        return res  
    else:  
        return 1/res  
print(power(float(input()),int(input())))
```

**Input :**

2  
-3

**Output :**

0.125

**Inference :**

Here a and x should be in float as the power any number may be in float so if we give int then it may not give the accurate answer.

**Statement :**

Write a function capitalize (low case words) that takes the lower case word and returns the word with the first letter capitalized. Eg., `print(capitalize('word'))` should print the word **Word**. Then, given a line of lowercase ASCII words (text separated by a single space), print it with the first letter of each word capitalized using the your own function `capitalize()`. In Python there is a function `ord(character)`, which returns character code in [the ASCII chart](#), and the function `chr(code)`, which returns the character itself from the ASCII code. For example, `ord('a') == 97`, `chr(97) == 'a'`.

**Objective :**

To understand the concept of recursion and capitalize 1st letter of every word in given input using a function in python.

**Code :**

```
a=str(input())
def cap(a):
    a=a.title(a)
    return a
print(cap(a))
```

**Input :**

harry potter

**Output :**

Harry Potter

**Inference :**

As `ord()` gives the ASCII value index and `chr()` gives the variable in that index so we split the sentence into words as elements in lists so we can access the starting letter and capitalize it.

**Statement :**

Given a positive real number  $a$  and a non-negative integer  $n$ . Calculate  $a^n$  without using loops, `**` operator or the built in function `math.pow()`. Instead, use recursion and the relation  $a^n = a \cdot a^{n-1}$ . Print the result from the function `power(a, n)`.

**Objective :**

To understand the concept of recursion and finding the  $a$  power  $b$  using a function in python. Where  $a$  and  $b$  integers.

**Code :**

```
def power(a,b):  
    if b == 0:  
        return 1  
    else:  
        return a*power(a,b-1)  
print(power(float(input()),int(input())))
```

**Input :**

2  
3

**Output :**

8.0

**Inference :**

Here  $a$  and  $x$  should be in float as the power any number may be in float so if we give int then it may not give the accurate answer.

**Statement :**

Given a sequence of integers that end with a 00. Print the sequence in reverse order.

Don't use lists or other data structures. Use the *force* of recursion instead.

**Objective :**

To understand the concept of recursion and to print given inputs in reverse order using a function in python.

**Code :**

```
def reverse():  
    a = int(input())  
    if a != 0:  
        reverse()  
    print(a)  
reverse()
```

**Input :**

1  
2  
3  
0

**Output :**

0  
3  
2  
1

**Inference:**

Rev()

l = 1

Rev()

l = 2

Rev()

l = 0

Print(0)

Print(2)

Print(1)

So first 0 will be printed and then 1 and then 2.

**Statement :**

Given a non-negative integer  $n$ , print the  $n$ th Fibonacci number. Do this by writing a function `fib(n)` which takes the non-negative integer  $n$  and returns the Fibonacci number.

Don't use loops, use the *flair* of recursion instead. However, you should think about why the recursive method is much slower than using loops.

**Objective :**

To understand the concept of recursion and to print  $n$ th element of Fibonacci series using a function in python.

**Code :**

```
def fib(n):  
    if n == 1 or n == 2:  
        return 1  
    else:  
        return fib(n-1) + fib(n-2)  
print(fib(int(input())))
```

**Input :**

6

**Output :**

8

**Inference :**

In Fibonacci series the  $n$ th term is equal to sum of  $(n-1)$ th +  $(n-2)$ th term.

First element of the series is 1,  $\text{Fib}(n-2) + \text{fib}(n-1)$  is recursion function



## **WEEK – 9 ( Two Dimensional lists (Arrrays ))**

### **Statement :**

Given two integers representing the rows and columns ( $m \times n$ ), and subsequent  $m$  rows of  $n$  elements, find the index position of the maximum element and print two numbers representing the index ( $i \times j$ ) or the row number and the column number. If there exist multiple such elements in different rows, print the one with smaller row number. If there multiple such elements occur on the same row, output the smallest column number.

### **Objective :**

To find the index of maximum element in a matrix(multi-dimensional list).

### **Code :**

```
n,m=[int(j) for j in input().split()]
a=[[int(j) for j in input().split()] for i in range(n)]
max_row = 0
max_col = 0
max=a[max_row][max_col]
for i in range(n):
    for j in range(m):
        if max < a[i][j]:
            max = a[i][j]
            max_row=i
            max_col=j
print(max_row,max_col)
```

**Input :**

3 4  
0 3 2 4  
2 3 5 5  
5 1 2 3

**Output :**

1 2

**Inference :**

Here in the above program we are using nested loop (one is for row and other for column). Within that code is written to find the index of maximum element in the matrix by iteration through the nested loop.

**Statement :**

Given an odd number integer  $n$ , produce a two-dimensional array of size  $(n \times n)$ . Fill each element with a single character string of ".". Then fill the middle row, the middle column and the diagonals with the single character string of "\*" (an image of a snow flake). Print the array elements in  $(n \times n)$  rows and columns and separate the characters with a single space.

**Objective :**

To fill the matrix (odd number  $n$  through users input  $(n \times n)$  matrix) in the above requirement pattern (mentioned in statement).

**Code :**

```
n = int(input())
k = n//2
a = [['.' * n for i in range(n)]]
for i in range(n):
    a[k][i] = '*'
    a[i][k] = '*'
    a[i][i] = '*'
    a[n-i-1][i] = '*'
for row in a:
    print(' '.join(row))
```

**Input :**

5

**Output :**

```
* . * . *
. * * * .
* * * * *
. * * * .
* . * . *
```

**Inference :**

Printing the pattern by taking users input (odd  $n$  value). And filling the characters based on the indexing method a code is written within a nested loop. In middle and diagonals "\*", otherwise "." is printed.

**Statement :**

Given two numbers nn and mm. Create a two-dimensional array of size  $(n \times m)$  and populate it with the characters "." and "\*" in a checkerboard pattern. The top left corner should have the character ".".

**Objective :**

To print the format in chess board pattern based on the statement provided.

**Code :**

```
def chessboard(n, m):
    if (n + m) % 2 == 0:
        return '.'
    else:
        return '*'

n, m = [int(j) for j in input().split()]
a = [[chessboard(i, j) for j in range(m)] for i in range(n)]
for row in a:
    print(' '.join(row))
```

**Input :**

3 4

**Output :**

```
. * . *
* . * .
. * . *
```

**Inference :**

To fill the pattern along with nested loop a recursion function is used to locate the "." at the corner leftmost part and in alternate pattern "\*" is printed based on the array size.

**Statement :**

Given an integer n, produce a two-dimensional array of size (n×n)(n×n) and complete it according to the following rules, and print with a single space between characters: On the main diagonal write 0. On the diagonals adjacent to the main, write 1. On the next adjacent diagonals write 2 and so forth. Print the elements of the resulting array.

**Objective :**

To print the matrix by placing all zeros at diagonal fashion later on increasing by 1 based on the size of the matrix.

**Code :**

```
n = int(input())
a=[[abs(i-j) for j in range(n)] for i in range (n)]
for row in a:
    print(' '.join([str(i) for i in row]))
```

**Input :**

5

**Output :**

```
0 1 2 3 4
1 0 1 2 3
2 1 0 1 2
3 2 1 0 1
4 3 2 1 0
```

**Inference :**

Here printing the matrix on placing 0's at main diagonal and adjacent to that it is increased further by using abs() function. And the code is written within the nested loop. And n value is given through user's input.

**Statement :**

Given an integer n, create a two-dimensional array of size (n×n) (n×n) and populate it as follows, with spaces between each character: The positions on the minor diagonal (from the upper right to the lower left corner) receive 1. The positions above this diagonal receive 0. The positions below the diagonal receive 2. Print the elements of the resulting array.

**Objective :**

To print the array based on the problem(arrangement in side diagonal fashion). 1's at the position (from the upper right to lower left corner) and position below diagonal receive 2.

**Code :**

```
n = int(input())
a = [0] * n
a = [[0] * (n - i - 1) + [1] + [2] * i for i in range(n)]
for row in a:
    print(' '.join([str(i) for i in row]))
```

**Input :**

4

**Output :**

0 0 0 1  
0 0 1 2  
0 1 2 2  
1 2 2 2

**Inference :**

Here printing the above matrix is done by iteration through for loop and also by join() method.

**Statement :**

Given two positive integers mm and nn, mm lines of nn elements, giving an m×n m×n matrix AA, followed by two non-negative integers ii and jj less than nn, swap columns ii and jj of AA and print the result. Write a function swap\_columns(a, i, j) and call it to exchange the columns.

**Objective :**

To swap the columns in a matrix using the data in the problem.

**Code :**

```
def swap_columns(a, i, j):  
    for row in a:  
        row[i], row[j] = row[j], row[i]  
    return a  
n, m = [int(k) for k in input().split()]  
a = [[int(j) for j in input().split()] for i in range(n)]  
i, j = [int(k) for k in input().split()]  
for row in swap_columns(a, i, j):  
    print(' '.join([str(i) for i in row]))
```

**Input :**

```
3 4  
11 12 13 14  
21 22 23 24  
31 32 33 34  
0 1
```

**Output :**

```
12 11 13 14  
22 21 23 24  
32 31 33 34
```

**Inference :**

In the above program the swapping of columns is done by recursive call which is called by the data in the nested for loop and later on index based the elements in the array are swapped.

**Statement :**

Given two positive integers mm and nn, mm lines of nn elements, giving an m×n matrix AA, followed by one integer cc, multiply every entry of the matrix by c c and print the result.

**Objective :**

Multiplying each element of a matrix by a certain number through users input (Scale of a matrix).

**Code:**

```
x,y=[int(i) for i in input().split()]
a=[]
for i in range(x):
    a.append([int(j)for j in input().split()])
n=int(input())
for i in range(x):
    for j in range(y):
        print(a[i][j]*n,end=' ')
    print()
```

**Input :**

```
3 4
11 12 13 14
21 22 23 24
31 32 33 34
2
```

**Output :**

```
22 24 26 28
42 44 46 48
62 64 66 68
```

**Inference :**

Scale of a matrix is done by entering m,n through users input an c is the constant to be multiplied for each element in the matrix here append and split() methods are used. Along within the nested loop.



**Statement :**

Given three positive integers mm, nn and rr, mm lines of nn elements, giving an  $m \times n \times n$  matrix A

A, and nn lines of rr elements, giving an  $n \times r \times r$  matrix BB, form the product matrix ABAB, which is the  $m \times r \times r$  matrix whose (i,k)(i,k) entry is the sum

$$A[i][1]*B[1][k]+\dots+A[i][n]*B[n][k]$$

$$A[i][1]*B[1][k]+\dots+A[i][n]*B[n][k]$$

and print the result.

**Objective :**

Multiply two matrices by entering row and column's values.

**Code :**

```
x,y,z=[int(i) for i in input().split()]
c=[0]*x
for i in range(x):
    c[i]=[0]*z
a=[[int(j) for j in input().split()]for i in range(x)]
b=[[int(l) for l in input().split()]for i in range(y)]
for n in range(x):
    for o in range(z):
        for p in range(y):
            c[n][o]+=a[n][p]*b[p][o]
for i in range(x):
    for j in range(z):
        print(c[i][j],end=' ')
    print()
```

**Input :**

3 4 2  
0 1 2 3  
4 5 6 7  
8 9 10 11  
2 3  
0 4  
5 -1  
1 1

**Output :**

13 5  
45 33  
77 61

**Inference :**

Here multiplication of two matrices is done through the the user's input of row and column value of both the matrices within the nested for loop.and othe methods used here are split()).

## Week - 10 ( Tuples )

### Statement :

Take two inputs from the user one is to create a tuple and another one is an integer n. Write a program to print the tuple n times.

Create another tuple with the user given elements and concatenate both the tuples and print the result as shown in the example.

### Objective :

To get familiar with concatenating two tuples.

#### Code :

```
e=int(input('Enter elemnent in tuple : '))
n=int(input('Enter no. times to be repeated : '))
t=(e,)
r=t*n
print(r)
e2=int(input('Enter element in another tuple : '))
t2=(e2,)
print('The concatenation of these two tuples are : ',r+t2)
```

#### Input :

Enter elemnent in tuple : 2  
Enter no. times to be repeated : 5  
(2, 2, 2, 2, 2)  
Enter element in another tuple : 5

#### Output :

The concatenation of these two tuples  
are : (2, 2, 2, 2, 2, 5)

### Inference :

A tuple should have commas after every element so if we have a single element it will take it as a string / number / float. To rectify this we will keep comma after the element i.e., ( e, )

**Statement :**

Write a program to add an element to a tuple based on the user-given value in a specific index, print the result as shown in the example. If the index is not in the range print the error message

**Objective :**

To get familiar with inserting an element into a tuple.

**Code :**

```
t=()
e=(int(i) for i in input('Enter elements to tuple : ').split())
for j in e:
    t=t+(j,)
print('Tuple : ',t)
n=int(input('Enter element to be inserted : '))
p=int(input('Enter position to be inserted : '))
if p > len(t):
    print('--- ERROR ---\n')
else:
    r=t[:p-1]+(n,)+t[p-1:]
print('The new tuple is : ',r)
```

**Input :**

Enter elements to tuple : 2 6 8  
Tuple : (2, 6, 8)  
Enter element to be inserted : 4  
Enter position to be inserted : 2

**Output :**

The new tuple is : (2, 4, 6, 8)

**Inference :**

The concept of slicing is similar to lists .to concatenate 2 tuples, simply we can use the ' + ' operator. We assign the inserted element into a new tuple after slicing.

**Statement :**

Create a tuple with the user given elements. Write a program to remove an element from the input tuple, print the result as shown in the example. If the element is not present in the tuple print the error message.

**Objective :**

To get familiar with deleting an element in a tuple.

**Code :**

```
t=()
k=0
e=(int(i) for i in input('Enter elements to tuple : ').split())
for j in e:
    t=t+(j,)
print('Tuple : ',t)
d=int(input('Enter element to be deleted : '))
for i in range(len(t)):
    if d is t[i]:
        t=t[:i]+t[i+1:]
        k=1
        break
if k==1:
    print('The new tuple is : ',t)
else:
    print('--- ERROR ---\n')
```

**Input :**

Enter elements to tuple : 1 3 4 5 7  
Tuple : (1, 3, 4, 5, 7)  
Enter element to be deleted : 4

**Output :**

The new tuple is : (1, 3, 5, 7)

**Inference :**

Similar to previous question but instead of adding we will not consider it and to know whether the element that need to be deleted is in the tuple we use a Boolean value i.e., k = 0.

**Statement :**

Create a tuple with the user given inputs. Take an index n from the user. Write a program to print the tuple element at index n, print the result as shown in the examples. If the given index range is not in the tuple print the error message

**Objective :**

To get familiar to display an element in a tuple.

**Code :**

```
t=()
e=(int(i) for i in input('Enter elements to tuple : ').split())
for j in e:
    t=t+(j,)
print('Tuple : ',t)
i=int(input('Enter the index to display the element : '))
if i<len(t):
    print('The element in the tuple is : ',t[i])
else:
    print('--- ERROR ---\n')
```

**Input :**

Enter elements to tuple : 1 2 3 4 5  
Tuple : (1, 2, 3, 4, 5)  
Enter the index to display the element : 4

**Output :**

The element in the tuple is : 5

**Inference :**

Here if the index given is not less than the tuple size then it prints ERROR as the elements are indexed from 0 to ( size - 1 ) so if we give more than this range it cannot take any inputs.

**Statement :**

Take two integers start index and end index as input from the console using input () function. Write a program to find the tuple elements within start index and end index, print the result as shown in the examples.

**Objective :**

To get familiar to display the elements between 2 inputs as index's in a tuple.

**Code :**

```
t=()
e=(int(i) for i in input('Enter elements to tuple : ').split())
for j in e:
    t=t+(j,)
print('Tuple : ',t)
s=int(input('Enter the starting index : '))
e=int(input('Enter the ending index : '))
print('The elements in between the index\'s are : ',t[s+1:e])
```

**Input :**

Enter elements to tuple : 1 2 3 4 5 6 7  
Tuple : (1, 2, 3, 4, 5, 6, 7)  
Enter the starting index : 1  
Enter the ending index : 5

**Output :**

The elements in between the index's  
are : (3, 4, 5)

**Inference :**

Here we use the concept of tuple slicing and the arguments are given as the starting and ending index's

## **WEEK – 11 ( Dictionaries )**

### **Statement :**

The text is given in a single line. For each word of the text count the number of its occurrences before it. A word is a sequence of non-whitespace characters. Two consecutive words are separated by one or more spaces. Punctuation marks are a part of a word, by this definition.

### **Objective :**

To get familiar with usage of dictionaries and functions like `get()`, `split()` etc.,

#### **Code :**

```
counter = {}  
for word in input().split():  
    counter[word] = counter.get(word, 0) + 1  
print(counter[word] - 1, end=' ')
```

#### **Input :**

one two one two one

#### **Output :**

00112

### **Inference :**

An empty dictionary can be created using the function `dict()` or an empty pair of curly braces `{}` (this is actually the reason the curly braces cannot be used to create an empty set ).



**Statement :**

You are given a dictionary consisting of word pairs. Every word is a synonym the other word in its pair. All the words in the dictionary are different. After the dictionary there's one more word given. Find a synonym for him.

**Objective :**

To get familiar with applications of dictionaries .

**Code :**

```
n = int(input())
d = {}
for i in range(n):
    first, second = input().split()
    d[first] = second
    d[second] = first
print(d[input()])
```

**Input :**

```
3
Hello Hi
Bye Goodbye
List Array
Goodbye
```

**Output :**

```
Bye
```

**Inference :**

From this program I am able to find out the synonym of the word mentioned after the dictionary by using the dictionaries data structure in python.

**Statement :**

As you know, the president of USA is elected not by direct vote, but through a two-step voting. First elections are held in each state and determine the winner of elections in that state. Thereafter, the state election is going: in this election, every state has a certain the number of votes — the number of electors from that state. In practice, all the electors from the state of voted in accordance with the results of the vote within a state the first line contains the number of records. After that, each entry contains the name of the candidate and the number of votes they got in one of the states. Count the total results of the elections: sum the number of votes for each candidate. Print candidates in the alphabetical order.

**Objective** : To print the candidates and their votes in alphabetical order

**Code :**

```
num_votes = {}
for _ in range(int(input())):
    candidate, votes = input().split()
    num_votes[candidate] = num_votes.get(candidate, 0) + int(votes)
for candidate, votes in sorted(num_votes.items()):
    print(candidate, votes)
```

**Input :**

```
5
McCain 10
McCain 5
Obama 9
Obama 8
McCain 1
```

**Output :**

```
McCain 16
Obama 17
```

**Inference :**

From this program I am able to arrange the elements in a dictionary in alphabetical order and can add same elements and print their result.

**Statement :**

Given the text: the first line contains the number of lines, then given the lines of words. Print the word in the text that occurs most often. If there are many such words, print the one that is less in the alphabetical order

**Objective :**

To print the most occurred word in the given words by iterating the dictionary.

**Code :**

```
counter = {}
for i in range(int(input())):
    line = input().split()
    for word in line:
        counter[word] = counter.get(word, 0) + 1
    max_count = max(counter.values())
    most_frequent = [k for k, v in counter.items() if v == max_count]
print(min(most_frequent))
```

**Input :**

1  
apple orange banana banana orange

**Output :**

banana

**Inference :**

From this program I am able to iterate the dictionary and find the most occurred word among the given words. In this min(),max(),get() functions were used.

**Statement :**

The virus attacked the filesystem of the supercomputer and broke the control of access rights to the files. For each file there is a known set of operations which may be applied to it:

write W,  
read R,  
execute X.

The first line contains the number N — the number of files contained in the filesystem. The following N lines contain the file names and allowed operations with them, separated by spaces. The next line contains an integer M — the number of operations to the files. In the last M lines specify the operations that are requested for files. One file can be requested many times.

You need to recover the control over the access rights to the files. For each request your program should return OK if the requested operation is valid or Access denied if the operation is invalid.

**Objective** : To iterate the dictionary by giving the key word and finding whether the access is given or denied.

**Code :**

```
ACTION_PERMISSION = {  
    'read': 'R',  
    'write': 'W',  
    'execute': 'X',  
}  
file_permissions = {}
```

```

for i in range(int(input())):
    file, *permissions = input().split()
    file_permissions[file] = set(permissions)
for i in range(int(input())):
    action, file = input().split()
    if ACTION_PERMISSION[action] in file_permissions[file]:
        print('OK')
    else:
        print('Access denied')

```

**Input :**

4  
 helloworld.exe R X  
 pinglog W R  
 nya R  
 goodluck X W R  
 5  
 read nya  
 write helloworld.exe  
 execute nya  
 read pinglog  
 write pinglog

**Output :**

OK  
 Access denied  
 Access denied  
 OK  
 OK

**Inference :**

Two for loops have been used in order to obtain the required output. if-else condition is used to print the required output.

**Statement :**

Given a list of countries and cities of each country. Then given the names of the cities. For each city specify the country in which it is located.

**Objective** : To iterate the dictionary by giving the key word and print the country in which given city is located.

**Code :**

```
n = int(input())
cities = {}
for _ in range(n):
    line = input().split()
    for city in line[1:]:
        cities[city] = line[0]
for _ in range(int(input())):
    print(cities[input()])
```

**Input :**

```
2
USA Boston Pittsburgh Washington
Seattle
UK London Edinburgh Cardiff Belfast
3
Cardiff
Seattle
London
```

**Output :**

```
UK
USA
UK
```

**Inference :**

Cities is declared as a dictionary and city consists the cities mentioned. If these are interchanged while writing the code we will obtain the wrong output.

**Statement :**

Given a number n, followed by n lines of text, print all words encountered in the text, one per line. The words should be sorted in descending order according to their number of occurrences in the text, and all words with the same frequency should be printed in lexicographical order

**Objective :**

To sort the words in descending order based on their occurrence.

**Code :**

```
counter = {}
for i in range(int(input())):
    line = input().split()
    for word in line:
        counter[word] = counter.get(word, 0) + 1
for i in sorted(counter.items(), key=lambda x: (-x[1], x[0])):
    print(i[0])
```

**Input :**

```
9
hi
hi
what is your name
my name is bond
james bond
my name is damme
van damme
claude van damme
jean claude van damme
```

**Output :**

```
damme
is
name
van
bond
claude
hi
my
james
jean
what
your
```

**Inference :**

Every word is counted and stored its occurrences and finally all the words are sorted in descending order based on the no. of occurrences.

**Statement :**

One day, going through old books in the attic, a student Bob found English-Latin dictionary. By that time he spoke English fluently, and his dream was to learn Latin. So finding the dictionary was just in time. Unfortunately, full-fledged language studying process requires also another type of dictionary: Latin-English. For lack of a better way he decided to make a second dictionary using the first one. As you know, the dictionary consists of words, each of which contains multiple translations. For each Latin word that appears anywhere in the dictionary, Bob has to find all its translations (that is, all English words, for which our Latin word is among its translations), and take them and only them as the translations of this Latin word. Help him to create a Latin-English. The first line contains a single integer N — the number of English words in the dictionary. Followed by N dictionary entries. Each entry is contained on a separate line, which contains first the English word, then a hyphen surrounded by spaces and then comma-separated list with the translations of this English word in Latin. All the words consist only of lowercase English letters. The translations are sorted in lexicographical order. The order of English words in the dictionary is also lexicographic. Print the corresponding Latin-English dictionary in the same format. In particular, the first word line should be the lexicographically minimal translation of the Latin word, then second in that order, etc. Inside the line the English words should be sorted also lexicographically.

**Objective :** To make a Latin to English dictionary.

**Code :**

```
from collections import defaultdict
latin_to_english = defaultdict(list)
for i in range(int(input())):
    english_word, latin_translations_chunk = input().split(' - ')
    latin_translations = latin_translations_chunk.split(', ')
    for latin_word in latin_translations:
        latin_to_english[latin_word].append(english_word)
print(len(latin_to_english))
for latin_word, english_translations in sorted(latin_to_english.items()):
    print(latin_word + ' - ' + ', '.join(english_translations))
```



**Input :**

3

apple - malum, pomum, popula

fruit - baca, bacca, popum

punishment - malum, multa

**Output :**

7

baca - fruit

bacca - fruit

malum - apple, punishment

multa - punishment

pomum - apple

popula - apple

popum - fruit

**Inference :**

From this program i am able to create a latin to english dictionary using dictionaries data structure.

## **WEEK – 12(Files)**

### **Statement:**

Write a program to print each line of a file in reverse order

### **Objective:**

To create a file and to print lines in the file in the reverse order.

#### **Code :**

```
print("Enter the Name of File: ")
fileName = input()
fileHandle = open(fileName, "w")
fileHandle = open(fileName, "r")
fileContent = ""
for content in fileHandle:
    fileContent = fileContent+content

print("\n----Content in Reverse Order----")
fileContent = fileContent[::-1]
print(fileContent)
```

#### **Input :**

Enter the Name of File:  
ravi.txt

#### **Output :**

----Content in Reverse Order----

### **Inference:**

From this program, I am learn how to print the content of the file on the output tab in reverse order. I got to know now to open a file and using ::-1 can print from reverse order. i.e., last line first .... first line last.

**Statement:**

Write a program to print the number of lines, words and characters in a file

**Objective:**

To get Familiar with file operations and print the number of lines, words and char in a file

**Code :**

```
l,w,c=0,0,0
fN=input("Enter the Name of File : ")
file=open(fN,"r")

for line in file:
    wordlist = line.split()
    l+=1
    w+=len(wordlist)
    c+=len(line)

print("\nNumber of lines present in given file : ",l)
print("Number of words present in given file : ",w)
print("Number of characters present in given file : ",c-1)
```

**Input :**

Enter the Name of File : ravi.txt  
(Hello world!  
How are you)

**Output :**

Number of lines present in given file : 2  
Number of words present in given file : 5  
Number of characters present in given file : 23

**Inference :**

From this program, I am able to learn how to print the number of lines, words And Characters in a file using for loop for counting n no of lines **For line in file** (is a recursive statement until the lines end) and closing the file and printing characters and words in a file.

**Statement :**

Write a Python program to copy one file to another file.

**Objective :**

To copy the content from one file to another file.

**Code :**

```
sFile = input("Enter the Name of Source File: ")
tFile = input("Enter the Name of Target File: ")

file1 = open(sFile, "r")
text = file1.readlines()

file2 = open(tFile, "w")
for s in text:
    file2.write(s)

print("\nFile Copied Successfully!")
```

**Input :**

Enter the Name of Source File: ravi.txt  
Enter the Name of Target File: kiran.txt

**Output :**

File Copied Successfully!

**Inference :**

From this program, I am able to learn how to copy the content of a existing file to a new file. I got to know different modes of opening a file. I got to know how to use for loop.